SLAP Widgets: Bridging the Gap Between Virtual and Physical Controls on Tabletops

Malte Weiss[†] **Julie Wagner**[†] **Yvonne Jansen**[†] **Roger Jennings**[‡]
Ramsin Khoshabeh[‡] **James D. Hollan**[‡] **Jan Borchers**[†]

[†]RWTH Aachen University
52056 Aachen, Germany
{weiss, wagner, yvonne, borchers}@cs.rwth-aachen.de

[‡]University of California, San Diego
San Diego, CA 92093, USA
{jennings, ramsin, hollan}@hci.ucsd.edu

ABSTRACT

We present **S**ilicone i**L**luminated **A**ctive **P**eripherals (SLAP), a system of tangible, translucent widgets for use on multi-touch tabletops. SLAP Widgets are cast from silicone or made of acrylic, and include sliders, knobs, keyboards, and buttons. They add tactile feedback to multi-touch tables, improving input accuracy. Using rear projection, SLAP Widgets can be relabeled dynamically, providing inexpensive, battery-free, and untethered augmentations. Furthermore, SLAP combines the flexibility of virtual objects with physical affordances. We evaluate how SLAP Widgets influence the user experience on tabletops compared to virtual controls. Empirical studies show that SLAP Widgets are easy to use and outperform virtual controls significantly in terms of accuracy and overall interaction time.

Author Keywords

Tangible user interfaces, transparent widgets, augmented virtuality, dynamic relabeling, tabletop interaction, multi-touch, toolkit

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Input Devices and Strategies*

INTRODUCTION

Beginning with the first computer interfaces, physical input devices have a long tradition in Human-Computer Interaction. They provide numerous benefits. Thanks to their physical, haptic nature, users can operate them in an eyes-free fashion, while looking at the screen. Graphical user interfaces, on the other hand, have the advantage of being easily positioned right at the locus of the user's attention, and configured to perfectly match a specific task. With the widespread interest in finger-operated multi-touch tabletop interfaces ([1], [3], [11], [14]), however, some shortcomings of on-screen controls have begun to show. For example, typing on a projected soft keyboard is difficult due to the lack of tactile feedback, but returning to physical input devices is not always an option. On a large table surface, a physical keyboard is either far away from the on-screen locus of attention, or it blocks part of the projection when put onto the table. On-screen buttons and scrollbars also lack tactile feedback, making it hard to operate them fluidly, but physical counterparts are not readily available.

SLAP (**S**ilicone i**L**luminated **A**ctive **P**eripherals) are transparent physical widgets made from flexible silicone and acrylic. As input devices, they combine the advantages of physical and virtual on-screen widgets. They provide a haptic operation experience with tactile feedback, supporting fluid and eyes-free operation. At the same time, thanks to their transparency, they support dynamic software-controlled labeling, using the rear projection of the interactive table they rest on. SLAP Widgets are also very simple hardware devices, without the need for tethering or any power, making them highly robust and affordable for research and prototyping. When made from silicone, they are even physically flexible and can literally be "slapped" on a table or tossed across the table from one user to another.

After a review of related research, the remainder of this paper introduces the hardware and software architecture be-

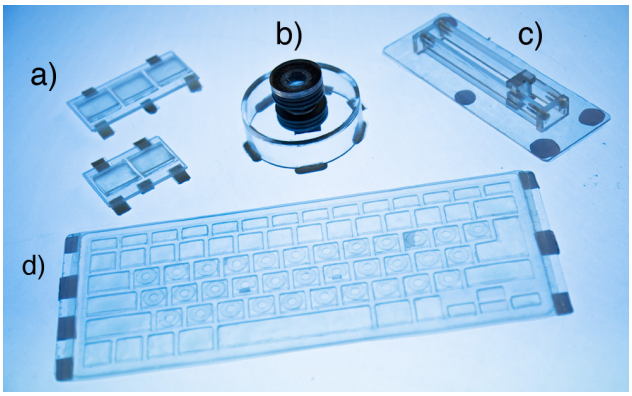

Figure 1. SLAP Widgets. a) Keypads with two and three buttons. b) Knob. c) Slider. d) Keyboard.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI 2009, April 4 - 9, 2009, Boston, MA, USA.
Copyright 2009 ACM 978-1-60558-246-7/09/04...$5.00.

hind the SLAP framework, presents our initial widget prototypes, and explains the interaction metaphors we chose for working with them. We conclude with several user scenarios that illustrate the potential of our toolkit, and a series of studies that provide more detailed qualitative feedback from users, and that show that SLAP Widgets outperform their on-screen counterparts in terms of interaction performance.

RELATED WORK

With the increasing interaction on flat surfaces without tactile feedback, research has focused more and more on compensating for this sensory lack. SenseSurface[1] recently introduced physical controls such as knobs to the desktop environment. Magnets are used to stick these controls onto the display, and internal sensors broadcast the manipulation data to the computer via Bluetooth. However, the controls of SenseSurface are opaque and cannot be relabeled dynamically.

The Optimus Maximus keyboard[2] provides dynamic relabeling of keys. Each key contains a colored 48×48 pixel OLED display that can be changed dynamically. However, the keyboard is tethered and it is rather expensive, making it inappropriate for use in tabletop environments.

With Bricks [4], Fitzmaurice et al. introduced graspable UIs on tabletops. Small bricks are used as handles attached to virtual objects, supporting two-handed interaction with physical objects. However, manipulation of data is limited to direct-manipulation, such as scaling, translation, and rotation. Object parameters cannot be changed.

