Supporting Responsive Cohabitation Between Virtual Interfaces and Physical Objects on Everyday Surfaces

ROBERT XIAO, Carnegie Mellon University
SCOTT HUDSON, Carnegie Mellon University
CHRIS HARRISON, Carnegie Mellon University

Systems for providing mixed physical-virtual interaction on desktop surfaces have been proposed for decades, though no such systems have achieved widespread use. One major factor contributing to this lack of acceptance may be that these systems are not designed for the variety and complexity of actual work surfaces, which are often in flux and cluttered with physical objects. In this paper, we use an elicitation study and interviews to synthesize a list of ten interactive behaviors that desk-bound, digital interfaces should implement to support responsive cohabitation with physical objects. As a proof of concept, we implemented these interactive behaviors in a working augmented desk system, demonstrating their imminent feasibility.

CCS Concepts: • **Human-centered computing → Graphical user interfaces**; • **Computing methodologies → Mixed / augmented reality**; *Graphics input devices*

KEYWORDS

On-world interaction; touch interaction; mixed physical-virtual systems; physical desktops

1 INTRODUCTION

Researchers and practitioners have long articulated the vision and promise of digitally augmented desks. Seminal work emerged in the early 1990's, most notably Xerox PARC's DigitalDesk [34, 48, 49]. Since then, dozens of systems have been proposed and built, demonstrating superposition of content onto physical artifacts [40, 48], the use of physical objects for tangible interaction [14, 45], in situ remote collaboration [22, 46, 48], and more generally, interactive applications on desk surfaces [23, 25, 36, 53].

However, a notable commonality of these futuristic systems is the minimalist nature of the surfaces used – often lacking keyboards, mice, mugs, papers, knickknacks, and other contemporary and commonplace items (Figure 1). Of course today's desk surfaces play host to a wide variety of items of varying shape and size. Moreover, these objects rarely conform to a grid or even common orientation. Desks are also constantly in flux, with items moving, stacking, appearing and disappearing. Example events include sliding a laptop out the way to make room for new work, or resting a fresh cup of coffee on the work surface.

If digital desks do not account for these basic physical actions, applications can become brittle (*e.g.*, does a mug placed on top of a virtual keyboard inject spurious touch input?) or inaccessible (*e.g.*, if a book is placed

Author's address: Carnegie Mellon University, Human-Computer Interaction Institute, 5000 Forbes Avenue, Pittsburgh, PA 15213

2573-0142/2017/06 - 11 $15.00
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
http://dx.doi.org/10.1145/3095814

over an interface, how does one access it?). Further, because physical objects cannot move or resize on their own, the burden of responsiveness falls to the digital elements. Thus, digital applications must employ a variety of strategies to successfully cohabit a work surface with physical artifacts.

To help close this gap, we conducted an elicitation study with ten participants at their personal desks to understand how applications could respond to different events. From this, we derived a list of ten fundamental interactive behaviors that responsive desk-bound virtual applications should exhibit. To demonstrate these behaviors can be achieved practically and in real time, we built a proof-of-concept system with the necessary technical advances to support each behavior. This system had to move beyond prior work in several key ways; for example, our system requires no calibration to the world, allowing the desk scene to be in flux (*i.e.*, there is no notion of a "background"). Further, our touch tracking approach distinguishes human "objects" (arms, hands, fingers) from other objects. This ability is critical for responsive interfaces, which must respond to user movement and input differently from changes to the physical environment (*e.g.*, interfaces should evade from your coffee mug, but not from your hands).

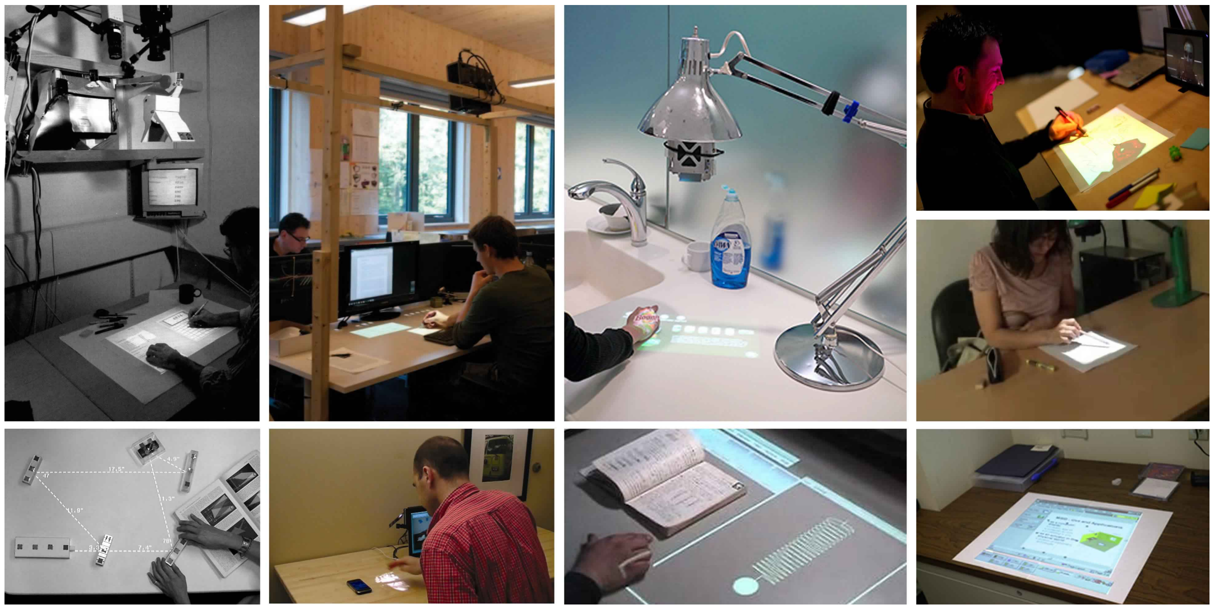

Figure 1. Clockwise from top left: Digital Desk [48], Hardy [18], LuminAR [30], IllumiShare [22], AR Lamp [25], Everywhere Displays [36], Enhanced Desk [27], Bonfire [23], I/O Bulb [46]. Note the general lack of items in the interactive area. The few physical objects that do have digital interactivity are either tagged (*e.g.*, fiducial markers), are special tangibles, or require a custom interactive table (*e.g.*, FTIR).

2 RELATED WORK

Behaviors surrounding physical desks and workplaces have long intrigued researchers from fields including management sciences, anthropology and ergonomics (see *e.g.*, [31, 42, 47]). More recently, HCI researchers have studied desk practice to better understand how to integrate digital workflows (*e.g.*, [7, 17, 18, 31, 43]). While this prior work provides great insight into the culture of desk practice, it tends to ignore the minutiae of small-scale, desk-level interactions. Instead, we more deeply review prior work on augmented desktop interactions relevant to our present research, as well as work on the technical underpinnings of such augmented desk systems.

3 AUGMENTED DESKTOPS

The concept of an augmented desktop – either projected [48] or through AR/VR technologies [33] – goes back to at least the 1970's with Knowlton's optically superimposed button array [26]. However, the full vision of the augmented desk did not appear until the early 1990s, with seminal systems such as Xerox PARC's digital desk [34, 48] and Ishii's TeamWorkStation [20]. The concept was rapidly expanded upon in the 90's with key systems including interactiveDESK [3], EnhancedDesk [27], I/O Bulb [46], Illuminated Light [45], Office of the Future [38] and the Everywhere Displays Projector [36].

