Improving drag-and-drop on wall-size displays

Maxime Collomb, Mountaz Hascoët
LIRMM, UMR 5506 CNRS
Univ. Montpellier II
Montpellier, France
{collomb, mountaz}@lirmm.fr

Patrick Baudisch
Microsoft Research
One Microsoft Way
Redmond, WA 98102
baudisch@microsoft.com

Brian Lee
Stanford Computer Graphics
Laboratory, Stanford University
California, United States
balee@graphics.stanford.edu

Abstract

On wall-size displays with pen or touch input, users can have difficulties reaching display contents located too high, too low, or too far away. Drag-and-drop interactions can be further complicated by bezels separating individual display units. Researchers have proposed a variety of interaction techniques to address this issue, such as extending the user's reach (e.g., *push-and-throw*) and bringing potential targets to the user (*drag-and-pop*). In this paper, we introduce a new technique called push-and-pop that combines the strengths of push-and-throw and drag-and-pop. We present two user studies comparing six different techniques designed for extending drag-and-drop to wall-size displays. In both studies, participants were able to file icons on a wall-size display fastest when using the push-and-pop interface.

Keywords: drag-and-pop, push-and-throw, wall-size display, drag-and-drop, pen input, touch-screen, interaction technique.

1 Introduction

With the emergence of wall-size displays (e.g., Liveboard [6] and SMART Board™), touch and pen input have regained popularity. Over the past years, more complex display systems have been created by combining multiple display units into wall-size display walls, such as the DynaWall [25], the iRoom SMART Board wall (iWall [14], shown in Figure 4), as well as display systems based on stitched projection, such as the Information Mural [11].

Touch/pen input requires users to physically display content in order to interact with it. This can become a problem when targets are out of reach, e.g., because they are located too high or too low or on a display unit that does not support touch/pen input [3]. Accessing icons located far away from the user may require users to physically walk over, requiring a target acquisition time roughly linear to distance [9]. Interactions that involve dragging objects tend to be particularly error-prone [22] and can be complicated further by the bezels separating screen units [3].

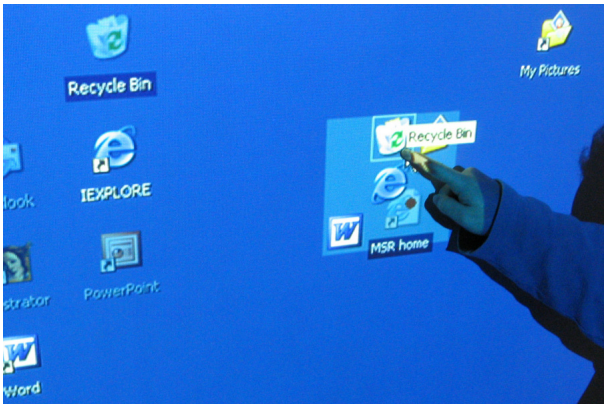

Figure 1: A user is dragging a web page icon into the recycle bin on a wall-size display. The proposed push-and-pop interaction technique has created a world-in-miniature around the user's finger that contains all valid target icons.

Researchers have proposed a variety of interaction techniques that simplify drag-and-drop from and to inaccessible screen locations, across long distances, and across display unit borders. Approaches include extending the user's reach (e.g., *push-and-throw* [12]) and bringing potential targets to the user (*drag-and-pop* [3], shown in Figure 4). Both techniques have their particular strengths, while facing their particular limitations, as we will discuss in detail in sections 3 and 4. We also present improvements addressing some of these limitations.

We then present a novel technique we call *push-and-pop* (Figure 1) that combines the world-in-miniature aspect from push-and-throw with the target-to-pointer approach of drag-and-pop. In two experimental comparisons with five competing drag-and-drop extension techniques, participants performed tasks fastest when using the push-and-pop interface.

2 Related work

Drag-and-drop is a well-known interaction technique for transferring or copying information using a pointing device, while avoiding the use of a hidden clipboard [26]. Several techniques have been proposed to adapt drag-and-drop, initially invented for use with a desktop-sized computer screen, to large screens.

The majority of work so far has focused on drag-and-drop with direct input devices, such as the mouse. Hyperdragging [21] allows extending drag-and-drop across physically separate displays. Manual And Gaze Input Cascaded (MAGIC) pointing [27] helps overcoming long distances by combining mouse input with gaze input. High-density cursors help users keep track of the mouse pointer during long-distance traversals [2]. Semantic pointing [4] and vector pointing [9] use a multi-scale navigation approach to allow users to cross very long distances with logarithmic access time [10].

