# SoD-Toolkit: A Toolkit for Interactively Prototyping and Developing Multi-Sensor, Multi-Device Environments

**Teddy Seyed, Alaa Azazi, Edwin Chan, Yuxi Wang, Frank Maurer**
University of Calgary, Department of Computer Science
2500 University Drive NW, Calgary, Alberta, T2N 1N4
{teddy.seyed, alaa.azazi, edwin.chan, yuxwang, frank.maurer}@ucalgary.ca

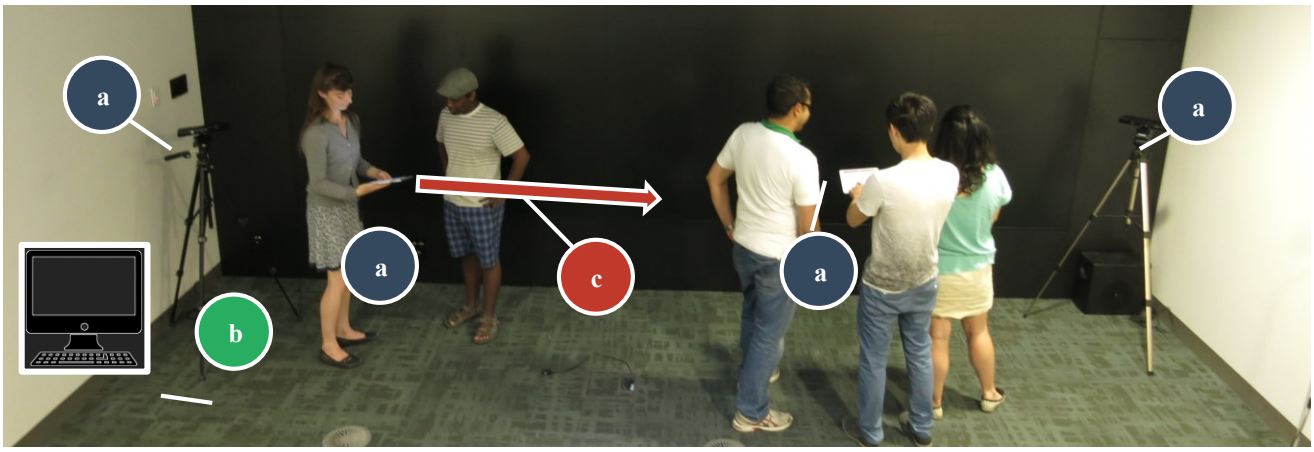


**Figure 1. The *SoD-Toolkit* consists of (a) numerous devices and sensors that are supported by (b) software tools and components providing information that facilitates (c) spatial interactions between devices and the environment.**

## ABSTRACT
As ubiquitous environments become increasingly commonplace with newer sensors and forms of computing devices (e.g. wearables, digital tabletops), researchers have continued to design and implement novel interaction possibilities. However, as the number of sensors and devices continues to rise, researchers still face numerous instrumentation, implementation and cost barriers before being able to take advantage of the additional capabilities. In this paper, we present the *SoD-Toolkit* – a toolkit that facilitates the exploration and development of multi-device interactions, applications and ubiquitous environments by using combinations of low-cost sensors to provide spatial-awareness. The toolkit offers three main features. (1) A "plug and play" architecture for seamless multi-sensor integration, allowing for novel explorations and ad-hoc setups of ubiquitous environments. (2) Client libraries that integrate natively with several major device and UI platforms. (3) Unique tools that allow designers to prototype interactions and ubiquitous environments without a need for people, sensors, rooms or devices. We demonstrate and reflect on real-world case-studies from industry-based collaborations that influenced the design of our toolkit, as well as discuss advantages and limitations of our toolkit.


ITS 2015, November 15–18, 2015, Funchal/Madeira, Portugal.

http://dx.doi.org/10.1145/2817721.2817750

### Author Keywords
Sensors; Toolkit; Kinect; Cross-device interactions; Gestural Interaction; Multi-Surface; Prototyping.

### ACM Classification Keywords
H.5.2. Information Interfaces. User Interfaces – input devices and strategies, prototyping.

## INTRODUCTION
The consumer space of computing technologies is experiencing a dramatic explosion of different size and form factors for devices (e.g. wearables, multi-touch wall displays). The capabilities of these devices can be expanded significantly when used collectively with other devices and sensors, effectively creating multi-device ubiquitous environments. Ubiquitous environments provide people with access to their information across many of their devices, with some devices being spatially aware of other devices in the environment. Spatial awareness in ubiquitous environments is directly linked to the sensors (either dedicated or device-based) and further magnifies existing challenges as to how information and tasks can be performed effectively across

different types of devices, within varying degrees of spatial awareness [32].

Research in ubiquitous environments primarily focuses on novel interaction forms between people and a set of devices [41] (e.g. digital tabletops, mobile devices, smart watches), with proxemics being a common method of conceptualizing the interaction space [1, 8]. As different form factors for devices becomes commonplace and the capabilities of sensors increase, novel forms of explorations between different devices (e.g. a Google Glass and a Smart Watch) is still fairly limited in the context of spatially-aware ubiquitous environments. Furthermore, exploration into real-world scenarios also faces limitations, much of which arises from the difficulty in building multi-device, spatially-aware environments, as many existing development kits are limited in support for different multi-sensor configurations, multi-device platforms and cross-connectivity and typically require complex software and hardware setups [13].

To bridge the gap and allow for richer explorations into spatially-aware ubiquitous environments, we introduce the Society of Devices (SoD) Toolkit (or *SoD-Toolkit)*, a toolkit that facilitates exploring and developing multi-device applications and interactions in spatially-aware ubiquitous environments (Figure 1). Overall, our primary research goal was *to allow for novel explorations of different types of multi-device, spatially-aware (through multi-sensor fusion) ubiquitous environments that can be augmented with a multitude of newer sensors and device platforms*. To address this goal, our toolkit abstracts sensor information from a multitude of sensors into a "plug and play" architecture, allowing researchers and developers to seamlessly fuse sensor information and utilize commercially available off-the-shelf tracking technologies to provide spatial awareness in ubiquitous environments. Researchers and developers can also create additional modules for future sensors through the modular architecture of the toolkit. The toolkit also provides a number of client libraries that are built upon existing operating systems and platforms which include iOS, Android, Windows, as well as web-based technologies such as HTML5, Node.js and Javascript. This allows for a wide range of skill sets and experience, letting researchers and developers implement and design interactions and ubiquitous environments in languages they are comfortable with, reducing many common platform, language, technology and device barriers [28]. In addition, the toolkit offers tools for researchers and developers to visualize and prototype interactions within varying levels of spatially-aware ubiquitous environments (due to limited hardware availability) or without the need of specialized hardware entirely. This reduces a significant hardware and cost barrier for researchers and developers, and can assist in more widespread research and application development for ubiquitous environments in both the research and consumer space.

