HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration

Roman Rädle[1], Hans-Christian Jetter[2], Nicolai Marquardt[3], Harald Reiterer[1], Yvonne Rogers[3]
[1] HCI Group, University of Konstanz, {roman.raedle,harald.reiterer}@uni-konstanz.de
[2] Intel ICRI Cities, University College London, h.jetter@ucl.ac.uk
[3] UCL Interaction Centre, University College London, {y.rogers,n.marquardt}@ucl.ac.uk

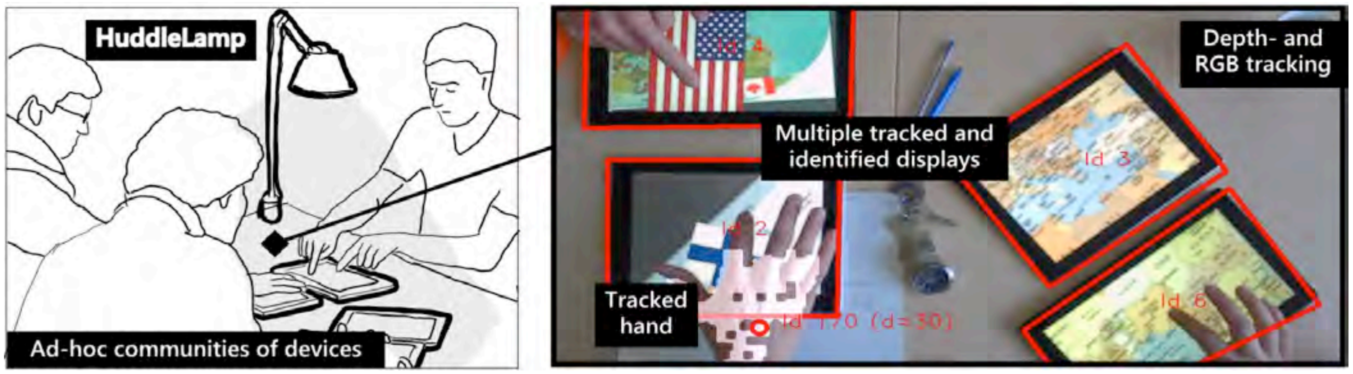

Figure 1. The HuddleLamp detects and tracks mobile devices and users' hands for ad-hoc multi-device collaboration on desks.

ABSTRACT

We present HuddleLamp, a desk lamp with an integrated RGB-D camera that precisely tracks the movements and positions of mobile displays and hands on a table. This enables a new breed of spatially-aware multi-user and multi-device applications for around-the-table collaboration *without* an interactive tabletop. At any time, users can add or remove displays and reconfigure them in space in an ad-hoc manner without the need of installing any software or attaching markers. Additionally, hands are tracked to detect interactions above and between displays, enabling fluent cross-device interactions. We contribute a novel hybrid sensing approach that uses RGB and depth data to increase tracking quality and a technical evaluation of its capabilities and limitations. For enabling installation-free ad-hoc collaboration, we also introduce a web-based architecture and JavaScript API for future HuddleLamp applications. Finally, we demonstrate the resulting design space using five examples of cross-device interaction techniques.

Author Keywords

Mobile devices; cross-device interactions; depth camera; ad hoc collaboration; society of devices; ubicomp ecologies.

ITS '14, Nov 16-19 2014, Dresden, Germany
ACM 978-1-4503-2587-5/14/11.
http://dx.doi.org/10.1145/2669485.2669500

ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: Input

INTRODUCTION

We are witnessing an explosive growth of the number and density of powerful mobile devices around us. However, their great majority are still blind to the presence of other devices and performing tasks among them is usually tedious [6] and lacks recognizable guiding principles [25]. We envision a future where mobile devices can contribute their interaction resources (e.g., their multi-touch displays) to a community of devices in their proximity that then serves as one seamless multi-device user interface (UI). At any time, users can dynamically compose and reconfigure this UI according to their current needs and the task at hand.

In this vision, users can seamlessly add or remove devices from the community in an ad-hoc fashion without explicit setup or pairing. Instead, this happens implicitly as a by-product of natural use in space, for example, by bringing multiple devices to the same room, placing them side-by-side on a table or desk, and moving them around as needed. Ideally, users will experience these co-located cooperating devices and reconfigurable displays as one seamless and natural UI for ad-hoc co-located collaboration.

As a first incarnation of this vision, we present *HuddleLamp*, a desk lamp with an integrated low-cost RGB-D camera that detects and identifies mobile displays (e.g., smartphones or tablets) on tables and tracks their positions and orientations with sub-centimetre precision. Users can add or remove off-the-shelf, web-enabled devices in an ad-hoc fashion *without* prior installation of custom hardware (e.g., radio modules, IR markers) or software. Because of HuddleLamp's web-based pairing, adding a new device is simply done by opening a URL on the device and putting it on the table so that it is visible to the camera. HuddleLamp also tracks users' hands to enable interactions between and above devices, such as 'pick, drag, and drop'.

This enables a new kind of computer-supported around-the-table collaboration *without* interactive tabletops. Users can still sit around ordinary tables that can remain cluttered with non-digital objects (e.g., printouts, maps, notebooks) while their digital collaborations happen using spatially-aware mobile screens that blend into existing spatial and social practices. Unlike traditional interactive tabletops, HuddleLamp needs only few *low-cost* and *off-the-shelf* hardware components, so it can be used in improvised settings or where the costs for hardware and administration typically prohibit the use of large interactive tabletops, e.g., public libraries, schools, community centres.

In the remainder of this paper, we focus on our following contributions:

- As our primary contribution, we introduce our *novel hybrid sensing approach* for HuddleLamp. This approach combines RGB *and* depth input for detecting and tracking movements of multiple mobile screens with sub-centimetre precision by exploiting their optical characteristics in both the RGB and IR range. We evaluate the tracking quality of our approach in terms of *accuracy*, *precision*, and *reliability* with a controlled experiment and discuss capabilities and limitations.
- As secondary contributions:
  - we introduce our web-based architecture and JavaScript API for enabling truly ad-hoc, walk-up-and-use applications with no need for installing any native mobile apps or instrumenting devices with any markers or hardware before collaborating.
  - we validate our architecture and API with five example interaction techniques that used to be possible only in instrumented rooms or on and above large tabletops. They demonstrate the future design space for spatially-aware multi-device around-the-table collaboration with HuddleLamp.

RELATED WORK

HuddleLamp relates to four different strains of previous research that we sample and discuss below.

