OmniTouch: Wearable Multitouch Interaction Everywhere

Chris Harrison[1,2] *Hrvoje Benko*[1] *Andrew D. Wilson*[1]

[1]Microsoft Research
One Microsoft Way
Redmond, WA 98052
{benko,awilson}@microsoft.com

[2]Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh PA 15213
chris.harrison@cs.cmu.edu

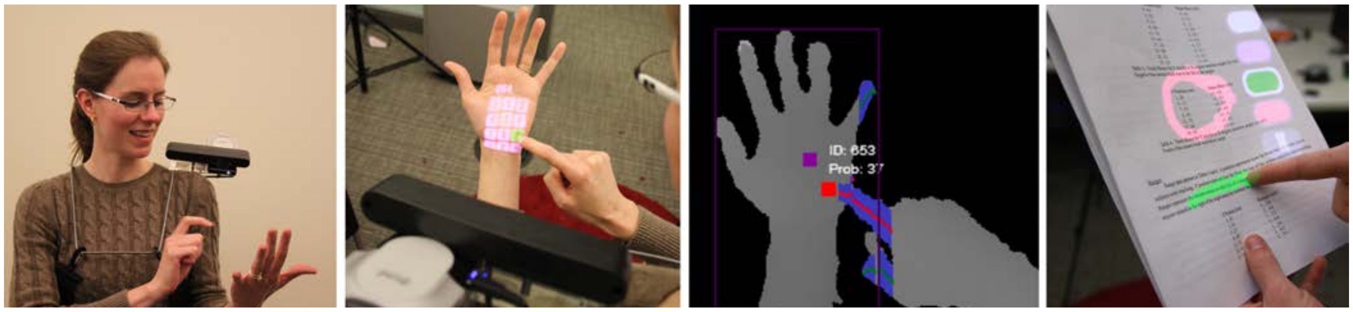

Figure 1. *OmniTouch* is a wearable depth-sensing and projection system that allows everyday surfaces - including a wearer's own body - to be appropriated for graphical multitouch interaction.

ABSTRACT
OmniTouch is a wearable depth-sensing and projection system that enables interactive multitouch applications on everyday surfaces. Beyond the shoulder-worn system, there is no instrumentation of the user or environment. Foremost, the system allows the wearer to use their hands, arms and legs as graphical, interactive surfaces. Users can also transiently appropriate surfaces from the environment to expand the interactive area (e.g., books, walls, tables). On such surfaces - without any calibration - *OmniTouch* provides capabilities similar to that of a mouse or touchscreen: X and Y location in 2D interfaces and whether fingers are "clicked" or hovering, enabling a wide variety of interactions. Reliable operation on the hands, for example, requires buttons to be 2.3cm in diameter. Thus, it is now conceivable that anything one can do on today's mobile devices, they could do in the palm of their hand.

ACM Classification: H.5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces; Input devices and strategies.

General terms: Human Factors

Keywords: On-demand interfaces, finger tracking, on-body computing, appropriated surfaces, object classification.

INTRODUCTION
Today's mobile computers provide omnipresent access to information, creation and communication facilities. It is undeniable that they have forever changed the way we work, play and interact. However, mobile interaction is far from solved. Diminutive screens and buttons mar the user experience, and otherwise prevent us from realizing their full potential.

In this paper we explore and prototype a powerful alternative approach to mobile interaction that uses a body-worn projection/sensing system to capitalize on the tremendous surface area the real world provides. For example, the surface area of one hand alone exceeds that of typical smart phone. Tables are often an order of magnitude larger than a tablet computer. If we could appropriate these ad hoc surfaces in an on-demand way, we could retain all of the benefits of mobility while simultaneously expanding the interactive capability. However, turning everyday surfaces into interactive platforms requires sophisticated hardware and sensing. Further, to be truly mobile, systems must either fit in the pocket or be wearable.

In this paper, we present *OmniTouch*, a novel wearable system that enables graphical, interactive, multitouch input on arbitrary, everyday surfaces. Our shoulder-worn implementation allows users to manipulate interfaces projected onto the environment (e.g., walls, tables), held objects (e.g., notepads, books), and their own bodies (e.g., hands, lap). A key contribution is our depth-driven template matching and clustering approach to multitouch finger tracking. This enables on-the-go interactive capabilities, with no calibration, training or instrumentation of the environment or the user, creating an always-available interface [8,24,28].

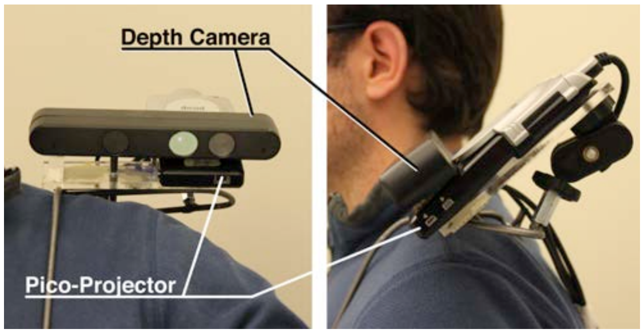

Figure 2. Our prototype shoulder-worn *OmniTouch* System.

RELATED WORK

OmniTouch draws from a variety of fields, including touch interaction techniques, surface computing, free-space gesturing, computer vision, wearables, and ubiquitous computing. Here we focus on highly related, influential work that intersects these fields.

Augmenting the environment with interactive projection interfaces has been a research vision for decades. A variety of approaches are possible, including installing projectors in the environment [20,32], using handheld projectors [3,22], and having users wear mounted projectors [11,17]. Providing touch-based interactivity on arbitrary projected objects is challenging, and generally requires a fixed non-tracking projection [8,17], careful calibration of objects in the environment [20,32,34], or objects to be instrumented with sensing elements [10,14,22]. A small number of such systems have attempted to be truly ad hoc, without requiring permanent instrumentation of the surfaces they operate on [12,17,29,33,34].

The emergence of pico-projectors has enabled a new class of worn, on-body projected interactive systems. SixthSense [18] and Interactive Dirt [17] both featured a worn camera/projector combination. Finger tracking was achieved by wearing fingertip markers (e.g., color or IR reflective). In contrast to our system, true touch interactions were not possible since the two systems could not differentiate between clicked and hovering fingers. This was partially due to the systems' inability to track surfaces in the environment, which also made it impossible to have the projected interface change and follow the surface as it moved. Also of note is Skinput [8], which uses bio-acoustics to detect finger tap events on the skin. Skinput's main limitations were its lack of support for any surface other than the user's body, its inability to detect touch drag movements, and the lack of support for multitouch. Although the system included a pico-projector, no surface tracking was performed, requiring users to position their arms at a predefined position.

Detecting fingers, hands, touches, and gestures has been an active research topic in computer vision [1,6,20,26]. Making computer vision-based tracking work in a wearable, ad-hoc context is challenging due to lack of control of the environment and the general inability to instrument users (e.g., markers [17,18], gloves [27]).