The remainder of this paper, is organized as follows. We first review related work and then introduce the design of the *SoD-Toolkit*, including its key features, architecture and components.

Next, we describe two real-world domain-specific case studies that demonstrate the flexibility of the toolkit, how they influenced the design rationale of the toolkit and lessons we learned. This is followed by a discussion and reflection on the design and features of the toolkit compared to other approaches. Finally, we conclude this paper describing limitations and future work.

## RELATED WORK

*SoD-Toolkit* is inspired by Weiser's vision of ubiquitous computing [41] and built upon prior work in three areas of research: (1) Proxemics and Ambient Interactions, (2) Multi-Surface Interactions and (3) Application Programming Interfaces (APIs) and Toolkit designs.

### Proxemics and Ambient Interactions

The research space of proxemic interactions is extremely rich and well explored. Greenberg et al. conceptualized proxemics in the context of ubiquitous environments with a number of investigations that focused on spatial relationships between users and objects, specifically applying five proxemic dimensions: orientation, distance, motion, identity and location [8]. The intention of much of the research in this space is to "leverage people's natural understanding of their proxemic relationships to manage the entities that surround them" [23]. Applying these proxemic theories to sensors in ubiquitous environments was further explored by Ballendat et al, who used sensors to detect people and their devices, and better understand different types of interactions (i.e. implicit and explicit) [1]. Follow-up work by Marquardt et al. examined the relationships between people in closer spaces through F-formations and micro-mobility and sociological constructs [25].

Marquardt and Greenberg also identified major challenges for proxemics, which included providing meaningful feedback, managing privacy and security and establishing connections between different types of devices [22]. Establishing connections with different types of devices was the focus of early work in proxemics by Vogel and Balakrishnan, who explored proxemics in relation to public ambient displays [38]. They defined four discrete areas in front of devices, similar to Edward Hall's proxemic zones surrounding a person [7]. In more recent years, research focused on devices such as whiteboards [15, 31] and other forms of displays [35].

While a majority of proxemics research has been heavily device-centric in relatively small or enclosed spaces, proxemics in larger more ambient spaces (and not around smaller displays) has been less researched. Active Badge by Want et al., is an early example of exploring larger scale proxemics [40]. Their system uses a beacon sensor to track position of users in a work environment. UbER Badge by

Laibowitz et al is another example, whose system uses proxemic badges [19]. The badges are used to facilitate social interaction in various large meetings. In the context of a ubiquitous home, the EasyLiving project explored multi-room proxemics by combining a number of technologies to track people and their devices on a larger scale [2]. Research into proxemic interactions in larger and more ubiquitous spaces is directly tied to different types of tracking technologies, a large research area within ubiquitous environments itself. *SoD-Toolkit* builds upon many of the concepts in proxemics and ambient spaces with a goal of providing lightweight tools and libraries for developing and exploring interactions and applications in larger scale ubiquitous environments and domains, without researchers and developers being restricted in room size or choice of sensors.

### Multi-Device Interactions

Multi-device interactions with a number of different device configurations is a rapidly growing research area, particularly with newer form factors of devices increasing in the consumer market (e.g. wearables). A major task in multi-device interactions involves the movement of information or content from one device to another [32]. Much of the early work in multi-device interactions is based upon Rekimoto's Pick and Drop, where pen input is synchronized across multiple computers, allowing a user direction manipulation of content between screens [30]. Hinckley explored the notion of bumping tablets and stitching tablets together (via pen stroke across displays) as multi-device interactions for transferring content [11, 2]. Lucero et al followed a similar approach, but instead used a pinching gesture across multiple mobile devices [21]. In these types of multi-device interactions, input is typically *synchronized* to facilitate smooth interaction and information transfer [3].

Multi-device interactions can also be impacted by *spatial awareness* and proxemic relationships [3]. Numerous sensors are employed to provide positioning and tracking of devices (and users) to take advantage of spatial relationships. Kortuem et al. used this approach when creating novel spatial widgets for user interfaces [17]. Kray et al., used a digital tabletop as a center of mediation for proxemic relationships between mobile devices [18]. More general work in these types of interactions were explored by Marquardt et al with the gradual engagement pattern, that maps device-to-device proximity as a function of different levels of information exchange [22]. LightSpace by Wilson and Benko [42], uses spatial awareness for interactions between and on physical surfaces.

Dividing information and interfaces across multiple devices results in interactions that are *distributed* [3]. The iLand system by Streitz et al [37] is an early example of interactions distributed in a ubiquitous environment. Roomware is another example that inter-connects smart artifacts in a room, to augment both individual and collaborative tasks [36]. More recently, interactions between newer forms of devices such as wearables has begun to appear in the research literature. Chen et al., explored interaction techniques and gestures for distributed interactions between a watch and smartphone [3]. In a similar fashion, Mayer et al. [26] explored head-mounted display to interact with objects within view of a user. Overall however, distributed interaction techniques for wearables and newer forms of devices is still extremely underexplored [3, 13, 39, 44].

Much of this work indicates the potential interaction techniques for ubiquitous environments, however a significant amount of implementation work is repeated for interactions that are *synchronized*, *spatially-aware* or *distributed*. *SoD-Toolkit* easily allows for the exploration of these types of interactions (or combinations thereof) and can facilitate researchers in exploring unconventional and yet-to-be explored multi-device spatially-aware interactions (e.g. between a head-mounted display and a smart watch).

### Application Programming Interfaces and Toolkits

A significant amount of recent research has explored creating multi-device toolkits, primarily designed to overcome different aspects of the engineering challenges that come with building ubiquitous environments. Toolkits, such as Conductor [9] and Panelrama [43] and others [4, 16, 33], focus specifically on multi-device application development through web-based interfaces. XDStudio also supports cross-device application development, through the use of a GUI builder [27].