Spatial Interaction in Instrumented Environments. The use of space and spatial configurations plays a key role in human cognition [11] and natural social interactions [6]. To leverage our spatial skills for interaction, one approach is to create instrumented rooms that capture the spatial interactions and configurations of users and objects as system input. For example, LightSpace uses multiple depth cameras and projectors to provide interactivity on and between physical surfaces [37]. The steerable Beamatron [36] projects directly into the environment, and the LuminAR actuated desktop lamp design allows touch detection and projection onto desks [17]. As a mobile solution, SurfacePhone [38] uses a similar approach to facilitate single or multi-user around-the-phone interactions through projected interfaces. The OmniTouch [8] wearable device introduces a high flexibility of where and when such interfaces are displayed. We focus on enabling ad-hoc device assemblages and interaction around tables with the mobile devices people already carry, such as phones and tablets. Conductor [7] recently introduced techniques for orchestrating such cross-device interactions, which HuddleLamp facilitates by automatically tracking spatial relationships between devices, allowing for fluent transfer and sharing.

Greenberg argued that we can enhance the interaction with ubicomp technology by designing systems that consider fine-grained inter-entity relationships, such as the distance or orientation between people and devices [6]. As proxemic interactions this has been applied in various contexts, such as the interaction with a large surface media player [1], games [6], or interactive advertisements [35]. Later, the GroupTogether system considers people's spatial F-formations [14] during small group collaboration to initiate cross-device interactions [21]. The majority of these systems rely on high-end motion capturing systems [20] that are ideal for prototyping interaction techniques but are difficult to deploy in environments out of the lab. To enable lightweight, ad-hoc scenarios of use, our system keeps the effort for instrumentation minimal, only uses a single RGB-D camera for tracking, and does not require any additional augmentations of devices.

Presence, Pairing, and Position of Co-located Mobile Devices. To enable cross-device interactions [19], the system first needs knowledge about the presence and position of other devices around it. Various techniques establish such connections: synchronous gestures [27] that pair devices when stitching stroke across the screens [10]; other approaches include shaking devices simultaneously [12], bumping [27], or performing pinching gestures [24]. Another approach is to use custom sensing hardware, for example infrared-, hall-, or radio-based position sensing. Siftables use infrared emitters and transceivers for detecting nearby cubes [23]. Likewise, mobile devices with magnets and hall sensors can sense presence [13], or instrumented phones can utilize custom-built radio tracking for positioning [18] – though often with relatively coarse-grained spatial resolution (~1m). Similarly, GroupTogether requires custom-built radio-based position trilateration for device positions [21]. As explained shortly, our novel sensor fusion approach brings this tracking to a new level by not requiring any radio-based trilateration hardware and allowing precise sub-centimetre tracking.

Alternatively, built-in cameras can infer device location information. Back facing phone cameras can infer relative positions from extracted features in the downward-facing camera images (e.g., legs, feet) [3]. Similarly, Schmitz et al. [29] and Li et al. [16] use the front facing camera of devices to detect fiducial markers on the ceiling. Improving this tracking for reliably handling different environmental conditions remains an on-going research challenge.

Alternatively, cameras positioned above a table can track the spatial layout of devices placed on the table below. Rekimoto's tracked devices' location by attaching fiduciary markers [28], an approach later also applied to track multiple devices for DynamicDuo [26]. Kray et al. later used a related approach to determine the position of phones in one of three discrete spatial zones that trigger sharing actions between the phones [15]. Taking these approaches further, we consider both depth- and RGB-tracking data to improve tracking quality for devices placed on a table.

Reconfigurable Tiled Displays. A special case of co-located mobile devices are (re)configurable tiled displays. ConnecTables [34] allow dynamically reconfigurable display assemblages when devices are in proximity. At a larger scale, Phone as a Pixel [30] creates ad-hoc large-scale displays composed of smaller devices, each serving as a pixel of a large virtual display. The position of each pixel/device is calculated by decoding a sequence of colour transitions that encodes the ID of that device with a camera [30]. Similarly, the web-based Junkyard Jumbotron system stitches together devices' screens to a single large virtual display [2]. Even though it is not built for real-time tracking, it allows a manual process where users take pictures of fiduciary markers shown on the screens and send it to a server for calibration processing. One approach towards allowing the display of *dynamic* digital content are Schmitz' [29] and Li's [16] ad-hoc multi-displays for mobile interactive applications, though their tracking performance or accuracy is not yet tested.

Above and around the surface interactions. For more expressive interactions, the interaction space with surfaces can be extended to include the space above and around the surface. Recent work explored this continuous interaction space with lenses above [32] or around tabletops [22], with techniques for picking up and manipulating content on an interactive touch surface [9]. TangibleLenses [31] and FlexPad [33] enable handheld interactions with rigid or flexible surfaces, and LightSpace allows bi-manual gestures to transfer content between multiple surfaces [37]. One important design goal of the HuddleLamp system is to enable similar cross-device interactions (including tracking people's gestures), around the ad-hoc assemblages of phones and tablets on a table.

HUDDLELAMP'S TECHNICAL SETUP

In the following, we describe HuddleLamp's technical setup, and its architecture and algorithms. To facilitate replication outside our lab, we used only off-the-shelf hardware and free or open source software components. We provide the source code as open source[1].

Hardware Components

HuddleLamp uses a low-cost *short range time-of-flight* (TOF) depth camera which delivers a 1280×720 RGB colour image and a 320×240 depth image at 25 or 30 fps. It shares its technical specifications with a Creative Senz3D or SoftKinetic DepthSense 325. The camera is fixed to an Artemide Tolomeo Tavolo desk lamp (Figure 2) in which it replaces the light bulb (top right corner of Figure 2). Using the lamp, users can conveniently move the camera into its downward-facing operating position that lies 0.8m above the horizontal surface to track. This results in a rectangular tracking region of approximately 1.0×0.6m from which the camera receives sufficient RGB and depth information to track mobile devices, their spatial configurations, and users' hands and to distinguish them from non-interactive objects.

The camera was chosen for its small size (110×30×25mm) and low-noise depth data. A further advantage of this particular camera and its Perceptual SDK by Intel is that its RGB and depth images can be aligned without calibration. Therefore it is easy to retrieve RGB values for depth pixels and vice versa. This facilitates the processing of data in our hybrid sensing approach. However, in principle, hybrid sensing should also work with other TOF cameras (e.g. Kinect v2) and a higher resolution and larger field of view.

For vision processing and for communicating with the mobile devices via a web socket server, we use a Windows PC or laptop (Figure 2, left). For better portability, we have considered integrating a single-board PC (e.g. Raspberry Pi) directly into the lamp. However, at this stage, the vision processing is still too computationally expensive to achieve our targeted tracking rate of 25-30 fps with ARM CPUs.