Recent advances in electronics and networking have created opportunities to refine the augmented desk experience. Magic Desk [5] prototyped an augmented desk interface using a physical touchscreen as the desk surface. IllumiShare [22] offers a sophisticated, end-to-end desktop remote collaboration experience, while Bonfire [23] takes the concept mobile with cameras and projectors operating behind the lid of a laptop. MirageTable [4] merges physical and virtual objects on a tabletop, with a physics-based interaction approach. Newer projects, such as LuminAR [30] and AR Lamp [25], put forward light-bulb-like implementations in attempts to achieve the technical vision proposed in the I/O Bulb [46] twenty years earlier.

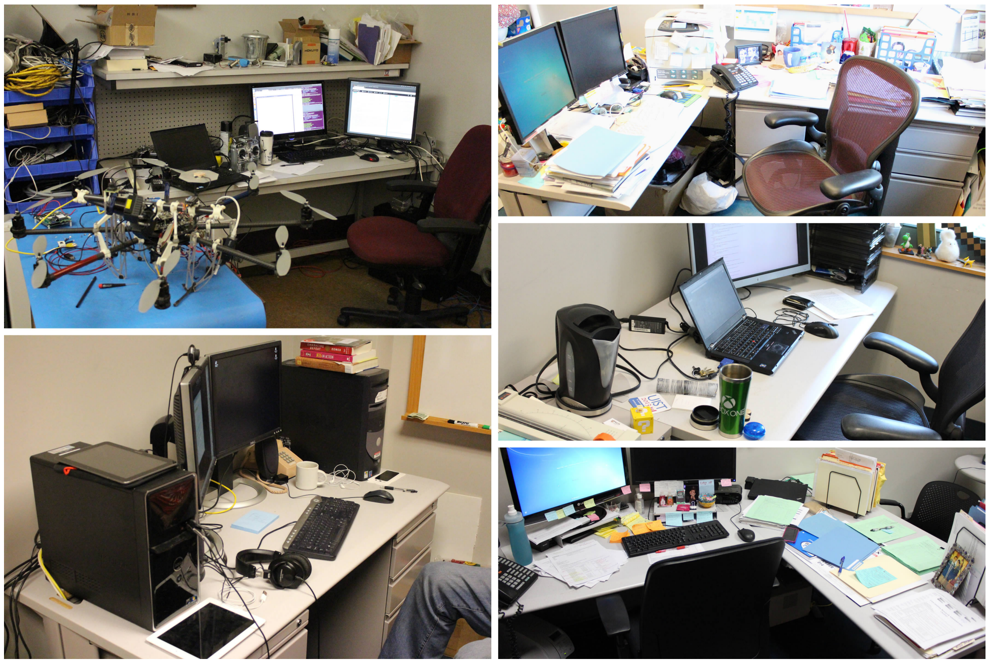

Figure 2. Example real-world desks of our participants.

Systems that superimpose rectified information onto physical artifacts (*e.g.*, [34, 45, 48]) might be described as using "following" or "snapping". However, there is an important difference between projected content that merely tracks with physical objects, and an interface that attaches to an object's 3D geometry and follows its movements. The closest related work is the "binding" behavior described in Live Paper [40].

Lastly, other efforts, such as WorldKit [53], aim to bootstrap on-world application development by offering an SDK to abstract away many of the complexities of operating on everyday surfaces (*e.g.*, touch tracking, rectified projected output). There have also been recent design-oriented efforts to study *e.g.*, augmented desk usage in the wild [18], as well as superior desk form factors [52].

3.1 Touch Sensing on the World

Numerous approaches have been proposed to enable touch- and gesture-sensitive work surfaces. One option is to have a specially instrumented surface that uses *e.g.*, acoustic [48], capacitive [5, 39], optical [44] or other sensing approaches. To enable touch interaction on conventional desks without instrumentation, systems typically use computer vision, generally requiring a camera operating above the surface. Many schemes have been proposed, including finger template matching [27], markers [32], skin-color segmentation [23, 25], residual thermal imprints [28] or contour-based classification and tracking [10].

Most similar to our own system are those employing depth cameras for touch sensing. Many such systems rely on a background model or background subtraction for touch sensing [21, 50, 51, 53], precluding their use in dynamic environments. Notably, however, OmniTouch [19] used a template-matching scheme for finding fingers and could thus operate on dynamic scenes.

3.2 Optimization-Based Layout

Our system handles layouts for irregular desk topographies using an optimization-driven approach. Optimization has long been used for spatial problems such as graph visualization [13], VLSI layout [11] and more recently, for problems in perceptually optimized display generation [2, 29]. More relevant are optimization based techniques employed for automatic generation of user interface layouts (see *e.g.*, [6, 15, 16, 41]). As far as we are aware, our work is the first to use optimization-based layout for on-world interactive purposes.

3.3 Projection Mapping

There are several approaches for projecting onto the complex and often irregular geometry of the world around us. For example, "The Office of the Future" [38] proposed using 3D head tracking and office-wide depth sensing to project imagery onto irregular surfaces, such that it would appear correct from the user's perspective. More recently, WorldKit [53] enabled the user to "paint" various widgets onto different flat surfaces in the environment. iLamps [37] used structured light to sense the 3D geometry of a projection surface (*e.g.* multiple walls or curved surfaces) and uses these data to minimize visual distortion of projected output. Finally, depth cameras have made it easier to sense the geometry of environments and perform projection mapping, enabling real-time rectification onto moving targets [19].

4 ELICITATION STUDY

To help identify useful interactive strategies, we recruited ten participants (three female, mean age 31) for a one-hour study. This study was conducted at participants' own desks (see Figure 2 for some examples) for ecological validity. Common desk-bound items included computers, monitors and related accessories; desk phones; stacks and files of papers; carried items such as wallets, phones and keys; personal items such as memorabilia, photographs and gifts; coffee mugs; and books.

We started with a general interview. Participants repeatedly articulated that frequently-accessed items gravitated to the front and center, while other items moved to the periphery. In general, however, the entire desk surface was in flux, with perhaps only a computer monitor and a few peripheral items (*e.g.*, picture frames) being stable on the time scale of months. Although our sample size was small, it reinforced our assumption (and findings from prior studies) that desks tended to be cluttered and dynamic.

Next – and the primary purpose of the study – was to elicit interactive behaviors that digital applications should support. We structured this elicitation study [35] around a think-aloud exercise with paper prototypes. Specifically, participants were given paper prototypes of four common applications – a calendar, map, music player, and number keypad – and asked to imagine them as though they were fully interactive (Figure 3).

Participants then placed the (paper) applications on their work surface as they saw fit, thinking aloud about their reasoning. We also prompted them with a series of hypothetical situations, such as "what should happen to this application if you put your phone down here?", "... push the cup to the left?", "... pack your laptop in your bag?" Participants could move, animate, fold or otherwise manipulate the paper interfaces, or explain verbally what should occur in response. The facilitator recorded comments for later analysis and affinity diagramming.