Closely related to our technical approach are systems using the depth-camera tracking without augmenting the user or environment. LightSpace [32] uses an array of depth cameras to track users and arm-level manipulations in an augmented room. At a smaller scale, a single camera can provide conventional touch events by using a per-pixel depth threshold determined from a histogram of the static scene [34]. Both approaches work on a variety of surfaces, but require careful calibration before they can operate.

HARDWARE

Our proof-of-concept *OmniTouch* system, seen in Figures 1 and 2, consists of three principal components. First is a custom, short-range PrimeSense [21] depth camera, which provides a 320x240 depth map at 30 FPS. Objects as close as 20cm can be imaged by this sensor, with relative error in the depth (Z) axis of approximately 5mm. Depth accuracy decreases and noise increases at larger distances. However, for our application, which chiefly considers interaction within a 1m "bubble" in front of the user, noise and accuracy loss was minimal. We initially constructed a prototype using a Microsoft Kinect with good results. However, a minimum sensing distance of ~50cm necessitated awkward placement high above the head to capture the hands.

The second key component is a Microvision ShowWX+ laser pico-projector [16]. This projector has the important property of wide angle, focus-free projection of graphical elements regardless of depth (i.e., distance from projector). Finally, the depth camera and projector are tethered to a desktop computer for prototyping purposes.

Both the depth camera and projector are rigidly mounted to a form-fitting metal frame, which is worn on the shoulders, and secured with a chest strap. We chose the shoulder as it provides a good vantage point (both for sensing and projection) of the arms and held objects, as well as proximate fixed surfaces, such as walls and tables. However, our approach is amenable to other locations (e.g., the upper arm [8] and chest [18]). Additionally, the shoulders tend to be very stable, allowing for projected interfaces with minimal sway and jitter (see Video Figure).

The first person body-stabilized perspective is desirable for sensing and processing, as many simplifying assumptions can be made about the location and orientation of fingers and hands. For example, it is physically impossible for the user's arms to enter the image from the top. The system's field of view also automatically translates with the wearer. Further, camera and projection occlusion issues are minimized, as their fields of view roughly coincide with the wearer's line of sight.

MULTITOUCH FINGER TRACKING

We present a unique approach to ad hoc finger tracking, which enables multitouch input on arbitrary surfaces, both flat and irregular, with no calibration or training. We can resolve the *X, Y and Z position* of fingers, and whether they are *touching* or *hovering* over a surface. Thus, *OmniTouch* produces input events similar to that of mice or touchscreens, enabling a wide variety of applications.

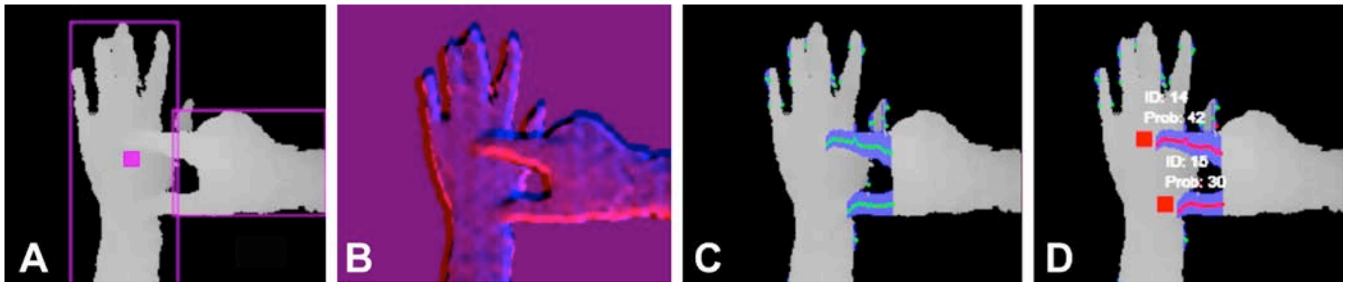

Figure 3. Left to right: depth map, derivative of depth map, finger slices overlaid in blue, path finding and tip estimation.

Finger Segmentation

Identifying finger input is a multistep process. First, we take a depth map of a scene (Figure 3A) and compute the depth derivative in the X- and Y-axes using a sliding, 5x5 pixel window (Figure 3B; X and Y derivative visualized using blue and red channels respectively). We then iterate over this derivative image, looking for vertical slices of cylinder-like objects. This is similar to template matching, but with some dynamic parameters. Put simply, for a slice of pixels to be a candidate, it must show a steep positive derivate, followed by a region of relative smoothness, and finally closed by a steep negative derivative (Figure 4). This ordering is critical; otherwise, concave features (e.g., gaps between fingers) would also be recognized. Also significant is that the depth camera we use represents sensing errors, out-of-range surfaces, and occlusion boundaries as holes in the depth image. As such, they appear as concavities in the derivative, which our process ignores.

To primarily isolate fingers, candidate slices must be between 5 and 25mm in height, a range we found to cover typical finger diameters, including the critical fingertip. Pixel distances can be converted into real world distances (mm) because the depth value is also known. The result of this finger-slice identification process is shown in Figure 3C.

Using the derivative of the depth map has several benefits that make it a key component of our sensing approach. Foremost, this approach suppresses absolute depth information, allowing the scene to be treated as a conventional 2D image, which is easier to process with standard computer vision techniques. Additionally, regardless of the surface the finger is operating on, the derivative profile is mostly invariant, greatly simplifying recognition.

Once all candidate finger slices are identified, we then greedily group proximate slices into contiguous paths. Paths that are shorter or longer than probable fingers are discarded. Even in noisy scenes, this process yields few false positives. The output, seen in Figure 3D, resembles a skeletal model of the fingers. Like other computer vision techniques, fingers that are occluded are not detected. Additionally, and usefully, fingers that are "tucked in" are not tracked. However, our technique is sensitive to approach angle (can neither be too steep nor too shallow) and generally requires fingers be outstretched for reliable recognition.

Many approaches are possible for disambiguating which end of the path is the fingertip. In our proof-of-concept system, we assume a right-handed user, and thus, in almost all cases, the leftmost point in a path is the fingertip. This worked well in practice for our left-shoulder mounted configuration. To eliminate sensing noise and pixel-boundary flicker, fingertip positions are smoothed by a Kalman filter.

Finger Click Detection

The finger segmentation process, described above, yields the spatial location (X, Y and Z) of fingers. A secondary process is used to determine whether these fingers - specifically the tips - are in contact with a surface (i.e., a "click").

