# Creepy Tracker Toolkit for Context-aware Interfaces

**Maurício Sousa**[1,*], **Daniel Mendes**[1,*], **Rafael Kuffner Dos Anjos**[1,2,*], **Daniel Medeiros**[1,*],
**Alfredo Ferreira**[1,*], **Alberto Raposo**[3,†], **João Madeiras Pereira**[1,*] **and Joaquim Jorge**[1,*]
[1]Inesc-ID / Universidade de Lisboa, [2]FCSH / Universidade Nova de Lisboa, [3]Tecgraf, PUC-Rio Brasil
[*]{antonio.sousa, danielmendes, daniel.medeiros, alfredo.ferreira, joao.madeiras.pereira,
jorgej}@tecnico.ulisboa.pt, [†]abraposo@tecgraf.puc-rio.br

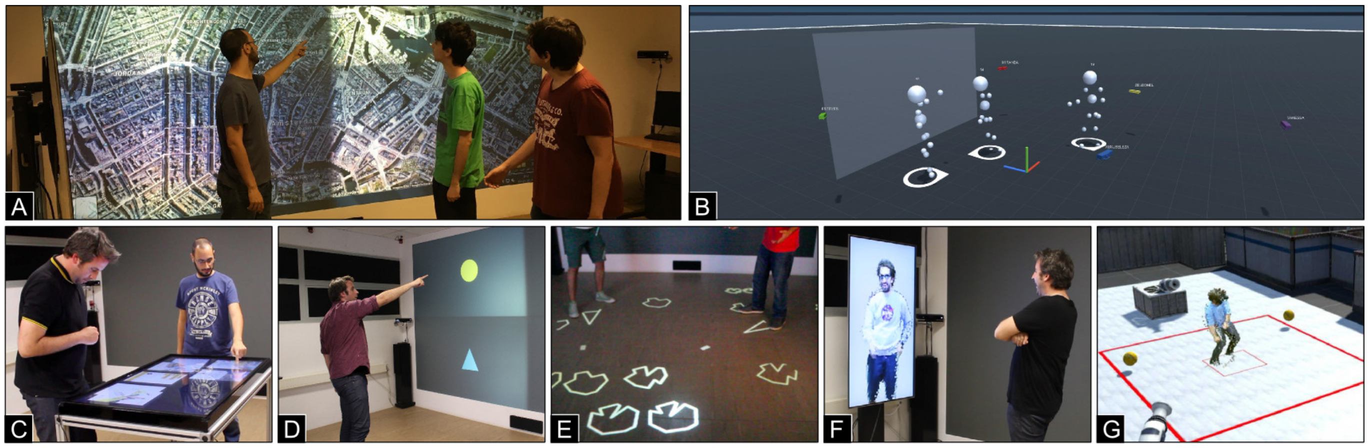


**Figure 1.** ***Creepy Tracker*** **is an open-source toolkit that provides spatial information about people and interactive surfaces. To do this, it resorts to multiple depth sensing cameras (A, B). It helps the design of systems that handle, for instance, (C) interactive tabletops, (D) vertical surfaces, (E) floor projections and even capture avatars for (F) telepresence or (G) virtual reality.**

## ABSTRACT

Context-aware pervasive applications can improve user experiences by tracking people in their surroundings. Such systems use multiple sensors to gather information regarding people and devices. However, when developing novel user experiences, researchers are left to building foundation code to support multiple network-connected sensors, a major hurdle to rapidly developing and testing new ideas.

We introduce *Creepy Tracker*, an open-source toolkit to ease prototyping with multiple commodity depth cameras. It automatically selects the best sensor to follow each person, handling occlusions and maximizing interaction space, while providing full-body tracking in scalable and extensible manners. It also keeps position and orientation of stationary interactive surfaces while offering continuously updated point-cloud user representations combining both depth and color data. Our performance evaluation shows that, although slightly less precise than marker-based optical systems, *Creepy Tracker* provides reliable multi-joint tracking without any wearable markers or special devices. Furthermore, implemented representative scenarios show that *Creepy Tracker* is well suited for deploying spatial and context-aware interactive experiences.

ISBN 978-1-4503-4691-7/17/10...$15.00
https://doi.org/10.1145/3132272.3134113

**Author Keywords**
Context-aware Computing; Toolkit; Rapid-Prototyping

**ACM Classification Keywords**
H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces – Input devices and strategies

## INTRODUCTION

Human-Computer Interaction (HCI) researchers strive to understand context to anticipate user requirements when designing novel experiences. Context-aware computing relates to interactive systems that leverage different sensing methods to gather understanding about their surroundings [31]. Moreover, context-awareness is an essential foundation of ubiquitous [40] and pervasive systems [4]. Recently, the interaction space and the physical relationships between people and interactive devices have been the focus of much research. Indeed, recent developments using spatially-aware ubiquitous environments can infer interactions and even people's intentions to interact using fused sensor data [17, 18]. The emergence of commodity depth sensors, such as the Microsoft Kinect, contributed out-of-the-box tracking approaches to focus ubicomp researchers' attention on the interaction design and user experience. However, depth cameras limitations can create barriers to the design and evaluation of new interaction approaches and techniques. In fact, single depth cameras cannot handle occlusions. While multiple cameras can mitigate this problem, they generate large volumes of network traffic for real-time data streaming. Furthermore, combining data from multiple coordinate systems (one per sensor) require additional processing and calibration methods. These issues must be addressed before any attempt to designing novel user experiences.

