EG UK Theory and Practice of Computer Graphics (2008)
Ik Soo Lim, Wen Tang (Editors)

A New Method for Interacting with Multi-Window Applications on Large, High Resolution Displays

Chris Rooney[†1] and Roy A. Ruddle [‡1]

[1]School of Computing, University of Leeds, Leeds, England

Abstract

Physically large display walls can now be constructed using off-the-shelf computer hardware. The high resolution of these displays (e.g., 50 million pixels) means that a large quantity of data can be presented to users, so the displays are well suited to visualization applications. However, current methods of interacting with display walls are somewhat time consuming. We have analyzed how users solve real visualization problems using three desktop applications (XmdvTool, Iris Explorer and Arc View), and used a new taxonomy to classify users' actions and illustrate the deficiencies of current display wall interaction methods. Following this we designed a novel method for interacting with display walls, which aims to let users interact as quickly as when a visualization application is used on a desktop system. Informal feedback gathered from our working prototype shows that interaction is both fast and fluid.

1. Introduction

Large, high resolution displays, such as the one in Figure 1, are becoming increasingly popular for visualization applications. The tens of millions of pixels of display real estate that is provided allows both the presentation of huge data sets, and the use of multiple viewpoints. However, these advantages are compromised by the cumbersome methods of interaction that are currently provided, which are very inferior when compared with use of a mouse and keyboard to interact with a desktop system [RCB*05].

This paper describes a new method for interacting with multi-window visualization applications that run on large, high resolution displays (also known as Powerwalls). To characterize how users interact with such applications, we recorded the tasks users performed when tackling three types of visualization problems with well known software (XmdvTool, Iris Explorer and Arc View). The motor actions required to perform four basic interaction tasks were analyzed in detail. A new taxonomy was developed and showed that existing interaction methods for large, high resolution displays are problematic because they involve more actions and, sometimes, greater precision than desktop interaction.

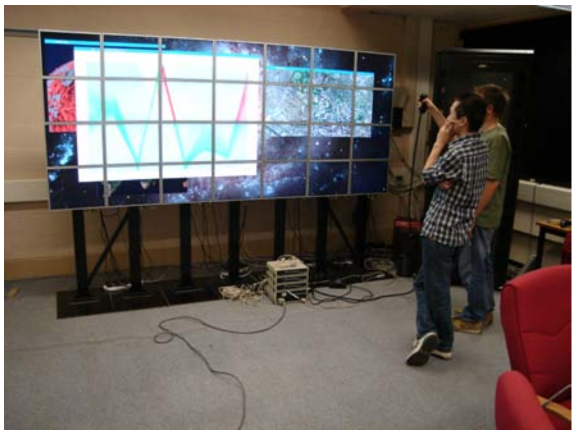

Figure 1: *The multi-window prototype on a 50 mega-pixel display.*

[†] rooney@comp.leeds.ac.uk
[‡] royr@comp.leeds.ac.uk

To address this, we developed a new method for interacting with large, high resolution displays.

2. Study of Interaction

This section describes a study of how people interact with three well-known visualization applications, using a desktop display and mouse and keyboard interface. The applications were XmdvTool, Iris Explorer and Arc View, which are widely used for information visualization, scientific visualization and geographic visualization, respectively. By understanding how users interact during desktop interaction, we know the functionality our interface will require when designing a solution for Powerwall interaction.

2.1. Method

Each application was used to solve a visualization problem, with the user's interactions being recorded for subsequent analysis. The XmdvTool problem was set as a practical exam question from an MSc module, which candidates were expected to spend an hour answering. The Iris Explorer problem was another question from the same exam and, again, candidates were also expected to spend an hour on their answer. The ArcView problem was a piece of coursework set for a third year undergraduate database module. Students were expected to spend five hours preparing for the coursework (e.g., reading background literature) and five hours actually doing it.

For all three problems, candidates had to open a data set, perform a set of actions to process and visualize the data, and take screen shots as part of their exam/coursework submission. However in the present study, rather than observe students, we solved the problems ourselves. This meant that the minimum number of steps was taken to find the answers, whereas a student would most likely have spent more time exploring the data or backtracking due to mistakes made along the way.