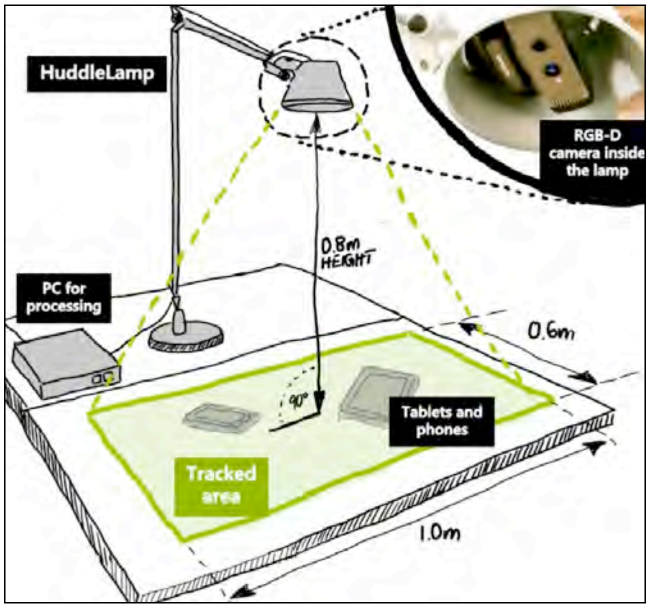

Figure 2. Technical setup of HuddleLamp with an integrated RGB-D camera (Tracking region: 1.0×0.6m).

Software Components & System Architecture

The computer vision application for processing the RGB and depth data was implemented in C# for WPF/.NET 4.5.1 and Emgu CV (OpenCV bridge for C#). For finding and decoding fiducial markers in the RGB stream we use the

[1] http://www.huddlelamp.org

glyph decoder of the AForge.NET library[2]. While the vision application is active, the incoming camera data is processed and all the identified device and hand positions, device orientations, and device occlusion states are streamed as JSON to connected mobile devices via a web socket server.

Client applications on the mobile devices access this stream using HuddleLamp's JavaScript API for browsers such as Safari (Mobile), Chrome, and Internet Explorer. If desired, the API also provides a shared virtual workspace that is spatially-situated in the physical tracking region and can be accessed with the devices. It uses a shared object storage implemented on top of the Meteor web framework[3] to distribute rendering and interaction across devices.

HUDDLELAMP'S HYBRID-SENSING

HuddleLamp's main purpose is to identify and track mobile screens of different sizes with sub-centimetre precision and to distinguish them from other objects or hands. To this end, we combine RGB and depth information and verify our results with virtual fiducial markers. This 'hybrid sensing' compensates for the limitations of sensing RGB, depth, IR, or fiducials alone and, unlike [16,18,20,21,23, 28], works without instrumenting devices or rooms with custom radio-hardware, markers, or tags.

Detection of Mobile Screens Using Low IR Reflectance

The first step of hybrid sensing is detecting the regions that possibly contain mobile screens. For this we use two optical characteristics that all mobile screens we worked with have in common. First, they are obviously rectangular and thus can be easily recognized as rectangles in a camera image. Our second characteristic, however, is not visible to the eye and we only learned about it during own experimentation:

We found that mobile screens generally have a very low reflectance for the modulated IR signal that is emitted by TOF cameras. Therefore, whenever a screen enters the view, the screen and its bezel absorb rather than reflect the IR signal. The reflected signal becomes so weak that the camera cannot reliably measure depth and returns 'low confidence' for most pixels inside of screens. However, depth values are available for pixels outside of screens, e.g., from the table's surface. Figure 3 shows this in side-by-side comparisons of RGB images (left) and depth confidence images (right). The right images are created simply by drawing all pixels that have a depth value in black and all 'low confidence' pixels in red ('low confidence' means signal intensity is < 87; intensity ranges from 1 to 32,767). The top row of Figure 3 shows low IR reflectance for devices of very different size, generation, screen type, screen brightness, and screen content (1. Samsung/Google Nexus 10; 2. LG/Google Nexus 5; 3. Nokia Lumia 620; 4. Apple iPad Air; 5. Lenovo ThinkPad Yoga; 6. Microsoft Surface 2 Pro; 7. Apple iPad 3; 8. Apple iPod Touch; 9. Samsung Galaxy S2; 10. Apple iPhone 4; 11. Nokia 106).

[2] http://www.aforgenet.com/
[3] https://www.meteor.com/

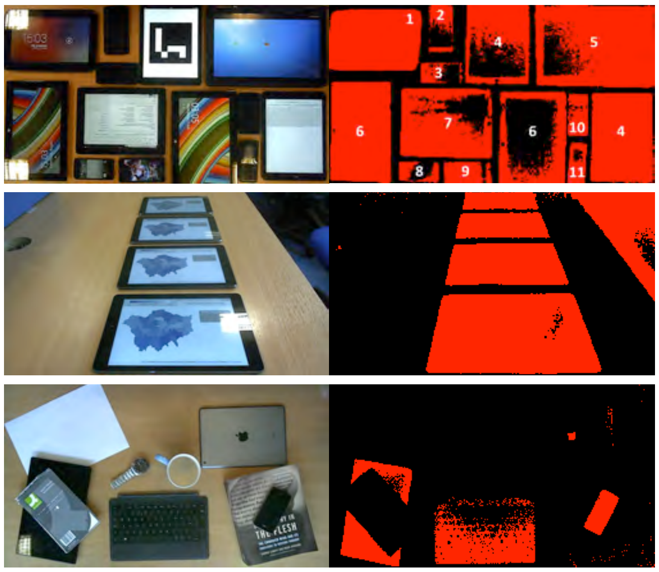

Figure 3. Low IR reflectance of mobile screens and objects.

The second row illustrates that low IR reflectance (here that of an Apple iPad Air) cannot only be observed when devices are perpendicular to the camera, but also for other angles (from bottom to top: 45°, 31°, 22.1°, 18.6°).

The third row shows that many everyday objects (from left to right: sheet of paper, notepad, watch, cup, back of Apple iPad, book) have normal to high IR reflectance and their depth is measured correctly. They therefore disappear among the other black pixels in the depth confidence image. The other objects (from left to right: Microsoft Surface tablet under notepad, Microsoft Surface keyboard, Apple logo on back of Apple iPad, Apple iPhone on book) have low IR reflectance and thus show in the right image as red shapes. Generally speaking, apart from screens, black or dark grey objects with a matte non-glossy finish are likely to have a low IR reflectance. However, as we explain below, such false positives can be easily identified as non-screen objects at a later stage.

The novelty of our screen tracking approach lies in not discarding but using these regions of low IR reflectance with missing depth values. The only other approach that we are aware of that uses IR reflectance (in that case structured light from a Kinect v1) to distinguish materials (e.g. skin from paper) is [33]. We can achieve a similar segmentation of screens from non-screens 'for free', by using a TOF camera and a threshold for signal strength without additional computation. Unlike Dippon et al. [4], we can therefore avoid the need for object detection and segmentation algorithms that use actual depth data.