In this work, we present the *Creepy Tracker* Toolkit, a set of open-source[1] software tools to aid rapid-prototyping of context-aware interactive systems using multiple depth cameras. The proposed tools allow seamless installation and offer a backbone for developing such systems, thus concealing the complexity inherent to current approaches. We followed a conceptual line parallel to the work from Seyed et al. [32]. Our tracker consists of a network server that combines data from multiple depth sensors to provide full-body positional tracking of people within a room-sized volume. The toolkit manages the spatial locations of interactive surfaces and can easily infer spatial relationships to the people surrounding them. Also, it supports flexible full-body point-cloud representations of people. For the purpose of this work, we consider the definition of context provided by Dey et al. [10], which is the knowledge of "location, identity and state of people, groups, and computational and physical objects". To this end, the *Creepy Tracker* toolkit combines multiple depth sensors to provide full-body positional tracking of people and tools to acquire precise locations of interactive surfaces, while providing a networked stream of context data front-end applications.

The contributions of this work are thus: (1) a set of tools for rapid-prototyping context-aware applications, that incorporates body tracking, interactive surfaces and point-cloud representation of people; (2) different scenarios that can be implemented using our toolkit; and (3) practical considerations when using *Creepy Tracker*, obtained from a system's performance evaluation. In this work, in addition to disclosing an open-source toolkit, we also detail how we overcame every major technical obstacle. In what follows, we review relevant related work, and detail the toolkit design and implementation. Furthermore, we demonstrate the scalability and performance of our toolkit using five Microsoft Kinect depth cameras to evaluate its accuracy and capabilities. We also provide a discussion of the evaluation results. Finally, we explore the design space of context-aware applications built on top of *Creepy Tracker* by demonstrating five different application scenarios.

## RELATED WORK

Our work builds on prior research in two main areas: (1) Context-aware and Ubiquitous Environments and (2) Marker-less Human Tracking. In the following subsections we discuss related work in these areas, focusing on the advantages and limitations of previous endeavours.

### Ubiquitous and Context-aware Environments

Mark Weiser [39] suggested that computing technologies would move beyond devices towards being embedded in the environment, in order not to intrude on people's daily tasks. Indeed, ubiquitous environments are becoming commonplace due to the rise of interactive devices and advances on sensing technologies [30]. Therefore, by sensing the situation of the environment, digital systems can use that knowledge to infer intent to interact [31]. Context-aware interactions can thus exploit what is happening in close proximity of a person or device. Matthews et al. [24] developed a toolkit that examines context by grabbing users' attention to peripheral displays. Annett et al. [2] demonstrated a proximity-aware tabletop that determines users' presence, position and the arm which is interacting with the display. Jota et al. [22] described the interaction design space on and above horizontal interactive surfaces. Interactive vertical display interfaces can also benefit from the environmental context. Vogel et al. [38] proposes an interaction framework for interactive public displays. The authors map out the area in front of a vertical display into ambient, implicit, subtle and personal interactive spaces. When a person is transitioning between those spaces, the ambient display can display different contextual information based on the person's proximity, ranging from public details at a distance to a more private information, when in close proximity. Also, with proper sensing technologies, context-aware systems can react to the normal everyday interactions of groups of people [21]. Edward Hall introduces the Proxemic Theory [14] and observed that space, distance and orientation between people impact the way they interact with each other. Ballendat et al. [7] suggested that knowledge about the way people position themselves can be exploited in ubicomp environments to start or end interactions, establish connections, and even, automatically transfer personal files between devices. Pushing this notion further, Marquardt et al [19] applied proxemics to explore the design space in ubicomp environments to mediate interactions between people and their personal and public devices.

### Marker-less Human Tracking

Our work builds on previous research on instrumenting the environment instead of requiring users to carry physical tracking devices [16]. Antifakos and Schiele [3] demonstrated that wifi networks are able to detect proximity. Fails and Olsen [11] proposed a technique to sense hand gestures to interact with surfaces via skin color detection using RGB cameras. However, recent developments in commodity depth cameras allow people tracking [41] in expeditious manners.Depth cameras can disambiguate color images with depth information [1] and allow devices to estimate human poses [33]. Wilson and Benko [42] presented LightSpace, a prototype that combines depth cameras and projectors to provide interactions on and between multiple surfaces. People can transfer and manipulate objects in one surface, "pick" and "drop" on another device. LightSpace employs multiple depth cameras calibrated to the same coordinate system. It is also the first approach to combine depth data from multiple depth cameras. Sousa et al. [35] also used multiple depth cameras to deal with body occlusions in a remote collaborative environment for groups of people. Yet, their tracking approach did not consider users' orientation and full body tracking. The Proximity Toolkit [20] uses sensor fusion to gather data from multiple different tracking