We start by computing the midpoint of the finger path, which roughly equates to the location of the minor knuckle. From this point, we flood fill towards the fingertip (i.e., all directions but rightward). This operation is performed on the depth map using a tolerance of 13mm in depth to determine if neighboring pixels can be filled. When the finger is hovering above a surface or in free space, the flood fill expands to encompass the entire finger (Figure 5, left). How-

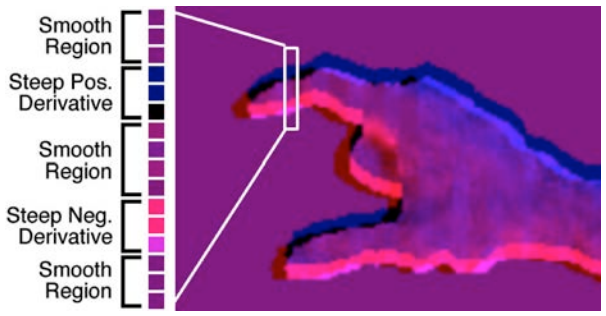

Figure 4. Close up example of a candidate finger slice.

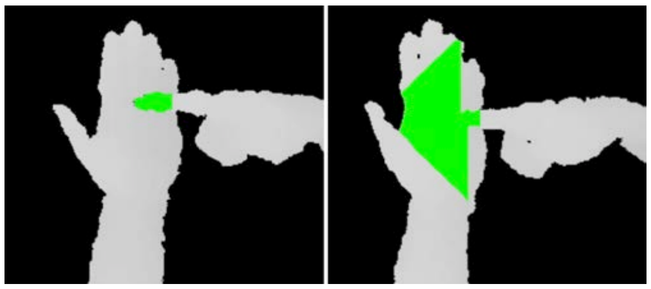
Figure 5. Flood filling result when finger is hovering (left) and "clicked" (right).

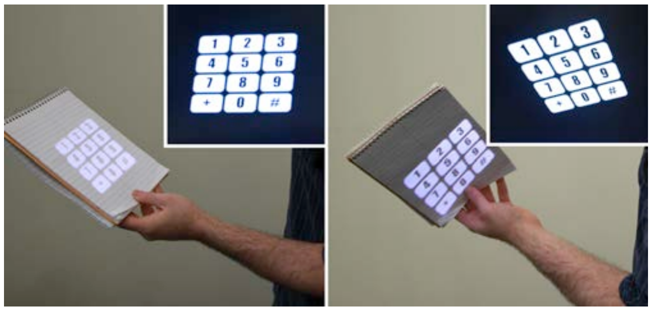
Figure 6. In order for interfaces to appear visually aligned and correct when projected onto moving surfaces, the projected image must be dynamically pre-distorted (see inset images).

ever, when the finger contacts a surface, the fill operation floods out into the connecting object (Figure 5, right). If a pixel count threshold is passed (e.g., 2000 pixels), the flood fill discontinues and the finger is determined to be clicked. Note, if the surface is very small or lies outside the camera's view, the threshold may not be passed, and the click missed.

This process detects finger clicks robustly, and also maintains a clicked state when dragging a finger across a surface, including irregular ones. In practice, a finger will be seen as "clicked" when its hover distance drops to 1cm or less above a surface; above 2cm is reliably seen as hovering. Hover distances between 1 and 2cm are ambiguous, and largely depend on local noise; we apply hysteresis to reduce flickering between click states. Anecdotally, users did not notice the ambiguity and generally "clicked through" this region on the way to their desired target.

ON-DEMAND PROJECTED INTERFACES

With finger tracking alone, it is possible to support interfaces lacking graphical feedback, or "invisible interfaces" [7]. For example, it would be possible to sketch simple figures or perform graffiti-like text entry on a notepad.

Infusing interactive graphical feedback expands the application space considerably. However, the inherent dynamic nature of objects in the real world makes this complex. Not only must interfaces track which objects they are rendered on, but they must be projected in such a way as to account for their host surface's position and orientation in 3D space (Figure 6). Without these considerations, interfaces would be rendered with inappropriate position, orientation and size, and be subject to perspective visual distortions.

Surface Segmentation and Tracking

In addition to finger tracking, the depth video stream is also used to track surfaces suitable for projection in front of the user. First, distinct surfaces are segmented by performing a 3D connected components operation on the depth map (Figure 7, right). Surfaces smaller than hand size are discarded.

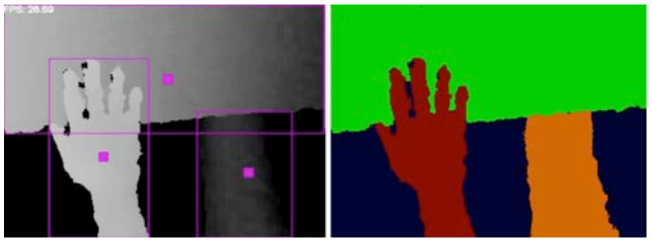
Figure 7. 3D connected components and their lock points.

For each surface, we compute the orientation about the Z-axis (orthogonal to the camera) by taking the covariance of the component's pixels in space, and computing the first and second moments. Orientation about the X- and Y-axes is estimated using the distribution of surface normals, which tend to be Gaussian over the primary orientations.

We also generate a central X/Y/Z "lock point", to which an interface can be attached (Figure 7). This point must be stable regardless of translation and rotation in 3D space. One approach is to take the centroid of an object's pixels. However, because part of the surface may be occluded when the user is interacting with their fingers, this is not reliable. Instead, we move inwards 10cm along the surface's major axis from its upper extent, centered on the midpoint of the minor axis (Figure 8, red). Although more sophisticated techniques are possible, this solution worked well. Finally, a Kalman filter is used to smooth all six degrees of freedom.

Projector/Camera Calibration

To enable authoring and interaction with projected interfaces, it is necessary to calibrate the projector and camera in a unified 3D space. Since our depth camera reports real-world depth values (mm), we chose that as our target coordinate system and calibrate the projector using camera values.

The process requires the intrinsic parameters of the projector, such as the field of view and the center of projection. To find the extrinsic projector parameters we require four non-coplanar calibration points. These four points must be identified by the depth camera and located in the projector image. Once the correspondence of the 2D points in the projected image and their actual 3D location in space (depth camera value) is established, we use the POSIT algorithm [5,32] to find the position and orientation of the projector. Note that this calibration only needs to be performed once, since the spatial relationship between the projector and the camera is fixed (i.e., both are mounted to a rigid frame).

Summoning & Defining Interactive Areas

Determining where to place an interface and how large it should be is non-trivial. For example, consider the hand: Do we center the interface in the middle of the palm, or the centroid of the surface? Or the midpoint between the wrist and finger tips? Or the absolute center of the bounds of the hand? Figure 8 depicts these four (of many possible) options. Sizing the interface has similar challenges: do we fit an interface to just the palm (which is attractive due to its relative flatness), the hand minus the thumb, or the full extent of the hand?

Previous approaches to on-body interfaces [8,17] projected a fixed-sized interface at a fixed image location. In order to use such an interface, a user must raise a physical object into this region at a specified distance, or walk up to a wall. This places the interface localization burden entirely on the user and is ill-suited for many on-the-go mobile scenarios.