Laser pointers [19] have been discussed for input on wall-size displays, but were found to be slower than touch input [18].

Techniques compatible with pen usage are based on a variety of different approaches. Pick-and-drop [22] and take-and-put [25] are based on point-and-click, but, unlike traditional drag-and-drop, they do not require users to maintain contact with the screen. The Frisbee technique by Kahn et al. [7] allows users to create a local view into a distant area of the screen, thereby allowing users to drag objects to arbitrary targets. These techniques also help users overcome obstacles, such as bezels separating display units. Marking menus [15] allow a user to perform a menu selection by either popping-up a radial (or pie) menu, or, for an experienced user, by making a straight mark in the direction of the desired menu item without popping-up the menu.

Other techniques offer additional functionality for interacting with targets that are out of reach. Throwing, for example, allows users to accelerate an object with a small gesture; the object then continues its trajectory based on its inertia [8]. The imprecision of human motor skills has prevented throwing from being used for reliable target acquisition.

Push-and-throw [12] and drag-and-pop [3] also fall into this category. In this paper, we focus on these two techniques and present extensions and design improvements. We describe these techniques in detail in the following section.

3 Push-and-throw and drag-and-pop

3.1 Drag-and-throw & push-and-throw

Drag-and-throw [12] was designed to address the limitations of throwing [8] by providing users with a real-time preview of where the dragged object will come down if thrown. This way, drag-and-throw allows users to tweak the throwing trajectory in order to assure that the right target is hit. Snap-to-target helps increase users' accuracy.

Push-and-throw [12] is a follow-up version of drag-and-throw. By letting users drag towards the target rather than away from it, push-and-throw was found to offer better affordance [12].

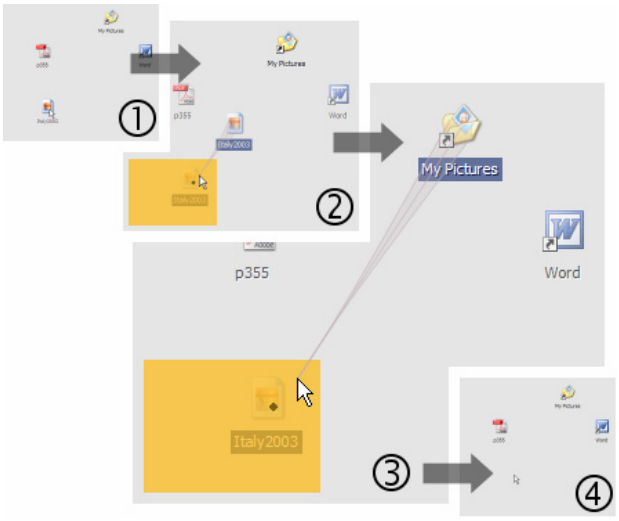

Figure 2: Push-and-throw walkthrough: the user is dropping the image icon located in the bottom left into the "My Pictures" folder located at the top right.

Figure 2 shows a walkthrough of push-and-throw. (1→2) Tapping the pen onto an icon causes the push-and-throw visuals to appear. The translucent rectangle, called the *take-off area*, represents a miniature of the desktop space. A translucent copy of the icon (the "tip icon") appears. It is connected to the cursor using a rubber band, which may be thought of as preview of the trajectory along which the icon will be "thrown" when released. Dragging the pen inside the take-off area causes the tip icon to move to the respective location on the real desktop. (3) As the user moves the tip icon over a target icon, such as a folder, the target icon provides the usual visual feedback, i.e., it changes its color. (4) When lifting the pen, the icon is filed.

Push-and-throw, originally inspired by the metaphor of the pantograph [5] is a combination of a go-go [20] and a world in miniature technique [24]. The main idea behind push-and-throw is to temporarily turn the pen/touch input, inherently a direct pointing device, into an indirect pointing device in order to traverse distances faster and to be able to reach locations further away or on a different screen unit. On the downside, it also leads to reduced resolution and accuracy.

Design improvements: rubber bands & acceleration

While the original version of push-and-throw [12] used simple lines to connect pointer and dragged icon, we created an improved version that uses a tapered rubber band inspired by drag-and-pop [3] as shown in Figure 2. As Baudisch et al. discuss, the shape of this type of rubber band provides users with an additional visual cue about distance. Unlike the rubber bands proposed by Baudisch et al., we used an asymmetric rubber band for push-and-throw, which looked more consistent

when linking the pointer (1 pixel wide) to an icon (32 pixels wide). This improved design was one of the interfaces compared in our user studies presented in Section 6 and 7.