[1]Github: `https://github.com/vimmi3D/CreepyTracker`

systems to gather proxemic data about people and devices. The toolkit combines skeleton data from depth cameras for body tracking with a marker-based system to track devices. Developers can create user experiences using both the tracked entities and the proxemic relationships between them. Despite having the same motivation, our approach focuses not on the relationships between entities, but rather deals with combining multiple depth sensors. In this work, we address resolving multiple skeletons into persons, choosing the optimal sensor for each person while avoiding orientation errors when tracking people from behind. More recently, Wu et al. [44] presented EagleSence, a top-view camera-based system for tracking people's position, orientation and activities. A top-view approaches minimize body occlusions by other people, although, as reported by the authors, this approach proves to be difficult when acquiring body skeleton joints.

Seyed et al. [32] tackled a similar challenge to ours. They introduced the SoD-Toolkit. It offers a set of tools for tracking people, interactions between them and devices using multiple sensors. Our *Creepy Tracker* follows some of the concepts from SoD-Toolkit, further exploring them. Indeed, we focus on continuous 3D spatial context and full body tracking of multiple people. Moreover, we allow for explicit and accurate surfaces' calibration, as well as point-cloud user virtual representation. In the following sections we thoroughly detail how *Creepy Tracker* operates and the main algorithms used. Finally, we evaluate our tracker regarding latency and accuracy against a maker-based optical system, developed specifically for motion capture and computer generated imagery.

## CREEPY TRACKER

The *Creepy Tracker* toolkit uses a network of distributed sensor units connected to a central hub. Each sensor unit is composed of a Microsoft Kinect depth camera and standalone C# application running on a single computer. The number of sensors is directly related to the area required by the interaction being designed. Interactions with a single typical (up to 4×2m) vertical surface may require one or two units, while interactions around a tabletop most commonly need several (up to 5) sensors surrounding that surface. Each sensor unit provides a continuous data stream. These converge on the tracker's central hub, which is responsible for synchronization, processing and merging the data, as depicted in Figure 2. The central hub also broadcasts the state of the tracked environment to client applications. Moreover, for the virtual model of the tracked people and surfaces to be precisely aligned with the physical topology of the room, the sensors' position and orientation must be first calibrated. Adding surfaces requires an active calibration for each new surface by defining the surface plane using 3D depth data of one sensor. After calibration, as people move in the tracked area, the virtual model gets updated in real-time, while broadcasting the updated data. In this section, we provide a detailed overview of the system's components, describing their implementation and application.

### Sensor Unit

All sensor units, each of them connected to an individual depth sensor, capture color, depth data and the body tracking model of every observed person in the tracked area. Each body

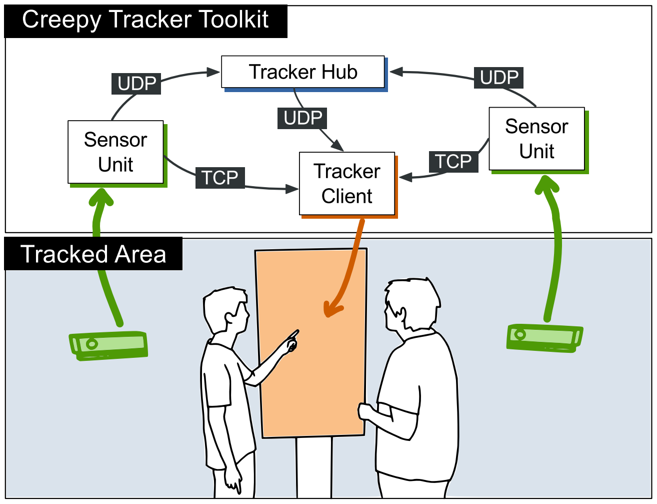


**Figure 2. Overall system's architecture.**

model is associated with a numerical factor to represent the estimated degree of confidence about the quality of the tracked person, which is sent together with the body model data. The confidence factor is calculated by adding all tracked body joints' weight, while discarding inferred ones. The weight of each joint can be customizable, so that the tracker can favor specific joints, useful for different scenarios. For instance, pointing tasks require far more importance given to hands' than feet' joints. Figure 3A shows a individual sensor client tracking two people, the person closer to the camera has a lower degree of confidence because half of the lower limbs' joints cannot be seen. Tracked people with confidence factors below a configurable threshold are ignored. Color and depth data are processed for the point-cloud representation of each person. The body tracking model is broadcast to the Tracker Hub using an UDP stream, while point-clouds are available via a concurrent TCP connection.