2.2. Analysis

The user interactions required to solve each problem were analyzed in three ways: at application-level, in terms of basic interaction tasks, and in terms of low-level motor actions.

There were 10 different application-level interactions (see section 2.3) but initial analysis showed that each of these could be broken down into a sequence of basic tasks, of which there were four categories:

- *Selection* involves positioning the cursor over an object (an icon, widget or item of data) and then either clicking with a mouse button or allowing the cursor to hover.
- *Dragging* is where the cursor is moved to an object, the object is moved to a different position on the display, and then released.
- *Symbolic Input* is the input of characters or symbols, and is usually performed with a keyboard [BKPL04].
- *Menu Navigation* is the process of traversing through the hierarchy of a menu system to select a particular option. Most systems allow users to navigate menus in several ways (e.g., using the mouse to click on/hover over an option (selection), or keyboard (symbolic input)), which is why interaction classifications generally place menu navigation in its own category (e.g., [SP05]). In the present study, menu navigation was performed using mouse clicks.

Each basic interaction task may be further broken down into the low-level motor actions of positioning and state changes (e.g., press, release or click a mouse button). Fitts' law [Fit54] is a well-known model for predicting the time taken to position a pointer (e.g., the cursor) over a target (e.g., a user interface widget) based on the distance to and size of the target, and has been widely adopted in human-computer interaction (HCI) to both improve and evaluate interaction [Ahl05] [GHA*90]. In the present study the majority of targets had a smallest dimension (width or height) of less than 25 pixels, and these were classified as requiring high precision positioning. Most of the remaining targets were much larger (e.g., over 300 pixels) and were classified as requiring low precision positioning.

Table 1: *Frequency of task occurrences within the study.*

Application-level task	XmdvTool	Explorer	ArcView
Open File	1	2	0
Filter Data	4	0	7
Format Visualization	0	3	6
Take Screen Shot	2	2	5
Query Data	0	0	8
Open a Module	0	10	0
Prepare Analysis	4	0	0
Analyze Data	5	3	0
Connect Pipeline	0	13	0
Save File	0	2	0
Basic interaction task	**XmdvTool**	**Explorer**	**ArcView**
Selection	26	38	141
Dragging	10	30	14
Menu Navigation	7	35	15
Symbolic Input	4	8	12
Motor action	**XmdvTool**	**Explorer**	**ArcView**
Total actions	139	408	439
Positioning	60	168	199
State Input	79	240	240

2.3. Results

There were ten different application tasks, four were specific to one application (see Table 1). These tasks involved a sequence of up to 14 basic interaction tasks, and up to 40 low-level motor actions.

The proportion of each category of basic task depended on the application, but in all three cases selection was the most common (see Figure 2). Iris Explorer required more dragging and menu navigation than the other applications, because dragging was used for the 'opening a module' and menu navigation was required for 'connecting a pipeline'. In all the applications, less than 10% of the user's low-level motor actions were for symbolic input.

As Figure 2 shows, the majority of the user's actions involved high precision positioning. This was particularly true for menu navigation, and was because menus display options as a list of thin, wide strips. In XmdvTool and ArcView, dragging was divided equally between high and low precision, and this was because dragging generally either started with low precision positioning and finished with high positioning, or vice-versa. For example, to move a boundary in XmdvTool the cursor had to be within a large area (low precision) to grab the boundary, but it needed to be placed at an exact point (high precision).

3. Basic Interaction Tasks

Two important factors that affect the time is takes to perform each basic interaction tasks are the number of motor of actions involved and the precision with which they have to be made. For the latter, Fitts' law [Fit54] allows predictions to be made about how increasing the size of a target (e.g., a user interface widget) will reduce the time required for interaction.

We have developed an interaction taxonomy that categorizes and shows the motor actions involved in basic interaction tasks (see Figure 3). Positioning is broken down into high or low precision, state input is broken down into a click, hold or release, and the taxonomy as a whole is presented in a similar style to Mackinlay's interface device taxonomy [MCR90].