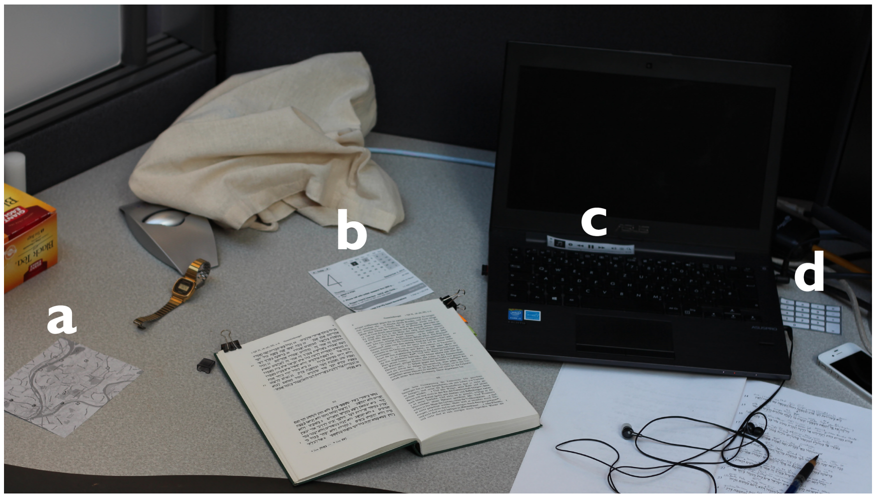

Figure 3. Sample arrangement of the paper prototypes in the elicitation study. a: The map is placed to the side, for reference but not frequent use. b: The calendar is placed nearby so it can be glanced at. c: The music player is snapped to the keyboard for quick access. d: The number keypad extends the keyboard.

5 DISTILLING INTERACTIVE BEHAVIORS

Following the study, we distilled our written notes and participant quotes into three functional categories:

5.1 Application Lifecycle

Summoning: Participants articulated a variety of potential strategies for summoning applications, including special gestures (*e.g.*, "double tapping the desk"), persistent buttons ("start button", "dashboard", "dock"), spoken commands, and using a conventional device ("dragging an interface off computer desktop or mobile phone", "keyboard shortcut").

Closing: Several methods for dismissing interfaces were proposed by participants: occluding an interface with an object (to make it "go away"), shrinking an interface substantially, moving an interface to a dedicated "trash can" area, or invoking a special gesture (*e.g.*, "scrunching up"), throwing an object ("flinging" it), or whimsically, miming a fireball-throwing gesture at it ("hadouken").

5.2 Layout Control

Repositioning: Participants uniformly expected to be able to reposition interfaces by holding their fingers or hand on the interface and dragging the interface around.

Reorienting: Objects on desks rarely conformed to rectilinear grids; rather, objects were typically oriented in a radial pattern centered on the user. We observed that users generally rotated interfaces to face themselves, and two users further wanted the ability to rotate interfaces to face visitors (*e.g.*, to show a map to someone).

Resizing: Most participants expected to be able to "pinch" to resize interfaces, though four participants noted that the pinch would be ambiguous (pinching content vs. resizing window). These participants suggesting resizing by using the corners, akin to desktop GUI applications. Finally, some participants noted that they would prefer unused applications to be shrunk and placed out of the way, to be retrieved and expanded on demand.

5.3 Cohabitation Behaviors

Snapping: Participants typically placed the number keypad or music player controls near the computer keyboard or mouse. Five participants also described wanting to "snap" or "link" these interfaces to the keyboard, letting it behave as though it was an extension of the keyboard.

Following: Four participants noted that they expected snapped interfaces to automatically "follow" the movements of their associated physical object, unless the interface was manually repositioned.

Detaching: Participants noted that interfaces should "unsnap" if they were covered up or manually "pulled apart" or "torn off" from the object they were snapped to.

Evading: When asked to describe what interfaces should do when occluded by an object, participants were divided. Six participants expected interfaces to "popup elsewhere" or "run away" when occluded, two suggested that interfaces could shrink down to fit the remaining space, and four participants expecting the interface to "nudge away" or "adjust" in response to partial occlusions. Conversely, four participants noted that they would simply expect the interfaces to misbehave or ignore input if occluded.

Collapsing: If the available desk space shrunk until interfaces could not find sufficient surface area to exist, participants expected them to shrink substantially or disappear entirely, as opposed to "climbing" on top of objects.

These ten verbs form the basis of the interaction techniques we believe are necessary to support responsive interfaces in mixed physical-virtual desk contexts. Notably, while some of our verbs are taken from desktop windowing operations (resizing, repositioning, closing), several are designed specifically for cohabitation with physical objects (snapping, following, evading). Next, we describe our proof-of-concept system that implements all ten behaviors in real time, with no fiducial tags (or equivalent), on conventional desks, using commodity hardware.

6 INTERACTIVE BEHAVIOR IMPLEMENTATIONS

We built a desktop projection and sensing system that offers *example embodiments* of our ten interactive behaviors. The specific technical implementations of these features are described in detail in the next section. Importantly, the interactive behaviors we describe are not specific to the particular manipulation scheme employed. For example, one alternative input method could be to treat digital items more like physical objects, and use physical metaphors for interface manipulation, such as pushing, throwing and stacking (as seen in *e.g.*, BumpTop [1] and ShapeTouch [9]).

6.1 Conventional Interactions

To support conventional multitouch interaction, we reserve one- and two-finger interaction for interacting with the application content itself. Users therefore use familiar touch gestures within an application content, such as one-finger swiping and panning, and two-finger pinching and rotating.

6.2 Summoning

A central feature of all interactive systems is the general ability to trigger actions, a subset of which is *summoning* applications. Conventional GUIs typically feature static application bars, docks, menus, launchers, or other schemes, which serve as a reliable and readily accessible access point. However, a physical desktop may not have sufficient space, let alone a universally available summoning point. Thus, we had to consider several mechanisms for instantiating applications.

A number of mechanisms were suggested during our study: special-purpose buttons or taskbars (*e.g.,* the Windows "Start" button); special-purpose hand gestures (*e.g.*, double-tapping the surface); or instantiation via transplantation from the computer itself. Ultimately, we decided to implement a special-purpose hand gesture for summoning, as it enabled launching an interface in-place without requiring a computer or dedicated desk space.

Executing the triggering gesture (tapping twice with four fingers on an unoccupied space on the desk) causes a radial menu to appear at the approximate centroid of the fingers (Figure 4a-c). The user then moves the fingers towards the desired application and lifts their fingers to confirm the selection (Figure 4d), which is then launched in place (Figure 4e). In a complete implementation, one of the menu options would permit voice or text input to search for less common applications not listed in the radial menu.

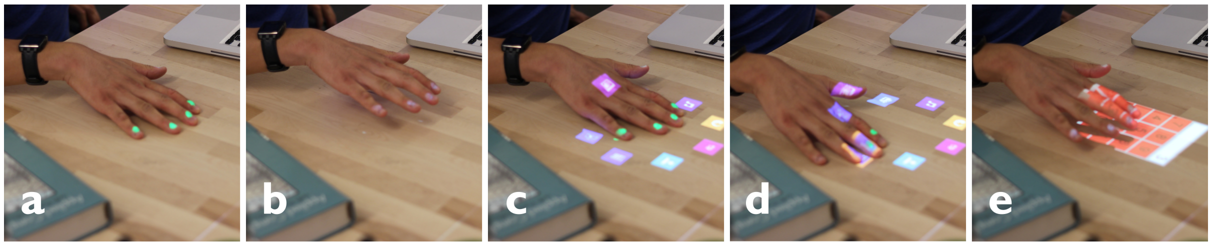

Figure 4. *Summoning* an application. a-c: The user taps twice with four fingers to bring up a launcher. d: The user moves to select the desired application. e: After lifting the fingers, the application is created.