Audiopad [12] combines knob-based controller "pucks" with multidimensional tracking using RFID tags. Each puck has two tags for determining angular orientation and position. Audiopad uses multiple pucks for selection and confirmation, which excludes single-handed operation. Furthermore, Audiopad pucks have no rotating axis making them drift while they are being turned.

VoodooSketch [2] supports the extension of interactive surfaces by either physically plugging widgets into a palette or drawing them. Yet this approach lacks the ability to label widgets on the fly. Furthermore, VoodooSketch tangibles need to be powered, making them more complicated and costly.

Tangible Workbench [9] provides a set of opaque and untethered objects for 3D applications. Their movement is visually tracked by moving markers underneath the widgets. However, the objects do not provide general-purpose controls since they are mapped to special functions, such as the camera-shaped objects for walking through a virtual room.

reacTable [8] implements low-cost widgets. It uses optical fiducials for tracking the position and orientation of tokens on a table. The system is created as a musician's interface. Although software could be implemented for different purposes, the reacTable tokens do only offer manipulation possibilities like positioning and turning, thus constraining the interaction. Another drawback with reacTable is that the fiducials are opaque and occlude the graphics underneath the token. Thus, custom labels can only be projected around each token.

DataTiles [15] introduce the idea of relabeling through the use of acrylic tiles. Although DataTiles mix graphical and physical interfaces, they do not fully explore the affordances of real world physical controls of the real world. Whereas DataTiles use engraved grooves usable in combination with a pen to manipulate data, they do not provide the tactile feeling of real-world controls.

In contrast to DataTiles, Tangible Tiles [16] are optically tracked acrylic tiles. They do not provide data manipulation by supporting the users' movement with grooves. Instead the user manipulates (e.g., rotating) the tile itself . The virtual object snaps to the tile when it is positioned on top of the object and its functionality is defined by the type of the tile, container or function tile. However, the user study revealed that Tangible Tiles should have distinguishable shapes to convey their functionality at first sight. Moreover, the tiles are labeled statically and cannot be used in a general-purpose context.

One of the most frequent actions people do in desktop environments is entering text. Hinrichs et al. [6] provides an overview of existing external and on-screen methods of entering text. We have yet to see keyboards that combine the advantages of physical keyboards (no visual attention required) with those of on-screen keyboards (no switching between table and external device). This would make blind tabletop typing possible, introducing new options for text entry to tabletop interfaces.

SYSTEM DESIGN

A multi-touch table provides our infrastructure for sensing physical SLAP Widgets (knobs, sliders, keyboards and keypads) as well as for displaying the virtual objects (e.g., movies, images, text fields) they modify. Widgets are transparent and utilize the rear projection display of the table to dynamically present labels and graphics around and beneath them. Associations between physical widgets and virtual objects are created and removed using synchronous tapping while halos around them indicate their status. These associations determine the labeling and graphics of the widgets. For example, a slider labeled "brightness" may have "0" and "255" at its extremes with gradations between black and white spanning its range of articulation.

Multi-touch Table

Our table uses a combination of infrared technologies to sense both surface pressures and reflected light using a single camera with an infrared filter and computer vision software. Rear projection displays graphics onto the matte touch surface without parallax errors. Silicone film between this projection/touch surface and the acrylic panel translates surface pressures to optical radiation by frustrating the total internal

[1] http://girtonlabs.googlepages.com/sensesurface
[2] http://www.artlebedev.com/everything/optimus/

reflection (FTIR) plate as described by [17] and popularized by [5]. Clusters of additional infrared LEDs are placed under the table to provide Diffuse Illumination (DI) as explained in [10]. This facilitate sensing of lightweight objects that do not register with FTIR.

The combination of FTIR and DI sensing technologies leads to a robust detection of both lightweight objects and contact pressures of fingertips. We use DI to detect the markers of objects placed on the table. FTIR is used for the detection of regular touches and keystrokes on the keyboard and the keypads.

The projector renders the graphics in a resolution of 1024×768 pixels on a 92 cm $\times$ 68 cm projection surface. The camera beneath the table captures touch events using a resolution of 640×480 pixels at 120 fps. Therefore, each pixel detected by the camera covers an area of approximately 2.03 mm^2. The table's roughly four inch wide edge allows the user to always keep the SLAP Widgets within reach.

Widgets

As shown in Figure 1, all widgets are constructed from transparent acrylic and silicone enabling the underlying graphics to shine through. Reflective markers of foam and paper create uniquely identifying "footprints", which are placed to minimize occlusion of the graphics. Reflective materials are also fastened to moving parts to track their position.

Figure 2 shows the footprints of our widgets as seen by the table's camera. SLAP Widgets are registered by the distinctive spacing of reflectors. The visual representations of widgets are aligned with these reflectors. Touches and moving parts such as the slider's handle (cf. Figure 2c) and the knob's arm (cf. Figure 2d) are tracked to update the widget state.

Keyboard

The SLAP Keyboard is a modified iSkin[3] silicone keyboard cover. Cemented onto each key is a transparent 0.01" PVC keycap providing rigidity for improved tactile feedback. These keycaps also convert the translucent matte texture of the iSkin to a transparent glossy texture improving the view of the projected graphics underneath. Two rigid strips of transparent acrylic are cemented on the edges of the keyboard to provide structural stability and reflective markers for an identifiable footprint. Keycap labels or graphics are displayed dynamically, tracking the location of the SLAP Keyboard.