The following sections show how basic interaction tasks are performed using: (a) desktop systems, and (b) existing techniques for large high resolution displays. Presentation of these techniques within our taxonomy helps explain why existing techniques for large displays are time consuming and, therefore, impede users' ability to work in an interactive manner.

3.1. Desktop Interaction

Examples of how the four basic interaction tasks are typically performed in desktop interaction (e.g., using XmdvTool, Iris Explorer or Arc View) are shown in Figure 3. Selection involves positioning the cursor over a target and clicking a mouse button, and in the majority of cases was a high precision activity. All the menu navigation in the study used a two-tier hierarchy (e.g., File -> Save), with each tier requiring high-precision positioning followed by a click. For dragging, the task of moving a window has been used as an example (high precision position and hold combination, followed by a low precision position and release action; see above). The symbolic input task is for the input of a single character.

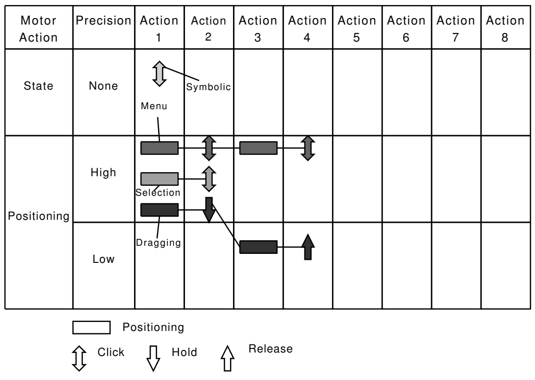

Figure 3: *Taxonomy showing how the four basic interaction tasks are performed on a desktop system.*

3.2. Existing Techniques for Large High Resolution Displays

A variety of interaction techniques have previously been proposed for use with large high resolution displays. This section reviews the most promising of those techniques and uses the taxonomy to show how they could be used to perform the basic interaction tasks of selection, dragging, menu navigation and symbolic input.

3.2.1. Selection

Two methods for performing selection on Powerwalls are *HybridPointing* [FVB06] which is based on a touch screen method, and the *Bubble Radar* [ANSG06] which provides interaction via a tablet PC. HybridPointing provides both absolute mapping between a user's position and the display (the user would have to physically walk to select an object that was outside arms' reach) and relative mapping (small arm movements traverse the cursor over a large distance). Relative mapping allows the user to select targets anywhere on the display but at the disadvantage of limited precision, which means that only large targets can be acquired from a distance.

The Bubble Radar has two modes (bubble and radar), with the former best suited to selection. A bubble cursor is an area surrounding the cursor, and it is manipulated on a Powerwall by dragging a pen across a tablet PC. When the cursor moves, the first target to meet the bubble is encompassed by it and selected if the pen is lifted off the tablet. This makes it straightforward to select small targets, but only if they are spaced out from other targets.

