VisionWand: Interaction Techniques for Large Displays using a Passive Wand Tracked in 3D

Xiang Cao, Ravin Balakrishnan
Department of Computer Science
University of Toronto
caox | ravin @dgp.toronto.edu
www.dgp.toronto.edu

ABSTRACT
A passive wand tracked in 3D using computer vision techniques is explored as a new input mechanism for interacting with large displays. We demonstrate a variety of interaction techniques that exploit the affordances of the wand, resulting in an effective interface for large scale interaction. The lack of any buttons or other electronics on the wand presents a challenge that we address by developing a set of postures and gestures to track state and enable command input. We also describe the use of multiple wands, and posit designs for more complex wands in the future.

Keywords: vision tracking, large displays, gestures, interaction techniques, input devices, buttonless input

INTRODUCTION
Large-format upright displays – ranging from 40" to 60" plasma panels to very large scale (>8') high-resolution displays driven by multiple projectors – enable us to work with very large quantities of simultaneously displayed visual data, and give multiple people the ability to work effectively together at a single display. Indeed, in recent years many researchers have recognized the value of such large scale displays and have explored a diverse set of applications for them, including collaborative groupware [12], electronic whiteboards [8, 21], and industrial design [3, 13]. Others such as Guimbretière et al. [15] have investigated more general interaction issues.

While the visual quality of commercially available large displays is already very high and continues to further improve, and the range of research applications available are quite impressive, the question of what input technology to utilize when interacting with displays of this scale remains an open one. The most promising and widely adopted input mechanism to date – single finger or pen input using a variety of technologies [8], [www.smarttech.com] – requires the user to stand up close to the display and is limited to single point two degree-of-freedom interaction much like when using a standard mouse. While this constraint is fine for many applications [12, 15, 21], others [3, 13] benefit from users operating the interface with higher degree-of-freedom input devices while standing further away from the display. Input technologies that have been used for such "from afar" interaction currently include a variety of 3D trackers [31], laser pointers [20, 22, 23], custom wands [32, 33], and the use of computer vision to track users' hands [6, 10, 29]. All these technologies, however, have a variety of limitations which we will discuss later in this paper.

In this paper, we explore the idea of using a passive wand that is tracked in 3D space using computer vision techniques as an alternative input device for interaction with large scale displays. This *VisionWand* is a simple plastic rod with colored ends (Figure 1a), without any embedded electronics, that is tracked by a pair of commodity (<$100) cameras (Figure 1b). The negligible cost of the wand allows for multiple versions to be available, used, or discarded at any time. Since both endpoints of the wand are tracked in 3D, the resulting input is a 3D ray, allowing for a richer vocabulary of actions than is possible with 2D point input. However, the lack of electronics presents a challenge in that there are no buttons or other means of directly providing state information from the device itself. We address this challenge by developing a set of gestures and postures to enable command input. The use of a physical wand rather than free-hand gestures not only simplifies the vision algorithms, but allows for interaction techniques that take advantage of the affordances of the physical tool, resulting in "rich-action" input as defined by Rekimoto and Sciammarella [25]. We present an exploratory set of interaction techniques that take advantage of all these features.

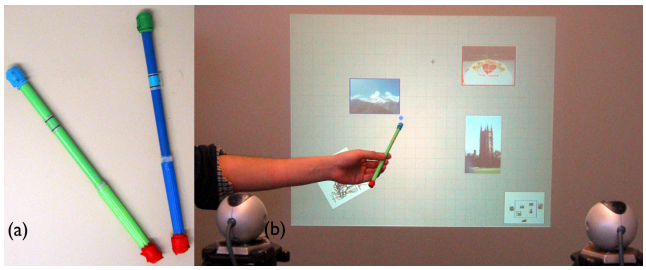

Figure 1. (a) VisonWands, (b) System setup.

UIST '03 Vancouver, BC, Canada

RELATED WORK

The most common and commercially viable input technique for large displays are systems that track a single point of 2-dof input, either by direct touch, use of a pen, or both. For example, the Liveboard system [8] used an optical pen, the Stanford Interactive Mural uses a wireless Ebeam pen [15], while the SmartBoard system (www.smarttech.com) supports both finger and pen input. The advantages of these systems are high resolution tracking and the single point input is backward compatible with existing GUI interfaces. Limitations include having only single point 2-dof input, and requiring the user to work up close to the display, reducing the user's ability to visualize the entire large display while interacting.

Another approach is to use optical, electromagnetic, acoustic, inertial, or radio tracking technologies to track one or more simultaneous points of input in 3D space. See Welch et al. [31] for an excellent survey of the various technologies. While these trackers enable us to prototype advanced interaction techniques [13, 14], their cost has remained rather high (ranging from a few thousand dollars for the cheapest trackers to a few hundred thousand dollars for more sophisticated motion tracking systems) for over a decade. As such, they are not practical for widespread use. Furthermore, the cheaper technologies are typically tethered, reducing the user's freedom of movement.

Recently, several researchers have explored the use of standard laser pointers as input to large displays [20, 22, 23]. These have the advantage of low cost, but have a fundamental limitation in that there are no buttons to augment the single point of tracked input. As such, even standard mouse operations are not possible. To overcome this, Olsen and Nielsen [23] explored the use of dwell time and other techniques to replace button presses. Oh and Stuerzlinger [22] have augmented laser pointers with buttons, resulting in a "from-afar" input technology that can operate the entire range of interaction techniques found in a standard GUI. Using clever multiplexing techniques, they were even able to differentiate between and track several laser pointers at the same time. However, the additional electronics required reduces the main advantage of laser pointers: low cost and ubiquity.