One of the main limitations of the original push-and-throw is its lack of precision due to the size reduction that occurs when mapping the desktop to the take-off area. We address this issue by introducing non-linear acceleration, as it is common with indirect input devices, to push-and-throw. In *accelerated push-and-throw*, moving the pointer slowly results in a much slower motion of the dragged icon, helping users acquire small targets. In addition, the acceleration factor is reduced when the dragged icon is close to a target (similar to semantic pointing [4]). Accelerated push-and-throw also allows *clutching*, i.e., lifting and repositioning the pen/finger within a drag interaction. This allows users to reach very distant targets.

With acceleration, there is no more immediate correspondence between physical pointer location and the location of the dragged icon. As a consequence, the technique does not have a clearly defined take-off area anymore and we cannot provide a preview of it. Since accelerated push-and-throw therefore does not completely subsume tradition push-and-throw, we decided to include both designs in our user study.

3.2 Drag-and-pop

Drag-and-pop [3] uses the opposite approach to drag-and-throw. Rather than sending the dragged object to the periphery, it allows users to bring a selection of likely candidates to the user. This allows users to complete drag interactions in a convenient screen location. There is no scaling of pointer motion, so users can make use of the full resolution of their motor skills.

Figure 3 shows a walkthrough of *drag-and-pop*. (1) The user intends to delete a web page by dragging it into the recycle bin. (2) As the user starts dragging the web pages icon towards the recycle bin, icons that are of compatible type and located in the direction of the user's drag motion "pop up". This means that each of these icons produces a "tip icon" that appears in front of the user's pen. Tip icons are connected to the respective original icon using a rubber band. (3) The user drags the web page over the recycle bin and releases the mouse button. The recycle bin accepts the web page. Alternatively, the user could have dropped the web page over the word processor or the web browser icon, which would have launched the respective application with the memo. (4) When the user drops the icon, all tip icons disappear instantly.

In order to reduce clutter, drag-and-pop creates tip icons only for icons that are of matching file type, located far enough away from the dragged icon, and located within a certain angle from the user's initial drag direction. Drag-and-pop compacts the layout of all tip icons by placing tip icons on a denser grid and by eliminating empty rows and columns from that grid [3]. Users can abort drag-and-pop interactions at any time by moving the pen away from the tip icon cluster. This allows users to rearrange icons on the desktop. The rubber bands connecting tip icons with their original are designed to help users follow the transition when the tip icons appear and to *re*-identify the desired targets among the other tip icons. Drag-and-pop can be extended to allow users to access content in the periphery (drag-and-pick [3]).

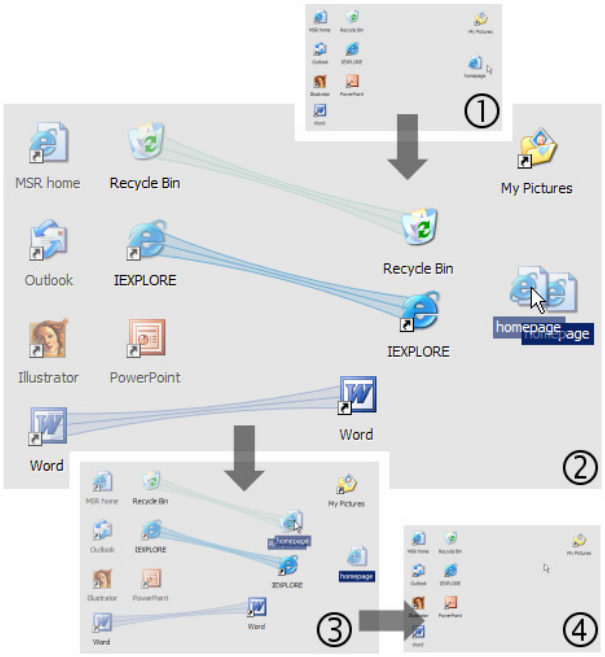

Figure 3: Drag-and-pop: Here the user drops the word file located at the right into the recycle bin.

The main limitation of drag-and-pop is that imprecise invocation gestures can cause the wrong tip icons to appear. In particular, Baudisch et al. found [3] that the arc-shaped full-arm drag motions users performed caused drag-and-pop to bring icons located in the extension of the first segment of that arc—this was typically *not* the direction to the target.