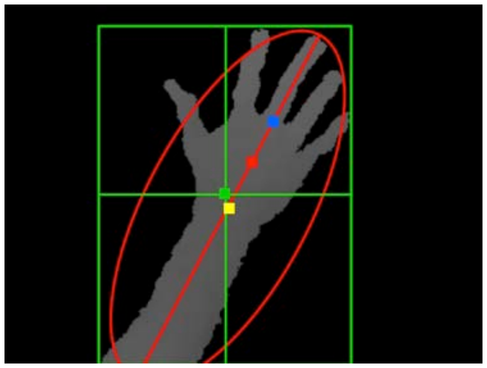
Figure 8. Possible lock points on the hand: Green: absolute center of surface bounds. Yellow: centroid of surface's pixels. Red: 10cm offset along major axis from upper extent. Blue: midpoint between wrist and middle finger tip.

In contrast, *OmniTouch* implements three distinct approaches to define, present, and track interactive areas:

One Size Fits All

OmniTouch can use a surface's lock point and orientation to provide an interface that tracks with a surface. However, because the bounds of the object in 3D space are unknown, the interface can only be as big as the smallest conceivable surface (generally the hand). Thus, even when a projecting on a large table, the interface will still be hand-sized. Additionally, every surface must use a generic lock point, which can lead to sub-optimal centering on asymmetric and organic surfaces, like the hands. These drawbacks motivated us to explore more sophisticated options.

Classification-Driven Placement

Classification-driven placement consists of two stages. First, the system differentiates between a small set of surfaces by performing surface classification. Second, the system automatically sizes, positions and tracks an interface given the available projection area and heuristics describing the appropriate location for that surface.

We perform surface classification among a set of five common surfaces (hand, arm, pad, wall, and table) by considering a variety of features derived from each surface's depth image. For example, to distinguish between planar and organic surfaces, we calculate the standard deviation of the surface normals. Planar objects inherently have a majority of their normals pointing in a common direction, yielding a low standard deviation. On the other hand, organic surfaces tend to be more "rounded" (often symmetrically so), leading to diverse distributions and higher standard deviations. Size

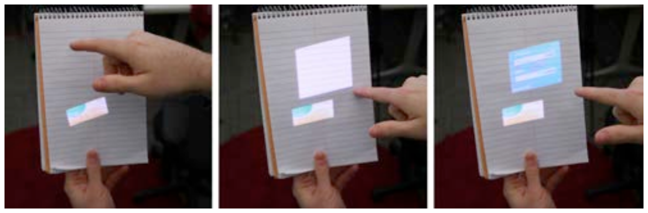
Figure 9. To sidestep complexities in automatically positioning and sizing interfaces, users can simply "click-and-drag" interfaces wherever desired.

is very also descriptive; depth data allows for reasonable approximation of real world size - a notepad is easily distinguished from a table. Additionally, aggregate surface orientation immediately disambiguates tables from walls. These simple features worked well in our prototype implementation given the small set of surfaces to distinguish, but a more general solution would require more sophisticated features (e.g., see [13] for depth-driven object recognition).

Each class of surface defines a unique graphics placement heuristic (an offset vector from the surface's lock point) and default size. For example, a hand has a hand-sized interface while a wall has a wall-sized interface. Lastly, once the surface is identified and the interface is placed, we track the surface change frame-to-frame and accordingly adjust the interface to reflect this change. This mimics the expectation of the user, that once the interface is established, it should remain "glued" to the surface it is projected on. Optionally, given the real-world depth data, the interface can be further refined and fitted to the available area on the surface, by performing depth-constrained flood filling from the surface's placement point.

Unfortunately, this classification-driven approach suffers from scalability issues, since it is simply not possible to build a classifier for every conceivable surface. However, for common surfaces that have unique placement considerations, this approach is attractive and viable.

User-Specified Placement

An entirely different approach is to let the user define the interactive area. This sidesteps much of the complexity described above, as users have a good innate sense of where interfaces should be centered and how big they should be. This exposes a high level of customization to users. However, this flexibility comes at the expense of requiring additional user interaction before an interface can be utilized.

In our prototype system, we provide two mechanisms for user specified placement, although many options are possi-

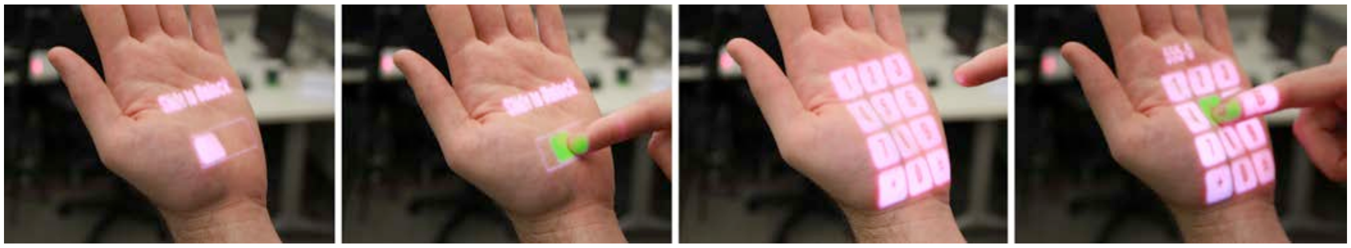

Figure 10. We created a simple phone keypad application; in this sequence, time progresses left to right.

ble. The simplest is for a user to "click" on a surface, causing a generically sized interface to be centered at that location. Alternatively a user can click and drag to position and size in one continuous action (Figure 9). As with the classification-driven approach, once the interface is established, we update its location and orientation frame-by-frame.

Compositing Interfaces in 3D Space

In our proof-of-concept implementation, we model interfaces as planar 2D surfaces, which are positioned and oriented in 3D space. Their 3D placement is computed in relation to the aforementioned lock points and surface orientations so that they are correctly updated as surfaces move. Displaying such interfaces on top of any available surface is straightforward since our projector is precisely calibrated to the depth camera coordinate system. We simply create a 3D scene containing all active surfaces and then render this scene from the perspective of the projector using the projector/camera calibration discussed earlier (Figure 6).While we currently render only planar interfaces, our approach easily lends itself to experimenting with 3D interfaces that take into account the true geometry of the projected surface.

By defining our interfaces in the 3D world space (i.e., using millimeters), they are projected with correct scale and distortion regardless of where the surface is with respect to the camera (as long as it is visible). Our aim is that the interactive surfaces appear to the user "glued" to the physical surface. 3D rendering also automatically takes into account the Z-ordering of our interfaces.

Simultaneously, we use the 3D scene to ray cast fingertip positions onto our planar interfaces. Finger inputs are reported as X/Y coordinates in their local 2D space, which simplifies interface development and enables detection and tracking of finger hover. Although it is possible to use Z distance for click detection, we found our flood-fill heuristic approach to be most accurate.