6.3 Resizing

During our study, every participant suggested using a pinch gesture to *resize* the interface. However, some noted that this would be ambiguous with respect to the content. Further, a pinch gesture does not naturally permit the aspect ratio to be adjusted. We therefore borrow a familiar mechanism for manual resizing – a draggable 1.5 cm grey box in the lower-right corner of running applications (Figure 5a). Because applications can (and must) exist at a variety of sizes on desktops, applications ideally implement responsive layout schemes [54] (Figure 5a,b). If applications are resized below a reasonable limit (*e.g.,* less than 4 cm in either direction), applications are "iconified" by our window manager, displaying only an icon of the application and the resize box (Figure 5c). This permits applications to be stored on the desk without taking up substantial space.

6.4 Deleting

We considered several possible *deletion* schemes, some suggested by our study. These included special-purpose gestures (*e.g.*, "scrunching up" the interface or "flicking" the interface away), special drag targets (*e.g.*, a "trash can"), or explicit close buttons (like those found on computer desktop windows). Ultimately, we chose to simply extend resizing to provide deletion capabilities: applications can be deleted simply by resizing them below 1.5 cm in either axis (Figure 5d). This approach is straightforward and requires no additional buttons or gestures.

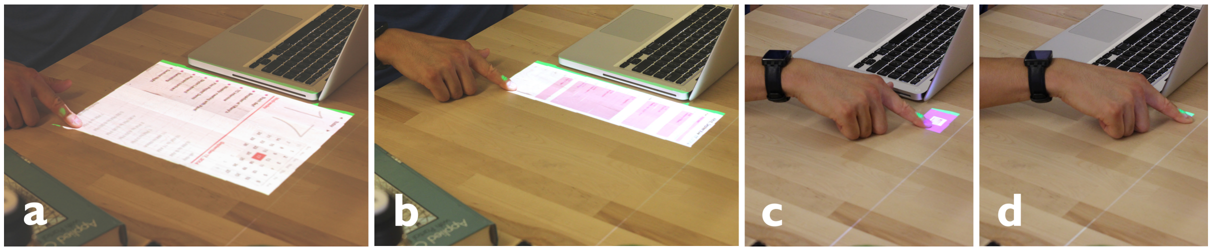

Figure 5. *Resizing* and *closing*. a: The user grabs the *resize* handle in the corner. b: Some interfaces can responsively alter their layout depending on their size; here, the calendar switches from week view to day view. c: Making the interface very small causes it to iconify. d: Shrinking the interface further will *close* it.

6.5 Repositioning and Reorienting

In order to *reposition* applications manually, we implemented a "drag" strategy. Users can press three fingers to the application and drag or rotate as desired (Figure 6a,b). For iconified applications, which are small, a single finger is used for dragging (as the application content is hidden, there is no interaction ambiguity in this case).

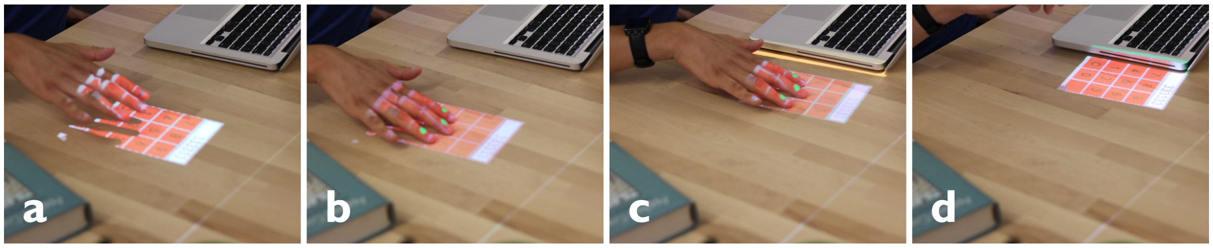

Figure 6. *Repositioning, reorienting* and *snapping*. a-b: Three fingers on the interface activates dragging. c: Dragging the interface near an edge highlights the edge in yellow. d: Upon releasing the interface, it snaps to the edge, which is highlighted in green.

6.6 Snapping

Users can *snap* applications to topological discontinuities (*i.e.*, "edges") of physical objects on the desk, such as the sides of laptops, books, keyboard, stacks of paper, edges of work surfaces and so on. To snap an interface to an edge, users drag the interface near to a target (Figure 6b). The nearest object edge to the interface (up to 50 mm away) is highlighted in yellow (Figure 6c). When an object edge is highlighted, releasing the drag will cause the interface to snap to the highlighted edge, repositioning and reorienting the interface as necessary. After successfully snapping, the interface edge is highlighted in green (Figure 6d).

6.7 Following

As noted during our study, users expect snapped interfaces to *follow* the objects they are snapped to. In our system, this “following” behavior is implemented by updating the object edges ten times per second (alongside the desk topography), and repositioning snapped interfaces to the new edge nearest to the previous edge (in both position and rotation angle). This causes interfaces to simply follow the objects naturally (Figure 7a-c). To avoid jitter due to noise in the edge measurements, interfaces only follow if the edge moves by more than 4 mm.

6.8 Detaching

Users may also want to disable snapping or following behaviors, in which case they can *detach* an interface from an edge. This can be achieved by either dragging the interface off of the object (Figure 7d), or by moving the object rapidly away from the interface (*i.e.*, “tearing” it off).

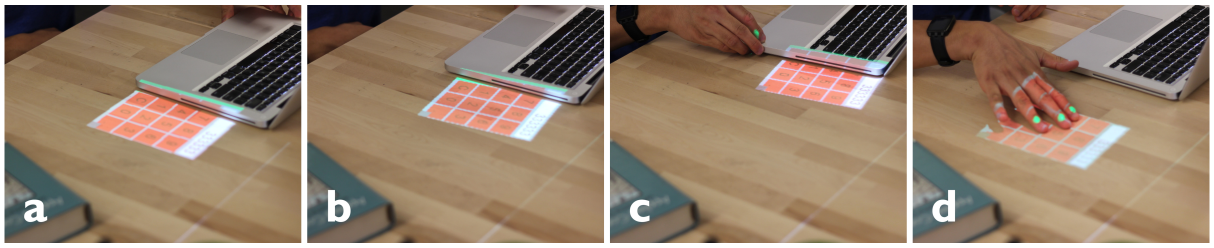

Figure 7. *Following* and *detaching*. a-c: Once a virtual interface is *snapped* to an object, it should *follow* if the object is moved. d: The snapped interface can be *detached* by dragging it off.

6.9 Evading

Interfaces will attempt to *evade* objects that suddenly occlude them (Figure 8b-c). This could happen if, *e.g.*, users rearrange their desk, or shuffle items around. When this occurs, the next topography update will detect a sudden increase in the interface’s “roughness”, and will include the interface in the subsequent optimization pass (discussed in the next section). This relocates the interface to a nearby open space on the desk (any relatively-flat area with low roughness), while avoiding other interfaces. Multiple interfaces may participate in optimization simultaneously.

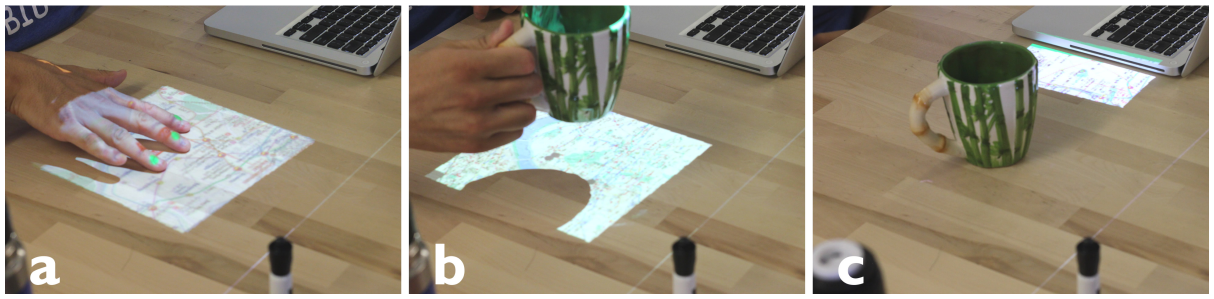

Figure 8. *Evading* and *collapsing*. a: An interface is positioned somewhere. b: The interface is occluded by an object. c: When the user moves his hands away, the interface *evades* the occlusion, and also *collapses* to fit available space.