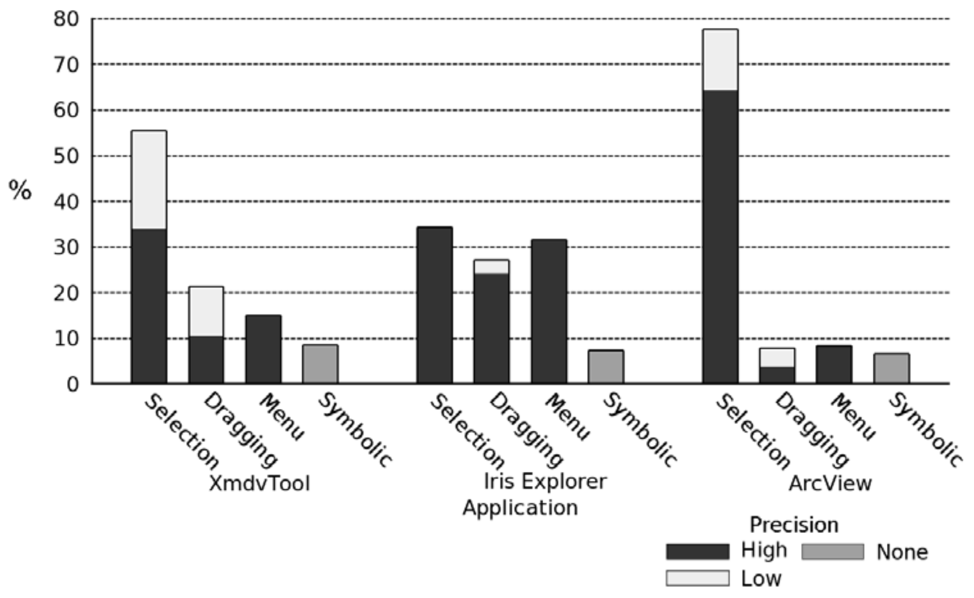

Figure 2: *The percentage of the four basic interaction tasks recorded for each application. Where appropriate, each task is separated into high and low precision.*

The Bubble Radar is likely to be faster than HybridPointing for selection tasks because it does not require any high precision motor actions (see Figure 4). However, both techniques involve twice as many actions as when selection is performed using a mouse on a desktop system (see Figure 3).

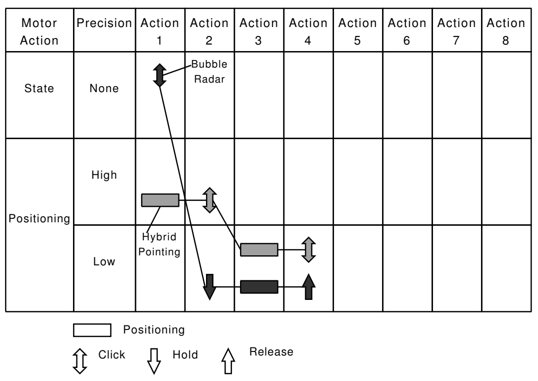

Figure 4: *Taxonomy showing the HybridPointing and Bubble Radar selection techniques.*

3.2.2. Dragging

Khan et. al. [KFA*04] developed a system known as a *Frisbee* to view and interact with distant regions of large displays. Similar to a portal, one end of the Frisbee (the telescope) is local to the user, and the other (the target) is positioned at a remote location. The telescope presents the content of the target. Various controls surrounding the telescope allow the user to both change the location of the target and interact with objects inside the target. Objects can be moved from local space to remote space (and vice-versa) by dragging them through the portal.

Another method for performing dragging tasks is the *Push-and-Pop* technique [CHBL05] designed for icon interaction. As the user begins to drag an icon, a copy of the related icons surround the cursor (e.g., the recycle bin would appear when a text file is dragged). The icon can be dragged directly to the near by copy.

Neither technique (see Figure 5) is as effective as desktop interaction. Although Push-and-Pop involves the same number of actions as desktop dragging (see Figure 3), both of Push-and-Pop's positioning actions are high precision and the technique is only designed for dragging icons. The Frisbee offers a more generic solution, but has twice the number of motor actions.

3.2.3. Menu Navigation

With desktop systems it is common to position a static menu at the top of a window and, due to the modest size of the display (say, a maximum of 1600 x 1200 pixels), it only takes a small amount of time to move the cursor to such a menu. With large displays, this 'return to base' style menu system is no longer suitable because the time required to move the cursor across the display is excessive.