The use of computer vision in HCI has long been a goal of the research community. Various excellent survey articles [6, 10, 29] discuss progress to date. Much of the research in this area that is relevant to large scale displays has focused on the relatively difficult task of using vision techniques to track freehand gestures. For example, Ringel et al. [26] describe a clever system that integrates vision tracking of hand poses with a SmartBoard to enable direct hands-on interaction with a large display that goes beyond single point input. Freeman et al. [11] describe a system for controlling a television using freehand gestures, while Segen and Kumar [28, 29] describe a VR system that uses vision tracking of a small set of hand gestures for spatial interaction.

Early advocates of virtual reality systems [17] and others [1, 2] have also explored the use of gestural input for interacting with large displays, typically tracking the hand and fingers using instrumented gloves. This approach, however, is unlikely to succeed in the long term given the significant inconvenience of having to put on a glove to enable interaction.

While interacting with computers with freehand gestures can appear appealing on the surface, upon deeper analysis it is apparent that they do not take advantage of inherent human abilities at using physical tools and the rich vocabulary of actions that are enabled by those tools, as discussed in Rekimoto and Sciammarella [25]. The physical form of the tools can often serve as haptic memory aids to the user as to what functions they can perform, whereas with freehand gestures the user has to rely completely on recall from memory. The sizeable literature on graspable [9] and tangible [16] interfaces provides further evidence of the value of physically manipulable entities in the user interface. While some researchers (e.g., Ringel et al. [26]) contend that their informal observations indicate a strong appeal towards implement-free interaction, we note that this preference is for direct touch, up-close, interaction on the display surface itself, and not to "from afar" interaction.

Some researchers have investigated implement-based gestural interfaces. Clark [5] describes a 3D CAD interface using a 3D wand with a button. Deering [7] describes a sketching and animation system using a 3D wand. Shaw and Green [30] describe a system for two-handed design using two 3D trackers with buttons, while Schkolne et al. [27] take a hybrid approach: combining hand gestures and instrumented physical tools for surface drawing. Note that all these systems used some form of tethered tracking technology to track the wands and other implements used.

Wilson et al. describe the XWand [33] and WorldCursor [32] systems, which use a wireless wand with buttons and sensors to control multiple electrical devices in a complex environment. However, their wand is used mainly as a pointing and command invocation tool, not for screen-based multi degree-of-freedom interactions.

In short, our analysis of the literature indicates that while a variety of different techniques for interacting with large displays have been investigated, they all have some drawbacks and none has yet emerged as the standard input mechanism. The second author and colleagues' previous experience in developing interaction techniques for large displays [3, 13, 14] also points to the need for more facile input techniques. A promising direction to explore is the use of computer vision tracking, which to date has focused on tracking freehand gestures. We believe that a more fruitful use of computer vision in this domain is to enable the tracking of simple, passive, physical tools around which sophisticated interaction techniques can be built. Our VisionWand is one attempt in this direction.

SYSTEM IMPLEMENTATION

Hardware

The VisionWand is a simple cylindrical piece of plastic with different colored ends (Figure 1a). Different wands can be distinguished by different colors of the bodies, the ends, or additional markers, allowing for different wands to be tracked using our camera setup. No buttons or wheels are attached to the wand. A pair of Logitech QuickCam Pro 3000 cameras are used for tracking. The cameras face a back-projected display. The user interacts with the display using the wand. (Figure 1b).

Tracking procedure

Standard stereo vision techniques are applied to track the wand in 3D. The camera pair is calibrated by projecting a calibration image on the display. Calibration needs to be done only once as long as the cameras, the projector and the screen are fixed. At each frame, the body of the wand, as well as the two ends, are detected by color in both of the images captured: a straight line is fitted to the wand body and the colored ends are searched for in the neighborhood of the line. A 3D ray, including the spatial coordinates of the two ends, is reconstructed from these observations, as illustrated in Figure 2.

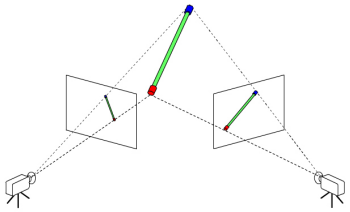

Figure 2. 3D reconstruction of wand from two cameras.

The tracking result is shown in Figure 3. The system displays the red and blue circles, which show the orthogonal projections of the wand ends on the screen. The black cross displayed by the system indicates the intersection of the 3D ray and the screen. This intersection denotes the screen position that the wand is pointing to. We display both colored circles simply to give the user an idea of how the wand is being tracked, while the black cross serves as a pointer. In addition to the spatial positions, we make use of the information of two angles: *orientation*, defined as the obliquity of the orthogonal projection of the 3D ray on the screen, and *tilt*, defined as the inclination between the 3D ray and the screen.

The tracking is achieved at approximately 20 Hz for a single wand. In our current system setup the user's actions are restricted in the space between the cameras and the screen, and the tracking works well when the majority of the wand body and at least one end can be seen by both cameras. We note that different camera configurations could be experimented with to reduce occlusions, for example, cameras from the top facing down towards the user. While our system can recognize different wands, this slows down the tracking speed with our current algorithm, mainly because we do all image processing on the main CPU. Dedicated image processing hardware would improve the tracking speed significantly. To maintain high update rates when doing our user tests, we use a keyboard switch to manually tell the system which wand to track.

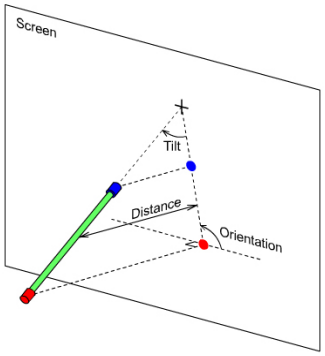

Figure 3. Mapping of wand to screen.