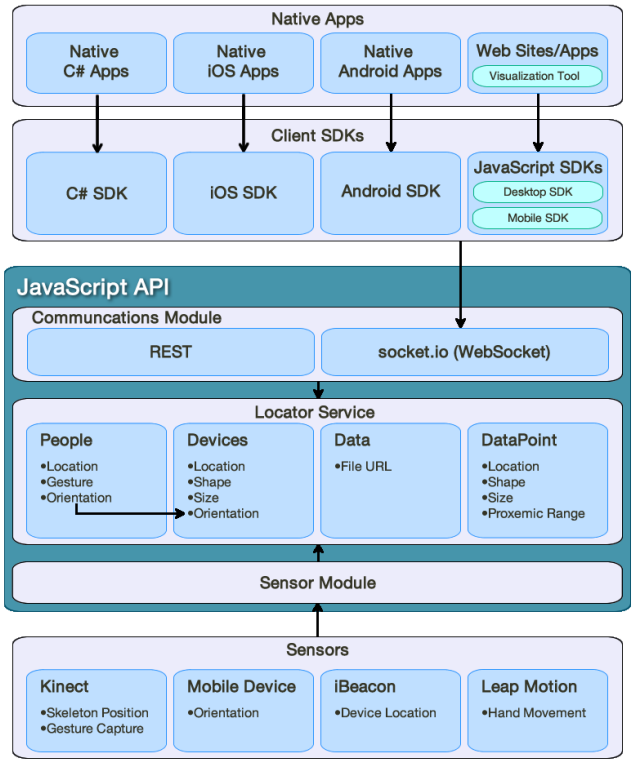


**Figure 2. An overview of the architecture of the *SoD-Toolkit***

The Proximity Toolkit is the canonical example of a toolkit focused on larger ubiquitous spaces [23]. The toolkit gathers data from various tracking sensors to allow for sensor fusion, and provides an easily accessible API. It also allows for interactions to be observed via a visual tool. A major challenge with the toolkit however, is its reliance on high-end tracking systems [27] that are ideal for prototyping but not feasible in real-world deployments [28] and its limited support for additional sensors. This inspired further work, such as XDKinect [28] which uses a single Kinect sensor to mediate interaction between different devices and also allows proxemic interaction and multi-modal input. In toolkits such as these, enabling multi-device interactions in ubiquitous environment requires knowledge about presence and position of devices, typically provided by a variety of sensors. Numerous approaches have been taken in the research literature, such as magnet based tracking [14], radio tracking [21] and sensor fusion approaches [23]. Much of the work in prototyping ubiquitous environments also faces issues of cost (for sensors and devices) and room size, making it difficult for researchers to properly explore novel types of ubiquitous environments and multi-device interactions.

Implementing ubiquitous environments that are deployable in real-world environments still requires substantial engineering efforts for researchers and developers. It is still possible to build applications for different devices, with different sensors in ubiquitous environments, but the effort required prevents richer explorations [29]. In contrast, *SoD-Toolkit* provides support for exploring multi-sensor, multi-device ubiquitous environments by providing (1) a "plug and play" architecture multi-sensor fusion, (2) native client libraries for major device, sensor and UI platforms and (3) tools that allow rapid prototyping without the need for people, sensors, rooms or devices.

## SOD-TOOLKIT

In this section, we provide an overview of the *SoD-Toolkit* architectural components, its visualization and prototyping tool, as well as its multi-sensor fusion approach to create larger ubiquitous environments. To facilitate real-world deployments and novel explorations of ubiquitous environments, we primarily focused on off-the-shelf sensor hardware and common devices, as well as free and open source software. Our source code is also freely available to download as open source[1].

### Architecture

The software architecture of *SoD-Toolkit* is composed of several components (Figure 2) but primarily, we discuss: (1) client libraries, (2) the locator service, and (3) the central server and communications module.

#### *Client SDK Libraries*

The client SDK libraries have two primary functions: (1) they provide necessary information for spatial awareness in the environment, depending on the sensor or device and (2) provide a platform to be built upon for native application development. Within each client library, a sensor module captures and sends data to the locator service frequently (discussed in the next section) and the data captured varies, depending on device or sensor type. The libraries support:

1. **Various form factors of devices** such as digital tabletops, wall displays, smart watches, head mounted displays and mobile devices (regardless of implementation platform). If a device has built in sensors (e.g. accelerometers, gyroscopes), this information is sent over the network to the locator service, which is discussed in the next section. Due to the modular nature of the architecture, spatial information from a device is easily fused with other supported sensors in the environment. For example, we allow for the position and orientation of a device to be determined by fusing skeletal information of a user (from a Microsoft Kinect) with orientation of a device (from gyroscopes and accelerometers). For devices that are stationary (e.g. a wall display), we allow researchers and developers to set the physical location in space, through the visualization tools we provide, as we discuss later.
2. **JS web client** that is both platform and browser agnostic, deriving information from a device if it has sensors available. This client is similar to the platform specific device clients, however, this allows for support of web-based multi-sensor, multi-device ubiquitous environments, which is not common in many toolkits.
3. **The Microsoft Kinect** provides skeletal, position, identity and gestural information through its available skeletal stream. The Kinect client library supports a single Microsoft Kinect (version 1 or 2) and sends information over the network to the locator service at a rate of 30 skeleton frames per second. A single Kinect sensor has a tracking range from 1.2 to 4.5 meters, which creates difficulties when trying to create larger ubiquitous environments. We address this by allowing multiple Kinects to track users and use sensor fusion to expand the tracking area, create novel tracking areas, as well as improve tracking accuracy. We discuss this sensor fusion in the Sensor Fusion approach section. Additionally, our Kinect client currently supports the "grab" and "release" gestures.
4. **The Leap Motion** provides detailed finger tracking of a user. Overall the sensor provides a more fine grained level of tracking that typically needs to be paired with a sensor that covers a larger area (e.g. Microsoft Kinect) to provide meaningful interactions. For example, if a user is further away from a Kinect and close to a wall display that isn't touch enabled, but is connected to a Leap motion, the locator service interprets the position of the user and