Fingertip forces are conveyed directly through the keys onto the multi-touch surface, detected as blobs in particular key regions, and interpreted as keystrokes.

Keypads

Unlike the keyboard, a keypad's base is rigid, and only the actual buttons are made of silicone. At 20mm x 15mm, its keys are also much larger. Otherwise it is quite similar; fingertip force is conveyed directly and labels/graphics are displayed dynamically. Two- and three-button variations of the keypad have been fabricated. Aggregates can be constructed by fastening multiple keypads together.

[3] http://www.iskin.com

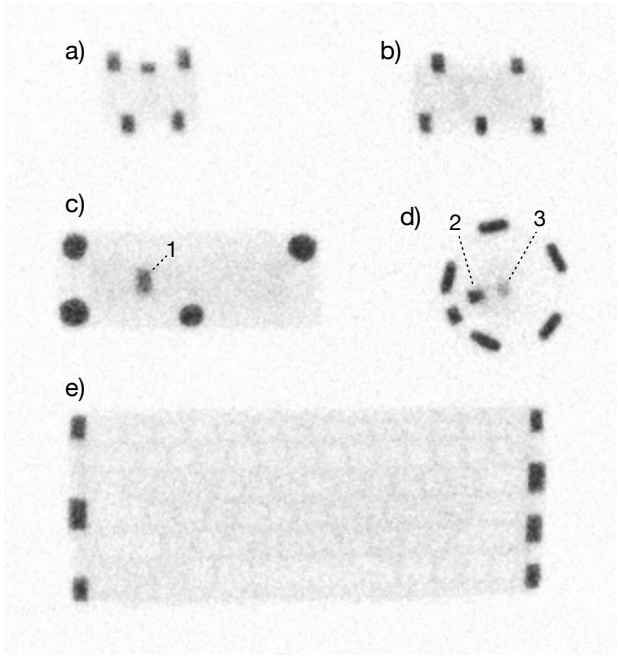

Figure 2. Footprints of SLAP Widgets (image has been inverted for better perception). a-b) Keypad with two and three buttons. c) Slider with sliding knob (1). d) Knob with angle indicator (2) and push indicator underneath the rotation axis (3). e) Keyboard.

Knob

An acrylic knob rotates on a clear acrylic base using steel ball bearings. The knob is vertically spring loaded and can be pressed as a button. An internal reflector arm orbits the axis and indicates an angular position to the camera. A reflector centered on the axis communicates the pushbutton function, and reflectors around the base provide information on its position and orientation. Using our hardware setup, we are able to detect about 90 different rotational positions.

Slider

Just as the knob, the slider is made entirely of acrylic. Two engraved sides act as rails guiding the linear motion of the sliding knob (see Figure 1c and 2c). For stabilization the slider is mounted onto an acrylic sheet. Reflective material cemented on the edges provides a footprint indicating location and orientation of the base. Reflective material placed on the slider knob indicates its linear position. According to the camera resolution and the size of the table, we can distinguish 20 different slider positions.

Pairing

A policy for connecting, or associating, physical widgets and virtual objects is implemented using a synchronous double tapping gesture. Typically a symmetric bimanual gesture, both the widget and virtual object are tapped twice in synchrony.

When first placed on a surface, a widget will display a wafting blue halo indicating the position and orientation of its footprint are successfully sensed, but the widget is lacking an association.

Associations between widgets and virtual objects are requested with synchronous double-taps of a virtual object and a widget halo. If a virtual object is not found, or it refuses the association, a red halo flashes around the widget indicating a problem. A successful association updates the widget's halo to green, associated graphics and labels are displayed in and around the widget, and it is ready for use.

If a previously associated widget is removed and returned to a surface, it will automatically restore its previous association. This permits collaborators to toss controls back and forth without loss of configuration. Associations may be removed by repeating the synchronous double tapping gesture, or replaced by associating the widget with a new virtual object. Multiple widgets may be associated with a single virtual object, but currently a widget may be associated only with one virtual object at a time.

Let's look at an example. A SLAP two-button Keypad widget, e.g., is thrown on the table and glows blue. A video virtual object is synchronously tapped with the Keypad's halo. The halo changes green before fading out, and graphics for *Play* and *Stop* are displayed under the keys. When the keypad is picked up by a collaborator and placed at a new position, its button labels are restored immediately.

Software architecture

As shown in Figure 3, the software architecture of our system consists of three layers: 1) the multi-touch framework, 2) the SLAP User Interface Toolkit, and 3) the application.

Multi-touch Framework

The lowest layer receives the camera image from the multi-touch table and detects touches by conventional computer vision algorithms using background subtraction, thresholding, and simple spot detection. Spots are converted into circles and sent as touch events to the next higher layer. The framework does not distinguish between spots created by surface pressure (FTIR) or reflections (DI).

SLAP User Interface Toolkit (SLAP UITK)

The SLAP UITK receives touch events, accumulates an active list, and looks for a matching footprint in the SLAP widget set. A widget footprint consists of three parts: the static *type footprint* specifying the kind of widget, a set of touches defining the widget's id (*id footprint*) and one or more touches that determine the widget's state (*state footprint*). When a footprint is detected, its id is extracted and a *SLAPWidget* object is created, providing a visual representation of the widget on the multi-touch table. The toolkit tracks the footprint and ensures that the visual representation is always aligned to the physical widget. When the state footprint changes, e.g., if a spot appears in the keyboard area indicating a keystroke, the UTIK notifies the *SLAPWidget* *object* which transforms the state change into a canonical event.