DESIGN PRINCIPLES

In designing a passive, buttonless, 3D wand input system, we have considered several important design issues:

Inferred actions: the lack of any buttons or other electronics on the wand itself implies that the device cannot actively communicate any information about its state to the computer. Rather, state and action information will have to be indirectly inferred by the system. We infer a set of *postures* based on the position and orientation of the wand in space, while a set of *gestures* are determined based on the dynamic characteristics of the wand's movement. Figure 4 defines these postures and gestures. The system actions associated with these postures and gestures will be described as we progress through the paper explaining the various interaction techniques and interface widgets.

Easily understandable actions: one could conceivably use the VisionWand to perform a huge set of functions by assigning meaning to every permutation of the sensed 3D ray's positions, orientations, and movements. However, unless users can easily understand and form a suitable mental model of the possible set of actions, the device will be essentially useless. To address this issue, we limit the number of possible actions to a small set, and provide appropriate visual feedback to aid in the comprehension of those actions. Where more complex interface behaviour is required than is afforded by this set of actions, we either compose a sequence of basic actions and/or use appropriate visual interface widgets operable by those basic actions.

Leveraging haptic memory: the literature on tangible interfaces [9, 16, 25] indicates that users can take significant advantage of haptic memory when using physical implements. As such, when selecting which postures and gestures of the myriad different possibilities afforded by the VisionWand to use, we have deliberately chosen those that have very different haptic profiles. These are more likely to be easily committed to the user's haptic memory, allowing for essentially eyes-free operation after sufficient practice.

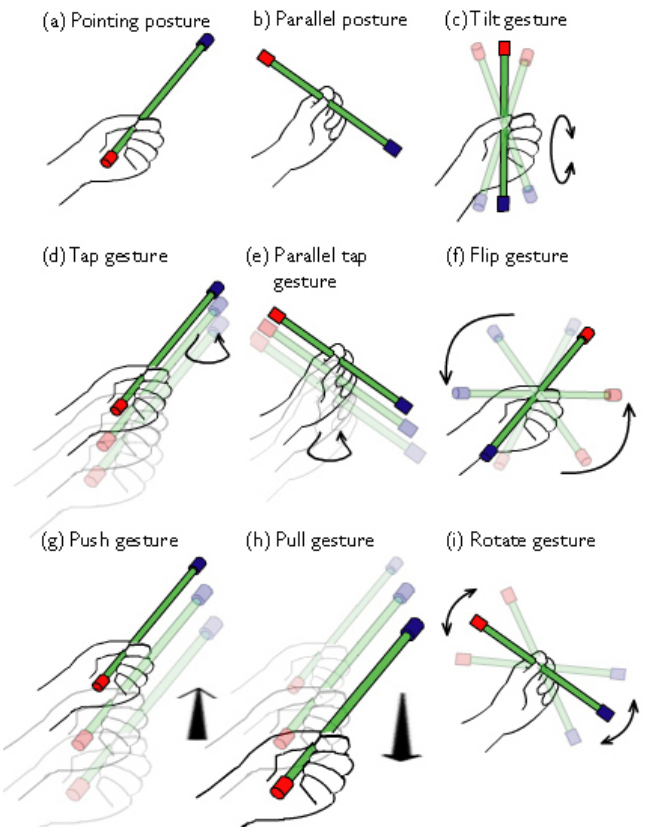

Figure 4. VisionWand postures and gestures. (a) Pointing posture: point to a position on screen; the end that is nearer to the screen is defined as the active end. (b) Parallel posture: keep the wand approximately parallel to the screen, in any orientation. (c) Tilt gesture: starting from a parallel posture, tilt the wand in either direction. (d) Tap gesture: quickly move the active end away from the screen and back again. (e) Parallel tap gesture: from parallel posture, quickly move the entire wand away from the screen and back again. (f) Flip gesture: quickly flip the wand end to end, keeping the orientation and tilt approximately the same as before the gesture. (g, h) Push and Pull gestures: change the distance between the wand and the screen. (i) Rotate gesture: change the orientation of the wand while keeping it in a parallel posture.

INTERACTION TECHNIQUES and INTERFACE WIDGETS

In the following sections we describe a variety of interaction techniques and interface widgets we have developed for the VisionWand. We stress that this is an exploratory set of techniques intended to investigate as thoroughly as possible the design space of VisionWand interactions. In some cases our techniques intentionally push at the extremes of the VisionWand's capabilities, allowing us to determine the limits of this new approach to interaction. We recognize that any real application for large displays that seeks to use the VisionWand will have to carefully select the most promising of these techniques, and perhaps iterate further on them. Initial user feedback presented at the end of this paper provides some direction in this respect.

We note that many standard GUI interaction techniques operated by a 2-dof cursor could also be directly used with the VisionWand. However, controlling a cursor requires a certain amount of precision, which can be difficult to perform with a wand operating in unconstrained 3D space. As such, we have deliberately attempted to use coarser granularity gestures where possible in the designs of the following techniques.

While we demonstrate these techniques and widgets within a picture manipulation and navigation application, our designs are clearly applicable to a broader set of large display applications. The present application is used merely as an illustrative example. In this application, the objects are pictures scattered on a canvas, and the screen displays a part of the canvas. We can move the objects around, scale/rotate them, change object properties, navigate around the canvas, etc.

We now describe the basic interactions and widgets associated with the VisionWand, followed by a discussion of additional functionality that can be achieved by using other wands.

Basic Interactions

Selection, Moving & Scaling, Deselection

An object has three possible states: *selected*, *captured*, and *unselected*. The captured state is similar to an object being dragged by mouse in a standard GUI. Because we do not have any buttons to indicate the state of the VisionWand itself, we switch between these object states using the tap gesture.

The blue end of the VisionWand performs the basic manipulations. While pointing at an object with the blue end, a tap gesture captures it (i.e. switches it into *captured* state). The captured object can be moved around by pointing the VisionWand at different positions on the screen.

In addition, the scale factor of the object is controlled by the distance between the wand and the screen. We can pull back the wand to enlarge the object, and push forward to shrink it (Figure 5). Since the wand is tracked in 3D, moving and scaling can be performed simultaneously.