EXAMPLE APPLICATIONS

With *OmniTouch* providing capabilities similar to that of mice and touchscreens, the application space is expansive. Over the course of development, we created many small, interactive applications that ran on top of our *OmniTouch* engine (Figures 1, 10 and 11). These served both as proof of concept and also as a gauge of input accuracy and real world applicability. We briefly describe a few exemplary applications; please also refer to the Video Figure.

Conventional Interaction

With the ability to click, buttoned interfaces are immediately possible; dragging enables interfaces to be scrolled. As a simple demonstration, we built a phone keypad application (Figure 10). To prevent accidental dialing, a "slide to unlock" feature is included. We also experimented with hierarchical menu navigation, visualized as a scrollable list with clickable items (Figure 11C). This class of interface is prevalent in contemporary mobile devices. Additionally, we built a full keyboard, potentially allowing for text entry on the go (Figure 11A). Lastly, a "post-it" application allows users to write quick notes on their palm (Figure 11E).

To showcase our system's multitouch capabilities, we developed a simple annotation application, where each finger generates a stroke; a small palette of highlight colors is provided (Figure 1). We also implemented the ubiquitous map panning and zooming demo, which is controlled by finger drags and pinching respectively (Figure 11D).

Multi Surface Interaction

As *OmniTouch* can track multiple objects within its field of view, it is possible to support interaction on multiple surfaces and levels. As a proof of concept, we created a painting application for walls. We use the left hand as the color pallet, which can be raised and lowered as needed (Figure 11G). Similarly, when working at a table, the hand may serve as an application switcher (Figure 11F).

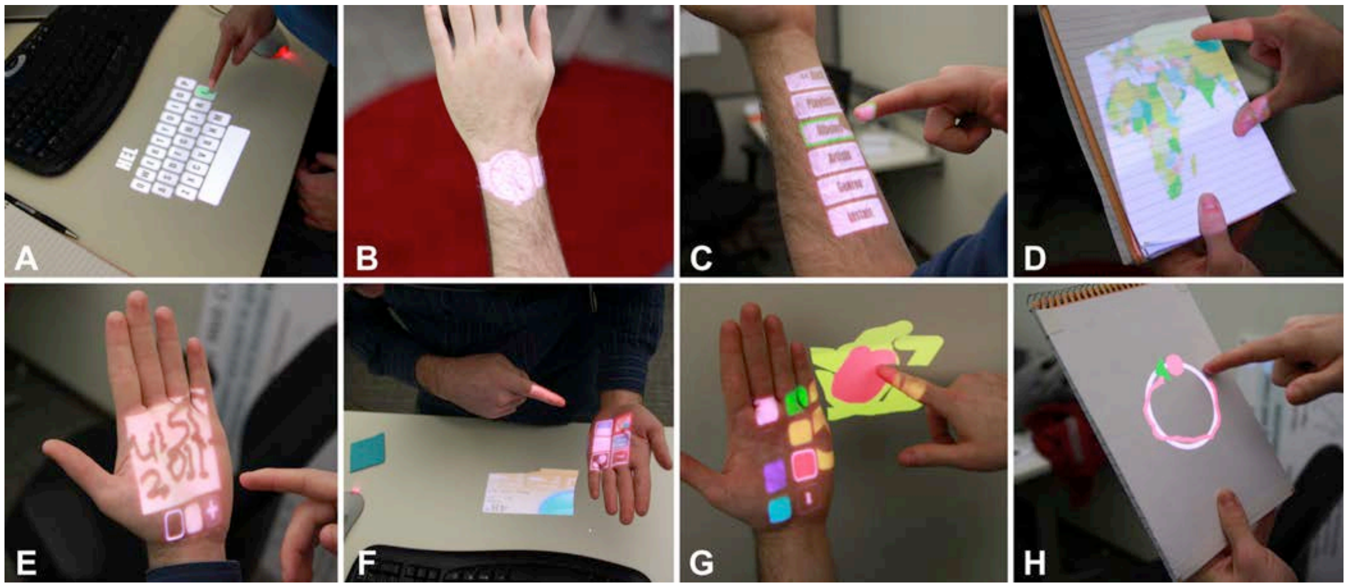

Figure 11. Some of the applications we developed that run on top of *OmniTouch*. See text for descriptions.

Making Use of Rich Metadata

Our surface tracking procedure generates a variety of metadata, such as whether a surface is public or private, organic or flat, big or small, and vertically or horizontally oriented – all of which are exposed to applications wishing to be more context sensitive. For example, the orientation of a hand or handheld surface could be used to determine if the content is visible to only the user or to others (Figure 12) [14]. Additionally, because the geometry between the shoulders and head is fixed, view-dependent VR [31] is possible. Finally, the position of surfaces within the field of view could be used to enable peephole displays [35].

USER STUDY

To evaluate and demonstrate the feasibility of our approach, we conducted a user study that sought to quantify the key performance characteristics of our system. At a high level, can *OmniTouch* correctly register touch events and how accurately can they be localized? At a meta-level, how large would interface elements have to be to enable reliable operation of an ad hoc interface rendered on the hand? To place our system's performance in context, we compare our method to the gold standard - capacitive touch screens - drawing performance results from the literature [9,15,25].

Participants

We recruited 12 participants from our local metropolitan area (6 female), ranging in age from 23 to 49, with a mean of 34. All participants were right handed and were required to have some experience with touch screen devices. The study took approximately one hour and included a gratuity.

Test Surfaces

Our goal with *OmniTouch* was to support interaction on three classes of surface: 1) on-body, 2) objects held in the hands, and 3) fixed surfaces in the environment. For our user study, we included one example from each class: the *hand*, a note *pad* held in the hand, and a *wall*. Additionally, we included the forearm (*arm*), as on-body interaction was a

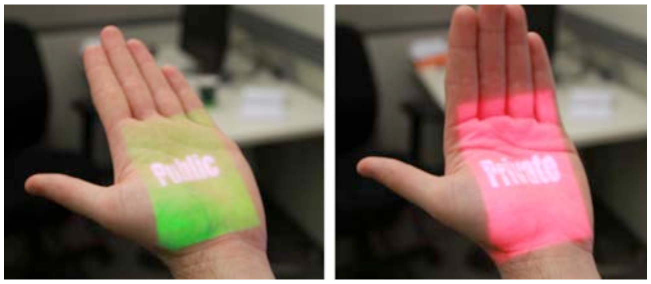

Figure 12. *OmniTouch* can infer if a surface is public or private from orientation features.

particular focus of our work and also challenging from a sensing perspective. Moreover, the *arm* served as a nice contrast to the *hand*, which, although highly irregular, is still fairly planar. Finally, these four surfaces, seen in Figure 13, represent ad hoc surfaces our system would likely use.