Design improvements: target sector and positioning

In order to address the limitations identified by Baudisch et al., we adjusted our version of drag-and-pop in two ways. First, we increased the size of the target sector and added extra tolerance for movements towards the top of the screen.

Second, in its original version, dragging towards another display unit sometimes makes the tip icon cluster appear fully or partially on that other screen unit.

The version of drag-and-pop used in our user study avoids this—it always places tip icon clusters in the display unit where the drag interaction was initiated.

Figure 4: Moving an object using the drag-and-pop technique on the iWall wall-size display.

4 Analysis and comparison of approaches

As listed in the previous section, push-and-throw and drag-and-pop have different strengths, but they also have different limitations. In order to allow creating a new technique that overcomes the limitations of both approaches, we take a closer look at these limitations and at the responsible design dimensions (Table 1).

4.1 Index of difficulty

All other factors kept constant, users can acquire closer targets faster than more distant targets (Fitts' law [17]). Push-and-throw and drag-and-pop both reduce the distance to the target in motor space. Push-and-throw amplifies pointer speed by a constant factor, but at the same time scales targets in motor space, so the only aspect that has an effect on the index of difficulty are snapping and acceleration. Drag-and-pop, in contrast, leaves target size unchanged. Packing targets more tightly therefore reduces the index of difficulty.

While index of difficulty does play a role in target acquisition in general, on wall-size displays the main factors are whether the target is in reach and how many bezels need to be crossed. In comparison to these factors, the impact of actual distance was found to be minor [3] and should therefore be expected to have only minor impact on the relative performance of push-and-throw with regard to drag-and-pop.

4.2 Need for reorientation

Push-and-throw-based techniques move the pointer all the way to the target, while drag-and-pop first moves potential targets to the pointer (Table 1). The difference, however, is merely a matter of the applied visuals. The underlying mechanism is similar: in motor space the user moves the pointer to a target that is within reach. In the case of push-and-throw the visuals appear in the target space, i.e., all over the wall-size display. In the case of drag-and-pop the visuals appear in the motor space, i.e., in the space reachable by the user.

The different visuals, however, have an impact on the interaction. Drag-and-pop involves a single fairly dramatic movement on the screen that requires users to reorient themselves. Drag-and-pop uses the rubber bands to minimize that impact, yet since different drag directions cause the tip cluster to be a little different every time, users need to pay attention when picking their target tip icon. Once users have identified the target tip icon, however, they can complete the interaction easily: the target is at a stable location and acquiring it requires only very little attention.

Push-and-throw, in contrast, requires users to constantly monitor the screen as it is virtually impossible for users to guess upfront where their finger has to be in order to acquire the target.

In our observation, the single motion caused by drag-and-pop impacts performance less than the continuous monitoring required by push-and-throw. We therefore paid close attention to avoid the need for continuous monitoring when designing *push-and-pop.*

technique	approach	need to reorient
drag-and-drop	—	never
pick-and-drop	—	never
push-and-throw	to target	constantly
accelerated…	to target	constantly
drag-and-pop	to pointer	once
New: push-and-pop	to pointer	once, later never

Table 1 : candidate techniques and design dimensions

5 Push-and-pop

Based on our analysis of push-and-throw and drag-and-pop, we created a new technique designed to combine the strengths of both techniques. We call this new technique *push-and-pop*—the name trying to convey that it builds on both *push-and*-throw and drag-and-*pop.*

Figure 5 shows a walkthrough. In the shown example, the user is dragging a word document into the recycle bin. The interaction proceeds as follows. (1) The user starts dragging the word document icon. (2) As a response, the system surrounds the pointer with a miniature representing the wall-size display—the *take-off area*—here containing the icons of the word application, a folder, and the recycle bin. (3) The user drags the word document icon over the recycle bin, which responds by highlighting itself with a rectangular frame. (4) The users lets go of the word document icon and the word document disappears in the recycle bin. The take

off area disappears. Figure 1 shows a photo of push-and-pop in use on a wall-size display.

In case users need to rearrange icons on the desktop, they can switch push-and-pop temporarily into a push-and-throw mode. Users invoke this functionality by moving the pointer back to the location of invocation, marked with a black circle in Figure 5.3.