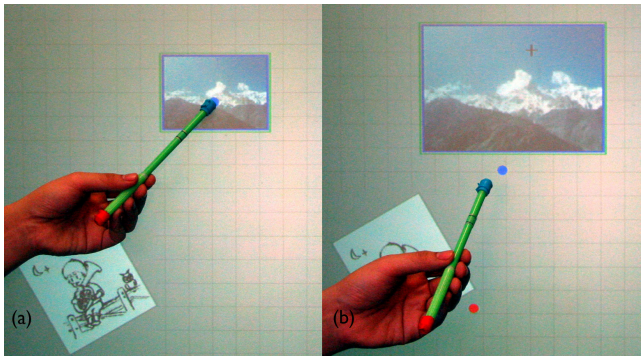

Figure 5. Scaling objects. (a) moving wand towards screen shrinks selected object, (b) moving away enlarges object.

A second tap gesture releases the object (i.e. switches it from *captured* to *selected*). A tap gesture in any blank area of the screen deselects all currently selected objects.

Note that individual deselection is not achieved here. A straightforward idea of implementing this is to let the object cycle through the three states by tap and tap again. But this will significantly reduce the efficiency of manipulating objects, since it takes more actions to enter the desired state. A possible solution to this trade-off is to define more gestures than we currently have. For example, a quick push for capture and a quick pull for release.

Undo

A flip gesture acts as an undo command. The most recent action is reversed.

Query

The red end of the wand acts as a query lens. Figure 6 illustrates. When the red end points at an object, a property sheet is displayed showing some information about the object. When the wand is pulled back from the screen, a spotlight is cast. All objects inside the spotlight show their property sheets. Again, the distance between the wand and the screen controls the scale factor (i.e. the radius of the spotlight).

Incidentally, the spotlight can also be used as a group selection tool. A tap gesture selects all objects inside the spotlight.

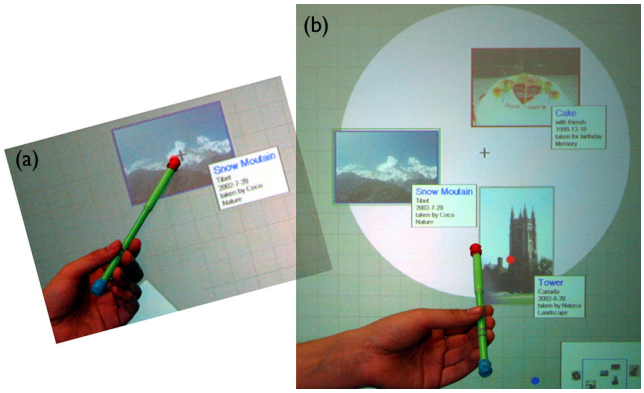

Figure 6. Querying. (a) pointing at object displays properties (b) pulling wand back casts query lens spotlight with radius proportional to distance of wand from screen.

Widgets

Defining a large set of gestures for a large set of functions is a burden not only to the system, but also to the users. Instead, we deliberately kept our gesture set small but designed a set of widgets which help users perform complicated tasks with the basic gestures, while still exploiting the rich 3D manipulation afforded by the wand.

Pie Menu

Menus are the standard widget for selecting from a large set of commands. In particular, pie menus [4, 18] with items arranged uniformly within a circle are especially suitable for our interface because it enables us to exploit the inherent orientation capabilities of the wand.

A context-sensitive pie menu is triggered by keeping the wand in the parallel posture for a short duration. After the menu is displayed, we perform rotation of the wand while keeping it parallel to the screen to move between different items. The current item, which is visually highlighted, corresponds to the orientation of the wand. Figure 7 illustrates. A parallel tap gesture selects the current item.

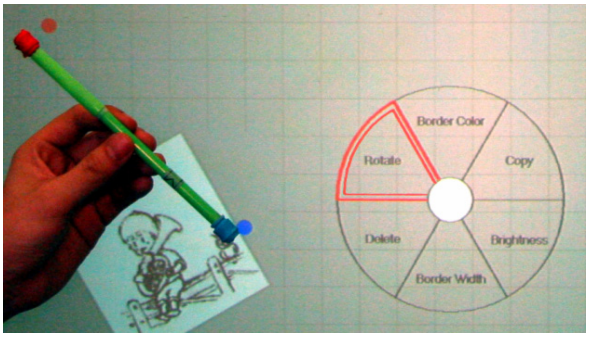

Figure 7. Wand orientation specifies active item in pie menu.

Note that the selection of menu items makes use of only one dimension of information: the wand's orientation. The spatial position and the tilt of the wand are still free for us to use for other functionality. With the basic pie menu, the menu itself can be moved around by following the spatial position of the wand. This allows us to position the menu such that it does not occlude objects of interest, while at the same time using the rotation of the wand to select items. This combination of menu positioning and selection cannot be achieved with more traditional 2-dof input techniques.

The wand's affordances can result in some interesting examples of smooth sequencing of several actions in a row, with similar but quite distinct types of movements. For example, a user can select an object by a tap gesture using the blue end of the wand, then use a parallel posture to pop up the pie menu, rotate the wand to the "rotate" menu item, select it by a parallel tap gesture, and then proceed to rotate the selected object by rotating the wand. The object will be rotated by the same relative angle as the wand's rotation. In this example, all the steps involved can be performed in a continuous, fluid, manner without requiring a pause or numerous button clicks. Guimbretière et al. [15] have also explored similar fluid techniques using pen input.

Because the menu item selected depends on the orientation of the wand, muscle memory can play an important role here. We expect that after some practice, the users will memorize the hand posture they are in with the most commonly used items. This will reduce the burden of visual attention, or even improve the efficiency of the interaction. The users would put the wand in the expected orientation before they actually trigger the menu, thus avoiding the necessity of a further rotation to switch to the desired item. Marking menus [18] similarly exploit repetitive practice to create a menuing system that is operable very quickly by expert users.