In Figure 4, the RGB image R1 and the depth confidence image D1 show the example of eight mobile screens on a table under realistic lighting conditions. Thereby D1 is almost completely unaffected by ambient light, screen content, screen backlighting, or even the bright reflections from overhead light sources in R1. Since IR tracking works with its own light sources, D1 would also look very similar in a room that appears completely dark to the human eye. In general, IR reflectance tracking is very reliable with regard to all kinds of other light sources and it is very easy to extract the positions and orientations of screens from an image like D1 just by using low-pass filtering, Canny edge detection, and finding contours and convex hulls. The result of this process is shown in image D2 and contains 7 out of 8 screens.

Sauron's Eye

Only using the depth confidence image would however have limitations. One limitation lies in D1's centre region, which we informally refer to as 'Sauron's eye' (Figure 4). Here the IR signal from the camera hits the screen surface with almost 90°. The IR signal is directly reflected back into the depth sensor with great intensity and saturates a small region of pixels in the centre. This results in a small red ellipse in the centre of D1 that is the 'pupil' of Sauron's eye (like low confidence pixels, saturated pixels in D1 are red). Around this inner ellipse, the reflectance remains high, but not high enough to saturate. Therefore there is another elliptical region around the saturated pixels, for which the depth values can be read. They show as a black 'iris' region around the pupil that only gradually turns into red pixels when moving further away from the centre. Finally, Sauron's eye is contained by the device bezels that typically have a lower IR reflectance than screens.

Hybrid Sensing: Fusing RGB and IR Detection

While the depth confidence image enables reliable tracking without strong interference from other light sources, Sauron's eye negatively affects the tracking quality in its centre. The closer screens get to it, the less visible they are in D1. For example, one smartphone from R1 becomes almost invisible in D1 with only parts of its bezels still visible. Another disadvantage of D1 is its low resolution of 320×240 or less compared to R1 with 1280×720. This reduces the accuracy of tracking positions and orientations. To compensate for these disadvantages, we employ a hybrid sensing approach that complements the results from depth confidence tracking with those of RGB tracking in R1. This RGB tracking method uses standard processing steps such as binary thresholding, low-pass filtering, Canny edge detection, and finding contours and convex hulls. R2 in Figure 4 shows the result with five of eight possible screens from the example RGB image R1.

It is important to notice the limitations of this simple RGB tracking: While it does not have a blind spot in the centre and has higher accuracy because of the higher resolution, the reflections from overhead light sources can easily deform device contours or cut through them, so that they are not recognized as screens anymore. Also, the devices' bezels must always have a colour that is clearly distinguishable from the table's surface. In summary, the RGB tracking is less reliable but more accurate and thus can serve to improve tracking whenever it can provide more accurate positions and orientations or the depth confidence tracking fails. Therefore RGB and depth tracking mutually complement each other. Consequentially, the merged result in H1 (Figure 4) contains all eight screens.

Identification of Displays, Display Size, and Orientation

After having determined the positions of rectangles that potentially are screens in the current frame, the vision processing associates them with the history of tracked rectangles from previous frames. This association is done by smoothing all previous positions of each rectangle using a Kalman filter and pairing these predicted points with the closest rectangle in the current frame. This enables the vision system to track the rectangles' movements over time, assign them with an internal ID, and look into their past.

However, the detected rectangles in Figure 4 are not necessarily actual screens. At this stage of the processing, the result contains regions that are only likely candidates for being a screen, but can also be false positives, e.g., the afore-mentioned dark non-glossy objects with low IR reflectance and a rectangular shape, e.g., tablet covers or protective pouches made from felt. To identify such false positives, we ask the screens to display fiducial markers and by this verify that they are actually interactive displays. To this end, the vision system broadcasts an identification request to all connected, but so far unidentified, devices. They react by displaying a fiducial marker containing a unique ID

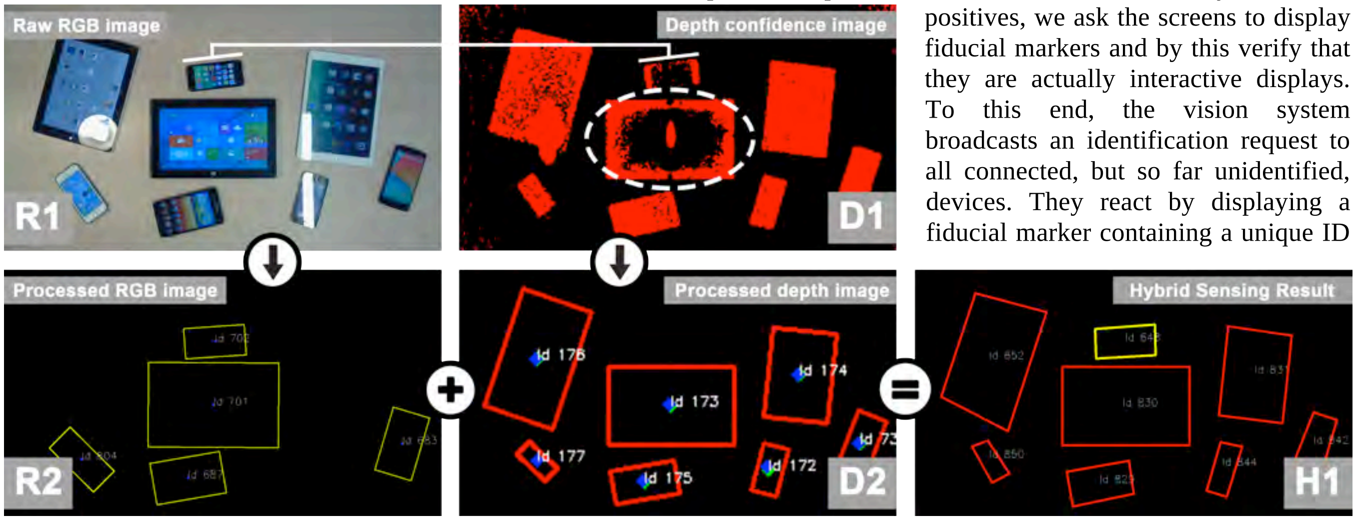

Figure 4. Hybrid Sensing. R1: RGB camera input. D1: Depth confidence image containing 'Sauron's eye' (dashed ellipse). R2: Mobile screens detected in R1. D2: Mobile screens detected in D1. H1: Merged result of hybrid sensing from R2 and D2.

on a white background that covers their screen (Figure 5). The vision system then looks for the presence of such a marker in the RGB image and decodes their value to verify that a new device has joined the huddle (and not a false positive) and to associate the corresponding region with the device ID. This process or pairing procedure usually takes only 1-2 seconds, and the fiducial markers disappear immediately once a device has been identified.

This identification process in the RGB image also enables us to determine the orientation and size of the screen in camera coordinates. The initial orientation is that of the found marker and the screen size is determined by the size of the white background surrounding it. This data is later used by the vision system to track the current orientation and also by our JavaScript API to compensate for different screen sizes, orientations, and resolutions (see below).