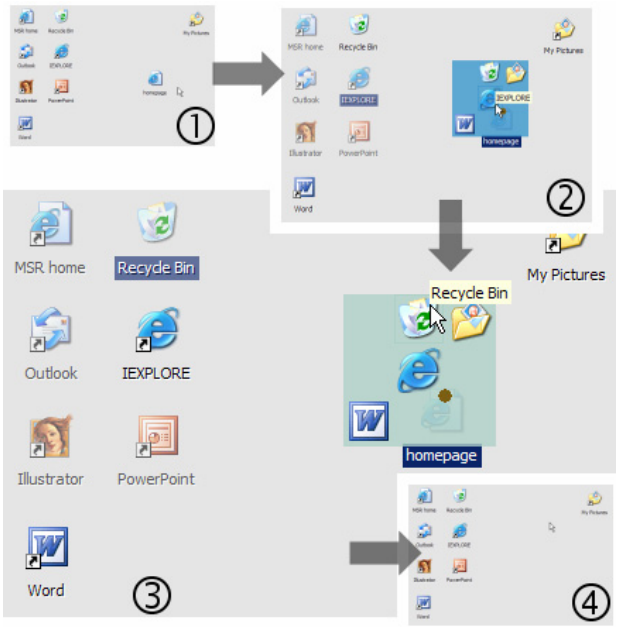

Figure 5: Push-and-pop walkthrough

The grid-like arrangement of the tip icons in the take-off area is taken directly from the drag-and-pop's layout algorithm [3]. It is created by placing icons on a small grid and by removing empty rows and columns. Unlike drag-and-pop which creates tip icons only for icons located in the drag direction, however, push-and-pop brings all target icons of matching type, independent of where they are located on the screen. This eliminates the risk of users invoking the wrong set of targets and also assures a stable, reproducible layout, thereby overcoming the two main limitations of drag-and-pop. Invocation of push-and-pop over the same icon type always results in the same take-off area, allowing users to perform the actual acquisition task based on muscle memory.

The resulting rectangular take-off area corresponds to the take-off area in push-and-throw. However, push-and-pop's take-off area offers two major benefits. First, the take-off area shows tip icons. This offers good readability even if the represented target is too far away to be readable. But most importantly it allows users to acquire the desired tip icon without the need for further reorientation. Second, rather than being a geometrically reduced version of the display, the miniature in the take-off is a *semantically* reduced version of the display in that only valid targets are contained. This allows push-and-pop to use full-size versions of the target icons allowing for an easier acquisition. However, to save space, we removed file names, instead revealing them as tool tips on hover.

6 First study: double wall-size screen

We conducted a user study comparing six drag-and-drop techniques for wall-size displays. The study served two main purposes. First, we wanted to learn more about the relative performance of the different techniques. We had several hypotheses. For sufficiently long distances, we expected all techniques to outperform traditional drag-and-drop. More specifically, we expected the techniques that required no or a one-time reorientation to outperform the techniques that required continuous tracking. Second, we wanted to validate the design of push-and-pop. Would it really outperform push-and-throw and drag-and-pop?

In order to extend our findings to longer distances, we later replicated the study on a three-unit display wall, as we report in Section 7. We will hold off with our discussion until after the second study.

6.1 Task

Participants' task was to perform drag-and-drop operations on a simulated Windows desktop. The task details corresponded largely to the original drag-and-pop user study reported in [3]. Figure 6 shows the icon layout used in the study. The icons to be filed appeared at the bottom right of the screen (the cluster of 10 icons at the bottom right of Figure 6). The target was successively displayed in one of 12 positions, which allowed us to obtain uniformly distributed indexes of difficulty. Unlike the original drag-and-pop study, we simplified the participants' task by always using the recycle bin icon as the target.

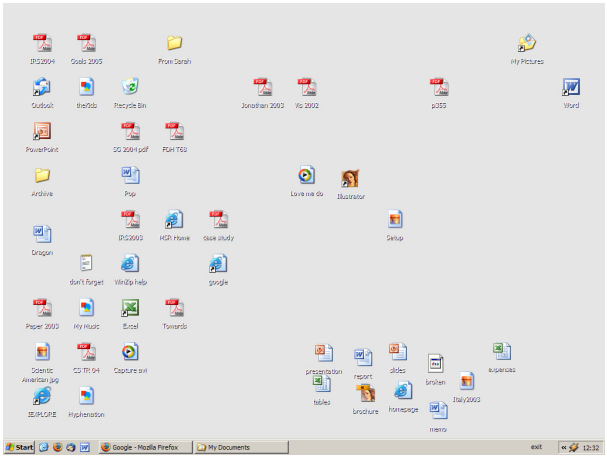

Figure 6: The icon layouts used in the study.