[1] *SoD Toolkit* - http://sodtoolkit.com/

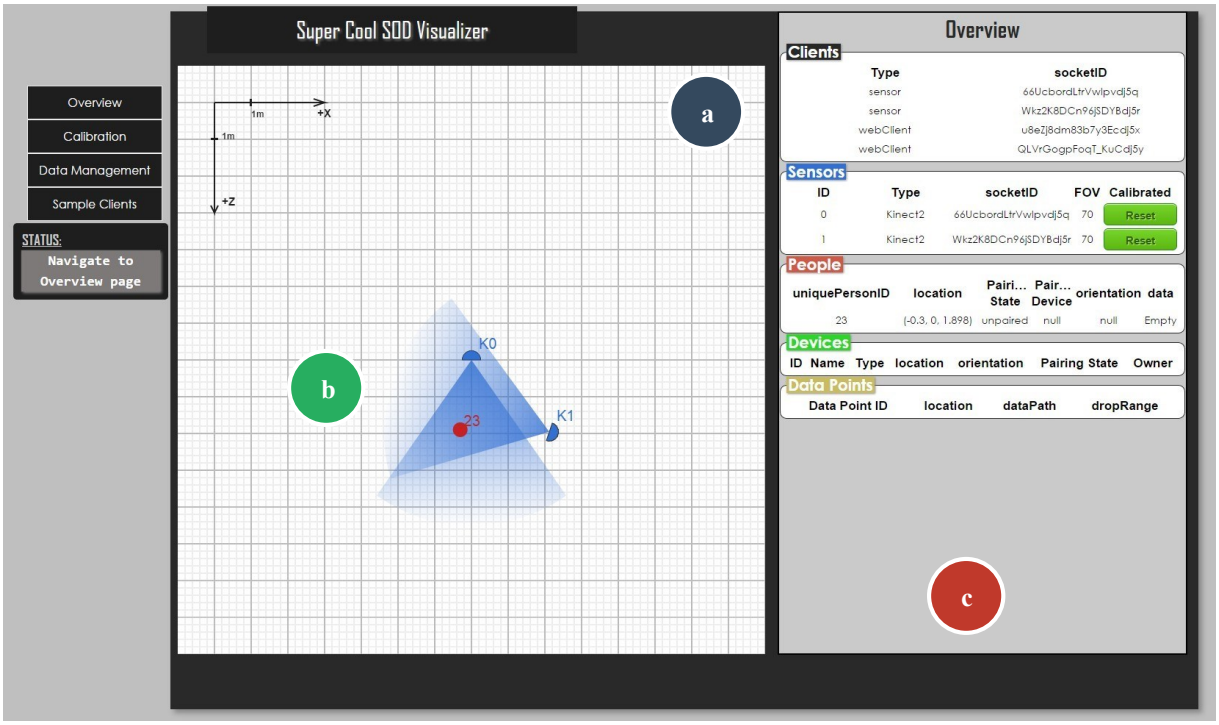


**Figure 3. The visualization and prototyping tool for the *SoD-Toolkit*. (a) The tracked environment, (b) Entities tracked in the environment, (c) List of available entities and their current state, as well as clients currently connected in the environment.**

prioritizes sensor information from the Leap motion over potentially inconsistent Kinect skeletal data, particularly for skeleton and joint data.

5. **Apple's iBeacon** which provide coarse grained positional information. The sensor provides position information of a device (either Android or iOS) or person (a person must have an iBeacon tag) in the form of close, near or far. While the sensor doesn't provide as accurate tracking as a Kinect sensor, it can facilitate a larger ubiquitous space and server as a mediator, transitioning between high and low spatially-aware regions.

All clients for sensors and devices support native development platforms which include Windows (C#), iOS (Objective-C and Swift), Android (Java) and HTML5/JS, as well as support for popular IDEs (Microsoft's Visual Studio, Android Studio and Apple's Xcode), significantly reducing start-up effort for researchers and developers. For other programming languages, environments, developers are able to write wrappers over the libraries and APIs we provide. We also provide a set of example applications using different sensors and devices to illustrate code required to implement novel spatially-aware multi-device interactions (e.g. "flick" to device) or multi-sensor ubiquitous environment.

### *Locator Service*

The Locator service is the hub that amalgamates spatial information from different sensors and devices, allowing the *SoD-Toolkit* to build multi-sensor spatially-aware ubiquitous environments. Information that is tracked includes different types of entities, such as sensors, devices (as well as their orientation) as well as users in the room. The locator service processes raw positional data from device and sensor clients that are distributed over the network, and transforms the data from a device-specific coordinate space into a locator service coordinate space, while also building higher level information about the state of the environment and the relationships between entities. Additionally, all entities that are tracked and processed by the locator service are in 3D.

As the locator service maintains position and distance between all entities, proxemic functions are readily available for researchers and developers to build upon. These functions include querying and filtering entities based on distance or within a certain distance range, and whether an entity is in the field of view of another entity, similar to [23]. We also allow researchers and developers to dynamically change proxemic properties of entities (e.g. location and orientation) through our visualization tool that we discuss in the next section.

The locator service uses an event-driven design, where clients subscribe to events that occur in the locator service, such as the proxemics previously discussed. For example, a wall display client can subscribe to an event that allows it to be aware of when a user approaches a certain range (similar to [23]) or when other entities are pointing in its direction. Lastly, the locator service also maintains information about data points, an entity unique to the *SoD-Toolkit*. The data point allows data to be assigned to specific physical locations in the room (through code or the visualization tool) and also supports proxemic properties already available to other entities. For example, a user can create virtual information spaces with different types of data in regions of a room or