Tilt Widgets

The pie menu interaction described previously only makes use of the wand's spatial position and orientation information. The wand's tilt information can also be used advantageously within a pie menu. We have designed two tilt widgets that enable tilt to be used either continuously to adjust a parameter: *tilt dial*, or discretely to select from a set of values in a sub-menu: *tilt menu*.

In our implementation, a tilt widget is associated with an item in a pie menu. If we keep the wand within a particular menu item for a short while, the associated tilt widget appears. We can then tilt the wand in either direction to change the widget's value. Rotating the wand out of that pie menu item dismisses the tilt widget and locks its value.

Figure 8 shows an example of a tilt menu, used to change the picture border's color, and Figure 9 shows an example of a tilt dial, used to adjust the picture border's width.

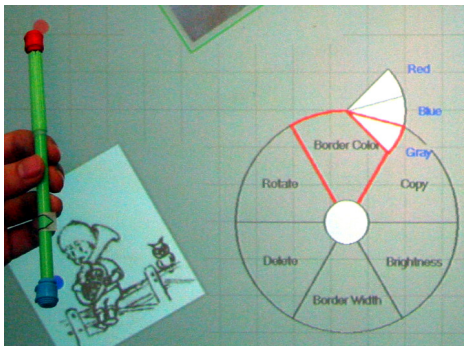

Figure 8. Tilt menu.

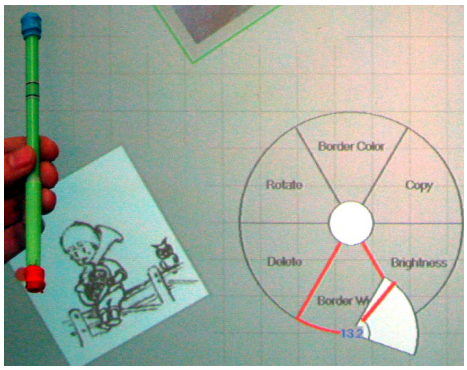

Figure 9. Tilt dial.

Given that the parent pie menu is always active when interacting with tilt widgets, we can modify several different parameters in a single continuous sequence of actions. For example, we can pop up the pie menu, adjust picture border width using the tilt dial associated with the "border width" menu item, and then change the color of the picture border using the tilt menu associated with the "border color" menu item.

The combination of pie menu and tilt widgets results in a unified compound widget for menu selection and parameter adjustment. Other researchers have also recently explored the idea of combining menu selection and parameter adjustment, resulting in several techniques such as FlowMenus [15], Control menus [24], and FaST sliders [19] that are operable using standard 2-dof input. However, we believe that ours is the first to use different input modalities – rotation for menu selection and tilt for parameter adjustment – for the two actions. It is also interesting to note that while many digitizing tablets on the market (e.g., Wacom tablets www.wacom.com) have long provided information about the tilt of the pen, as far as we know no significant exploration of the use of tilt in the interface has been conducted. As such, our work contributes in this regard as well.

We also note that these tilt widgets could be used independently from a menu, should such functionality be desirable within a particular application.

Dial Panel

Although tilt widgets can be used to adjust continuous parameters, the valid range of tilt is relatively small (about -60° to 60°), thus making it unsuitable for delicate adjustment of parameters with a large range. We have designed a *dial panel* to address this issue.

Rotation of the wand while keeping it parallel to the screen dials the arm in the panel, thus modifying the parameters. Unlike the fixed mapping between tilt angle and parameter value in tilt widgets, the parameter controlled by a dial panel is modified according to the relative change of orientation angle. In this way, we can rotate the wand cycle after cycle, and reach an infinite range of value in theory.

Even so, there is trade-off between the efficiency and precision of the adjustment of parameters. Again, we exploit the distance between the wand and the screen to switch between different granularities of adjustment. When the wand is farther from the screen, a larger panel is displayed, and the adjustment is faster and coarser. When the wand is nearer, a smaller panel is displayed, and the adjustment is slower and finer. We can start from the coarsest scale, then push forward the wand a little to enter a finer scale, and so forth. This results in a very facile technique, allowing the user to simultaneously adjust the parameter and pick the optimal tradeoff between speed and precision in a single fluid interaction. Figure 10 demonstrates using a dial panel to adjust picture brightness.

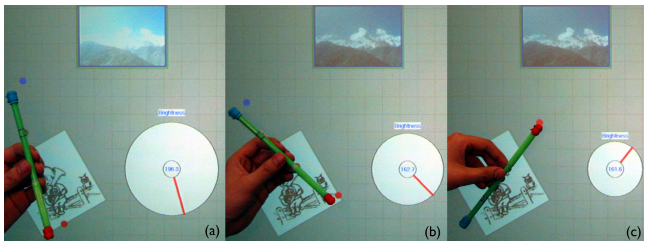

Figure 10. Dial panel. Granularity is controlled by distance of wand from screen; wand orientation dials the value. (a) Wand is farthest away from screen, resulting in coarse grain adjustment, (b) mid-grain, (c) fine-grain.

While the granularity of the dial panel could be controlled in a continuous manner, in practice we have found this difficult to use. As such, we use a discrete set of possible granularities. Again, there's a tradeoff: too small a set cannot provide the expected efficiency, while too large a set will increase the difficulty of selecting a granularity and staying in it. Preliminary experiments indicate that a set of three to four different granularities works well.