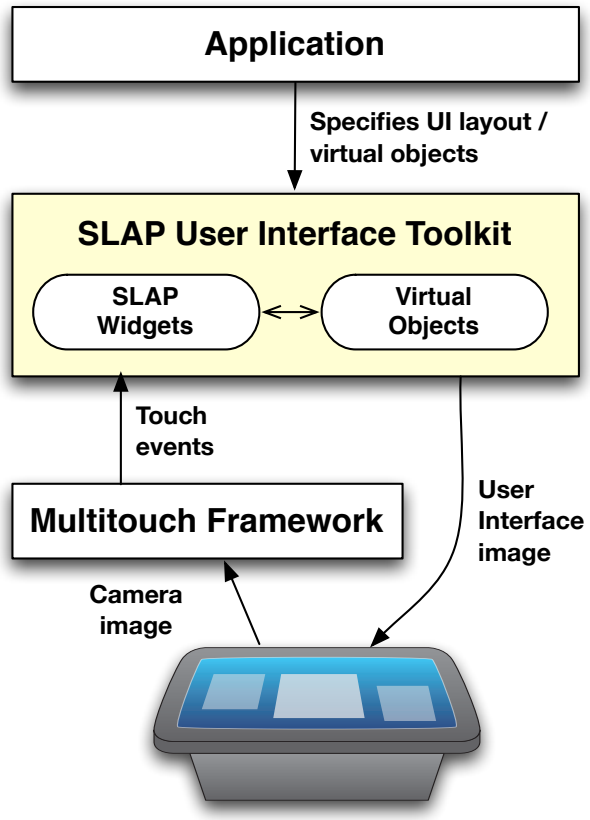

Figure 3. SLAP Software architecture.

The toolkit is also responsible for storing and drawing the virtual objects. It provides conventional direct manipulation interaction methods for tabletops: objects can be dragged using one finger and rotated/scaled by dragging with two fingers. We developed a small library of virtual objects for text fields, images, and movies to quickly prototype various usages and support initial experiments.

Finally, the SLAP UITK handles the pairing mechanism. If the user wants to pair a widget with a virtual object, the SLAP UITK sends a pairing request to the object. The virtual object can either accept or reject the widget depending on its type. When accepting, the virtual object configures the visual representation of the widget, e.g., by setting the button images of a keypad or by defining the menu items of a property menu. Accordingly, the widget is fully functional and all events, such as pushing a button or selecting a menu item, are sent to the virtual object.

Application

On the highest layer, developers specify the user interface of their applications. Since the SLAP UITK encapsulates communication with the widgets, developers can easily set up the design by creating and arranging virtual objects.

Figure 4. SLAP Knob user interface. a) Selecting image property from menu. b) Setting continuous value. c) Relative navigation for frame stepping in videos.

Extending the framework

The object-oriented nature of the framework simplifies creating new widgets. *SLAPWidget* provides a base class for instantiating widgets, encapsulating communication with virtual objects, and providing standard methods to visualize a widget on the screen. New widgets are registered with the framework by subclassing from this base class, specifying the type footprint, and overwriting the drawing and communication methods. In a similar manner, new virtual objects are developed by subclassing from class *SLAPVirtualObject*.

USER INTERFACE

Keyboard

Keyboards are arguably the most necessary input devices for computers. Virtual keyboards have gained popularity with the emergence of multi-touch technology. However, they lack the haptic feedback traditional keyboards offer. This leads to problems for touch typists who rely on the sense of touch to guide text input.

The SLAP Keyboard attempts to alleviate the problems introduced by virtual keyboards, while taking advantage of the capabilities of multi-touch technology. The user can place the widget anywhere on the surface, pair it with an application, and begin to enter text as if using a traditional keyboard.

However, there is no reason that this keyboard should be limited to just matching normal keyboard behavior. When a user presses the "<CTRL>" (control) modifier, keys can be dynamically relabeled to indicate what the key combinations mean (see Figure 5). For example, the "c" key can display the word "copy" or possibly a small image to illustrate that a "<CTRL>+C" combination performs a copying operation. Thus, the keyboard becomes a dynamic medium that can support a specific application's usage needs.

Keypad

Applications frequently do not require a full keyboard for their manipulation. For example, a video player may need only a few buttons for playing, pausing, rewinding, and fast-forwarding. It is important that a keypad exists whose buttons can be relabeled with ease. Moreover, users may find a full keyboard that is completely relabeled for a task to be quite confusing since the form factor may suggest that it provides normal keyboard functionality. Additionally, fewer buttons are easier to locate than arbitrarily assigned keys on a full keyboard.

For these situations, we designed the SLAP Keypad. With several keys in series, the keypad suggests that users define its mode of operation according to their specific application needs. We built keypads with two and three keys. They can be combined for tasks where more buttons are needed. A typical application for a three-keypad would be the video navigation we just mentioned. When paired with a video object, this is the default layout. It is also possible to pair a keypad with an application controller to provide shortcuts to often used functions for all objects in the application on dedicated keys, e.g., cut/copy/paste as known from the Xerox Star [7].