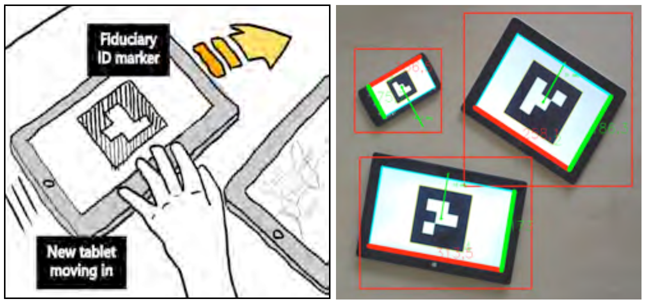

Figure 5. The vision system determines ID, orientation, and screen size using fiducial markers on a white background.

Hand Detection and Tracking

HuddleLamp's hand detection and tracking happens entirely in the depth image. Using background subtraction, depth thresholding, and flood fill segmentation, we identify contours in the depth image that could contain an arm and hand, based on the assumption that users always reach into the camera from outside (Figure 6). The hand position can then be approximated by the centroid of the hand's depth minima. Like for the screens, these positions are tracked over time by using a Kalman filter and a distance threshold.

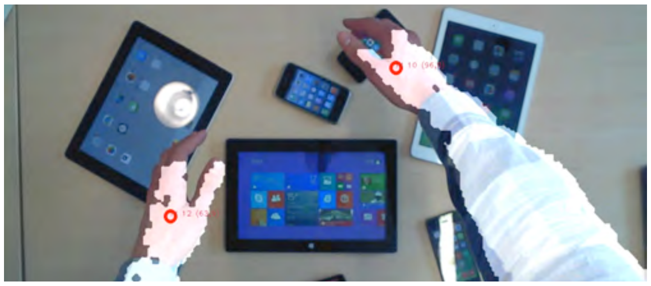
Figure 6. Segmentation of hands and arms (white overlays) and estimated location and height of hands (red circles).

Our hand tracking is not intended for providing a detailed representation of finger positions or detecting hand gestures, but to provide light-weight low-precision information about hands' locations and depths for cross-device or above-the-table interaction techniques. For more precise manipulations of on-screen objects, users can continue to use the low-latency and high-precision multi-touch detection of their capacitive touchscreens. For example, in the interaction techniques described below, we only used our hand tracking for low-precision tasks, such as using the hand position to determine the destination device for a cross-device 'pick, drag, and drop' gesture. The exact on-screen position for dropping the object is determined using the device's touchscreen and not the hand tracking.

Technical Evaluation of Hybrid Sensing

We conducted a controlled experiment to evaluate precision, accuracy, and reliability of hybrid sensing and RGB-only tracking. They are defined as follows: 1.) *precision* is the standard deviation of the tracked position and orientation of a fixed tablet over time and thus measures noise and jittering (units: mm or degree); 2.) *accuracy* is the spatial accuracy of a tablet's position compared to a ground truth (unit: mm); 3.) *reliability* is the percentage of frames in which a present tablet was tracked (its tracking state is 'true') during measurement (unit: %).

As apparatus we used a setup like in Figure 2 with a tracking region of 1020mm×570mm and a camera height of 780mm. The effective usable resolution for the RGB image was 1280×720 pixels and for the depth image 283×159 pixels. We used a black Apple iPad 2 as mobile device.

For the experiment, we studied seven different conditions divided into three categories: *Lighting*, *Occlusion*, and *RGB-only*. The *Lighting* and *Occlusion* conditions used hybrid sensing. In *RGB-only* hybrid sensing was turned off.

The *Lighting* conditions had different levels of illumination: 20 lux, 1600 lux, 2200 lux. For comparison: 400 lux is the recommended illumination for offices and classrooms and 1000 lux for hospital examination and treatment or for difficult industrial assembly. In the 2200 lux condition we simulated strong ambient light using two R7 halogen lamps (400W) with a peak emission in IR (approx. 800nm).

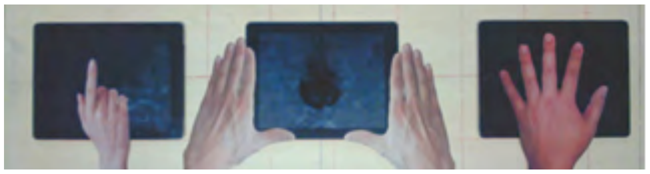
Figure 7. Picture taken from camera input stream shows the three levels of occlusion: *1 finger*, *2 hands*, and *1 hand*.

The illumination for the three *Occlusion* conditions was held constant at 1600 lux. The conditions simulated occlusion with *1 finger*, *1 hand*, and *2 hands* (Figure 7). The *RGB-only* condition was also at 1600 lux but without simulating any occlusion.

In each condition, a tablet was systematically moved to 21 different positions on a table where it was fixed for a frame-by-frame measurement of tracking state, position, and orientation for 10 seconds. The positions were defined by a 3x7 grid on the table that had a grid distance of exactly 100mm and was centred in the camera image. The grid distance served as ground truth for accuracy measurement.

Results

The results show sub-centimetre precision and accuracy for all conditions (Table 1). As expected, the best precision and accuracy was achieved for *RGB-only* due to the higher resolution of the RGB image but with a low reliability of only 89.5% due to the reflections of ceiling and ambient light sources. Under the same conditions, switching to hybrid sensing increased the reliability to 100% with only a small decline in precision and accuracy due to increased dependence from the lower resolution depth image when RGB tracking failed. Still, accuracy remained below 2.3mm and thus well inside the sub-centimetre range.

For the *1 hand* and *2 hands* occlusion, the reliability of hybrid sensing decreased to 89.8% since it sometimes failed to track occluded tablets close to Sauron's eye. In contrast, 20 lux or 2200 lux did not negatively affect the 100% reliability of hybrid sensing, but its accuracy and precision. Nonetheless, the worst accuracy was still below 3.5 mm and well inside the sub-centimetre range. As we discuss below, there is however an upper limit for ambient light.

Condition	**Precision** [in mm or degree]			**Reliab.** [in %]	**Accuracy** [in mm]			
	Mean (of all SD points)			**SD**	**Mean**		**SD**	
Lighting	**X**	**Y**	**Angle**		**X**	**Y**	**X**	**Y**
20 lux	1.22	1.26	.32	100.0	3.34	3.40	2.16	1.48
1600 lux	.78	1.05	.29	100.0	2.10	2.24	1.47	1.67
2200 lux	1.25	1.24	.37	100.0	3.29	2.80	2.25	1.41
Occlusion								
1 finger	.95	.74	.17	100.0	2.33	2.68	1.65	1.92
1 hand	1.08	1.02	.32	99.9	2.27	2.43	1.97	1.71
2 hands	1.36	1.81	.44	89.8	3.61	3.49	2.88	2.25
RGB-only								
1600 lux	.60	.63	.14	89.5	1.78	1.13	1.24	.70

Table 1. Precision and accuracy of HuddleLamp's hybrid sensing and RGB-only tracking.