### Tracker Hub

The Tracker Hub component handles the unified model of the tracked area by combining the data streams from all sensor units. To create a reliable model, the Tracker Hub requires a calibration process to transform all received data into a single coordinate system. Figure 3A shows three calibrated sensors with both position and orientation matching the physical cameras. Data received from each of those sensor units will

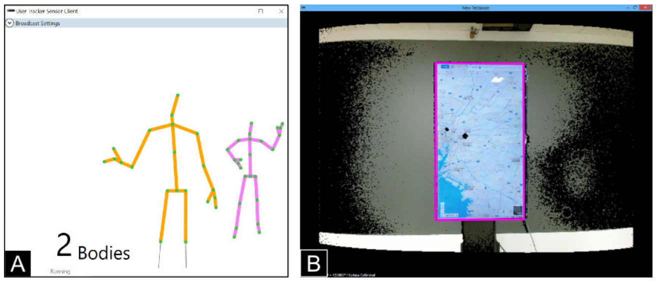


**Figure 3. *Creepy Tracker*'s (A) sensor unit and (B) surface calibration application.**

be spatially correct in the unified model's coordinate system. Analogously, a surface calibrated on one depth camera's coordinate system also is transformed to match the area of the physical one, as shown in Figure 1A. Since a surface is a collection of four fixed points in space, the setting up and calibration process needs to occur only once. The Tracker Hub is a Unity3D application that acts as broadcast server of the unified model to application clients.

## Calibration Method

A calibration process is required to unify all data streams into a single coordinate system. For this *Creepy Tracker* relies on the body tracking model of a person from each sensor unit to calculate the new global coordinate system and all cameras' position and orientation. The calibration process requires one standing person to be seen by all sensor units, in two discrete steps. Figure 4A shows five uncalibrated sensor units before the calibration process.

*Creepy Tracker* requires calibration parameters from body tracking models at two distinct locations separated by the distance of a step to calculate the origin and forward and up vectors of the new calibrated coordinate system. In the first step, the position of the person is used to define the origin. The up vector, defined by the spine base and spine shoulder joints of the body model, is also stored, as well as the position of both feet. The second calibration step can be performed after the person moves a step forward (Figure 4B). This new position is used in conjunction with the first to define the coordinate system forward vector. The second up vector is averaged with the first to minimize the impact of incorrect poses when calculating the final up vector. Finally, the minimum height according to the up vector of the four feet positions is used to define floor's position. Figure 4C shows five calibrated sensors around the coordinate system's origin.

This calibration is usually enough for most interactive scenarios. However, we reckon that a more precise calibration might be needed for more demanding cases. For such situations, we created an additional calibration step. It consists of capturing a depth data frame of each sensors and displaying them using point-clouds, with a simple object placed in the middle of the tracked area. For this, we resort to a cardboard cube with a coloured checkerboard in each face (Figure 4D). Then, is possible to manually adjust position and rotation parameters of each sensor, so that the point clouds match as well as possible.

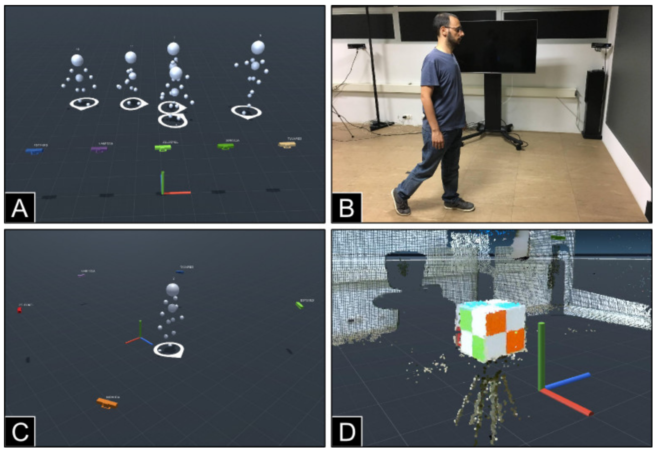


**Figure 4. Calibration process: (A) center; (B) step forward; (C) result; and (D) calibration cube for manual adjustments.**

A new calibration process is required when the setup undergo any adjustment or modification in sensor units.

## Adding Surfaces

The *Creepy Tracker Toolkit* also contributes with a standalone C# surface calibration tool (Figure 3B). We consider surfaces with three common standard aspect ratios: 4:3, 16:9 and 9:16, for surfaces in portrait mode. Our calibration tool treats a surface as a subspace of an infinite geometric plane restricted by the size and aspect ratio of the physical surface. The surface calibration method requires the mouse click input (on the calibration application) of two corner points from the same edge (surface's bottom left and bottom right) to define size and position, and a third inner point to calculate the surface's normal vector. Only three points spare the need for a depth sensor to have the entirety of the surface in its field-of-view. Figure 3B shows a calibrated 9:16 vertical surface with the manually selected points. By knowing the position and orientation of the calibrated sensor from which the surface's points originated, the surface is then transformed to the tracker's coordinate system.

## Tracking People

The Tracker Hub formalizes a body tracking model from the sensor unit into a *Body* entity and a person into an instance of *Human*. Overlapping individual body tracking models from different sensor units map into a single person. Consequently, a *Human* preserves a set of *Bodies*, one for each seen by a sensor unit.