6.10 Collapsing

Our optimization-based layout algorithm will also automatically shrink an interface if the available space is insufficient to host a displaced interface. This permits an interface to “squeeze” into a space where it can be displayed (Figure 8c). If the available space is very limited, the interface may simply iconify; the user can then choose to clear off space and re-expand the interface if desired.

7 TECHNICAL IMPLEMENTATION

Achieving our ten interactive behaviors required combining features described in many disparate research efforts into a single novel system, as no prior system has the necessary technical capabilities. More specifically, our system provides rectified projection onto irregular surfaces. To support our cohabitation behaviors, it needed to perform object tracking and edge finding in real-time. We also employ an optimization-based layout engine to automatically relocate evading interfaces. For finger touch tracking, we extend the approach described in OmniTouch [19], which allows fingers to be disambiguated from other objects.

7.1 Hardware

Our hardware prototype consists of a paired depth camera (Asus Xtion) and pocket projector (700-lumen Optoma ML750), a configuration similar to that found in [19, 53] (Figure 9, right). The hardware was enclosed in a ceiling-hung lampshade fixture (Figure 9, left) suspended 90 cm above a desk surface. At this distance, the projected image is 62x39 cm in size. The camera and projector are rigidly affixed together and calibrated to each other “at the factory” using a seven-point calibration routine [53]. The hardware is designed to be easily portable, requiring no calibration to the desk surface.

The depth camera produces 320x240 pixel depth images at 30 frames per second. Natural variation due to noise is 4-7 mm at a distance of 90 cm [24]. Our software was developed using the OpenFrameworks C++ library, and runs on a 2011 Macbook Pro. The complete implementation requires less than 15% of the CPU in full operation. It runs touch tracking at camera frame rate (30 fps), geometry updates at 10 fps, and interface rendering at projector frame rate (60 fps). Please also see the Video Figure.

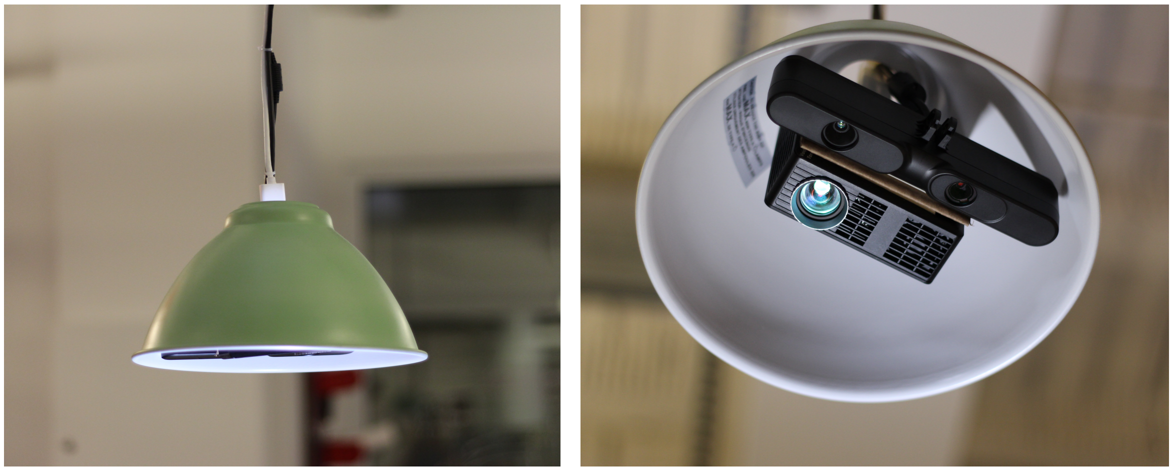

Figure 9. Our proof-of-concept projected augmented desk system fitted into a lampshade.

7.2 Disambiguating Object and Human Movement

In order to enable interaction on the ever-changing desk surface, our system needs to readily and accurately distinguish fingers and hands from other objects. For example, to implement the *evade* behavior, we need to have interfaces move away from objects that are placed on top, but not move away from hands or fingers that appear over the interface in attempts to interact with it. Additionally, interfaces must be able to respond to hand and finger input without being affected by other objects that may be moving in the scene.

We therefore detect human arms in addition to fingers during our touch tracking procedure, and use the arm and finger data to label fingers, hands and arms. These human "objects" are then excluded from most topographical considerations. Since the field-of-view of the depth sensor is approximately twice as large as the projection area in our system, we can guarantee that fingers appearing within the projection area will always be accompanied by visible arms in the larger depth image.

7.3 Touch Tracking

We did not want to augment our desks with additional sensors (since the system was intended to operate over any unmodified desk surface), and therefore opted to use a depth camera to track touches. Further, while background subtraction and in-air touch tracking enable reliable hand tracking (by identifying and removing the background), we determined that such an approach would be infeasible in a desk environment where objects are liable to move.

Instead, we decided to use the "sausage-finding" finger tracking algorithm proposed in OmniTouch [19], which we extend in several significant ways:

- We extract both horizontal and vertical slices, enabling detection of fingers oriented in any direction.
- Our algorithm works on messy surfaces with a variety of objects and upon multiple surfaces.
- We reject finger-like objects by also detecting arms and requiring co-occurrence.
- We operate at a significantly increased range with similar hardware, necessitating stronger error tolerance.
- We can detect fingers at a wide range of vertical angles, from completely flat (Figure 10) to 70° vertical.
- The algorithm is optimized, requiring less than 10 ms to detect any number of touches on the surface.

Additionally, our algorithm requires no calibration, no background subtraction, and makes no decisions based on (potentially lighting-dependent) color information.

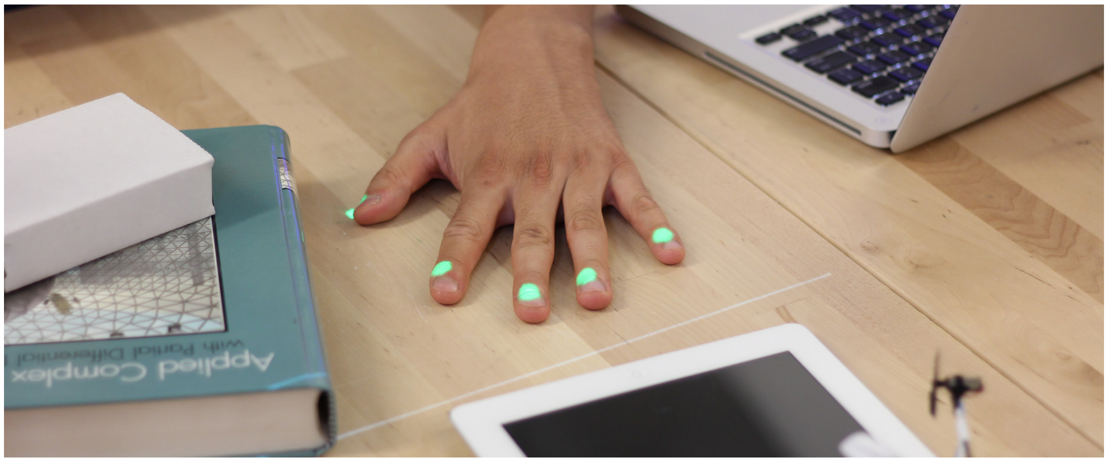

Figure 10. Our touch tracking algorithm can find fingertips even when the hand is flat against the surface.