To overcome this, the *VisionWand* [CB03], a small rod

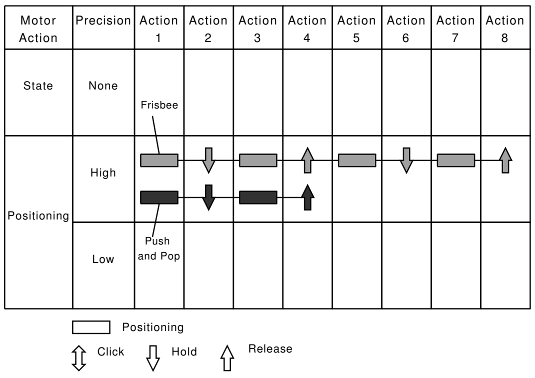

Figure 5: *Taxonomy showing the Frisbee and Push-and-Pop dragging techniques.*

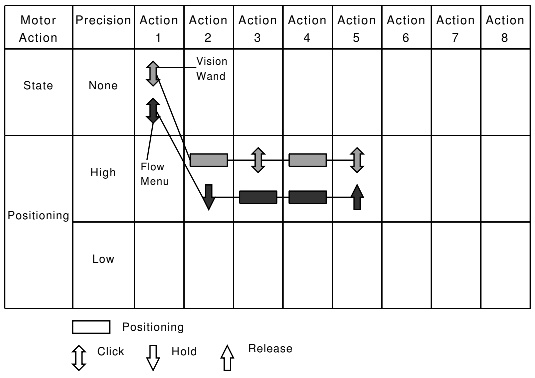

Figure 6: *Taxonomy showing the VisionWand and FlowMenu menu navigation techniques for a two tier menu hierarchy (e.g., File -> Save).*

tracked in 3D space, and *FlowMenu* [GW00], a pen based interaction technique, both offer an 'at cursor' menu system. The VisionWand uses a pie widget with each slice representing a menu option. The menu is activated by turning the rod 90°, and then rotated again to the position of the slice to be selected. A downwards gesture makes the selection. A large number of menu options can be divided into a series of pie hierarchies.

Similar to the Bubble Radar, the FlowMenu is activated using a button press on the non-dominant hand. The menu is shaped like a small octagon with trapeziums connected to each edge. Interaction begins by pressing the pen inside the octagon, and dragging it towards an outer trapezium. In a multi-hierarchical menu, this action would change the value in each trapezium to an option from the next hierarchy. Whilst keeping the pen pressed down, it is then dragged to the next trapezium. This is a repeated action until the bottom of the hierarchy is reached, final selection is made by releasing the pen from the surface.

For a two tier menu hierarchy, both devices require five motor actions (see Figure 6). However, for each additional tier of the hierarchy, the VisionWand requires two motor actions (positioning followed by a click), in comparison the FlowMenu only requires one (just positioning). So for menu hierarchies of three tiers or greater, the FlowMenu requires fewer motor actions for menu interaction. As with desktop interaction, all the positioning actions for the VisionWand and FlowMenu are high precision.

3.2.4. Symbolic Input

Previous work on symbolic input has used devices such as the *forearm keyboard* [TTG97], *pinch keyboard* [BRP01] and *pen & tablet* [BRP01]. For each of these devices, one motor action is required to input a character, which is the same as when symbolic input is performed using a keyboard (see Figure 3). This indicates that devices such as a forearm keyboard are adequate for large high resolution displays and, therefore, symbolic input is not considered further in the present paper.

4. Design

This section describes our new methods for selection, dragging and menu navigation. In each case, our methods are an improvement on existing ones because the number of motor actions has been reduced and/or less precision is required for positioning.

For positioning, a user's hands are tracked in 2D space (the plane of the display), so the user may alter their distance from the display without interfering with interaction. This also allows the user to physically walk from one part of the display to another, which is known to be preferable to tethering the user to a stationary device [BNB07]. Alternatives to our approach are use of a pointing device [BSC06] [CT06] and use of pen & tablet techniques. However, pointing devices tend to be error prone when the user wishes to select small, distant targets [VB05] and tablets are large and heavy and so induce fatigue. Tabletop solutions have been proposed [MAB05], but leave the user tethered and unable to physically navigate.

State input requires a system for inputting clicks, hold and releases. Desktop systems achieve state inputs for selection, dragging and menu navigation using only two mouse buttons. However, large display interaction benefits from additional state inputs, which we achieve using pinch gloves [Fak07]. These have been chosen because they offer a free hand technique and provide kinetic feedback. A tap between finger and thumb is recognized as an input, allowing up to four inputs per hand. The dominant hand is used for selection and dragging state changes, and the non-dominant hand for menu navigation. Although we have chosen to use pinch gloves, our solution works with any device that can offer four state inputs per hand (e.g. gesture recognition).

4.1. Selection

We have developed a selection technique called Fast n' Fine. In 'fine' mode, the user's hand is mapped to the cursor with a gain of 1.0, allowing precise movements to be made. A switch to 'fast' mode increases the gain so the user is able to traverse the whole display using only arms' reach (the increase in gain is based on the size of the display, for our Powerwall a gain of 3.0 was appropriate). Use of the fine mode on its own allows targets that are in the immediate vicinity of the user on the display to be selected, and combining the modes allows distant targets to be selected with precision.