Challenges and Limitations of Hybrid Sensing

The natural segmentation of mobile devices using their low IR reflectance worked robustly for different lighting conditions and we achieved to process the hybrid RGB and depth input at the maximum frame rate of the camera (30 fps) with an Intel Core i7 laptop. As we discuss below, there is still a perceptible latency on the UI, but it is not originating from hybrid sensing, but from web sockets and rendering on mobile devices which are slower than on PCs.

Since we use an overhead camera, users' arms and hands can sometimes occlude devices and deform device contours during touching or moving mobile screens. As the experiment shows, this can decrease reliability to 89.8%, so that devices and their ID are sometimes lost and a fiducial marker must be flashed again for device identification. This can interrupt the users' flow of interaction, especially when reflections of overhead light sources inhibit marker recognition and make it necessary to move the device in a reflection-free area. We could potentially improve this by using alternative means of optical device identification, e.g., flashing full screen colour sequences as in [30].

Finally, there is an unavoidable upper limit of ambient light for all consumer TOF cameras with an integrating CMOS detector. They stop working as soon as the IR light that is reflected into the detector from the sun or other light sources becomes many times stronger than the modulated camera signal. Therefore HuddleLamp's depth confidence tracking does not work outdoors on bright days or if sunlight shines directly on an indoor table. However, this upper limit was not reached even in our 2200 lux condition.

HUDDLELAMP'S JAVASCRIPT API

For enabling truly ad-hoc multi-device collaboration, it is necessary to provide an API that lets devices of all operating systems easily join the devices on the table without prior installation of native apps or applications. HuddleLamp achieves this by providing an API for writing collaborative applications with HTML5, CSS3, and JavaScript, so that the application becomes simply a web page that can be opened in every device's browser. As soon as this web page is loaded, users can put their device on the table into the camera's view and by doing so add that device to the huddle. For the users' convenience, a QR code with that URL can be attached to the lamp or desk. After joining a huddle, the web-based application can access the JavaScript API to make optional use of three key features:

1) It provides a web socket connection from the application to the vision server that returns a JSON data stream with device and hand positions, device orientations, and device occlusion states. This data is also provided as events using the JavaScript observer pattern (Listing 1).

```
var huddle = Huddle.client()
  .on("devicefound", function() {
    console.log("devicefound");
  })
  .on("devicelost", function() {
    console.log("devicelost");
  })
  .on("proximity", function(data) {
    console.log(data);
  })
  .connect(host, port);
```

Listing 1. HuddleLamp's JavaScript API to access the data stream from the vision server.

2) The API provides a shared virtual workspace that can be accessed from all mobile devices. This workspace contains objects that can be arranged in space using the well-known multi-touch user manipulations from tabletops, e.g., drag, pinch-to-zoom-and-rotate, flick. However, in contrast to tabletop SDKs (e.g. the ScatterView control of the Microsoft Surface SDK), it is synchronized for all connected devices via a server. We implemented a shared object storage on top of the Meteor web framework, so that all manipulations on one device become instantly visible on all other devices to enable collaborative cross-device work.

3) The API enables a homogeneous rendering of the workspace across different devices with respect to their different locations, screen sizes, and resolutions. Each device can become part of a multi-device display that renders the virtual workspace on the individual screens as if the workspace was physically situated in the tracked region

on the table. Only that part of a workspace (e.g. a map) and its objects (e.g. images, videos) is rendered that lies underneath the device, correctly preserving absolute positions and orientations. The API achieves this by translating, rotating, and scaling the local rendering of the workspace based on the devices' individual screen size, position, orientation, and aspect ratio that were determined by hybrid sensing. There is no need to create a database of devices with their screen sizes. The API thus enables a fully interactive and reconfigurable Junkyard Jumbotron [2].

Validation of API with Example Interaction Techniques

To validate the design of our API, we implemented five examples of existing and novel cross-device gestures and interaction techniques that used to be possible only in instrumented rooms, as part of fixed installations or exhibits, or above instrumented surfaces. These examples serve as validation, but they also demonstrate the possible design space for future HuddleLamp applications.

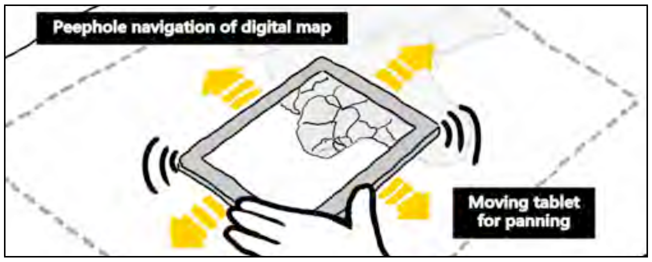

Figure 8. Peephole navigation with HuddleLamp: physically navigating a large virtual map with a tablet.

1.) Peephole Navigation

Peephole navigation is a classic technique for navigating large information spaces such as maps using spatially-aware displays. A mobile device acts as a lens [32] or peephole [5] to physically navigate virtual information as if it was situated in physical space, e.g. by moving or rotating a device to control what is shown on the screen. HuddleLamp's API with a spatially-situated workspace enables peephole navigation with one or more mobile displays of different sizes. For example, we built a demonstrator in which users can move one or more tablets to physically navigate in a virtual world map (Figure 8).

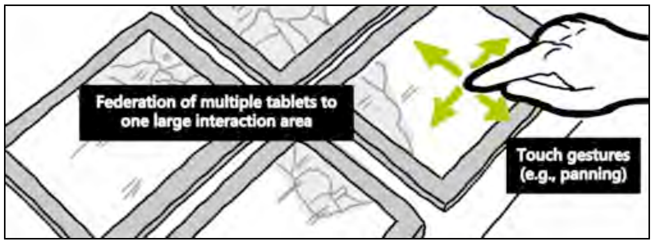
Federation of multiple tablets to one large interaction area
Touch gestures (e.g., panning)

Figure 9. An illustration of synchronous huddle navigation.

2.) Huddle Navigation

We extended peephole navigation to the novel technique of 'huddle navigation'. It enables users to create large multi-device displays that are similar to interactive tabletops simply by moving multiple tablets or smartphones side-by-side (Figure 9). By this, users let devices join into a 'huddle'. Users can zoom, rotate, and pan all screens in the huddle synchronously with multi-touch. They can also create two or more huddles based on device proximity. Each huddle then zooms, rotates, and pans independently to support collaboration of multiple users or user groups.