Procedure

We first fit participants with our shoulder-mounted system. Once the frame was secured and comfortable, participants were allowed to play with the phone keypad example application. This let them find comfortable positions to hold their arms, both for being rendered on and for pointing, and also to practice using the system. During this period, the experimenters provided feedback to help them become more accurate. This training period lasted a maximum of 10 minutes, though most participants felt confident using the system after just a few minutes of use.

Our primary user study interface consisted of 9 crosshair targets, laid out in a 3x3 pattern (Figure 13). Columns and rows were spaced 3cm apart; the crosshairs were 2x2cm in size (9x9cm total size). In each trial, one crosshair was rendered in red. Users "clicked" this crosshair as accurately as they could. If a click was detected, the system would beep and a green circle was placed around the target crosshair (see Figure 13, wall). The experimenter advanced the inter-

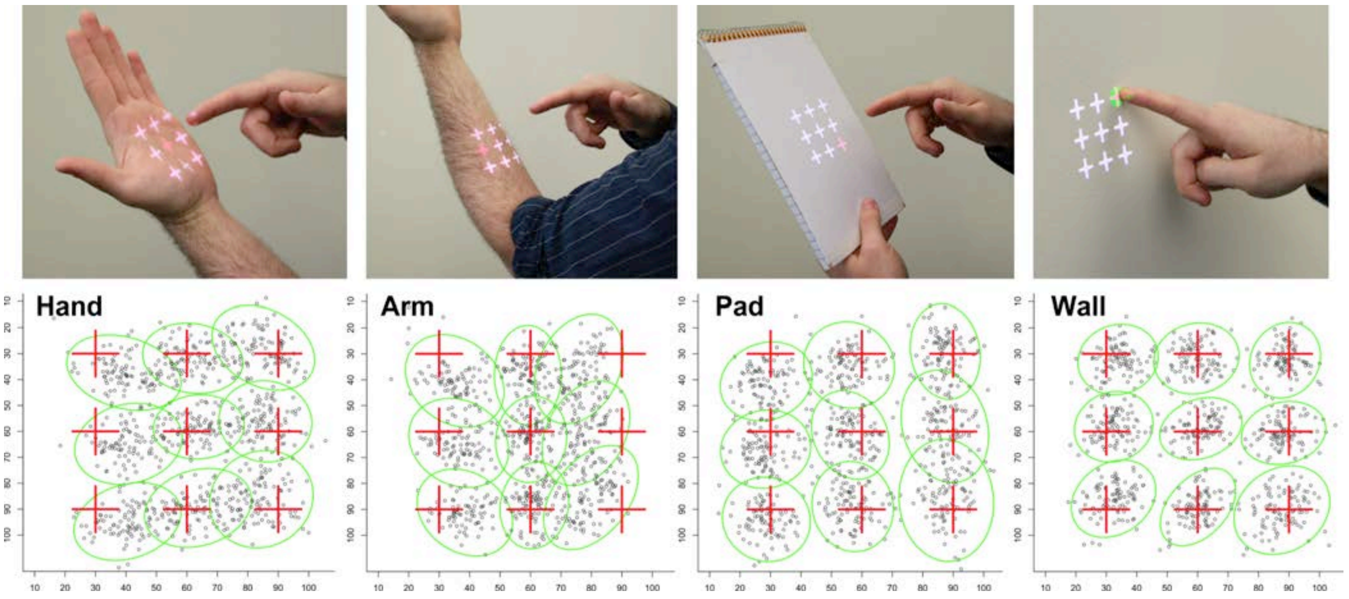

Figure 13. The four surfaces we tested and user click distributions. 95% confidence ellipses shown in green. Axis units in mm.

face to the next trial after each click attempt, regardless of whether or not it was detected. Each of the 9 crosshair locations was repeated 4 times, for a total of 36 click trials; presentation order was randomized.

This interface and procedure was used for each of the four test surfaces: *hand*, *arm*, *pad* and *wall*. Before each surface, users were allowed to briefly practice before data collection began. For the *wall* condition, participants were asked to stand approximately 30cm from the wall. For the other three surfaces, users found a comfortable position. The ordering of the surfaces was randomized to compensate for any order effects. We ran two rounds of data collection to investigate if there were any effects from learning, fatigue, or slight variations in posture. This produced 288 trials (2 rounds x 4 surfaces x 36 click trials) per participant.

To quantify how our system performed at different distances, we included two additional rounds of data collection. Participants were asked to hold their *hands* at arm length (*far*), at an "average and comfortable" distance (*average*), and as close to the system as possible, while still being able to click with their right hand (*close*). We also tested these three distances with the *pad* surface; the ordering of the *pad* and *hand* distance trials was alternated between participants. This procedure produced 216 trials (2 surfaces x 3 distances x 36 click trials) per participant.

The tests described above were primarily designed to isolate click segmentation and spatial accuracy. A key feature of *OmniTouch* is its ability to track fingers while dragging. To better understand the spatial performance of finger drags, we created a drawing experiment interface (Figure 11H). In this application, users were presented one of six possible shapes: up line, down line, left line, right line, clockwise circle, counterclockwise circle. Each shape was repeated 4 times, for a total of 24 drawing trials per participant.

Direction of the stroke was indicated using a green arrow and a red "stop". Participants were asked to draw as closely to the white path as possible, balancing speed and accuracy. Unlike in the crosshair experiments, users received graphical feedback in the form of a red path illustrating their stroke. This allowed participants to compensate for any inaccuracies in their movement and the system's fingertip estimation. We chose to conduct this experiment on the *pad*, as a flat surface minimized external confounds (e.g., user inaccuracy caused by the irregular surface of the hands).

RESULTS & DISCUSSION

Our 12 participants produced 3456 click trials on our four surfaces, a further 2592 in our distance experiment, and 288 drawn shapes. No effect was found between the two rounds of crosshair trials (e.g., from fatigue or learning). Additionally, there were no significant performance differences between participants. Thus, data was combined for each surface. Data from the distance and the dragging trials was kept separate for independent analysis. Ultimately, these results should be considered a performance baseline, as significant improvements in depth camera resolution and sensitivity are forthcoming.

Finger Click Detection

We combined data from all of our crosshair-clicking experiments (two rounds of four surfaces and two rounds of three distances) – a total of 6048 click trials. Of these, 96.5% correctly received exactly one finger click event. Regarding errors, 50 trials (0.8%) had no click event (i.e., the system missed the participant's finger click), 154 trials (2.5%) had two click events (i.e., the system incorrectly thought the user clicked twice, or believed a secondary finger to have clicked), and 8 trials (0.1%) had three click events.

For the user study, we configured *OmniTouch* to record all input events, without any high-level mechanism for click rejection, as typically found in interactive systems. Of the 162 trials receiving double and triple clicks, 94.8% percent occurred within 500ms of the first click event. Thus, with a simple timeout, single finger click segmentation accuracy would be 98.9%.