Knob

Knobs are often found in software audio applications because they mimic the physical mapping found in production-level mixers. Their familiarity helps the user to grasp their intended function. However, using them is difficult when there is no tangible control.

Our SLAP Knob physically features turning and pushing. These two simple functions are mapped onto different virtual representations depending on the object with which it is paired. Once paired with an application, the knob enables the user to manipulate it much like a traditional knob. Volume controls intuitively map to the knob. In one user test, we

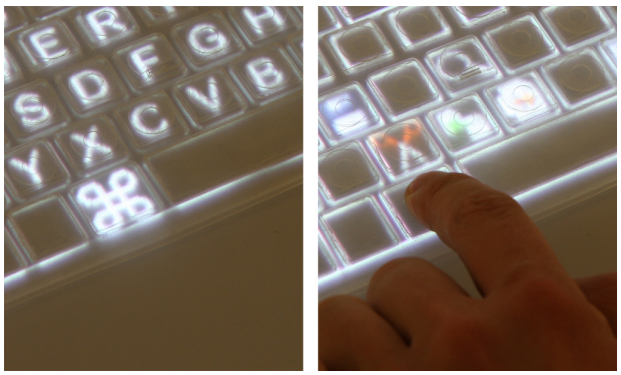

Figure 5. Dynamic relabeling of SLAP Keyboard.

used a knob for fine-navigation of video, i.e., frame stepping (Figure 4c), in another for setting image parameters (Figure 4a-b).

Additionally, since our widgets are not limited to a single object, a more complex interaction is possible when it is paired to an object with several properties, e.g., an image, as a property editor. By rotating it, the user shuffles through a circular menu of properties. To select a property to change, the knob is pressed and released once. The current value is then displayed underneath it and can be changed by turning the knob. The property can be adjusted with a high degree of precision. A second push confirms the new value and lets the user choose another property.

We explored a quasi-modal interaction [13] requiring the user to turn the knob while pushed-down to change values. However, quick tests showed several interactions where the user accidentally stopped pushing while turning and hence selected a different property whose value was then inadvertently changed.

Slider

Slider bars are quite common in graphical user interfaces, from scrollbars to parameter adjustment bars. Our slider is unique in that it facilitates the use of a single slider bar for all applications. The pairing/un-pairing mechanism allows for quick application switching by a pair of quick double taps. Furthermore, the property value is projected directly below the slider to aggregate all slider-related activity to one particular location.

The slider can be used for any interaction in which an absolute value needs to be set. It could be used as a physical timeline for fast navigation in a video object, or as an analog slider for setting volumes in an audio context. As with all the other SLAP widgets, the possibilities are numerous and depend solely on the virtual object. The slider can also complement the knob's functionality if a frequently changed property is assigned to it.

USAGE SCENARIOS

SLAP Widgets offer versatility and ease-of-use. Having no electronic parts, they are simple, affordable, flexible, and robust. The user can literally slap a widget onto the multi-touch surface and is ready to interact with it. Versatility is seen with respects to pairing and relabeling. Although each widget has its rigid nature, cast from silicone or built from acrylic, its functionality can vary significantly based upon which application pairs with it.

SLAP Widgets can be used in any application that requires parameter changing functionality, expert shortcuts or text entry. Since it is desirable to have a large number of virtual objects on the touch surface but not to have a multitude of physical controls linked to each one cluttering the surface, SLAP fades controls into view when they are required on a virtual surface, and lets them disappear when they are physically removed from the table, avoiding the cognitive load of remembering different gestures. SLAP supports flexible interaction through a small number of controls for an arbitrary number of virtual objects. The following usage scenarios will emphasize the flexibility of SLAP Widgets, since the same physical widgets are used in all scenarios.

Collaborative Usage Scenario

One of the primary advantages of multi-touch tables is to support collaborative work of multiple people. Situated around the multi-touch table, several collaborators may work together. An individual can be typing annotations with the keyboard when a second person will want to enter something interesting that comes to mind. In this situation, a normal keyboard would require the first person to hand over the tethered keyboard which might be complicated by cable length. The cable may also reach over another person's workplace disrupting their work. It could also require the annotator to walk away from the table to the location of the first person to enter the annotation. With SLAP, however, it becomes a trivial matter. The first user grabs the flexible silicone keyboard and simply tosses it to the second person with no fear of damaging anything, and the annotation can be made with little effort.

Video Ethnography Scenario

Video ethnographers often need to analyze immense amounts of video data. Typically they work on desktop workstations using existing tools, such as QuickTime and Microsoft Excel, to do their analysis. Multi-touch tables pose an alternative to the current ethnographic working environment as presenting the user with much larger screen real estate, providing a collaborative space and an opportunity for new methods of interacting with the data.

We have developed an application for video ethnography as part of our user study. A major task that all ethnographers undertake is fine-scale navigation in videos. To assist navigation, we implemented frame-by-frame navigation using the SLAP Knob. Alternatively, we also let the user manipulate the SLAP Slider for rough navigation. For annotations related to video clips or images, the SLAP Keyboard is used. Linked with the object, the table projects the keyboard layout, and then the user can quickly enter relevant notes. We also implemented a function using the SLAP Keypad to "bookmark" frames of interest.The keypad button changes to a small thumbnail of the bookmarked frame. The slider can be used to browse through the bookmarked scenes.