6.2 Apparatus & interfaces

The study was conducted on a combination of a wall-size back-projected display and a front-projected SMART Board™. Each of the two displays measured 5'/1.7m across and ran at 1024*768 pixel resolution. Pen input on the back-projected display was supported by a Mimio™ capture bar and a specialized pen. The Wall-screens were connected to a PC with a Pentium4 1.5GHz processor and 512MB of RAM.

Participants used six different interfaces: drag-and-drop, pick-and-drop, push-and-throw, accelerated push-and-throw, drag-and-pop, and push-and-pop. The pick-and-drop interface corresponded to [22] (A first click selects the object that has to be moved and a second one selects the target) with the difference that the interface in the study required users to briefly drag an icon in order to pick it up. This allowed differentiating drag operations from object selection.

6.3 Participants

Twelve participants (all male, one left-handed, students and researchers) were recruited internally. All participants but one had little or none experience with using wall-size displays.

6.4 Experimental design

We used a within-subjects design. Each participant performed 36 trials per interface, resulting in a total of 216 trials. The 36 trials were grouped in three blocks of 12 trials, with each trial corresponding to a different target positions. A Latin square was used to counterbalance order of interfaces and order of presentation. Target positions within each block were randomized. Participants received up to 5 minutes of training per interface before beginning the timed tasks.

We measured *Movement Time*, i.e., the time from the moment the participant tapped onto the icon to be filed to the moment the participant lifted the pen. We also measured *Error Rate*, i.e., the percentage of cases where the user released the dragged icon over the wrong target. In cases where participants accidentally released the dragged object over empty space, they had to pick it up again and complete the trial. We kept track of that as well.

6.5 Results

Task completion time: An ANOVA on median values for time showed significant differences with the type of technique ($p<0.001$). A Dunn's pair-to-pair comparison showed that all comparisons were significant except the comparison of accelerated push-and-throw and pick-and-drop.

Overall, participants performed the task fastest when using the push-and-pop interface, confirming our hypothesis. Drag-and-pop was only slightly slower. Next came accelerated push-and-throw and pick-and-drop. Participants performed worse when using push-and-throw. Confirming the findings by Baudisch et al. [3], drag-and-drop worked well as long as the target was located within the same display unit, but performed poorly when the task required participants to cross the bezel between screens. Averaged across distance, traditional drag-and-drop performed worst.

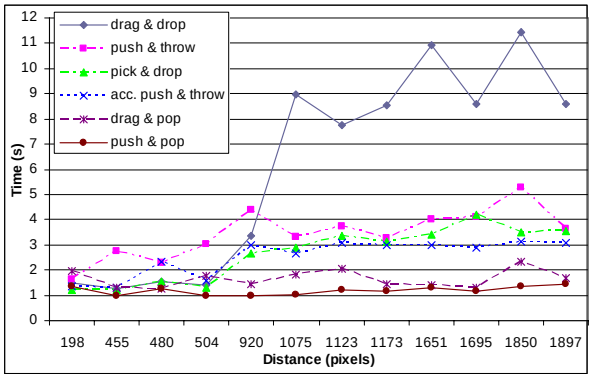

Figure 7: Mean task completion time for each technique depending of the ID of the task.

Errors rates: Figure 8a shows error rates. The number of cases where participants dropped an icon into the wrong target was generally low (less than two cases out of 36 trials per interface). The number of cases where participants temporarily dropped an icon, but then successfully filed it was much higher for push-and-throw and drag-and-pop (between 9% and 12%). Pick-and-drop, push-and-pop and accelerated push-and-throw offered much better results here (below 2%). Drag-and-drop does not count here, because participants had to temporarily drop the icon whenever crossing the bezel.

Subjective satisfaction: At the end of the experiment, participants ranked all six techniques by preference. Figure 8b shows a summary created by assigning 5 points for each first ranks, down to 0 points for last rank and averaging the results. In particular, 9 of the 12 participants ranked push-and-pop first.

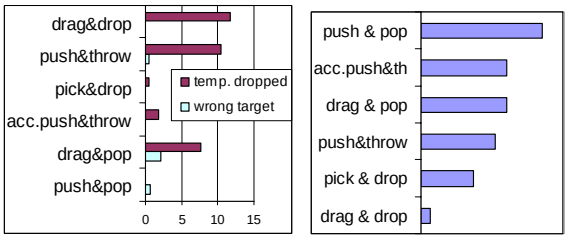

Figure 8: (a) Error rates for each technique. (b) Subjective preferences (higher is better)

7 Second study: triple wall-size screen

We replicated the study on a corresponding three-unit display wall in order to extend our findings to longer distances.