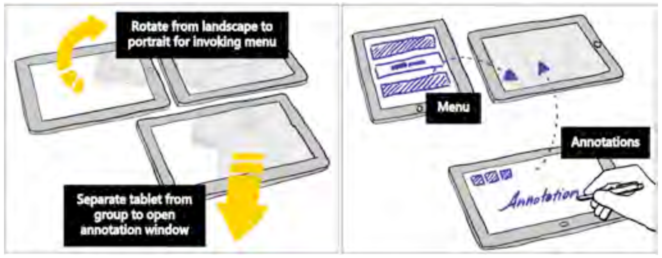
Rotate from landscape to portrait for invoking menu
Separate tablet from group to open annotation window
Menu
Annotations
Annotation

Figure 10. Spatially-aware menus and modes change the role of devices based on their orientation or distance.

3.) Spatially-Aware Menus and Modes

In the previous example, a tablet that was moved close to a map huddle automatically turned into an extension of the map. But we also used proximity and orientations for more complex behaviour such as 'spatially-aware menus and modes'. For example, when a user rotates a tablet in a huddle from landscape to portrait orientation, this reveals a tool palette or menu where display parameters for the entire huddle can be altered, e.g., visible layers or data points of a map. When moving the tablet away from the huddle and closer to the user, the tablet automatically switches into notetaking mode and users can use a stylus to take personal notes or for annotating content (Figure 10). With a good mapping of spatial relations to switching menus or modes, a HuddleLamp application can achieve a seemingly intelligent spatially-aware behaviour that proactively supports the users during collaboration. This also enables smooth transitions between tightly-coupled collaboration (tablet is shared in the huddle) and loosely-coupled parallel work (tablet is picked up and used as personal display).

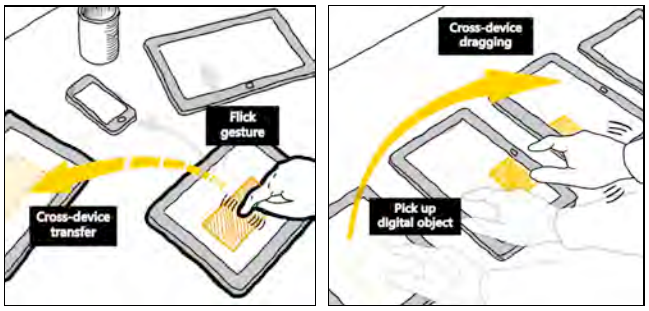

Figure 11. Cross-device flicking and Touch&Flick Browsing.

4.) Cross-Device Flicking and Touch&Flick Browsing

Cross-device flicking is an interaction technique to enable users to flick objects with their fingers (Figure 11 left). A simple physics simulation makes objects accelerate and stop at new locations in the workspace. Objects can stop on the same screen, but also fly to the screen of another device.

A variation of cross-device flicking is 'touch-and-flick browsing'. We implemented this technique for web browsing with multiple tablets or smartphones. Content items from a web page such as a link, video, or image can be opened on another screen simply by touching and flicking them towards a neighbouring device. The content is then shown on the destination device using the entire available screen space, similar to how links can be opened in new tabs or new windows in desktop browsers.

5.) Pick, Drag, and Drop

A different way of moving objects between devices is the 'pick, drag, and drop' hand gesture. An example of such an above-the-table object manipulation technique that uses 3D graphics and physics simulations has been discussed in [9]. It has inspired us to implement our own 'above-the-tablets' technique, but with using a simpler 2D implementation. The gesture is initiated by tapping an object on a source mobile screen to pick it up and then lifting the hand and moving it above the desk to drag it to another destination device (Figure 11 right). During 'pick, drag, and drop', HuddleLamp continuously tracks the hand position and the picked object travels with the hand over the surface and across the screens lying beneath it. To drop the object, users move their hand above the destination device and then tap its touchscreen at the destination location. Inspired by Hilliges et al. [9], the current height of the hand above the surface is also used to increase the rendered size of the object to create a simple illusion of 3D movement in space.

Challenges and Limitations of the HuddleLamp API

During implementation of the examples, we identified two limitations of the API that we will address in future:

First, there is a small but noticeable delay between the physical movement of a screen and the corresponding reaction of the UI. While the vision processing works at the maximum frame rate of the camera (30 fps), the web socket connection, the synchronization of devices with Meteor, and particularly the different rendering performances of browsers and devices induce a noticeable latency. Best results were achieved using Safari Mobile on Apple iPad 2/3/Air, Apple iPhone 4/5S, and Microsoft Surface 2 Pro with Chrome. Less successful were tests with mobile devices of older generations (e.g. Apple iPad 1, Apple iPhone 3G) and surprisingly with Google Nexus 5&10.

Second, the recognition of multi-touch gestures such as pinch-to-zoom is limited when they span device boundaries. Currently, gestures are only recognized if all fingers are on a single device. They cannot be performed by putting one finger on one device and other fingers on a different device. Nonetheless, for each individual device they are correctly recognized and processed and their results are immediately synchronized with all other devices.

DISCUSSION AND FUTURE WORK

There are also more higher-level questions raised by HuddleLamp which lead us to directions for future work:

A clear limitation of HuddleLamp compared to traditional tabletops is the absence of a large screen that also displays touch-enabled interactive content around the mobile screens on a table. Does this inhibit its usefulness in real-world scenarios? This disadvantage has to be weighed against HuddleLamp's advantages in terms of low-cost portability, ad-hoc interaction, and distributed rendering. Furthermore, it supports high-resolution output and responsive input on a capacitive multi-touch screen, sometimes with a pressure-sensitive stylus (e.g. on a MS Surface tablet). This is often not the case for large tabletops with optical touch detection.

Furthermore, although we already realized a series of spatially-aware cross-device interaction techniques, we have only scratched the surface of what can be done in future with ad-hoc communities of spatially-aware and reconfigurable displays and devices. We consider HuddleLamp only as the first step towards realising this greater vision. One of the great advantages that we see in HuddleLamp or similar approaches is that they make digital tools for co-located collaboration available to the masses. They can be used where motion-capturing tracking systems and large tabletops or surfaces are too expensive to buy, setup, and maintain. Consequentially, we are now working on applications for engaging a great variety of users, e.g., playful exploration of urban data in public libraries or community centres, collaborative search tools for schools, or novel kinds of informational games in museums.

CONCLUSION

We have described HuddleLamp, a sensing system in the form of a desk lamp with an integrated RGB-D camera that tracks the movements of multiple mobile displays on a table for around-the-table collaboration. We described our implementation and our approach of hybrid sensing, i.e., detecting mobile screens by exploiting their optical properties in the IR range and additionally using RGB images and fiducial markers to better track screens and to distinguish them from other objects or users' hands. After a technical evaluation of our hybrid sensing approach, we also introduced HuddleLamp's web-based architecture and JavaScript API for ad-hoc collaboration that enables users to add or remove displays and reconfigure them in space at any time without installing any software. We discussed how this can be used to create large multi-device tiled displays for multi-user and multi-touch interaction. Beyond our five demonstrated examples of different interaction techniques, we believe that HuddleLamp's setup and hybrid sensing tracking will allow the rapid exploration of future cross-device interactions supporting group collaborations.