We have chosen Fast n' Fine over related work such as Area Cursors and Sticky Icons [WWBH97] because the majority of targets should require only low precision. Any remaining high precision targets are likely to require the cursor to be placed at an exact point that is known only to the user, making target prediction difficult.

The Fast n' Fine technique is implemented using a device such as a pinch glove or a 3D mouse, which is tracked in the plane of the display and has at least three 'buttons' (two are used for selection and one for dragging; see below). One of the buttons is used for target acquisition and another for switching between fast and fine movements. Over lengthy periods of interaction, the physical position of a user's hand and cursor may get out of sync. To resolve this, dwelling on the switching button resets the cursor back to the central position.

Figure 7 shows the Fast n' Fine selection technique plotted on the taxonomy for both local and distant target acquisition. The default state is 'fine', so local targets can be selecting using just a positioning-click combination, which is the same number of actions as desktop selection (see Figure 3). To reach small, distant targets the mode is switched to 'fast', so the cursor can be positioned close to the target at speed, and then the switched back to the 'fine' for final target acquisition.

4.2. Dragging

In a multi-window environment, windows are often moved and resized. On a desktop system, the most common way of moving a window is to hold and drag the title bar, which involves high precision positioning. Resizing a window requires the cursor to be in an area of five pixels square, requiring even more precise positioning.

Our new interface reduces the time required for dragging tasks by superimposing a manipulation layer on top of the window (see Figure 8). Two of the buttons on the dominant hand are reserved for selection (see above), and a third is

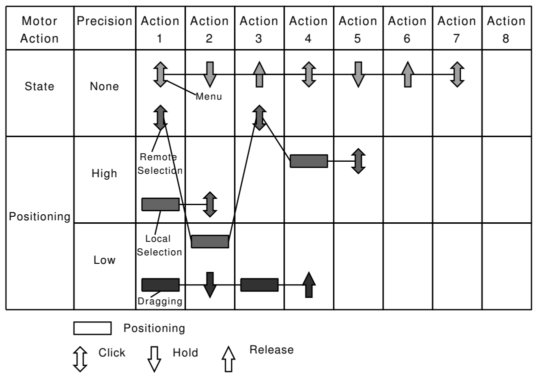

Figure 7: *Taxonomy showing Fast n' Fine (local and remote selection), the OnHaP menu and window manipulation techniques.*

reserved for interaction with the manipulation layer. Distinct regions of the manipulation layer are used for each operation (e.g., moving or resizing) but the size of the regions means that, unlike desktop interaction (see Figure 3), these operations only require low-precision positioning (see Figure 7).

The manipulation regions could be shown using color highlighting (similar to the ToolGlass approach [BSP*93]), but that would obscure data. Instead, the cursor icon changes to indicate the type of operation that may be performed.

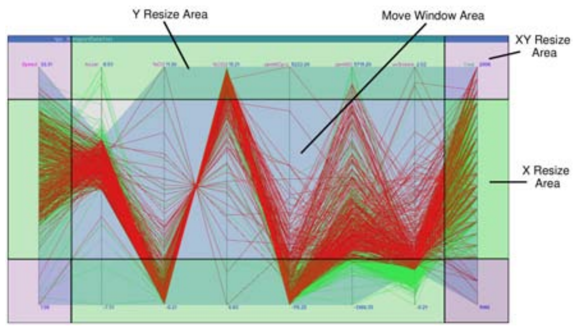

Figure 8: *The manipulation layer highlighted to show where window manipulation can occur. Resizing in the X or Y plane is performed in the rectangular areas at the edges of the window, resizing in both planes is performed in the corner areas, and moving is performed in the middle of the window.*

4.3. Menu Navigation

A pinch glove menu system called *TULIP* was designed by Bowman for use in immersive virtual environments (VEs) [BW01]. The system was two-handed (some menu options were accessed via a user's non-dominant hand and others via their dominant hand) and, on each hand the index, middle and ring fingers were used to select three of the menu options, whilst the little finger was used to access a list of additional options that were displayed in the palm of the user's virtual hand.