When information regarding a new *Body* arrives, the Tracker Hub will try to fit that body in a *Human* within a parameterizable distance threshold. This threshold is set by default to 30 cm, to account for different sensors' perspectives, as it is impossible for two people to have their Spine Base joints closer than this threshold without intimate space violations. The distance is calculated according to Spine Base joints of both *Bodies*. If there is no suited *Human*, a new one is created. When a *Body* is no longer seen by its sensor, it is dissociated from the corresponding *Human*. If a *Human* has no more associated *Bodies*, it will enter a waiting period of 1 second. During that period, if a new *Body* appears within the distance threshold from the *Human*'s last position, it is associated to that *Human*, which exits from the waiting period. Otherwise, if no *Body* is associated with the *Human* until the waiting period expires, the *Human* is removed from the tracker.

Each *Human* entity is constantly choosing the most appropriate *Body* by selecting the one with the highest confidence value. It is not always easy to acknowledge for sure where people are turned to, as some sensors may be facing each other and perceiving mirrored body models for the same person, because the Microsoft Kinect cannot distinguish between people facing forwards or backwards. To overcome this, we follow two approaches. Firstly, we consider a disambiguation pose consisting of having at least one forearm approximately parallel to the floor. The direction one is pointing at, can be used to define that person's forward vector, as it would be both unnatural and very difficult to accomplish such a pose with the arm pointing backwards. When this vector's direction is opposite from the current *Human*'s forward, we automatically mirror *Body*'s left and right data. Secondly, as front and back switching occurs mainly when a *Body* from a different sensor is chosen, we also mirror the *Body* when the *Human* is detected to rotate faster (approximately 180 degrees in two consecutive frames) than it is humanely possible.

To deal with the known noisy skeleton information from the Microsoft Kinect, we implemented a double exponential smoothing filter [6]. The filter's parameters can be configured to achieve a compromise between smoothness and added latency. This filter is applied to *Human*'s joints, not to *Bodies*, and helps when dealing with sensor switching in setups with coarse calibrations.

## Point-cloud Representations

Using real-time body representations can enhance interactive scenarios for collaborative telepresence [27], computer-assisted rehabilitation [36] and immersive virtual reality [34]. Our approach relies on processing separate streams of point clouds. Each individual sensor unit first captures the skeletal body model data for each person in its field-of-view. We create a point-cloud by combination depth and color values captured by the camera. Then the person's relevant points are segmented from the background. Figure 5 shows the resultant streamed body representation. When interacting with applications using the body representation, different cameras will be sending very similar information. However, due to network, processing and rendering constrains, integration of different streams or redundancy resolution is not performed. We implement a task-oriented decimation, taking into account what body parts are more relevant. To this end, we attribute different priorities to each joint of the available *Body* information according to user-defined parameters. To wit, in collaborative telepresence scenarios, head, face and hand non-verbal communication cues are more relevant than capturing other body parts. Whereas in first-person virtual reality scenarios, detailed hands are more valuable and head information can be totally discarded.

Thus, for each point in the segmented cloud, we calculate its euclidean distance to the body joints marked as relevant on

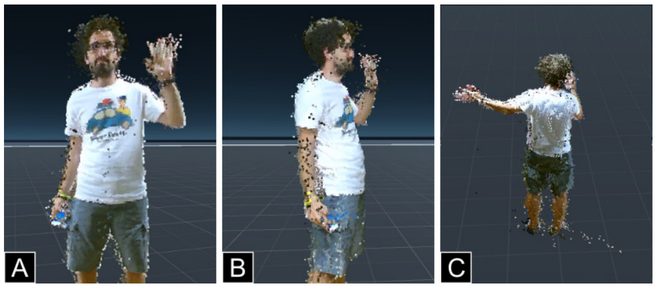


**Figure 5. Point-cloud body representation: (A) front, (B) side and (C) top views.**

the sensor units. If this distance is smaller than a threshold value (proportional to the user's body height), this point is marked as a high quality point for transmission. Points not marked as high quality, are sampled at a lower frequency to reduce data redundancy on less relevant areas. For each point, sensor units transmit 3D point coordinates, color, and a data bit to indicate high or low quality. This last bit is used to adjust the rendering parameters in the application client. Each coordinate in a 3D point is multiplied by 1000 and rounded to the closest integer, to assure millimeter precision, and packed into two bytes. Data read through the network is then parsed and rendered in the environment using surface-aligned splats at 30 frames per second. While higher quality points are more tightly spaced, requiring smaller splat sizes, we use larger splats in under-sampled regions to create closed surfaces.

## Using Tracker Data

*Creepy Tracker* offers a client-side C# API with a layer to render network communication transparent and provide updated encapsulated abstractions of tracking data. Figure 6 shows the data model put up by the API.

An independent tracker client, upon connection, continuously receives a list of *Humans* and can request at any time a list of available *Surfaces*. A *Human* is a representation of a real person in tracked area. It holds an unique identification provided by the tracker, a point in space correspondent to the person's position, a client-side calculation of the person's direction and a list of all body joints. This is specially useful a application scenario requires the spatial distribution and proximity relationships between people. We defined a surface as a geometric plane. The *Surface* entity is a combination of a center point, a normal vector and the four vertexes to define the surface's edges. Making it easy to calculate if a person is near, approaching or facing the surface's front side. Next, we demonstrate how to exploit the tracker's API and resources to develop five different sample scenarios.