The second study was run on the iWall [14], a three back-projection SMART Board™ displays, each with resolution 1024x768 (Figure 4). Participants interacted with the board via touch, either using their fingers directly or holding a pen. The display was driven by a Pentium® II 500MHz with 256MB of RAM.

Six volunteers (4 females, 2 left-handed) participated in this study. All participants but one had very little experience with using wall-size displays.

The experimental design was identical to the first study, but instead of 3 blocks of 12 trials, participants performed 4 blocks or 12 trials per interface.

7.1 Results

Task completion time: An ANOVA on median task times showed significant differences with the type of technique ($p<0.001$). A Dunn's pair-to-pair comparison found all differences significant except the comparison of push-and-throw and accelerated push-and-throw.

Figure 9 shows the results. The resulting ranking corresponds largely with the first study. Participants were fastest when using push-and-pop, followed by drag-and-pop, pick-and-drop, and drag-and-drop. In this study participants performed worst when using push-and-throw/accelerated push-and-throw. Only for very long distances did these techniques perform better than traditional drag-and-drop. Note drag-and-drop and pick-and-drop performance depended on the target distance, while the performance all other techniques is largely distance independent.

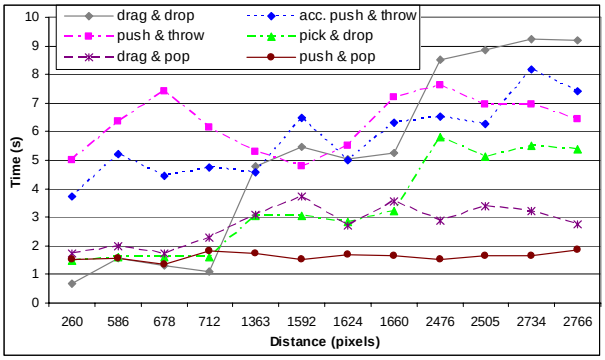

Figure 9: Mean task completion time for each technique depending of the ID of the task.

Error rates: Error rates corresponded to the first study with the exception that this time participants using accelerated push-and-throw performed a higher number of accidental drops (Figure 10a).

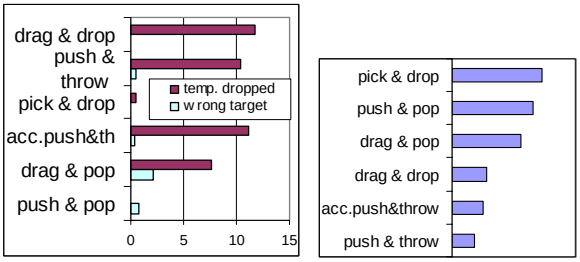

Figure 10: (a) Error rates for each technique. (b) User's subjective preferences (higher is better)

Subjective satisfaction: This time pick-and-drop scored slightly higher that push-and-pop (Figure 10b). The push-and-throw techniques scored last.

8 Discussion

The two user studies presented above provide some supporting evidence for the claims made earlier in this paper.

Confirming findings by Baudisch et al. [3] drag-and-drop performed well as long as source and target icons were situated in the same display unit, but failed quickly when long distances and bezels were involved. In addition, we found that pick-and-drop is affected by distance in a similar way, though to a lesser extent.

For all other evaluated techniques, target distance had comparably little impact on task performances. However, our studies seem to indicate a performance benefit of acquisition techniques that require a one-time reorientation over techniques that require continuous tracking. The two techniques that required only require a one-time reorientation (drag-and-pop and push-and-pop) achieved the best task times.

Overall, the study indicates that push-and-pop is indeed a useful technique. Push-and-pop outperformed all other techniques, including its ancestors, drag-and-pop and push-and-throw. Participants' subjective preference reflected this. Push-and-pop also offered a very low error rate, which is one possible explanation for the performance benefit in comparison with drag-and-pop. While participants using drag-and-pop needed to pay close attention to the directionality of their invocation gesture in order to avoid error, push-and-pop avoided this additional burden on users by always displaying all possible tip icons.

Among pointer-to-target techniques, accelerated push-and-throw performed significantly better than traditional push-and-throw. Despite the unexceptional performance of both techniques, this indicates that our design improvements were beneficial.

9 Conclusion

We presented an experimental comparison of six drag-and-drop techniques for wall-size displays and found

significant performance benefits for techniques that do not require users to continuously track their interaction, in particular the push-and-pop technique introduced in this paper.