The algorithm detects fingers and arms, which are characterized by cylindrical geometry within a certain narrow range of widths. To start, it computes the depth derivatives in both the horizontal and vertical directions using a sliding 8x8 window (Figure 11a), then locates cylindrical slices in both the vertical and horizontal axes (Figure 11b, red and blue lines) which are characterized by a pattern of a steep negative depth gradient (rising towards the camera), followed by relatively flat (near-zero) depth gradients, and then closed by a steep positive depth gradient.

Adjacent slices are connected into cylinders (Figure 11b, white), skipping short gaps to improve robustness. Vertical and horizontal slices can coexist in the same cylinder, to provide detection of diagonal fingers. Only cylinders in the typical length range for an arm or finger are kept. Finger cylinders are then paired with arm cylinders based on proximity. The detection of arm cylinders both serves to reject non-fingers and to orient the finger cylinders correctly (with the fingertip pointing away from the arm).

For touch detection, the algorithm starts a depth-based flood fill (Figure 11b, green) at a point 90% of the way from the base of the finger to the fingertip (roughly the position of the distal knuckle joint). The flood fill only fills into points having depth near to the seed point, and only fills up to a set maximum number of pixels (*i.e.,* by flooding out onto the surface). In the latter case, the finger is determined to have touched a surface. Finally, the touch tip positions (Figure 11c) are smoothed with an exponentially-weighted moving average (EWMA) filter, reducing jitter substantially.

This touch algorithm also provides several useful pieces of information about each finger: the height of the finger above the surface, the 3D position and 3D orientation of the finger (since the 3D positions of the finger cylinder ends are both known), whether the finger is bent or straight, and the approximate length of the finger. These features could be used to expand the interactive capabilities of virtual desk applications.

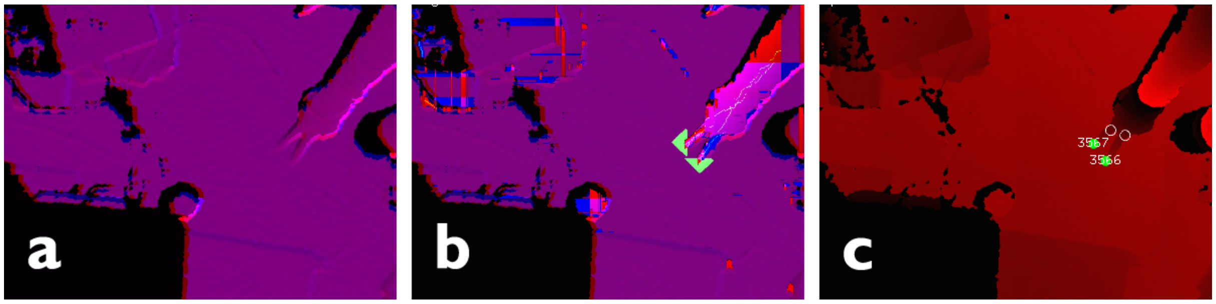

Figure 11. Touch tracking steps. a: Depth gradients (red=dx, blue=dy). b: Unfiltered cylinder slices (red, blue), slices connected into cylinders (white lines) and touch-detection flood-fill (green). c: Detected touches overlaid on depth image. White circles represent the base of the finger.

7.4 Handling Irregular Surfaces

The desk topography is sensed by converting each depth pixel in the projection area into a height value, which is used to construct a mesh. The topography mesh is then re-rendered from an orthographic top-down view, and the depth component of this rendering is extracted into a height map. This height map is used to identify flat areas suitable for placing interfaces.

The mesh is also used for projection mapping. The desktop imagery is texture-mapped onto the mesh such that the resulting projected output appears correct from overhead, even if desktop-bound objects are tall or irregularly-shaped. This also ensures the projections maintain a consistent scale (so that *e.g.,* 1 mm on the virtual desktop is always 1 mm on the surface regardless of surface height).

The system attempts to update the desk topography ten times per second, but will skip the update if fingers are detected over top of, or on an interface to avoid occlusion and interference from the user's hands. After completing a topography update, the system will search for edges in the new topography, and then initiate an interface optimization pass to support evading and following behaviors.

7.5 Edge Finding

In order to support our snapping and following behaviors, our system required the ability to detect edges of physical objects. The process starts by filtering the desk height map with a Canny edge filter [8] to extract depth discontinuities. The parameters were tuned to extract discontinuities of at least 7 mm without excessive noise (the smallest possible difference given the noise performance of our depth sensor). The binary edge image is then fed to OpenCV's contour-finding routine, which connects adjacent edge pixels into contiguous contours.

Contours are then filtered to remove artifacts (*e.g.*, doubled-up paths) and short noise-induced contours. We then extract segments from the contours by walking over the pixels in each contour and outputting a segment whenever the contour abruptly changes direction (by more than 45° in an 8-pixel window). Short segments (less than 20 mm long) are deleted, and the remaining segments are converted to straight lines by using linear interpolation.

When a user drags an interface near a discovered edge, the edge highlights (Figure 6c); the closest edge to the interface center is selected in case of ambiguity. If the user releases the drag while an edge is highlighted, the interface will snap to that edge (Figure 6d). Snapping locks one edge of the interface to the target object edge; the chosen interface edge is the closest edge that makes a 45°-or-less angle with the object edge. In subsequent update frames, the system selects the nearest object edge to the previous snapped edge, and moves the interface accordingly. If the nearest object edge is more than 40 mm away, the system assumes that a detach event has occurred and the snap is released.

7.6 Optimization-Based Layout

In order to support evading and collapsing behaviors, the system must be able to automatically identify open spaces on the surface, and to decide whether the available space is sufficient to host an application. However, the irregular and complex topology of desks yields innumerable corner cases that make building a rule-based or heuristically driven layout approach unwieldy and brittle. Thus, we use an optimization-based layout engine to support evading and collapsing interfaces.

This approach defines a certain "cost" to every interface configuration, depending on the desk topography and interface positions. It then aims to minimize the total cost of the configuration, subject to a number of "penalties" which guide the process away from undesirable behaviours (*e.g.*, to avoid large jumps in interface position). Furthermore, the optimization-based layout engine can be used, for example, to automatically position new interface elements (if they are summoned *e.g.*, using a voice command alone) or to support recalling a set of interfaces onto a changed desk layout.

The layout engine uses the topographical height map to determine the surface *roughness* as follows. The roughness at each desk point is calculated as the standard deviation of the height map in a 41 × 41 mm window centered on the desk point. This is efficiently computed by using sliding windows over summed-area tables [12]. The "roughness" of a particular interface area is the average roughness across each point in the region; this is efficiently calculated by using a polygon scanline approach coupled with a summed-area table of the roughness map.

Every time the topography is updated (~10 FPS), the roughness map is also updated. Interface positions are adjusted to follow any snapped edge boundaries (to support following behaviors), and the optimization algorithm is executed. The optimizer generates and evaluates a number of candidate interface configurations, starting with the previous configuration. Interfaces which were recently positioned by hand, or whose roughness has not significantly increased since the most recent optimization run are excluded from optimization.