### *Tabletops*

Multitouch tabletops with large screens enable simultaneous interactions from multiple people and foster collaboration by

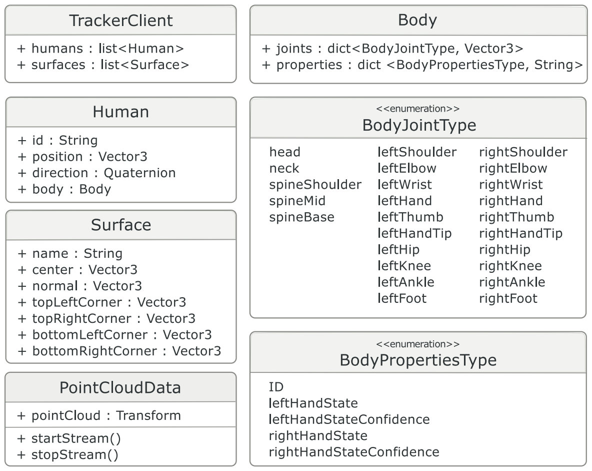


**Figure 6. *Creepy Tracker* client-side API classes + properties.**

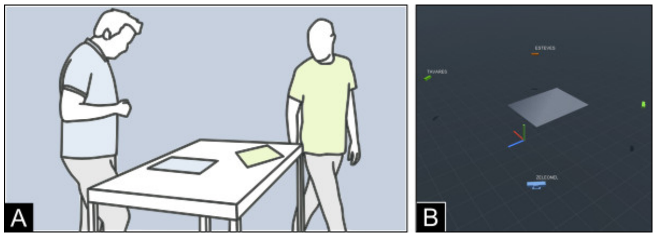


**Figure 7. Tabletop context-aware example: (A) idealized interaction design and (B) calibration setup.**

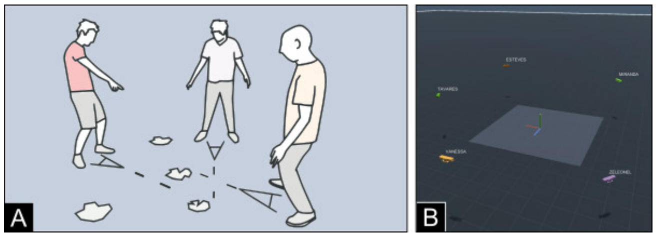


**Figure 9.** ***Asteroids*** **game on the floor: (A) idealized interaction design and (B) calibration setup.**

allowing individual interactions with common content [15, 23]. To explore this scenario, we devised an interactive tabletop application that enables people to take temporary ownership of digital content. Figure 7A depicts the interaction design rational for a tabletop application that utilizes context-aware information provided by *Creepy Tracker*. Users can get hold of digital content by touching it and the selected content starts following the person around until another touch breaks the temporary lock. For this, four depth cameras were distributed around the room with a calibrated surface right in the center of the tracked area, as shown in Figure 7B. Since a tabletop can provide multitouch inputs, associating content to a person during a selection requires the distance of all the users' hands spatial position to the point of touch. The person with the nearest hand takes ownership of the digital content. From here, the digital content follows the user around, until another touch gesture is detected.

### *Wall-sized Displays*

Bolt's Put-that-there [9] multimodal interface is a canonical approach to interact with objects in large scale displays. Put-that-there combines pointing gestures with speech recognition to select objects and define target locations. We designed an adaptation of Bolt's seminal system using body pose information to create and move objects with different shapes and colors, as depicted in Figure 8A. To avoid the occlusions in front of the large display, we opted for placing depth cameras on both sides of the screen, as demonstrated in Figure 8B. Instead of a laser, we utilized the image-plane pointing technique [28]. Therefore, when the tracker client detects a relevant utterance (”that” or ”there”), intersecting the vector, that starts in the user's head through the raised hand, with the surface plane, the target position can be determined.

### *Floor Projected Surface*

Using large floor projected interfaces provide sufficient space to promote interactive visualizations [5, 37] and shared social user experiences [12, 13]. We design a playful projection-based collaborative game based on the classic arcade space shooter *Asteroids*, released by ATARI, Inc in 1979. A player can use their own body position and orientation to destroy asteroids, as depicted in Figure 9A. While the projected floor surface serves to display the game view and players can control a spaceship placed exactly below them, automatically firing space bullets in the users' forwards direction at a fixed rate. Figure 9B shows the tracker's setup with five cameras and a calibrated 4:3 surface on the floor. For this experience, we resorted to *Human*'s position and orientation for steering the ship and utilized the *Surface* to map the players' positions.