We have focused on icon displacements. As future work, we plan to optimize the design of push-and-pop (e.g., by reintroducing rubber bands and solving scalability problems, such as those presented by desktops with numerous folders) and extending the presented techniques to other types of interactions, such as activation of menus and buttons.

Acknowledgements

We would like to thank the participants in our experiments, who spent significant amounts of time testing the interaction models.

References

[1] Accot, J and Zhai, S. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proc.CHI'97*, pp.295-302.

[2] Baudisch, P., Cutrell, E., and Robertson, G. High-Density Cursor: A Visualization Technique that Helps Users Keep Track of Fast-Moving Mouse Cursors. In *Proc. Interact'03*, pp. 236–243.

[3] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P. Bederson, B., and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch and Pen-operated Systems. In *Proc Interact'03*, pp. 57–64.

[4] Blanch, R., Guiard,Y., and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. CHI'04*, pp. 519–526.

[5] Coxeter, H. S. M. *Introduction to Geometry*, 2nd ed. NewYork: Wiley, pp. 69–70, 1969.

[6] Elrod, S., et. al. Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In *Proc. CHI'92*, pp. 599–607.

[7] Khan, A., Fitzmaurice, G. Almeida, D., Burtnyk, N., and Kurtenbach, G. A remote control interface for large displays. In *Proc. UIST'04*, pp. 127–136.

[8] Geißler, J. Shuffle, Throw or Take It! Working Efficiently with an Interactive Wall. In *CHI '98 Late-Breaking Results*, pp. 265–266.

[9] Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Vector Pointing: Object vs. Pixel Selection in Graphical User Interfaces. In *Proc. GI'04*, pp 9–16.

[10] Guiard, Y., Beaudouin-Lafon, M., Bastin, J., Pasveer, D., Zhai, S. View size and pointing difficulty in multi-scale navigation. In *Proc. AVI'04*, pp. 117–124

[11] Guimbretiere, F., M. Stone, and T. Winograd. Fluid interaction with High Resolution Wall-Size Displays. In *Proc. UIST'01*, pp. 21–30.

[12] Hascoët, M. (2003). Throwing models for large displays. In *Proc. HCI'03*, pp. 73–77.

[13] Johanson B., Hutchins, G., Winograd, T., and Stone, M. PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. In *Proc. UIST'02*, pp. 227–234.

[14] Johanson, B., Fox, A., and Winograd, T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing*, 1 (2), 67–74.

[15] Kurtenbach, G., Buxton, W. User learning and performance with marking menus. In *Proc. CHI'94*. pp. 258–264.

[16] MacKenzie, I.S., and Jusoh, S. An evaluation of two input devices for remote pointing. In *Proc. EHCI'01*, pp. 235–249.

[17] MacKenzie, I.S., Fitts' law as a research and design tool in human-computer interaction. *Human Computer Interaction, 1992*, 7, pp. 91–139.

[18] Myers, B., Bhatnagar, R., Nichols, J., Peck, C., Kong, D., Miller, R., and Long, C. Interacting At a Distance: Measuring the Performance of Laser Pointers and Other Devices. In *Proc CHI'02*, pp 33–40.

[19] Olsen, D. R. and Nielsen, T. Laser pointer interaction. In *Proc. CHI'01*, pp. 17–22.

[20] Poupyrev, I., Billinghurst, M., Weghorst, S., Ichikawa, T. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proc. UIST'96*, pp. 79–80.

[21] Rekimoto, J. and Saitoh, M. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *CHI'99*, pp. 378–385.

[22] Rekimoto, J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proc. UIST'97*, pp. 31–39.

[23] Smith, D.C., Irby, C., Kimball, R., Verplank, W., and Harslem, E. Designing the Star user interface, *Byte* 7(4):242–282, 1982.

[24] Stoakley, R., Conway, M., and Pausch, R. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proc. CHI'95*, pp. 265–272.

[25] Streitz, N.A., Tandler, P. Müller-Tomfelde, C., Konomi, S. Roomware. In: Carroll, J.A. (Ed.), *Human-Computer Interaction in the New Millennium*, Addison Wesley, pp. 553–578, 2001.

[26] Wagner, A., Curran, P., O'Brien, R. Drag me, drop me, treat me like an object. In *Proc CHI '95*. pp. 525–530.

[27] Zhai, S., Morimoto, C. Ihde, S. Manual And Gaze Input Cascaded (MAGIC) Pointing. In *Proc. CHI '99*, pp. 246–253.