The cost of a particular configuration is computed from the total sum of the interface area roughness values and an assortment of penalties (cost increases) to discourage particular behaviors. A penalty is imposed for moving an interface at all, which biases the algorithm towards keeping applications still. Another penalty is applied based on the distance moved and any applicable size change, which biases the algorithm towards small movements (preserving spatial locality) and minimal adjustments to the window size. Finally, a large penalty is imposed for any overlap between interfaces, to prevent the algorithm from stacking interfaces on top of one another in ideal areas.

The optimization itself is performed using a simulated annealing algorithm, which maintains a "temperature" parameter that decreases with each iteration of the algorithm. In each iteration, each interface in the current best configuration is "mutated" to generate a list of new candidate placements; these mutations can consist of moving, rotating or resizing objects. The expected magnitude of each mutation is controlled by the "temperature" parameter, with higher temperatures resulting in more extreme mutations (*e.g.*, longer distance movements). Candidates are combined into new configurations and tested for acceptance according to the current timestep's "temperature" parameter (which controls the simulated annealing algorithm's willingness to accept new "best" configurations). The configurations are generated exhaustively starting with the candidates having the lowest roughness values. If this process yields too many combinations, the algorithm switches to generating configurations randomly to avoid combinatorial explosion.

Our optimization algorithm terminates after reaching a predefined number of iterations, or if it runs for more than 60 ms. This bounds the optimization algorithm and prevents it from running endlessly. The best configuration found so far is then returned (which may be the same as the initial configuration, as that is heavily weighted, if no alternative was clearly superior).

8 CONCLUSIONS

We have presented a set of ten interaction behaviors for augmented desk interfaces that facilitate digital interfaces co-existing with and responding to physical objects. These behaviors were derived from an elicitation study using paper prototypes. We then described our proof-of-concept projector-camera system, designed with a unique set of technical features necessary to implement our behaviors. In general, the interaction behaviors presented in this paper serve as a basis for a more complete, modern digital desk system, in which the work surface and its contents are not replaced with digital equivalents, but augmented to add new capabilities. While similar ideas have been explored in prior work, we draw them together in a holistic and grounded manner, and additionally demonstrate their technical feasibility.

ACKNOWLEDGEMENTS

This research was supported by funding from Intel and the David and Lucile Packard Foundation.

REFERENCES

[1] Anand Agarawala and Ravin Balakrishnan. 2006. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06). ACM, New York, NY, USA, 1283-1292. http://doi.acm.org/10.1145/1124772.1124965

[2] Maneesh Agrawala and Chris Stolte. 2001. Rendering effective route maps: improving usability through generalization. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '01). ACM, New York, NY, USA, 241-249. http://doi.acm.org/10.1145/383259.383286

[3] Toshifumi Arai, Kimiyoshi Machii, Soshiro Kuzunuki, and Hiroshi Shojima. 1995. InteractiveDESK: a computer-augmented desk which responds to operations on real objects. In *Conference Companion on Human Factors in Computing Systems* (CHI '95). ACM, New York, NY, USA, 141-142. http://doi.acm.org/10.1145/223355.223470

[4] Hrvoje Benko, Ricardo Jota, and Andrew Wilson. 2012. MirageTable: freehand interaction on a projected augmented reality tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 199-208. http://doi.acm.org/10.1145/2207676.2207704

[5] Xiaojun Bi, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2011. Magic desk: bringing multi-touch surfaces into desktop work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 2511-2520. http://doi.acm.org/10.1145/1978942.1979309

[6] François Bodart, Anne-Marie Hennebert, Jean-Marie Leheureux, and Jean Vanderdonckt. 1994. Towards a dynamic strategy for computer-aided visual placement. In *Proceedings of the workshop on Advanced visual interfaces* (AVI '94). ACM, New York, NY, USA, 78-87. http://doi.acm.org/10.1145/192309.192328

[7] Olha Bondarenko and Ruud Janssen. 2005. Documents at Hand: Learning from Paper to Improve Digital Technologies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 121-130. http://doi.acm.org/10.1145/1054972.1054990

[8] J Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679-698. http://dx.doi.org/10.1109/TPAMI.1986.4767851

[9] Xiang Cao, Andrew D.Wilson, Ravin Balakrishnan, Ken Hinckley, and Scott E. Hudson. ShapeTouch: Leveraging Contact Shape on Interactive Surfaces. In *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, 2008 (TABLETOP '08)*, 129-136. http://dx.doi.org/10.1109/TABLETOP.2008.4660195

[10] Jae Sik Chang, Eun Yi Kim, KeeChul Jung, and Hang Joon Kim. Real time hand tracking based on active contour model. In *Proc. ICCSA '05*, 999-1006. http://dx.doi.org/10.1007/11424925_104

[11] Jason Cong, Lei He, Cheng-Kok Koh, and Patrick H. Madden. 1996. Performance optimization of VLSI interconnect layout. *Integr. VLSI J.* 21, 1-2 (November 1996), 1-94. http://dx.doi.org/10.1016/S0167-9260(96)00008-9

[12] Franklin C. Crow. 1984. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '84). ACM, New York, NY, USA, 207-212. http://doi.acm.org/10.1145/800031.808600

[13] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. 1998. *Graph Drawing: Algorithms for the Visualization of Graphs* (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

[14] Jerry Alan Fails and Dan Olsen Jr.. 2002. Light widgets: interacting in every-day spaces. In *Proceedings of the 7th international conference on Intelligent user interfaces* (IUI '02). ACM, New York, NY, USA, http://doi.acm.org/10.1145/502716.502729

[15] James Fogarty and Scott E. Hudson. 2003. GADGET: a toolkit for optimization-based approaches to interface and display generation. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (UIST '03). ACM, New York, NY, USA, 125-134. http://doi.acm.org/10.1145/964696.964710

[16] Krzysztof Z. Gajos, Daniel S. Weld, and Jacob O. Wobbrock. 2008. Decision-theoretic user interface generation. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3* (AAAI'08), Anthony Cohn (Ed.), Vol. 3. AAAI Press 1532-1536.

[17] Christoph Gebhardt, Roman Rädle, and Harald Reiterer. 2014. Integrative workplace: studying the effect of digital desks on users' working practices. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '14). ACM, New York, NY, USA, 2155-2160. http://doi.acm.org/10.1145/2559206.2581186

[18] John Hardy. 2012. Experiences: a year in the life of an interactive desk. In *Proceedings of the Designing Interactive Systems Conference* (DIS '12). ACM, New York, NY, USA, 679-688. http://doi.acm.org/10.1145/2317956.2318058

[19] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 441-450. http://doi.acm.org/10.1145/2047196.2047255

[20] Hiroshi Ishii. 1990. TeamWorkStation: towards a seamless shared workspace. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work* (CSCW '90). ACM, New York, NY, USA, 13-26. http://doi.acm.org/10.1145/99332.99337

[21] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 559-568. http://doi.acm.org/10.1145/2047196.2047270

[22] Sasa Junuzovic, Kori Inkpen, Tom Blank, and Anoop Gupta. 2012. IllumiShare: sharing any surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 1919-1928. http://doi.acm.org/10.1145/2207676.2208333

[23] Shaun K. Kane, Daniel Avrahami, Jacob O. Wobbrock, Beverly Harrison, Adam D. Rea, Matthai Philipose, and Anthony LaMarca. 2009. Bonfire: a nomadic system for hybrid laptop-tabletop interaction. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09). ACM, New York, NY, USA, 129-138. http://doi.acm.org/10.1145/1622176.1622202