Of the 50 trials (0.8%) with missed clicks, 33 were contributed by the three left-most crosshairs in the *arm* condition (see Figure 13, arm). As noted in [23], participants tend to hook their fingers when targeting items on reverse slopes, which is the case for the right hand targeting the left most side of the left forearm. One possible explanation for this increased error is that hooking occludes the contact point and also shortens the finger's profile from the camera's perspective, which can cause tracking loss. Otherwise, the distribution of missed-click and multi-click errors was evenly spread over all crosshair positions and surface conditions.

Finally, we compared click segmentation performance at the three distances tested in the user study (*hand*/*pad* surfaces at *close*/*average*/*far* distances). However, no significant effects were found.

Finger Click Spatial Accuracy

Importantly, our results represent the cumulative error of the system and the user. There are three primary sources of error: 1) misalignment and non-linearities in the projector/camera calibration (e.g., a button is projected somewhere slightly different from where the camera believes it to be), 2) inaccuracy in the fingertip estimation, especially when the tip fuses with the surface during clicks, and 3) user inaccuracy when clicking targets, (e.g., due to "fat fingers" and varying perception of one's finger input point [9]). Although some of these factors are outside of our control, they model the *real-world performance* of our system.

There are two important and independent measures for analyzing targeting performance: offset and spread [4,9,25].

Finger Click Spatial Offset

Analysis revealed there was a small systematic offset between where *OmniTouch* believed the user clicked and where the user believed they clicked. Specifically, we found an average offset of 11.7mm to the left of targets across all conditions and participants, in agreement with previous findings in the touchscreen literature [15,25]. Y-offset for the *hand*, *arm* and *pad* surfaces was similarly an average of 1.1mm above the true target. Finger touches on the *wall*, however, were offset downwards an average 10.0mm, pos-

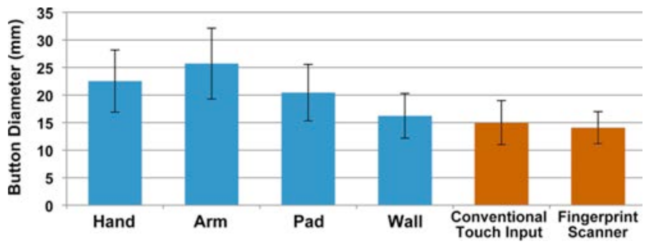

Figure 14. Button diameter necessary to encompass 95% of touches. Error bars denote standard deviation across all trials. Results in orange from [9].

sibly due to its extreme angle (at roughly chest height, and oriented vertically). Finally, distance appears to have no significant effect on offset.

Because offsets are systematic across-users and across-surfaces, we simply apply a post-hoc X/Y offset to our subsequent data analysis. These offset values could be trivially added to our system's real-time finger point estimations. The only case we handle specially is the *wall*, which is recognized by *OmniTouch* using size and orientation information. With the wall, points are shifted upward 10.0mm. For maximum generality, we did not compute or apply any per-user offset, though this has been shown to significantly increase accuracy [9,30].

Finger Click Spatial Precision

The spatial precision of *OmniTouch* is visualized in Figure 13, which depicts 95% confidence ellipses for the nine crosshair targets on our four test surfaces. For analysis, we removed 93 outliers in order to plot our results side-by-side with those in [9]. Outliers were defined as points lying greater than three standard deviations away from the mean difference between points and the target. Similar to [9], outliers were a mix of user error, user inaccuracy, and tracking errors.

Figure 14 displays the minimum button diameter necessary to correctly capture 95% of touches for each surface. We also include two points of comparison from [9] - an estimation of conventional touch input (derived from a capacitive touchpad) and results from crosshair trials using a high-resolution optical fingerprint scanner. Exceeding our expectations, *OmniTouch* on a *wall* appears to be nearly as accurate as conventional touchscreens (16.2mm vs. 15.0mm). The *hand* requires buttons to be 22.5mm in diameter, offering the highest un-instrumented, on-body button density to date [8,24,32]. The *pad* performs similarly to the *hand*.

The arm is our least accurate surface, requiring targets approximately 70% larger than a conventional touch screen to achieve the same 95% touch accuracy (25.7mm vs. 15.0mm). This degradation in error comes chiefly from buttons located on the sides of the arms, where curvature is high (see Figure 13). Given that the arm is well suited to narrow, tall interfaces (Figure 11C), we also computed the accuracy using only the center column of crosshair targets, which proved to be quite accurate (20.5mm, SD=5.1mm).

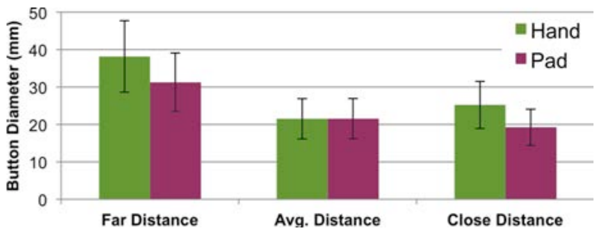

Figure 15. Button diameter needed to encompass 95% of touches. Error bars show standard deviation across trials.

Effects of Distance on Spatial Precision

We previously reported that distance had no significant effect on click segmentation accuracy or on spatial offset. However, there does appear to be a significant loss of precision when interacting at *far* distances. Using a Bonferroni-corrected all-pairs *t*-test, we found no significant difference in performance between the *hand* and *pad* at our three test distances. We then combined *hand* and *pad* distance trials into aggregated *far*, *average* and *close* data sets. Overall, the *far* condition is significantly worse performing than both *average* and *close* distances (both $p<.001$) – requiring buttons to be roughly 60% larger to operate at around arm's length. There is no significant difference between *average* and *close* distances. Figure 15 illustrates these results as button diameters necessary to encompass 95% of touches.

Finger Drag Spatial Accuracy

In our dragging (drawing) experiment (Figure 11H), we included vertical and horizontal lines, as well as circles, in order to assess X, Y and X+Y dragging performance. For both lines and circles, we use the absolute Euclidian distance from the closest point on the desired path as our error distance metric. We did not apply our global X/Y offset to stroke points as live graphical feedback was provided. Such feedback allowed participants to compensate for any system inaccuracies as they performed the task.

On average, participants deviated from the desired path by just 6.3mm (mean SD=3.9mm). There is no significant performance difference between shapes, 1D or 2D trials, or in X- or Y-axes.

FUTURE WORK

Several immediate avenues of future work exist. Foremost, we focused on interfaces that were 2D and primarily rectilinear. Our bodies, however, are neither. If we are to have interfaces on organic surfaces [10], it will be valuable to rethink classic interface paradigms, and consider how our unique form can contribute to a computing experience.