### *Telepresence Portals*

Taking inspiration from the Office of the Future by Raskar et al. [29], we designed a telepresence experience to connect two remote locations. Indeed, depth sensors have already been used to create highly realistic avatars in remote collaboration scenarios [26], but with custom hardware and cluster-based rendering. *Creepy Tracker* can be used to show real-time realistic user representations using commodity hardware, easing the development of approaches similar to the work of Beck et al. [8]. This approach allows for verbal and non-verbal communication, and at the same time, creates a seamless visual continuity from the local to the remote location, as depicted in Figure 10A. The portals implicitly establish or cease the link between them by allowing a person to transition between a non-interactive space to a explicit interactive space, similarly to Vogel et al. [38]. Therefore, to initiate link between two portals, someone simply needs to walk into close proximity of the surface, triggering a presence notification on the other side. Analogously, the receiver of the notification can proceed by walking towards the portal to accept the connection.

For this, sensors were placed on each side of the surface to capture a point-cloud of the user, combining information from both sides of the body, as shown in Figure 10. Then, the

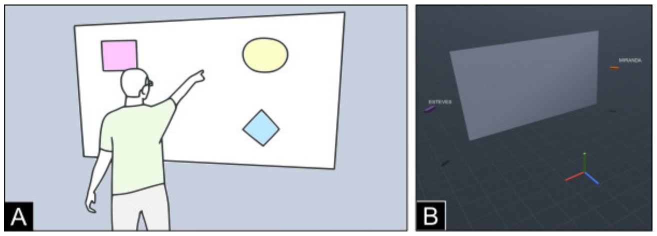


**Figure 8. Bolt's Put-That-There with the** ***Creepy Tracker*****: (A) idealized interaction design (B) calibration setup.**

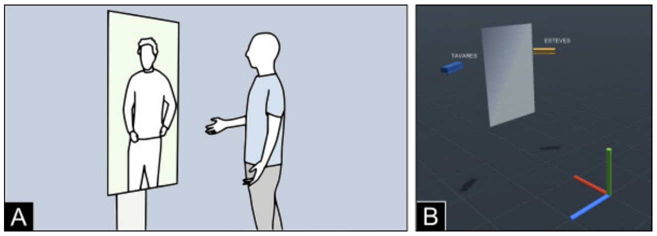


**Figure 10. Telepresence scenario: (A) idealized interaction design and (B) calibration setup.**

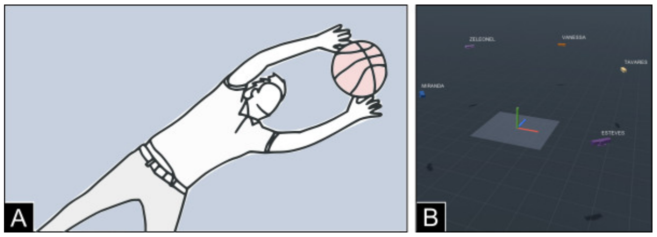


**Figure 11. Virtual Reality game: (A) idealized interaction design and (B) calibration setup.**

setup was replicated in another remote location. The distance between *Humans* and *Surfaces* were used to established and destroy the communication link. While *PointCloudData* was used to start streaming remote people.

### *Virtual Reality Interactions*

Multiple depth sensors setups have been deployed to capture full body data to animate generic avatars and explore novel interaction techniques in virtual reality [25]. Using several sensors allow users to freely navigate the interactive space. In such immersive virtual environments, realistic self representation enhance the perception of being there [34]. We designed a virtual reality gaming experience that takes advantage of full body point-cloud representations aided by body joints' positions. Therefore, we idealized a gaming zone where people have to catch basketballs thrown at them by three surrounding cannons. Figure 11A depicts a person trying to catch a basketball. We placed five cameras around a room for maximum body coverage as demonstrated in Figure 11B. *PointCloudData* was used to render the user's body and the tracker was set up to ignore the user's head when streaming. Human's hand joints were used to distinguish touching on a basketball. Still, all joints were used to build the person's bounding volume for the basketballs to collide and bounce back. Also, a *Surface* was used to define the gaming area.

## EVALUATION

We carried out a performance evaluation against a marker-based infrared tracking system as a baseline. The goals of our evaluation were to determine how the system's differs from a highly accurate marker-based tracker and determine body tracking consistency using a stress test. Besides, this evaluation serves to inform design decisions when developing future context-aware interfaces.

### Design

We devised three different evaluation tasks, depicted in Figure 12. For this, a logger application was developed to combine data from the two tracking streams by matching both coordinate systems. The logger was able to record, into file, sessions containing timestamps per frame, the spatial position from a person's right hand from our tracker and the position of a rigid-body marker attached to that person's right hand from the baseline tracker.