Layered Menu

When dealing with complicated tasks that require a large set of functions, a single level pie menu may not be sufficient. Simply increasing the number of items in the pie menu will result in increasing the difficulty of positioning the wand in the desired orientation. Our solution is a layered menu that increases the number of available menu items. Similar to the different layers of sub-menus in standard GUI, the layered menu organizes the items in a tree-structure. Using the distance between the wand and the screen again, the layered menu uses the metaphor of several layers of menus stacked perpendicular to the screen. Thus, we first pop up the root menu, switch to the desired item, pull back the wand a little to enter a deeper layer (if the current item has sub-menus, marked with "+"), and so forth. We can also go back to higher layers by simply pushing forward the wand to the proper depth. A parallel tap gesture executes the current item if it is a leaf in the tree. Figure 11 shows an example of the layered menu. We also apply transparency and perspective when drawing the menus, in order to reinforce the perception of layering. Note that we explored this notion of pushing and pulling in depth to activate different layers, rather than simply moving the pointer an appropriate distance from the centre of the menu as is done in traditional multi-level pie menus, because we have already used the X-Y position of the wand to control the spatial position of the menu. In other words, our VisionWand and layered menu combination affords more functionality in a single gestural action than is possible with regular 2-dof input techniques.

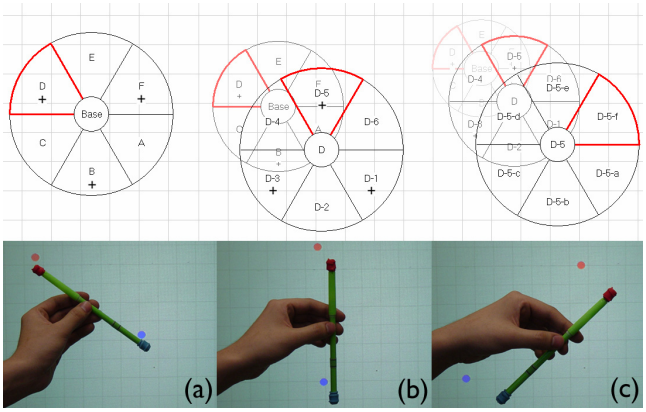

Figure 11. Layered menus. (a) Single menu, (b) If the current active menu item has a sub-menu, which is indicated by a "+" symbol, pulling the wand back from the screen activates the relevant second layer, (c) Similarly, pulling even further back activates a third layer. The wand's orientation determines the active menu item.

One concern that arises when utilizing several different gestures – tilting, rotating, pulling back – for the various widgets is that users may be confused as to which gesture to use at any given time. To mitigate this concern, we deliberately provided very different visual feedback cues (e.g., "+" symbol in layered menus to indicate the presence of sub-menus; section-shaped sub-widget with clearly demarcated sections for tilt menus) to indicate the appropriate gesture to use for the given context. In practice, we found that these cues were effective in enabling users to employ the correct gesture.

Additional Wands

By having a set of wands that can be tracked and distinguished by the system, we can significantly enlarge the interaction possibilities. One application is to give different wands different privileges, which is a natural requirement in supporting collaborative applications. For example, we may have a teacher's wand which has more privileges than a students' wand. Or we can assign the objects in the system ownership by different wands. Another application is to assign a different set of functions to different wands, thus enriching the tasks that can be achieved. In this way the wands themselves act as different physical widgets. In our current system we have partly implemented the latter idea. In addition to the basic wand that achieved the interactions we have talked about, we have another two wands with more specific functions: the navigation wand and the drawing wand.

Navigation Wand

In our application, the screen displays only part of a larger canvas. A navigation window displayed in the bottom-right of the screen shows a thumbnail of the whole canvas, and indicates the region that is being displayed by a blue rectangle. Using the navigation wand, we can navigate through the whole canvas in two ways: using direct manipulation, and via a compass widget.

Direct Manipulation: Pointing to the screen, a tap gesture grasps the canvas (the black cross changes to a hand icon). Then the whole canvas can be moved and zoomed, similar to how we manipulate an object with the basic wand. A second tap gesture releases the canvas.

Compass Widget: To achieve more smooth and precise navigation, we designed a compass widget (Figure 12). By keeping the navigation wand in parallel posture for a short while, the compass widget is triggered. Instead of being directly dragged by the user, here the canvas moves according to the posture of the wand. The orientation of the movement is the same as the orientation of the wand, and the velocity of the movement is proportional to the tilt of the wand. Thus we can navigate throughout the canvas by only rotating and tilting the wand. Moving the wand out of the screen dismisses the compass widget and locks the canvas.

As with the basic wand, a flip gesture restores the canvas to the initial position and scale.

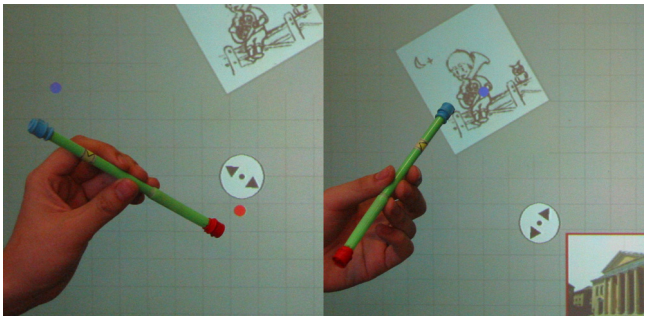

Figure 12. Compass widget allows navigation by tilting and orienting the wand

Drawing Wand

The drawing wand simulates the standard function of a pencil. The blue end acts as the tip, and the red end act as the eraser. When the blue end is touching the screen, a stroke can be drawn. Pointing the red end at a stroke for a short while deletes that stroke.

Simultaneous Tracking of Multiple Wands

Currently our system tracks only one wand at a time, but conceptually there is no reason why multiple wands could not be tracked simultaneously, further enriching the interaction space. A straightforward application of this information is to use two hands to work cooperatively. For example, use one wand to indicate the current function and state, and use another wand to perform the real manipulation. Or, use two hands jointly to manipulate large objects. Cooperative work by more than one user is also a potential area for exploration, especially for large displays.