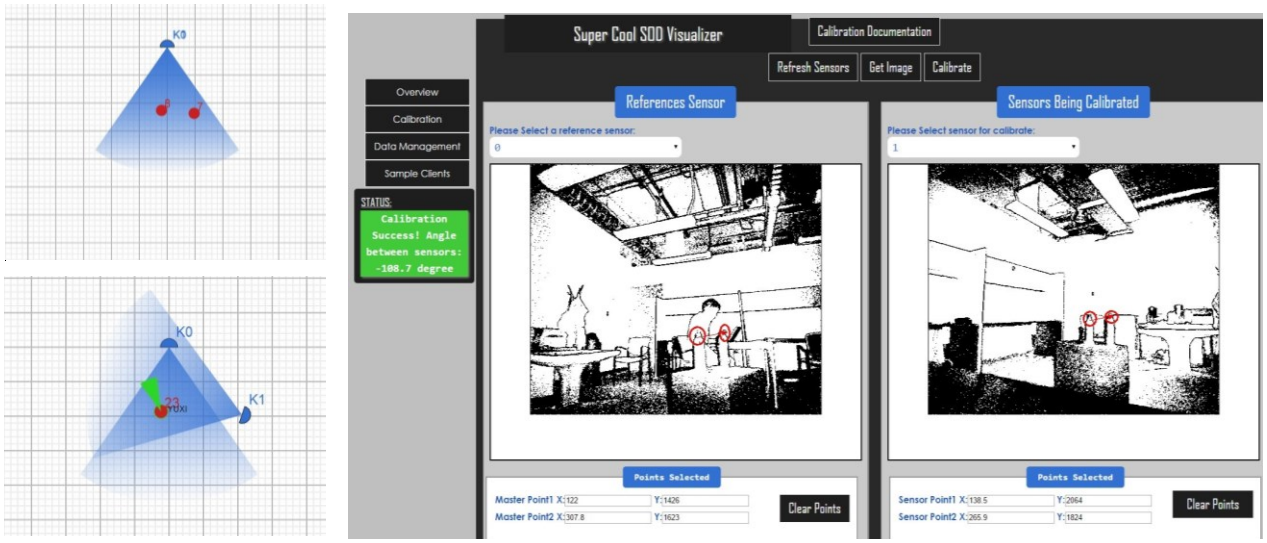


**Figure 4. Sensor Fusion using multiple Kinect sensors. (a) 2 uncalibrated Kinect sensors (overlapping) showing skeletons tracked in the environment. (b) Interface for calibration. (c) Properly calibrated environment with overlapping Kinect areas.**

assign data to physical objects (e.g. a table) that can be interacted with via gestures or a device-based interactions.

*Central Server and Networking*

The central server is built into the locator service, and follows a client-server architecture. The server is implemented using the highly scalable and efficient Node.js platform and the underlying communication between the server and client libraries (on all platforms) use the Socket.IO networking module. These components are based upon the WebSocket protocol, which uses duplex bidirectional communication with significantly less overhead than other traditional methods based on HTTP. All information and data exchanged between the server and clients uses the standard JSON format.

Through the modular design of the networking and client library components, the *SoD-Toolkit* allows researchers and developers to easily extend support for future devices and sensors with minimal development effort. As we discuss in the upcoming case studies section, this was a direct result of collaboration with industry partners building real-world deployable ubiquitous environments.

### Visualization Tool

The visualization tool for the *SoD-Toolkit* (Figure 4) has three primary functions, (1) to allow researchers and developers to monitor and understand entities – sensors, users, data points and devices – that are being tracked in the environment, as well as provide the state of fused sensors, (2) to allow for quickly prototyping ubiquitous environments without the need for hardware, people or room space and (3) maintain calibration data for multiple sensors (discussed in the next section).

In the visualizer tool, an overview list is provided, detailing the connected clients in the system, a list of available sensors in the environment, people and details on their location information, pairing state and if they currently hold any data, a list of devices and according details, and finally the location of data points and what data is contained in the data point. All entities that are visible in the tool, are user moveable components and easily configurable, allowing researchers and developers to physically remap sensors in the environment and dynamically change the layout of the environment with no additional code-required.

The tool also allows for quickly prototyping environments by allowing sample sensor and device clients to be created and connected into the system. This allows researchers and developers to conceptually build applications, environments and multi-device interactions independent of hardware and sensors and physical space. However, should they later decide to add in real-world sensors and hardware into their virtual environment, their built application or environment will already support the hardware. Furthermore, researchers and developers can also prototype mixed fidelity ubiquitous environments, allowing for a mixture of real and virtual sensors and devices (e.g. a virtual leap motion and a real-world Kinect sensor, with a fixed virtual wall display). Overall, the visualization tool is meant to be used throughout the development process – from development to deployment.

### Sensor-Fusion Approach

As discussed earlier, one of our research goals was the novel exploration of ubiquitous environments with an expanded tracking space and multiple newer sensors. This goal requires sensor fusion techniques that are seamless in nature, as well as modular. The *SoD-Toolkit*, primarily relies on the Kinect (version 1 or 2) as the means for providing room-

based tracking, as opposed to more expensive and harder to setup motion-tracking systems [25]. Given the limitations of a single Kinect tracking system, we use a method of image comparison between multiple Kinect sensors to expand tracking area and improve accuracy.

Generally, the sensor fusion method for multiple Kinect sensors in *SoD-Toolkit* works as follows: (1) A common object (e.g. a water bottle) is placed in the common viewing and tracking area of two Kinects, (2) Two points from each sensor's view are selected (based on the common object) in the Calibration interface in the Visualization tool and (3) One Kinect sensor's vector (the user selected reference sensor) is translated to another Kinect sensor's vector through a custom algorithm and saved. Figure 4 illustrates this specific process from the perspective of the Visualizer tool. The translation of the Kinect data is also reflected in the skeletal tracking, as the Location service compares skeleton data with the list of people being tracked in the environment. The location service then ensures that a person being tracked by multiple Kinect sensors is presented only once by comparing its position with the relative position of tracked users. This process is for two sensors at a time, and for larger spaces with multiple Kinect sensors, the process is repeated for each additional Kinect sensor and the reference sensor.

Information from Kinect sensors and individual device orientations is also fused by the Location service to provide position and orientation of a device. This type of fusion, which we call a paired state – where a device and user are paired – then allows for multi-device interactions such as "flicking" and "pouring" [32] that require more accurate spatial-awareness. We also allow for sensor data from other sensors such as the iBeacon and Leap motion to augment existing information that can be inconsistently provided by Kinect sensors in certain situations (e.g. a user is too far away from the sensor for fine grained finger tracking or is out of consistent tracking range to determine distance). In code, a researcher or developer can simply choose which sensor to use, either based on confidence levels or preference.