ACKNOWLEDGMENTS

This work was conceived of at Dagstuhl Seminar 13452. We would like to thank Katherine Isbister (and others of the Game Innovation Lab at NYU) and Ashley Karr for their valuable comments. We also thank Mario Schreiner for his JavaScript implementation of Touch&Flick Browsing and Oscar Robinson and Jonny Manfield for parts of the Huddle JavaScript API. Finally, we thank Francesco de Gioia for his thoughts and work on our tracking algorithms.

REFERENCES

1. Ballendat, T., Marquardt, N., and Greenberg, S. Proxemic Interaction: Designing for a Proximity and Orientation-Aware Environment. Proc. of ITS '10, ACM (2010).
2. Borovoy, R. and Knep, B. Junkyard Jumbotron, http://jumbotron.media.mit.edu. 2011.
3. Dearman, D., Guy, R., and Truong, K. Determining the Orientation of Proximate Mobile Devices Using Their Back Facing Camera. Proc. of CHI '12, ACM (2012), 2231–2234.
4. Dippon, A., Wiedermann, N., and Klinker, G. Seamless Integration of Mobile Devices into Interactive Surface Environments. Proc. of ITS '12, ACM (2012), 331–334.
5. Fitzmaurice, G.W. Situated information spaces and spatially aware palmtop computers. Comm. of ACM 36, 7 (1993).
6. Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., and Wang, M. Proxemic Interactions: The New Ubicomp? ACM Interactions 18, 1 (2011), 42–50.
7. Hamilton, P. and Wigdor, D.J. Conductor: Enabling and Understanding Cross-device Interaction. Proc. of CHI '14, ACM (2014), 2773–2782.
8. Harrison, C., Benko, H., and Wilson, A.D. OmniTouch: wearable multitouch interaction everywhere. Proc. of UIST '11, ACM (2011), 441–450.
9. Hilliges, O., et al. Interactions in the air: adding further depth to interactive tabletops. Proc. of UIST '09, ACM (2009).
10. Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., and Smith, M. Stitching: pen gestures that span multiple displays. Proc. of AVI '04, ACM (2004), 23–31.
11. Hollan, J., Hutchins, E., and Kirsh, D. Distributed Cognition: Toward a New Foundation for Human-computer Interaction Research. ACM Trans. Comput.-Hum. Interact. 7, 2 (2000).
12. Holmquist, L., et al. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. Proc. of UbiComp, Springer (2001), 116.
13. Huang, D.-Y., Lin, C.-P., Hung, Y.-P., et al. MagMobile: Enhancing Social Interactions with Rapid View-stitching Games of Mobile Devices. Proc. of MUM '12, ACM (2012).
14. Kendon, A. Conducting Interaction: Patterns of Behavior in Focused Encounters. Cambridge Press, Cambridge, 1990.
15. Kray, C., et al. Group Coordination and Negotiation through Spatial Proximity Regions around Mobile Devices on Augmented Tabletops. Proc. of TABLETOP, IEEE (2008).
16. Li, M. and Kobbelt, L. Dynamic Tiling Display: Building an Interactive Display Surface Using Multiple Mobile Devices. Proc. of MUM '12, ACM (2012), 24:1–24:4.
17. Linder, N. and Maes, P. The Design Evolution of LuminAR: A Compact and Kinetic Projected Augmented Reality Interface. CHI ’12 Ext. Abs., ACM (2012), 1435–1436.
18. Lucero, A., Holopainen, J., and Jokela, T. Pass-them-around: Collaborative Use of Mobile Phones for Photo Sharing. Proc. of CHI '11, ACM (2011), 1787–1796.
19. Lucero, A., et al. Mobile Collocated Interactions: Taking an Offline Break Together. interactions 20, 2 (2013), 26–32.
20. Marquardt, N., et al. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. Proc. of UIST '11, ACM (2011).
21. Marquardt, N., Hinckley, K., and Greenberg, S. Cross-Device Interaction via Micro-mobility and F-formations. Proc. of UIST '12, ACM (2012), 13–22.
22. Marquardt, N., Jota, R., Greenberg, S., and Jorge, J.A. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. Proc. of INTERACT, Springer (2011), 461–476.
23. Merrill, D., Kalanithi, J., and Maes, P. Siftables: towards sensor network user interfaces. Proc. of TEI, ACM (2007).
24. Ohta, T. and Tanaka, J. Pinch: An Interface That Relates Applications on Multiple Touch-screen by Pinching Gesture. Proc. of ACE '12, Springer-Verlag (2012), 320–335.
25. Oulasvirta, A. When Users “Do” the Ubicomp. interactions 15, 2 (2008), 6–9.
26. Piazza, T., et al. Dynamic Duo: Exploring Phone-Tablet, Combinations for Mobile Usage. WIP, TEI '13, ACM (2013).
27. Ramos, G., et al. Synchronous Gestures in Multi-Display Environments. Human-Computer Interaction 24, 1 (2009).
28. Rekimoto, J. and Saitoh, M. Augmented surfaces: a spatially continuous work space for hybrid computing environments. Proc. of CHI '99, ACM (1999), 378–385.
29. Schmitz, A., Li, M., Schönefeld, V., and Kobbelt, L. Ad-Hoc Multi-Displays for Mobile Interactive Applications. The Eurographics Association (2010), 45–52.
30. Schwarz, J., et al. Phone As a Pixel: Enabling Ad-hoc, Large-scale Displays Using Mobile Devices. Proc. of CHI '12, ACM (2012).
31. Spindler, M., et al. Tangible Displays for the Masses: Spatial Interaction with Handheld Displays by Using Consumer Depth Cameras. Personal Ubiquitous Comput. 18, 5 (2014).
32. Spindler, M., Stellmach, S., and Dachselt, R. PaperLens: advanced magic lens interaction above the tabletop. Proc. of ITS '09, ACM (2009), 69–76.
33. Steimle, J., Jordt, A., and Maes, P. Flexpad: Highly Flexible Bending Interactions for Projected Handheld Displays. Proc. of CHI '13, ACM (2013), 237–246.
34. Tandler, P., et al. Connectables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. Proc. of UIST '01, ACM (2001), 11–20.
35. Wang, M., Boring, S., and Greenberg, S. A Public Advertising Display that Captures and Preserves the Attention of a Passerby. Proc. of PerDis '12, ACM (2012).
36. Wilson, A., et al. Steerable Augmented Reality with the Beamatron. Proc. of UIST '12, ACM (2012), 413–422.
37. Wilson, A.D. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. Proc. of UIST '10, ACM (2010), 273–282.
38. Winkler, C., et al. SurfacePhone: A Mobile Projection Device for Single- and Multiuser Everywhere Tabletop Interaction. Proc. of CHI '14, ACM (2014), 3513–3522.