Compared with desktop menus, pinch glove menus have the advantage that users only have to make state changes (e.g., clicks) and not perform position actions. We have modified the TULIP design to produce a one-handed menu (OnHaP) that is suitable for Powerwalls, and leave a user's dominant hand free to perform selection and dragging tasks.

OnHaP works as follows. A single click activates and presents the menu, with the first option already highlighted. The index and ring fingers scroll up and down the menu, the middle finger selects an option and the little finger exits from the menu. Scrolling can be performed either with series of clicks, or holding a pinch until the appropriate option is reached. Once highlighted, selection is achieved using the middle finger, which in turn either performs an action or presents the next tier of the menu hierarchy.

Although OnHaP is designed to work with pinch gloves, it can in fact be controlled by any four button device (e.g., a 3D mouse). The technique is plotted on the taxonomy in Figure 7. The first action is the activation of the menu, then for each hierarchy three actions are required. A hold to scroll through the options, a release to stop at the chosen option and a click to select it. So although OnHaP requires an extra action per hierarchy compared to desktop menu navigation (see Figure 3), no positioning tasks are required.

5. Implementation

A prototype of the design was implemented to work on a display consisting of 28 20-inch monitors tiled seven by four. Each monitor has a resolution of 1600x1200 pixels, giving a total real estate of 11200x4800. The display is powered by seven nodes (one master and six slaves). The master node contains two AMD Opteron 270 processors, two XFX nVidia 7800 GTX graphics cards and 8GB of DDR400 RAM. The slave nodes each contain an AMD Athlon X2 4400+ processor, two XFX nVidia 7800 GTX graphics cards and 4GB of DDR400 RAM. The prototype was coded in C++ using the OpenGL and VRJuggler libraries. An eight button stylus was used for state input, and was tracked in 3D space using a Flock of Birds tracking system for positioning input. A multi-window desktop was created (one window was an interactive parallel coordinate application, and the others were images; see Figure 1). The Fast n' Fine selection and window dragging/resizing functionality was implemented.

5.1. Evaluation

The prototype was informally tested during a demonstration afternoon that was run for students in the School of Computing. The students were split into three groups, and for each group the functionality of the prototype was demonstrated. Each student was then given a turn at interacting with the prototype.

The overall feedback given by the students was positive, with the majority finding the window manipulation technique pleasing to use. The students had no difficulty in moving the cursor to one of the outer manipulation areas of the window. One student suggested that as well resetting the cursor to the middle position, there should be the option to detach ones self from the cursor (similar to lifting the mouse off the desk to reposition it). Another student suggested the small arrows indicating manipulation could actually be positioned inside the cursor, causing less obstruction of the data behind.

6. Conclusion

This paper describes a new method for interacting with multi-window applications on large, high resolution displays. A study of desktop interaction identified four basic interaction tasks and, for each of these, the precision and number of motor actions involved were plotted onto a taxonomy. This allowed deficiencies of existing techniques for interaction with large high resolution displays to be identified, and a new method for interacting with such displays to be developed.

In our new method a user's dominant hand is tracked in the plane of the display and used for selection and dragging tasks, and the user's non-dominant hand is used for menu navigation. Local selection tasks now involve the same number and precision of motor actions as when selection is performed on a desktop system, and targets that are on distant parts of a large display may be selected accurately by movements no larger than the users arms' reach. Dragging tasks such as moving or resizing a window now only involve low precision actions and, therefore, are likely to be faster on a large display than a desktop. Menu navigation for large displays is achieved solely by state changes (e.g., finger pinches or button presses), which is similar to the way menus on devices such as digital cameras and video recorders are accessed. Although this means that more actions are required than with desktop menus, the latter involve high precision position tasks which are more time consuming than state changes.

Finally, the next stage of our research is to implement our interface in a full, working visualization application and conduct formal evaluations.

7. Acknowledgments

Chris Rooney is funded by a PhD scholarship from the University of Leeds.