Image Editing Scenario

Editing images represents another use of SLAP Widgets. The SLAP Knob provides an intuitive facility for browsing and modifying image properties. We implemented a menu to cycle through parameters like brightness, contrast, saturation, etc. (see Figure 4a). This eliminates the need for complicated menus and submenus that often mask crucial abilities from the novice user. When pushing down the knob, the user can change the specific parameter (see Figure 4b). Pushing again returns to the menu selection. A crucial benefit of SLAP Widgets for image editing is that the user can focus on the images as the properties are adjusted since tactile

information provides sensory input outside the visual locus of attention.

Interface Designer Usage Scenario

Our widget framework can also serve as a toolkit for interface designers working on tabletop applications. They can take advantage of the available widgets and develop a SLAP-based facility for their work. For instance, a designer fashioning an audio mixing application may want to place sliders to represent volume and equalizer levels, knobs to represent gain and fader settings, and keypads for playback controls. In fact, designers may even choose to use SLAP Widgets on a table to cooperatively prototype a traditional application for the desktop.

USER STUDIES

Knob Performance Task

Video navigation and annotation require users to manipulate controls while visually attending to video. However, virtual controls typically also require visual attention. In contrast, tangible controls may be manipulated without any visual attention. Compared to virtual controls, we anticipated that the SLAP Knob would improve performance of video navigation and annotation. Performance measurements include elapsed times for participants to complete the whole task, elapsed time to navigate to a particular frame in the video, and the number of navigational overshoots past a target frame.

Hypothesis 1: navigation times for SLAP Widgets are less than their corresponding virtual controls.

Hypothesis 2: navigational overshoots for SLAP Widgets are less than their corresponding virtual controls.

Hypothesis 3: task completion times for SLAP Widgets are less than their corresponding virtual controls.

Experimental Design

Our experiment consisted of two conditions that differed only by the use of SLAP Widgets or virtual controls. In both conditions all controls were fixed at the same positions in the same orientation.

1. *Condition "SLAP".* All controls, two keypads and a knob, were SLAP Widgets with their respective rear-projections.
2. *Condition "Virtual".* All controls were purely virtual, that is, no widgets were placed on the table, but the graphics were the same as in the first condition. The keypad buttons were triggered by regular touches. The virtual knob used the standard method of tracking as established in today's desktop applications: When the user holds down her (index) finger in the knob area, knob rotation follows the finger until it is released, even if dragged outside the area.

Each condition consisted of four trials, and each trial consisted of three instances of navigating to a frame in a video and marking it. A set of eight video clips was randomly sequenced for each participant; four for the first condition and four for the second. Each participant was randomly assigned to a condition.

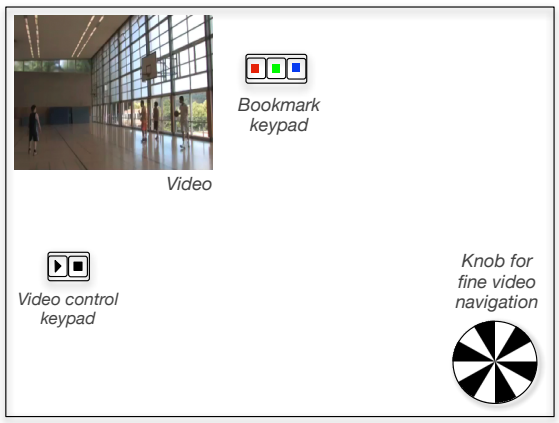

Figure 6. Layout of quantitative test setup

	condition	N	mean	std. dev.
overshoots	Virtual	191	3.13	1.88
	SLAP	189	2.11	0.91
knob interaction time	Virtual	191	6.46	5.35
	SLAP	189	4.47	3.23
overall interaction time	Virtual	126	10.57	5.55
	SLAP	124	9.11	3.82

Table 1. Results for quantitative analysis of knob performance

	T	p
overshoots	6.7	< 0.01
knob interaction time	4.4	< 0.01
overall interaction time	2.4	0.016

Table 2. t-test for results

Participants

Volunteer participants were recruited from a university campus using a general posting in a cafeteria and from a presentation on multi-touch technology. A total of 21 volunteers participated, 19 male and 2 female, between the ages of 22 and 36 with an average age of 25.7. Three were left-handed, 18 right-handed, and none reported color vision deficiency.

Method

Participants were presented with a multi-touch table with a video window, a bookmark pad, a control pad, and a navigation knob (see Figure 6). Depending on the condition, widgets were or were not in place. The goal of finding and tagging three frames in a video clip was explained. The task was to navigate the video using a knob and keypad, locate tinted frames, and tag them using a bookmark keypad. Frames tinted in red were to be tagged with a red bookmark, similarly for green and blue. A host computer recorded all actions in a time-coded log file for later statistical analysis.

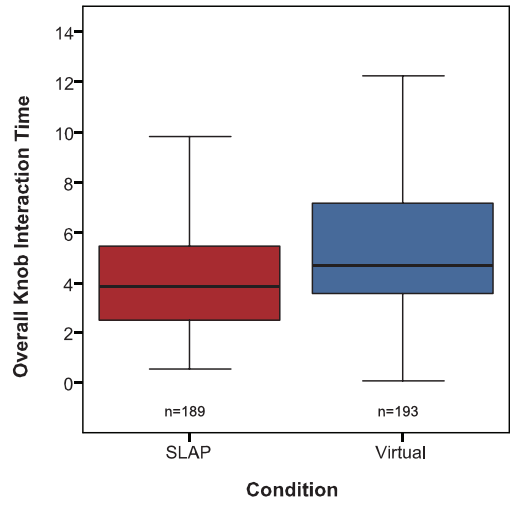

Figure 7. Overall knob interaction time depending on input type.