[24] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2), 2012, 437-454. http://dx.doi.org/10.3390/s120201437

[25] Jeongyun Kim, Jonghoon Seo, and Tack-Don Han. 2014. AR Lamp: interactions on projection-based augmented reality for interactive learning. In *Proceedings of the 19th international conference on Intelligent User Interfaces* (IUI '14). ACM, New York, NY, USA, 353-358. http://doi.acm.org/10.1145/2557500.2557505

[26] K. C. Knowlton. Computer Displays Optically Superimposed on Input Devices. *Bell Systems Technical Journal*, 53(3), 1977, 367-383. http://dx.doi.org/10.1002/j.1538-7305.1977.tb00514.x

[27] Hideki Koike, Yoichi Sato, and Yoshinori Kobayashi. 2001. Integrating paper and digital information on EnhancedDesk: a method for realtime finger tracking on an augmented desk system. *ACM Trans. Comput.-Hum. Interact.* 8, 4 (December 2001), 307-322. http://doi.acm.org/10.1145/504704.504706

[28] Eric Larson, Gabe Cohn, Sidhant Gupta, Xiaofeng Ren, Beverly Harrison, Dieter Fox, and Shwetak Patel. 2011. HeatWave: thermal imaging for surface user interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 2565-2574. http://doi.acm.org/10.1145/1978942.1979317

[29] Joonhwan Lee, Jodi Forlizzi, and Scott E. Hudson. 2005. Studying the effectiveness of MOVE: a contextually optimized in-vehicle navigation system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 571-580. http://doi.acm.org/10.1145/1054972.1055051

[30] Natan Linder and Pattie Maes. 2010. LuminAR: portable robotic augmented reality interface design and prototype. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 395-396. http://doi.acm.org/10.1145/1866218.1866237

[31] Thomas W. Malone. 1983. How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. Inf. Syst.* 1, 1 (January 1983), 99-112. http://doi.acm.org/10.1145/357423.357430

[32] Daniel C. McFarlane and Steven M. Wilder. 2009. Interactive dirt: increasing mobile work performance with a wearable projector-camera system. In *Proceedings of the 11th international conference on Ubiquitous computing* (UbiComp '09). ACM, New York, NY, USA, 205-214. http://doi.acm.org/10.1145/1620545.1620577

[33] Jurriaan D. Mulder, Jack Jansen, and Arjen van Rhijn. 2003. An affordable optical head tracking system for desktop VR/AR systems. In *Proceedings of the workshop on Virtual environments 2003* (EGVE '03). ACM, New York, NY, USA, 215-223. http://doi.acm.org/10.1145/769953.769978

[34] William Newman and Pierre Wellner. 1992. A desk supporting computer-based interaction with paper documents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '92). ACM, New York, NY, USA, 587-592. http://doi.acm.org/10.1145/142750.143007

[35] Michael Nielsen, Moritz Störring, Thomas B. Moeslund, and Erik Granum. 2004. A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In *Gesture Based Communication in Human-Computer Interaction* 2915, 409-420. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-540-24598-8_38

[36] Claudio S. Pinhanez. 2001. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. In *Proceedings of the 3rd international conference on Ubiquitous Computing* (UbiComp '01). Springer-Verlag, London, UK, UK, 315-331. http://dx.doi.org/10.1007/3-540-45427-6_27

[37] Ramesh Raskar, Jeroen van Baar, Paul Beardsley, Thomas Willwacher, Srinivas Rao, and Clifton Forlines. 2003. iLamps: geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2003 Papers* (SIGGRAPH '03). ACM, New York, NY, USA, 809-818. http://doi.acm.org/10.1145/1201775.882349

[38] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. 1998. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '98). ACM, New York, NY, USA, 179-188. http://doi.acm.org/10.1145/280814.280861

[39] Jun Rekimoto. 2002. SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02). ACM, New York, NY, USA, 113-120. DOI=10.1145/503376.503397 http://doi.acm.org/10.1145/503376.503397

[40] Charles Robertson and John Robinson. 1999. Live paper: video augmentation to simulate interactive paper. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)* (MULTIMEDIA '99). ACM, New York, NY, USA, 167-170. http://doi.acm.org/10.1145/319878.319923

[41] A. Sears. 1993. Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout. *IEEE Trans. Softw. Eng.* 19, 7 (July 1993), 707-719. http://dx.doi.org/10.1109/32.238571

[42] Abigail J. Sellen and Richard H.R. Harper. 2003. *The Myth of the Paperless Office.* MIT Press, Cambridge, MA, USA.

[43] Jürgen Steimle, Mohammadreza Khalilbeigi, Max Mühlhäuser, and James D. Hollan. 2010. Physical and digital media usage patterns on interactive tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces* (ITS '10). ACM, New York, NY, USA, 167-176. http://doi.acm.org/10.1145/1936652.1936685

[44] Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. 2010. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces* (ITS '10). ACM, New York, NY, USA, 91-94. http://doi.acm.org/10.1145/1936652.1936668

[45] John Underkoffler and Hiroshi Ishii. 1998. Illuminating light: an optical design tool with a luminous-tangible interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '98). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 542-549. http://dx.doi.org/10.1145/274644.274717

[46] John Underkoffler, Brygg Ullmer, and Hiroshi Ishii. 1999. Emancipated pixels: real-world graphics in the luminous room. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '99). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 385-392. http://dx.doi.org/10.1145/311535.311593

[47] Dhaval Vyas and Anton Nijholt. Artful surfaces: an ethnographic study exploring the use of space in design studios. *Digital Creativity*, 23(1), 2012, 1-20. http://dx.doi.org/10.1080/14626268.2012.658522

[48] Pierre Wellner. 1993. Interacting with paper on the DigitalDesk. *Commun. ACM* 36, 7 (July 1993), 87-96. http://doi.acm.org/10.1145/159544.159630

[49] Pierre Wellner. 1991. The DigitalDesk calculator: tangible manipulation on a desk top display. In *Proceedings of the 4th annual ACM symposium on User interface software and technology* (UIST '91). ACM, New York, NY, USA, 27-33. http://doi.acm.org/10.1145/120782.120785

[50] Andrew D. Wilson and Hrvoje Benko. 2010. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 273-282. http://doi.acm.org/10.1145/1866029.1866073

[51] Andrew D. Wilson. Depth sensing video cameras for 3D tangible tabletop interaction. In *Proc. Tabletop '07*, 201-204. http://dx.doi.org/10.1109/TABLETOP.2007.35

[52] Raphael Wimmer, Fabian Hennecke, Florian Schulz, Sebastian Boring, Andreas Butz, and Heinrich Hußmann. 2010. Curve: revisiting the digital desk. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries* (NordiCHI '10). ACM, New York, NY, USA, 561-570. http://doi.acm.org/10.1145/1868914.1868977

[53] Robert Xiao, Chris Harrison, and Scott E. Hudson. 2013. WorldKit: rapid and easy creation of ad-hoc interactive applications on everyday surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 879-888. http://doi.acm.org/10.1145/2470654.2466113

[54] Clemens Zeidler, Christof Lutteroth, Wolfgang Sturzlinger, and Gerald Weber. 2013. The auckland layout editor: an improved GUI layout specification process. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (UIST '13). ACM, New York, NY, USA, 343-352. http://doi.acm.org/10.1145/2501988.2502007