INFORMAL USER FEEDBACK

While our system is still in an exploratory stage, we asked five graduate students in our department to try it out for some informal early feedback. Each participant was given a 10 minute demo of how the system and interaction techniques work. Then they had 20 minutes to practice the gestures and try out all the system functions. The practical restriction of the tracking system was explained to them, thus they were cooperative users. For example, they knew not to block the cameras during the test.

All users got used to the system in the given 20 minutes of practice. They did not have difficulties in learning the basic gestures within a few tries. The idea that they can interact with the system without necessarily touching the screen was found to be appealing. They also found the relationship between depth (distance between the wand and the screen) and scaling factor easy to grasp. They all became skilled in basic manipulation of the pictures in the first few minutes.

The pie menu used in the system was easily understood even by those who had never seen pie menus before. As we expected, the users began to remember the menu items even in the short 20 minutes' trial. For the most commonly used functions, they tended to put the wand in the correct orientation before the menu was actually triggered. While many of them expressed the need for the most commonly used functions be placed in the most comfortable orientations, they were not unanimous in what orientations are most comfortable. This may be an interesting topic for more extensive user testing. One participant felt it a little awkward to rotate the wand with one hand. He suggested adding a small handle perpendicular to the wand, with which the user could rotate the wand by very small finger movements.

All the participants felt comfortable when using the tilt to adjust parameters or select items using our tilt widgets. They also liked the different granularities provided by the dial panel to do very delicate adjustment of parameters. One of them even suggested having different scales of gain as well when rotating the pictures, although this may seem less obvious. With regards to the layered menu, participants were observed easily navigating between different layers by simply moving the wand forward or backward.

The compass widget used with the navigation wand was also well received, partly due to familiarity with similar widgets in standard GUI interfaces. Due to the noise in the tracking system, the drawing functionality is not yet satisfactory. The subjects suggested a list of possible widgets that may help to improve the drawing, however we feel that simply improving the tracking quality would obviate the need for these additional widgets.

Based on our observations of participants using our system, we suspect that the size of the gesture set we defined is appropriate. The participants all easily memorized the basic gestures, and were also able to perform the more complicated tasks as well.

DISCUSSION

An interesting property of the VisionWand is that when the wand is held in different ways, it generates different functionality. For example, pointing with the different ends of the wand invokes manipulation and query actions respectively, while a parallel posture of the wand activates a menu. With sufficient practice, users can begin to commit these different postures and gestures to haptic memory, allowing for very rapid execution. Rekimoto and Sciammarella [25] have argued that such "rich action" tools can significantly enhance our interaction with computational systems.

The distance between the wand and the screen is a very important dimension of information. In our design, we make a consistent association between the distance and a scaling factor. This consistency is important for the user to easily understand the interaction.

In a basic pie menu, only the orientation is used to select the item. In our tilt widgets, we demonstrated the use of both orientation and tilt, while in the dial panel and layered menu we used orientation and distance. While these dual combinations have already increased the input bandwidth as compared to standard 2-dof devices, we could go further. Given that the wand has 5 degrees-of-freedom: 2-dof position, distance, orientation and tilt, theoretically this provides the possibility of a 5-dof widget. However, we

suspect that there is some limit beyond which the complexity of the widget will overcome the advantages of simultaneously using multiple degrees-of-freedom of the wand. It would be interesting to experimentally evaluate the limits to which we can push the complexity of our wands.

In our system, gesture recognition occurs continuously without the user having to explicitly enter a "recognition" mode. This continuous recognition may raise the concern of misrecognizing ordinary movement of the wand as commands, often referred to as the "clutching problem" in virtual reality systems. However, we deliberately kept our gesture/posture set small and simple, and the interactions are highly dependent on context. Thus the clutching problem is to some extent reduced. Although in a production system, having a button would be an "easy" solution for the user to control when recognition is occurring, this would require that the wand have some active components. Our insistence on a buttonless wand for our research system allowed us to push on the boundaries of the design space of passive wands.

In addition to a standard pen-shaped wand, wands of other shapes may also provide different possibilities of interactions. A cross-shaped wand that has more than two ends may be assigned more functions, each function to one end. Alternatively, more than one end may be used simultaneously. Wands with manipulable elements, such as foldable arms, or spring-loaded barrels within wands could also be interesting possibilities for future exploration.

Finally, one could imagine designing wands with more dramatic haptic features. For example, we could make one half of the wand have a rough texture or be thicker. This would allow the user to determine the wand's orientation by tactile feel alone. We caution, however, that one of the nice features of the wands used in our system is its very low cost. Adding additional physical properties may increase the cost, another trade-off to consider.

CONCLUSIONS

Our work has explored a variety of techniques and interface widgets for interacting with large scale displays using a buttonless passive wand tracked in 3D. While our tracking implementation could be improved, it was more than sufficient to explore a wide range of alternatives. Our own experience with using the system, and observations during our informal user study, indicate that the gestures and postures of wand based interaction is easily understood and used, particularly when the set of such actions is kept reasonably small. As our interaction with computers increasingly moves away from the standard desktop to other form factors, including large displays, it is critical that we continue to explore alternative input and interaction modalities that are well suited to the new media, rather than relying by default on techniques designed for the previous generation of technology. The work presented here is one step in this exploration.

ACKNOWLEDGEMENTS

We thank Michael Wu and Gonzalo Ramos for help with image and video production; Allan Jepson, Joe Laszlo and members of the Dynamic Graphics Project laboratory (www.dgp.toronto.edu) at the University of Toronto for valuable ideas and discussions.