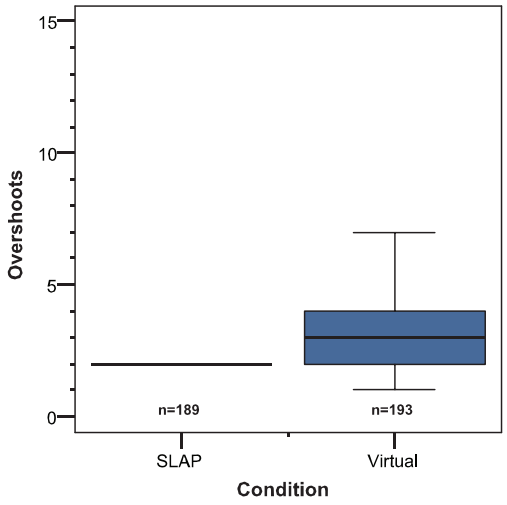

Figure 8. Overshoots depending on input type.

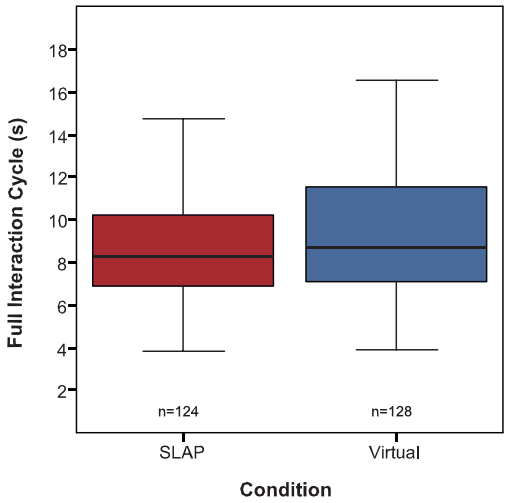

Figure 9. Duration of full interaction cycle depending on input type.

Typically, a participant would press the *Play* button to start the video, press the *Stop* button when a tinted frame was noticed, navigate frame by frame using the navigation knob until the exact tinted frame was displayed, press a bookmark button to tag it, and press *Play* to continue searching for any remaining tinted frames.

Results

Our quantitative results are summarized in Tables 1 and 2. Fine video navigation to specific target frames was significantly faster using the SLAP Knob compared to virtual graphics only (averages of $4.47s$ vs. $6.46s$, $p < 0.01$, cf. Figure 7), and produced fewer overshoots (averages of 2.1 vs. 3.1, $p < 0.01$, see Figure 8). Moreover, it took significantly less time to complete a task using SLAP Widgets than with their virtual counterparts (average $8.6s$ vs. $10.75s$, $p < 0.05$, cf. Figure 9).

Discussion

Our study revealed that navigation using virtual knobs required more time and produced more overshoots of the target keyframe compared to the SLAP Knob. We believe the reason for this difference to be that the virtual knobs need visual attention and lack tactile feedback. Participants needed to look to position their fingers at the virtual knob. Also, when their finger drifted away from the central point, the irregular scrolling speed of the video forced participants to correct their finger position. The SLAP Knob instead was grabbed and turned mostly without any visual attention, leading to less overshoots and shorter interaction times.

Qualitative evaluation

Are SLAP Widgets easy to associate and manipulate? What do people like, dislike, or want to change about them? These are important questions that were approached by using a set of tasks to familiarize participants with the widgets.

Participants

All participants were expert computer users experienced with using graphical user interfaces and recruited from a university campus. 7 male and 3 female, between ages of 21 and 28, volunteered to participate and consented to video recording.

Method

Participants were presented with a multi-touch table displaying a video window, an image window, and a text field. The experimenter introduced the SLAP Widgets and provided a 5-minute demonstration of their use including synchronous pairing gestures. Participants were requested to perform the following series of control, navigation and editing tasks followed by an interview to provide feedback. The tasks and interview were recorded and reviewed.

1. Video Control: place a keypad widget on the table, associate it with the video window, and control the video using *Play* and *Pause* buttons of the keypad widget.
2. Video Navigation: place SLAP Slider and SLAP Knob on the table, associate them with the video window, scroll

through the video using the slider for gross navigation and the knob for fine steps between frames.

3. Image Editing: re-associate the SLAP Slider and SLAP Knob to the image window, adjust brightness with the slider and saturation with the knob.

4. Text Editing: place a SLAP Keyboard on the table, associate it with the text field, type your name, re-associate the knob to the text field, and modify text color with the knob.

Results

Most (9/10) participants declared manipulating the SLAP Widgets was intuitive and self-evident. One participant emphasized that widgets map well-known physical control elements to their virtual equivalents and may be particularly well adapted for people not familiar with virtual controls. Another participant commented on how the widgets permit resting her hands on them while not using them (something not possible with virtual keyboards and controls). Associating gestures were immediately understood by all participants and used readily. Comments also indicated that it felt similar to setting a foreground GUI window. Some (4/10) participants suggested alternative association gestures such as placing a widget on a virtual object and sliding it to a comfortable position not occluding any virtual objects ("grasp and drag" of control properties), but also felt that synchronous double-tapping was particularly appropriate for the keyboard.