There are also many opportunities to enhance the approaches in this work. For example, it is possible to create 3D meshes of objects, allowing for distortion free projection onto non-planar surfaces (i.e., projective texturing). Additionally, postures and gestures of the arms and hands could be used for input. For example, a "telephone" gesture with hands could summon a keypad on the arm, while a "back of hand" pose could render a watch on the wrist (Figure 11B).

Finally, there are many fascinating qualitative questions surrounding on-body interaction. For example, how comfortable are people using their own bodies as interactive platforms? What are the social implications if other people want to use these interfaces? Is it acceptable to control an interface on someone else's body? Such interactive capabilities may open new questions in proxemics research.

CONCLUSION

In this paper, we described and evaluated our proof-of-concept implementation of *OmniTouch*. Results suggest interactions traditionally reserved for dedicated touch surfaces can now move into the environment, in an ad hoc fashion. Although our current prototype is fairly large, there are no significant barriers to miniaturization. It is entirely possible that a future incarnation of *OmniTouch* could be the size of a box of matches, worn as pendent or watch. Thus, the benefit of extreme portability can be combined with the ease and accuracy of interaction on large, physical surfaces.

The chief goal of the present work is to demonstrate that touch input can be achieved on everyday surfaces, including the human body. This brings to reality many intriguing interactions proposed in earlier work. In this paper, we only graze the surface of interactions enabled by making touch input available everywhere. We answer the fundamental question of whether or not it is possible, but in many ways, the hardest work lies ahead.

ACKNOWLEDGMENTS

We thank PrimeSense and Ohad Shvueli for the short range depth camera used in this research.

REFERENCES

1. Argyros, A.A., and Lourakis, M.I.A. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Proc. ECCV '06 Workshop on Computer Vision in HCI*. LNCS 3979. 40–51.
2. Benko, H. Saponas, T.S., Morris, D., and Tan, D. Enhancing input on and above the interactive surface with muscle sensing**.** In *Proc. ITS '09*. 93–100.
3. Cao, X., and Balakrishnan, R. Interacting with dynamically defined information spaces using a handheld projector and a pen. In *Proc. UIST '06*. 225–234.
4. Chapanis, A. Theory and methods for analyzing errors in man-machine systems. Annals of the New York Academy of Science 51, *Human Engineering* (1951), 1179–1203.
5. DeMenthon D. and Davis, L.S. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15, (1995). 123–141.
6. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., and Twombly, X. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*. 108, (2007), 52–73.
7. Gustafson, S., Bierwirth, D., and Baudisch, P. Imaginary interfaces: spatial interaction with empty hands and without visual feedback. In *Proc. UIST '10*. 3–12.
8. Harrison, C., Tan, D., and Morris, D. Skinput: appropriating the body as an input surface. In *Proc. CHI '10*. 453–462.
9. Holz, C. and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proc. CHI '10*, 581–590.
10. Holman, D. and Vertegaal, R. Organic user interfaces: designing computers in any way, shape, or form. *Comm. of the ACM*, 51, 6 (2008). 48–55.
11. Hua, H., Brown, L., and Gao, C. Scape: Supporting stereoscopic collaboration in augmented and projective environments. *IEEE Comput. Graph. Appl.* 24, 1 (2004), 66–75.
12. Kane, S., Avrahami, D., Wobbrock, J., Harrison, B., Rea, A., Philipose, M. and LaMarca, A. Bonfire: A nomadic system for hybrid laptop-tabletop interaction. In *Proc. UIST '09*. 129–138.
13. Lai, K., Bo, L., Ren, X., and Fox, D. Sparse Distance Learning for Object Recognition Combining RGB and Depth Information. In *Proc. ICRA '11*.
14. Lee, J.C., Hudson, S.E. and Tse, E. Foldable interactive displays. In *Proc. UIST '08*. 287–290.
15. Lewis, R.J. Literature review of touch-screen research from 1980 to 1992. *IBM Technical Report*, 54.694. Aug 20, 1993.
16. MicroVision, Inc. http://www.microvision.com
17. McFarlane, D., and Wilder, S. Interactive dirt: Increasing mobile work performance with a wearable projector-camera system. In *Proc. UbiComp '09*, 205-214.
18. Mistry, P., Maes, P., and Chang, L. WUW - wear Ur world: a wearable gestural interface. In *CHI '09 Ext. Abst.* 4111–4116.
19. Park, Y. Han, S., Park, J. and Cho, Y. Touch key design for target selection on a mobile phone. *Proc. MobileHCI '08*. 423–426.
20. Pinhanez, C. S. The Everywhere Displays projector: A device to create ubiquitous graphical interfaces. In *Proc. UBICOMP '01*. 315–331.
21. PrimeSense Ltd. http://www.primesense.com.
22. Raskar, R., Beardsley, P., van Baar, J., Wang, Y., Dietz, P., Lee, J., Leigh, D., and Willwacher, T. RFIG lamps: interacting with a self-describing world via photosensing wireless tags and projectors. In *Proc. SIGGRAPH '04*. 406–415.
23. Roudaut, A, Pohl, H. and Baudisch, P. Touch input on curved surfaces. In *Proc. CHI '11*. 1011-1020.
24. Saponas, T.S., Tan, D.S., Morris, D., Balakrishnan, R., Turner, J., and Landay, J. A. Enabling always-available input with muscle-computer interfaces. In *Proc. UIST '09*. 167–176.
25. Sears, A. Improving touchscreen keyboards: Design issues and a comparison with other devices. *IEEE Computer*, 3 (1991), 253–269.
26. Starner, T., Auxier, J., Ashbrook, D., and Gandy, M. The Gesture Pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proc. ISWC* '00, 87–94.
27. Sturman, D.J. and Zeltzer, D. A Survey of glove-based input. *IEEE Comp Graph and Appl*, 14, 1 (1994). 30–39.
28. Tan, D., Morris, D., and Saponas, T.S. Interfaces on the go. *ACM XRDS*, 16, 4 (2010), 30–34.
29. Tomasi, C., Rafii, A. and Torunoglu, I. Full-size projection keyboard for handheld devices. *Comm. of the ACM*, 46, 7 (2003), 70–75.
30. Wang, F. and Ren, X. Empirical evaluation for finger input properties in multi-touch interaction. In *Proc. CHI '09*. 1063–1072.
31. Ware, C., Arthur, K. and Booth, K.S. Fish tank virtual reality. In *Proc. CHI '93*. 37–42.
32. Wilson, A. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc. UIST '10*. 273–282.
33. Wilson, A.D. PlayAnywhere: a compact interactive tabletop projection-vision system. In *Proc. UIST '05*. 83–92.
34. Wilson, A.D. Using a depth camera as a touch sensor. In *Proc. ITS '10*. 69–72.
35. Yee, K. Peephole displays: pen interaction on spatially aware handheld computers. In *Proc. CHI '03*. 1–8.