VIDEOS

A video demonstrating this system can be downloaded from www.dgp.toronto.edu/research/visionwand

REFERENCES

1. Baudel, T., & Beaudoin-Lafon, M. (1993). Charade: remote control of objects using free-hand gestures. *Communications of the ACM, 36*(7). p. 28-35.
2. Bolt, R. (1980). Put-that-there: Voice and gesture at the graphics interface. *ACM SIGGRAPH Computer Graphics, 14*(3). p. 262-270.
3. Buxton, W., Fitzmaurice, G., Balakrishnan, R., & Kurtenbach, G. (2000). Large displays in automotive design. *IEEE Computer Graphics and Applications, July/Aug 2000*. p. 68-75.
4. Callahan, J., Hopkins, D., Weiser, M., & Shneiderman, B. (1988). A comparative analysis of pie menu performance. *ACM CHI Conference on Human Factors in Computing Systems*.
5. Clark, J.H. (1976). Designing surfaces in 3D. *Communications of the ACM, 19*(8). p. 454-460.
6. Crowley, J., Coutaz, J., & Bérard, F. (2000). Perceptual user interfaces: things that see. *Communications of the ACM, 43*(3). p. 54-64.
7. Deering, M. (1995). HoloSketch: a virtual reality sketching/animation tool. *ACM Transactions on Computer-Human Interaction, 2*(3). p. 220-238.
8. Elrod, S., *et al.* (1991). Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. *ACM CHI Conference on Human Factors in Computing Systems*. p. 599-607.
9. Fitzmaurice, G.W., Ishii, H., & Buxton, W. (1995). Bricks: Laying the foundations for graspable user interfaces. *ACM CHI Conference on Human Factors in Computing Systems*. p. 442-449.
10. Freeman, W., Beardsley, P., Kage, H., Tanaka, K.-I., Kyuma, K., & Weissman, C. (2000). Computer vision for computer interaction. *ACM SIGGRAPH Computer Graphics, 33*(4). p. 65-68.
11. Freeman, W., & Weissman, C. (1995). Television control by hand gestures. *International Workshop on Automatic Face and Gesture Recognition*. p. 179-183.
12. Greenberg, S., & Rounding, M. (2001). The notification collage: posting information to public and personal displays. *ACM CHI Conference on Human Factors in Computing Systems*. p. 514-521.
13. Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., & Buxton, B. (2001).

Interaction techniques for 3D modeling on large displays. *ACM I3DG 1999 Symposium on Interactive 3D Graphics*. p. 17-23.

14. Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., & Buxton, B. (2002). Creating principal 3D curves with digital tape drawing. *ACM CHI Conference on Human Factors in Computing Systems*. p. 121-128.
15. Guimbretière, F., Stone, M., & Winograd, T. (2001). Fluid interaction with high-resolution wall-size displays. *ACM UIST Symposium on User Interface Software and Technology*.
16. Ishii, H., & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. *ACM CHI Conference on Human Factors in Computing Systems*. p. 234-241.
17. Krueger, M., Artificial Reality II. 1991: Addison-Wesley.
18. Kurtenbach, G., & Buxton, W. (1993). The limits of expert performance using hierarchical marking menus. *ACM CHI Conference on Human Factors in Computing Systems*. p. 35-42.
19. McGuffin, M., Burtnyk, N., & Kurtenbach, G. (2002). FaST Sliders: Integrating marking menus and the adjustment of continuous values. *Graphics Interface*.
20. Myers, B., Bhatnagar, R., Nichols, J., Peck, C.H., Kong, D., Miller, R., & Long, C. (2002). Interacting at a distance: measuring the performance of laser pointers and other devices. *ACM CHI Conference on Human Factors in Computing Systems*. p. 33-40.
21. Mynatt, E., Igarashi, T., Edwards, W., & LaMarca, A. (1999). Flatland: New dimensions in office whiteboards. *ACM CHI Conference on Human Factors in Computing Systems*. p. 346-353.
22. Oh, J.-Y., & Stuerzlinger, W. (2002). Laser pointers as collaborative pointing devices. *Graphics Interface*. p. 141-149.
23. Olsen, D.R., & Nielsen, T. (2001). Laser pointer interaction. *ACM CHI Conference on Human Factors in Computing Systems*. p. 17-22.
24. Pook, S., Lecolinet, E., Vaysseix, G., & Barillot, E. (2000). Control menus: Execution and control in a single interactor. *ACM CHI Conference on Human Factors in Computing Systems (Extended Abstracts)*. p. 263-264.
25. Rekimoto, J., & Sciammarella, E. (2000). ToolStone: Effective use of the physical manipulation vocabularies of input devices. *ACM UIST Symposium on User Interface Software and Technology*. p. 109-117.
26. Ringel, M., Berg, H., Jin, Y., & Winograd, T. (2001). Barehands: implement-free interaction with a wall-mounted display. *ACM CHI Conference on Human Factors in Computing Systems (Extended Abstracts)*. p. 367-368.
27. Schkolne, S., Pruett, M., & Schroeder, P. (2001). Surface drawing: Creating organic 3D shapes with the hand and tangible tools. *ACM CHI Conference on Human Factors in Computing Systems*. p. 261-268.
28. Segen, J., & Kumar, S. (1998). Gesture VR: Vision-based 3D hand interface for spatial interaction. *ACM International Conference on Multimedia*. p. 455-464.
29. Segen, J., & Kumar, S. (2000). Look ma, no mouse! *Communications of the ACM, 43*(7). p. 102-109.
30. Shaw, C., & Green, M. (1994). Two handed polygonal surface design. *ACM UIST ACM Symposium on User Interface Software and Technology*. p. 205-212.
31. Welch, G., & Foxlin, E. (2002). Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications, special issue on "Tracking", 22*(6). p. 24-38.
32. Wilson, A., & Pham, H. (2003). Pointing in intelligent environments with the World Cursor. *INTERACT International Conference on Human-Computer Interaction*.
33. Wilson, A., & Shafer, S. (2003). XWand: UI for intelligent spaces. *ACM CHI Conference on Human Factors in Computing Systems*. p. 545-522.