## REAL-WORLD CASE STUDIES

To demonstrate the functionality and test the feasibility and applicability of the toolkit for real-world deployments of spatially-aware ubiquitous environments, we present two real-world industry case studies. The implementations we discuss were built in parallel with the toolkit itself, and shaped many of the decisions that led to the features and design of the toolkit. Both of the systems described in this section were built with different teams of developers and industry partners with whom we collaborated closely with. For each case study, we briefly describe the system and reflect on lessons learned from the domain that influenced the design of the toolkit.

### Oil and Gas Exploration

The very first ubiquitous environment built in parallel with the *SoD-Toolkit* was [34]. We worked closely with SkyHunter Exploration Ltd. who are located in Canada, and specialize in oil and gas exploration, and have proprietary technology that collects a variety of geo-spatial data. The data they collect is multidisciplinary, and ultimately increases the chances of discovering oil and gas significantly. Prior to building the ubiquitous environment, much of their collaboration with the stakeholders in the exploration and decision-making processes (i.e. geophysicists, geologists) was paper-based and ineffective due to large volumes of geo-spatial data, as well as the reliance on single user non-collaborative tools. Upon building the environment with spatial-awareness and various multi-device interactions, we received positive feedback in areas such as collaboration and interaction with geospatial data [34].

### Emergency Response

Following the first industry case study for the toolkit and the adjustments made based on developer and industry feedback, we then continued to the emergency response domain. We collaborated with C4i Consultants, also located in Canada, who specialize in training software for military operations and emergency response. They provide a software tool ePlan, which is designed to simulate large scale emergencies, and train civic operators on how to respond with different types and scales of emergencies. Given the collaborative nature of emergency response planning environments and the number of individuals and devices that can be involved in the decision-making process, this was an ideal candidate for building a ubiquitous environment. We built a ubiquitous emergency response environment that built upon ePlan to drive simulations [4]. It allowed for larger groups of different stakeholders (e.g. fire, police, hazmat) to collaborate and communicate within a spatially-aware environment that contained a large wall display, tablets and a digital tabletop. Spatial interactions such as "flick" and "pour" allowed transfer of vital emergency information amongst the different stakeholders and their devices.

### Lessons Learned

As we began to develop the toolkit in collaboration with our first industry partner, we relied exclusively on the Microsoft Kinect sensors for spatial-awareness and tracking in the environment. However, when developers deployed the system built with the toolkit and received feedback from system users about the general uncomfortableness of multiple highly visible tracking sensors in an environment (particularly in an office or meeting context), this facilitated our change in the toolkit from *single sensor rigidity* to *multi-sensor flexibility*. We also provided a set of common gestures from multi-device interactions (flick, pour, bump from [32]) for the developers to use for information transfer tasks, but our feedback from them indicated *the need for accessing raw information* to customize/ explore more detailed interactions, particularly as the skill on the development team was varied.

After the subsequent changes made in the toolkit based on feedback from developers in the first industry project, we then worked closely with a second industry partner in a

different domain, emergency response. A primary challenge for developers of ubiquitous environments in this domain, is information is distributed across the room, usually in highly concentrated regions around specific personnel. At the time, the toolkit wasn't optimized enough to handle regions of coarse and fine grained tracking. This lead to the change of allowing for *sensor transition*, depending on the sensors available in the environment. The developers and industry partner also expressed interest in applying of *proxemic interactions to region specific data*, which led to the development of data points in the toolkit. Lastly, the developers and industry collaborators faced challenges in developing their ubiquitous environment due to limited access to highly confidential and private emergency response environments. To help facilitate developers, we found it important to enable them to *develop unencumbered by room access, devices or sensors*. This created features in the toolkit for easy prototyping, allowing developers to mock ubiquitous environments and multi-device interactions.

## DISCUSSION

Implementing and designing larger ubiquitous environments, applications and multi-device interaction techniques is an extremely complicated task. We introduced the *SoD-Toolkit* that provides researchers and developers with a number of tools that support the design, prototyping and implementation of deployable multi-device, multi-sensor ubiquitous environments, applications and interactions. We allow researchers and developers to choose their own sensors and devices to build ubiquitous configurations (regardless of owning hardware) and use the tools provided to explore their ubiquitous designs. In this section, we compare *SoD-Toolkit* to other approaches using Olsen's thematic framework approach [6], similar to [13].

### Problem not previously solved

A number of toolkits such as XDStudio [27], Conductor [9] and Panelrama [43] reduced the barrier of entry for researchers and developers in creating and exploring ubiquitous environments, applications and interactions. *SoD-Toolkit* builds upon many of the concepts in these toolkits, and further extends the work with greater support for a diverse set of devices and sensor platforms, multi-sensor mappings, and the ability to explore novel multi-device interaction combinations. We also focused on providing a toolkit that is easily configurable and deployable in real-world contexts, a common criticism of toolkits for ubiquitous environments [28, 29]. The toolkit also draws upon previous work in proxemic interactions and applications (such as Proximity Toolkit [23] and Grouptogether [24]), generalizing ubiquitous approaches, and spatial awareness, allowing for the exploration and development of different facets of research in ubiquitous environments, such as multi-device interaction techniques and sensor fusion approaches.

Another limitation of several existing toolkits lies in the physical reliance on sensor and device hardware for prototyping and developing ubiquitous environments and multi-device interactions. We allow researchers and developers to explore multi-device, multi-sensor ubiquitous environments without the need for all hardware or sensor components, which becomes increasingly important and relevant as the underlying technology moves rapidly both in terms of cost and capability. Reducing this limitation allows for an expansion of research into ubiquitous environments.

### Generality

To demonstrate the expressivity of the building block components of the *SoD toolkit* [6], we discussed two industry case-studies that included interaction techniques and scenarios considered state of the art [4, 34]. We also highlighted the versatility and generality with these industry case studies, but through them recognized the limitations and strengths of the toolkit. A number of limitations of the toolkit, particularly in its focus for providing larger ubiquitous environments stem from the usage of the Kinect hardware. The Kinect does not provide the same level of extremely accurate tracking by large scale systems (e.g. Vicon) even with multiple Kinect sensors. This is further enhanced when coarser grained sensors (e.g. iBeacons) are used entirely. However, based on our collaborations with industry partners, we believe that for more real-world deployments of ubiquitous environments, this is a necessary trade-off. This provides an advantage, as hardware and sensor limitations are nullified by simply using existing hardware and sensor setups to develop novel and more futuristic multi-sensor, multi-device ubiquitous environments, applications and interactions. Furthermore, the toolkit can easily be extended to support newer sensor and hardware platforms as they emerge.