References

[Ahl05] AHLSTRÖM D.: Modeling and improving selection in cascading pull-down menus using fitts' law, the steering law and force fields. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), pp. 61–70.

[ANSG06] ALIAKSEYEU D., NACENTA M., SUBRAMANIAN S., GUTWIN C.: Bubble radar: Efficient pen-based interaction. In *Proceedings of the working conference on Advanced visual interfaces* (2006), pp. 19–26.

[BKPL04] BOWMAN D. A., KRUIJFF E., POUPYREV I., LAVIOLA J.: *3D User Interfaces: Theory and Practice*. Addison Wesley, 2004.

[BNB07] BALL R., NORTH C., BOWMAN D.: Move to improve: promoting physical navigation to increase user performance with large displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2007), pp. 191–200.

[BRP01] BOWMAN D., RHOTON C., PINHO M.: Text input techniques for immersive virtual environments: An empirical comparison. In *Proceedings of the Human Factors and Ergonomic Society Annual Meeting* (2001), pp. 2154–2158.

[BSC06] BI X., SHI Y., CHEN X.: upen: A smart pen-liked device for facilitating interaction on large displays. In *Proceeding of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems* (2006), p. 7.

[BSP*93] BIER E., STONE M., PIER K., BUXTON W., DEROSE T.: Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), pp. 73–80.

[BW01] BOWMAN D., WINGRAVE C.: Design and evaluation of menu systems for immersive virtual environments. In *Proceedings of the Virtual Reality 2001 Conference (VR'01)* (2001), p. 149.

[CB03] CAO X., BALAKRISHNAN R.: Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. In *Symposium on User Interface Software and Technology* (2003), pp. 173–182.

[CHBL05] COLLOMB M., HASCOËT M., BAUDISCH P., LEE B.: Improving drag-and-drop on wall-size displays. In *ACM International Conference Proceeding Series; Vol. 112, Proceedings of Graphics Interface 2005* (2005), pp. 25–32.

[CT06] CHENG K., TAKATSUKA M.: Estimating virtual touchscreen for fingertip interaction with large displays. In *Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments* (2006), pp. 397–400.

[Fak07] FAKESPACE: http://www.fakespace.com/, [Last accessed: 21-Dec-07].

[Fit54] FITTS P.: The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47* (1954), 381–391.

[FVB06] FORLINES C., VOGEL D., BALAKRISHNAN R.: Hybridpointing: fluid switching between absolute and relative pointing with a direct input device. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (2006), pp. 211–220.

[GHA*90] GILLAN D., HOLDEN K., ADAM S., RUDISILL M., MAGEE L.: How does fitts' law fit pointing and dragging? In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people* (1990), pp. 227–234.

[GW00] GUIMBRETIÈRE F., WINOGRAD T.: Flowmenu: combining command, text, and data entry. In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (2000), pp. 213–216.

[KFA*04] KHAN A., FITZMAURICE G., ALMEIDA D., BURTNYK N., KURTENBACH G.: A remote control interface for large displays. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (2004), pp. 127–136.

[MAB05] MALIK S., ABHISHEK R., BALAKRISHNAN R.: Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (2005), pp. 43–52.

[MCR90] MACKINLAY J., CARD S., ROBERTSON G.: A semantic analysis of the design space of input devices. *Human-Computer Interaction 5* (1990), 145–190.

[RCB*05] ROBERTSON G., CZERWINSKI M., BAUDISCH P., MEYERS B., ROBBINS D., SMITH G., TAN D.: The large-display user experience. *IEEE Computer Graphics and Applications 25, issue 4* (2005), 44–51.

[SP05] SHNEIDERMAN B., PLAISANT C.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley, 2005.

[TTG97] THOMAS B., TYERMAN S., GRIMMER K.: Evaluation of three input mechanisms for wearable computers. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers* (1997), p. 2.

[VB05] VOGEL D., BALAKRISHNAN R.: Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (2005), pp. 33–42.

[WWBH97] WORDEN A., WALKER N., BAHRAT K., HUDSON S.: Making computers easier for older adults to use: area cursors and sticky icons. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1997), pp. 266–271.