Some (4/10) participants felt the SLAP Widgets were too quiet and could benefit from auditory feedback, particularly the keyboard. Feedback on the keyboard was mixed, some commented on improvements to feel the edges of the keys and keycap contours as well as a more traditional tactile response.

Discussion

The participants felt generally positive about the SLAP widgets. After a short demonstration they were able to complete basic sample tasks with the widgets. Based on this user feedback, we will continue to iterate on our widget designs.

The concept of our haptic SLAP Keyboard was appreciated. However, most users still felt more comfortable with the virtual version. We identified two reasons for that: first, the use of DI yielded false positives due to hover effects, and second, the pressure point of the silicone keys was not clear enough, i.e., users had problems to determine how hard a key had to be pressed. Both issues will be addressed in future iterations.

CONCLUSION AND FUTURE WORK

Our studies showed that users enjoyed using the SLAP Widgets. The mechanism of pairing SLAP Widgets with virtual objects was easily understood. However, most users stated that the association technique could be simpler, for example, by placing widgets directly on the virtual object to link them. We will explore alternative pairing strategies in future work.

In our qualitative user test, we investigated $n : 1$ mappings, that is, multiple SLAP widgets were mapped to single virtual objects. We will explore more general mappings ($n : m$) in future experiments.

The quantitative studies exposed that SLAP Widgets improve tasks in which the visual attention is not focussed on the control but on the virtual object that is modified, that is, SLAP Widgets support eyes-free controls. Although they do not have the same resolution as widgets using real potentiometers, SLAP Widgets are still usable for relative adjustments of values. There is potential to further improve the performance of SLAP widgets.

It might be necessary to rebuild the keyboard using a custom keyboard mold, rather than a modified iSkin. Furthermore, we will include auditory feedback for keystrokes. In addition, we will build multiple SLAP Knobs and SLAP Sliders with different form factors and further investigate their usefulness in different multi-touch applications. Currently, the SLAP widget set represents "verbs" that allow users to manipulate the parameters of virtual objects. We are planning to introduce tokens that represent passive "nouns". For example, these could be used as containers to store pairings between widgets and virtual objects, such that they can be quickly restored later. Finally, we will implement further applications as explained in the user scenarios and focus on the usability of our SLAP Widget in collaborative contexts.

ACKNOWLEDGEMENTS

This work was funded by the German B-IT Foundation, NSF Grant 0729013, and a UCSD Chancellor's Interdisciplinary Grant.

REFERENCES

1. H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1263–1272, New York, NY, USA, 2006. ACM.

2. F. Block, M. Haller, H. Gellersen, C. Gutwin, and M. Billinghurst. VoodooSketch: extending interactive surfaces with adaptable interface palettes. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 55–58, New York, NY, USA, 2008. ACM.

3. P. L. Davidson and J. Y. Han. Synthesis and control on large scale multi-touch sensing displays. In *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*, pages 216–219, Paris, France, France, 2006. IRCAM — Centre Pompidou.

4. G. W. Fitzmaurice, H. Ishii, and W. A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

5. J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on*

User interface software and technology, pages 115–118, New York, NY, USA, 2005. ACM.

6. U. Hinrichs, M. S. Hancock, M. S. T. Carpendale, and C. Collins. Examination of text-entry methods for tabletop displays. In *Tabletop*, pages 105–112, 2007.
7. J. A. Johnson, T. L. Roberts, W. Verplank, D. C. Smith, C. H. Irby, M. Beard, and K. Mackey. The Xerox Star: A retrospective. *IEEE Computer*, 22(9):11–29, 1989.
8. S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 139–146, New York, NY, USA, 2007. ACM.
9. T. Kienzl, U. Marsche, N. Kapeller, and A. Gokcezade. tangible workbench "TW": with changeable markers. In *SIGGRAPH '08: ACM SIGGRAPH 2008 new tech demos*, page 1, New York, NY, USA, 2008. ACM.
10. N. Matsushita and J. Rekimoto. HoloWall: designing a finger, hand, body, and object sensitive wall. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210, New York, NY, USA, 1997. ACM.
11. M. R. Morris, A. Huang, A. Paepcke, and T. Winograd. Cooperative gestures: multi-user gestural interactions for co-located groupware. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1201–1210, New York, NY, USA, 2006. ACM.
12. J. Patten, B. Recht, and H. Ishii. Audiopad: a tag-based interface for musical performance. In *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6, Singapore, Singapore, 2002. National University of Singapore.
13. J. Raskin. *The humane interface : new directions for designing interactive systems*. Addison-Wesley, Reading, Mass. [u.a.], 2. print. edition, 2000.
14. J. Rekimoto. SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, New York, NY, USA, 2002. ACM.
15. J. Rekimoto, B. Ullmer, and H. Oba. DataTiles: a modular platform for mixed physical and graphical interactions. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 269–276, New York, NY, USA, 2001. ACM.
16. M. Waldner, J. Hauber, J. Zauner, M. Haller, and M. Billinghurst. Tangible tiles: design and evaluation of a tangible user interface in a collaborative tabletop setup. In *OZCHI '06: Proceedings of the 18th Australia conference on Computer-Human Interaction*, pages 151–158, New York, NY, USA, 2006. ACM.
17. W. White. Method for optical comparison of skin friction-ridge patterns, 1965. U.S. Patent 3,200,701, 13.12.2002.