### Reducing Solution Viscosity

Compared to other methods to develop real-world deployable multi-sensor, multi-device ubiquitous environments, applications and interactions, the *SoD-Toolkit* dramatically lowers the development viscosity [34] by providing a flexible architecture that allows for "expressive leverage" [13]. The design of the architecture also allows for the potential to transform several existing multi-device toolkits (e.g. XDStudio [27]) into multi-sensor and spatially-aware toolkits. Furthermore, as the toolkit abstracted and simplified many challenges in creating complex ubiquitous environments (as discussed previously), novice researchers and developers can focus on creating novel systems and interactions with minimal overhead, while those with more experience can freely modify different aspects of the toolkit, such as sensor-fusion approaches and sensor support.

### Empowering new design participants

The lack of real-world deployable examples of larger multi-device ubiquitous environments (prototypes or otherwise) despite widely available sensors and devices, indicates a need for adequate toolkit support for researchers and developers. In the research literature, a number of elicitation studies have been performed to study interactions in ubiquitous environments, but very little follow-up work is typically performed, particularly from an implementation

perspective. Additionally, multi-device explorations with newer forms of devices in ubiquitous environments (e.g. wearables) is still very minimal in the research literature, particularly in multi-sensor spatially-aware ubiquitous environments [13]. *SoD-Toolkit* focuses on lowering the threshold for non-expert programmers, researchers and interaction designers and empower them to explore with new and existing sensor and device technologies for real-world deployments and contexts. To date, we have not performed an in-depth study with developers for the toolkit, but we have worked closely with developers and industry from the onset of implementation and design of the toolkit. This remains as future work, but we believe the toolkit is an important step in exploring novel larger scale ubiquitous environments with multiple sensors and multiple devices, as well as necessary for exploring novel multi-sensor, multi-device interactions.

### Power in combination

Similar to prior toolkits, *SoD-Toolkit* combines hardware and sensor management, sensor fusion techniques, networking, disributed interfaces and prototyping into a single toolkit. As mentioned earlier, the architecture design allows for these components to be extremely decoupled, allowing for the addition or subtraction of different approaches. Additionally, as the *SoD-Toolkit* supports multiple platforms, it already integrates with several major UI frameworks (e.g. Objective C and Xcode), meaning developers can both prototype and build upon commercially available hardware.

### Can it scale up?

In building multi-sensor, multi-device spatially-aware ubiquitous environments, particularly ones designed for deployment, the scalability of the sensors that provide spatial awareness for multi-device interactions becomes an issue. For toolkits such as Panelrama [43] and Conductr [9], the solutions are scalable as they do not rely on sensors and use scalable web-based technologies (e.g. HTML5). Alternatively, toolkits such as Proximity Toolkit [23] and XDKinect [28], are either scalable but face challenges for real-world deployment due to complex setups or are limited in functionality due to limited sensor support. Undoubtedly, balancing spatial awareness and scalable solutions is difficult, especially when not using motion-tracking systems that aren't feasible for deployment [28, 29]. The design of *SoD-Toolkit* supports multiple low-cost off the shelf sensors that can be used in concert, for a larger spatially-aware ubiquitous environment. Additionally, as *SoD-Toolkit* allows researchers to choose their sensors and devices, researchers and developers can choose their granularity of spatial-awareness in a ubiquitous environment.

## CONCLUSION

The *SoD-Toolkit* offers a rich set of tools and software libraries for researchers and developers to prototype and develop deployable multi-device, multi-sensor interactions, applications and environments. We allow researchers and developers to easily "plug-and-play" popular off-the-shelf hardware (sensors and devices) to build their ubiquitous environments, while removing network and device management, sensor fusion and complex algorithmic challenges. The toolkit abstracts these challenges into an easily accessible API and events, integrating with several existing UI development software tools. As a result, the toolkit reduces complexity and lowers the barrier of entry for researchers and developers to design larger and more intricate multi-device ubiquitous environments. The toolkit also allows for future explorations of novel multi-device interaction techniques, multi-sensor designs and other unique ubiquitous environments that can be deployed in new and unexplored contexts and domains. Future work for the *SoD-Toolkit* includes integrating powerful device-centric sensors like the Google Tango, and gesture detection for different types of interactions in ubiquitous environments.


## ACKNOWLEDGEMENTS

We would like to thank our many colleagues from the Agile Surface Engineering (ASE) Lab and our industry partners for their useful discussions and advice as we developed the toolkit. We also thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported in by the NSERC SurfNet Research Network, NSERC and Alberta Innovates Technology Futures (AITF).


## REFERENCES


1. Ballendat, T., Marquardt, N. and Greenberg, S. Proxemic interaction: designing for a proximity and orientation-aware environment. In *Proc of ITS 2010*, 121-130.
2. Brumitt, B., Meyers, B., Krumm, J., Kern, A. and Shafer, S. A. EasyLiving: Technologies for Intelligent Environments. In *Proc of UbiComp 2000*, 12-29.
3. Chen, X. A., Grossman, T., Wigdor, D. J. and Fitzmaurice, G. Duet: exploring joint interactions on a smart phone and a smart watch. In *Proc of CHI 2014*, 159-168.
4. Chi, P. and Li, Y. Weave: Scripting Cross-Device Wearable Interaction. In *Proc of CHI 2015*, 3923-3932.
5. Chokshi, A., Seyed, T., Rodrigues, F. M. and Maurer, F. ePlan Multi-Surface: A Multi-Surface Environment for Emergency Response Planning Exercises. In *Proc of ITS 2014*, 219-228.
6. Dan R. Olsen, J. Evaluating user interface systems research. In *Proc of UIST 2007*, 251-258.
7. Edward, T. *Hall, The Hidden Dimension*. Garden City, NY: Doubleday, City, 1966.
8. Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R. and Wang, M. Proxemic interactions: the new ubicomp? *interactions*, *18*, 1 (2011), 42-50.
9. Hamilton, P. and Wigdor, D. J. Conductor: enabling and understanding cross-device interaction. In *Proc of CHI 2014*, 2773-2782.
10. Hartmann, B., Beaudouin-Lafon, M., and Mackay, W. HydraScope: creating multi-surface meta-applications through view synchronization and input multiplexing. In *Proc of PerDis 2013*, 23-28.
11. Hinckley, K. Synchronous gestures for multiple persons and computers. In *Proc of UIST 2003*, 149-158.

12. Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P. and Smith, M. Stitching: pen gestures that span multiple displays. In *Proc of AVI 2004*, 23-31.
13. Houben, S. and Marquardt, N. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proc of CHI 2015*, 1247-1256.
14. Huang, D.-Y., Lin, C.-P., Hung, Y.-P., Chang, T.-W., Yu, N.-H., Tsai, M.-L. and Chen, M. Y. MagMobile: enhancing social interactions with rapid view-stitching games of mobile devices. In *Proc of MAM 2012*, 1-4.
15. Jakobsen, M., Haile, Y., Knudsen, S. and Hornbæk, K. Information Visualization and Proxemics: Design Opportunities and Empirical Findings. *IEEE Transactions on Visualization and Computer Graphics*, *19, 12* (2013), 2386-2395.
16. König, A., Rädle, R. and Reiterer, H. Squidy: a zoomable design environment for natural user interfaces. In *Proc of CHI'EA 2009*, 4561-4566.
17. Kortuem, G., Kray, C. and Gellersen, H. Sensing and visualizing spatial relations of mobile devices. In *Proc UIST 2005*, 93-102.
18. Kray, C., Rohs, M., Hook, J. and Kratz, S. Group coordination and negotiation through spatial proximity regions around mobile devices on augmented tabletops. In *Proc. Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, IEEE (2008), 1-8.
19. Laibowitz, M. and Paradiso, J.A. The UbER-Badge, a versatile platform at the juncture between wearable and social computing. In: *Fersha A, Hortner H, Kostis G (eds) Advances in Pervasive Computing*. Oesterreichische Computer Gesellschaft, Wien, pp 363-368.
20. Lucero, A., Holopainen, J. and Jokela, T. Pass-them-around: collaborative use of mobile phones for photo sharing. In *Proc of CHI 2011*, 1787-1796.
21. Lucero, A., Keränen, J. and Korhonen, H. Collaborative use of mobile phones for brainstorming. In *Proc of MobileHCI 2010*, 337-340.
22. Marquardt, N., Ballendat, T., Boring, S., Greenberg, S. and Hinckley, K. Gradual engagement: facilitating information exchange between digital devices as a function of proximity. In *Proc of ITS 2012*, 31-40.
23. Marquardt, N., Diaz-Marino, R., Boring, S. and Greenberg, S. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proc of UIST 2011*, 315-326.
24. Marquardt, N. and Greenberg, S. Informing the Design of Proxemic Interactions. *IEEE Pervasive Computing*, *11*, 2 (2012), 14-23.
25. Marquardt, N., Hinckley, K. and Greenberg, S. Cross-device interaction via micro-mobility and f-formations. In *Proc. of UIST 2012*, 13-22.
26. Mayer, S. and Sörös, G. User Interface Beaming -- Seamless Interaction with Smart Things Using Personal Wearable Computers. In *Proc. Proceedings of the 2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops*, IEEE Computer Society (2014), 46-49.
27. Nebeling, M., Mintsi, T., Husmann, M. and Norrie, M. Interactive development of cross-device user interfaces. In *Proc of CHI 2014*, 2793-2802.
28. Nebeling, M., Teunissen, E., Husmann, M. and Norrie, M. C. XDKinect: development framework for cross-device interaction using kinect. In *Proc of EICS 2014*, 65-74.
29. Rädle, R., Jetter, H.-C., Marquardt, N., Reiterer, H. and Rogers, Y. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proc Proceedings of ITS 2014*, 45-54.
30. Rekimoto, J. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proc of UIST* 1997, 31-39.
31. Schmidt, D., Seifert, J., Rukzio, E. and Gellersen, H. A cross-device interaction style for mobiles and surfaces. In *Proc of DIS 2012*, 318-327.
32. Seyed, T., Burns, C., Sousa, M. C., Maurer, F. and Tang, A. Eliciting usable gestures for multi-display environments. In *Proc of ITS 2012*, 41-50.
33. Schreiner, M., Rädle, R., Hans-Christian, J. and Reiterer, H. A Framework for Cross-Device Web Applications. In *Proc of CHI'EA 2015*, 2163-2168.
34. Seyed, T., Sousa, M. C., Maurer, F. and Tang, A. SkyHunter: a multi-surface environment for supporting oil and gas exploration. In *Proc of ITS 2013*, 15-22.
35. Snibbe, S. S. and Raffle, H. S. Social immersive media: pursuing best practices for multi-user interactive camera/projector exhibits. In *Proc of CHI 2009*, 1447-1456.
36. Streitz, N., Prante, T., M, C., ller-Tomfelde, A., Tandler, P. and Magerkurth, C. Roomware: the second generation. In *Proc of CHI EA 2002,* 506-507.
37. Streitz, N. A., Holmer, T., Konomi, S. i., M, C., ller-Tomfelde, Reischl, W., Rexroth, P., Seitz, P. and Steinmetz, R. i-LAND: an interactive landscape for creativity and innovation. In *Proc of CHI 1999*, 120-127.
38. Vogel, D. and Balakrishnan, R. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proc of UIST 2004*, 137-146.
39. Wagner, J., Nancel, M., Gustafson, G., Huot, S. and Mackay, W. Body-centric design space for multi-surface interaction. In *Proc of CHI 2013*, 1299-1308.
40. Want, R., Hopper, A., Falc, V., #227 and Gibbons, J. The active badge location system. *ACM Trans. Inf. Syst.*, *10*, 1 (1992), 91-102.
41. Weiser, M. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, *3*, 3 (1999), 3-11.
42. Wilson, A. D. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc of UIST 2010*, 273-282.
43. Yang, J. and Wigdor, D. Panelrama: enabling easy specification of cross-device web applications. In *Proc of CHI 2014*, 2783-2792.
44. Zadow, U., Büschel, W., Langner, R. and Dachselt, R. SleeD: Using a Sleeve Display to Interact with Touch-sensitive Display Walls. In *Proc of ITS 2014*, 129-138.