Therefore, the evaluation consisted of a latency task, and two tasks to measure the accuracy and consistency of our tracker by observing a single person holding an infrared marker. To calculate the average latency we recorded a raising gesture

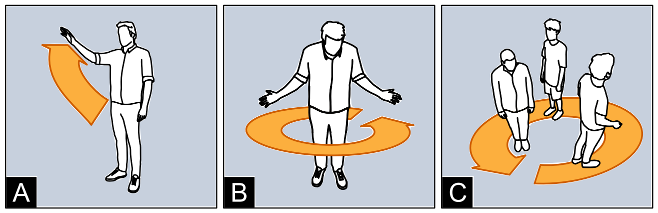


**Figure 12. Evaluation tasks: (A) hand raise, (B) spin about oneself, and a (C) circular path with multiple people.**

of the right hand five times. Figure 12A depicts the first task. The second task, depicted in Figure 12B, required a tracked person to fully spin about oneself to test for accuracy, with the purpose of forcing our tracker to switch to different sensors when choosing the adequate body model. Finally, the last task consisted of walking a circular path with two meters of diameter (Figure 12B). This required multiple sessions and incrementally adding a different person, until the tracked area was congested to the point that the observed person's tracking data was erratic due to body occlusions.

### Setup

The evaluation was conducted in a controlled laboratory environment with four by five square meters of tracked area. We employed five Microsoft Kinect version 2 depth cameras, evenly distributed around the room roughly forming a geometric pentagon. For the marker-based tracking system, we resorted to 12 Optitrack Flex 3 infrared cameras evenly placed around the room at 2.30 meters above the floor, providing a tracking volume surrounding the Kinect sensors' setup. To minimize the effects of network communication, both tracking systems and logger applications were running in the same desktop computer. Also, to obtain accurate positional data, all smoothing filters were disabled. Still, all sensor units were remotely streaming data within the same local wired network. We limited Optitrack's send rate to 100 frames per second and *Creepy Tracker* was set to 20 frames per second due to both Kinect and local network limitations.

### Results

Using the data thus collected we calculated the latency and position error of *Creepy Tracker* as measured against Optitrack. These allow us to assess how best to explore the flexibility vs accuracy tradeoffs.

### *Latency*

To measure latency we used the logs from the raising hand task, which are depicted in Figure 13. Logs had data recorded at an average of 170 frames per second. The difference in

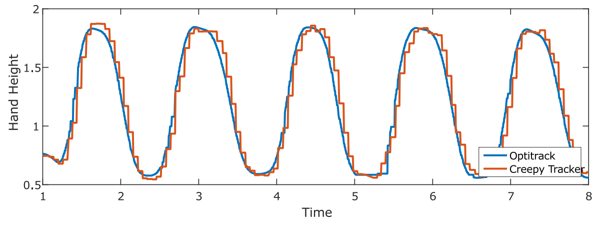
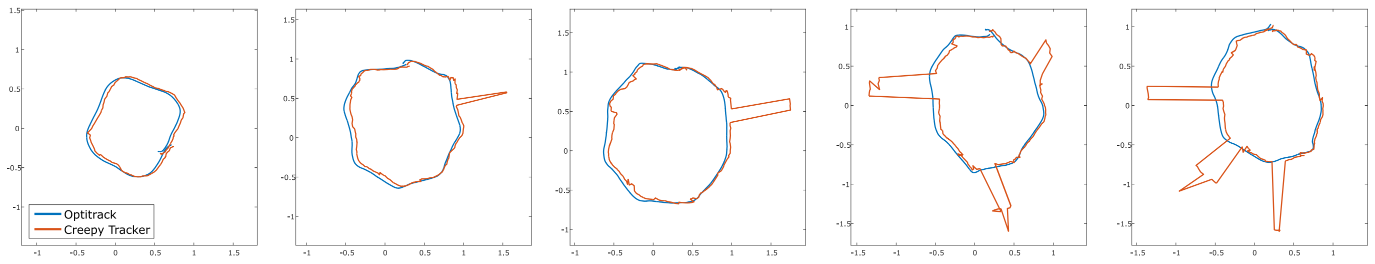


**Figure 13. Latency comparison between Optitrack and *Creepy Tracker*.**



a) Spinning about oneself (avg error = 62mm).
b) Circular path with 1 person (avg error = 80mm).
c) Circular path with 2 people (avg error = 93mm).
d) Circular path with 3 people (avg error = 134mm).
e) Circular path with 4 people (avg error = 127mm).

**Figure 14. Hand tracking data for each condition, and average position error.**

send rates justifies the less smooth data of our tracker, as visible in the chart. We calculated the difference between local maxima and minima of both trackers. *Creepy Tracker* had a delay averaging 76 milliseconds in comparison to Optitrack. Although this latency alone might not be enough for real-time performance in VR, combining *Creepy Tracker*'s positional data with the orientation provided by current head-mounted displays appears sufficient to fulfill the illusion of being there.

### *Accuracy*

We compared the hand position from each system, in five different conditions, illustrated in the plots of Figure 14. Although the plots are 2D we stored full 3D data. We averaged the Euclidean distance between both trackers' data in each frame. As evidenced in the spinning task, *Creepy Tracker* is capable of accurate results, where switching between sensors is not noticeable. While the average error was of 62mm, we estimate that it is in part due to latency. However, in more demanding scenarios, inaccuracies may arise. In all circular path tasks, spikes occasionally occurred. These correspond to sensor switching, where the new sensor perceives the tracked person on his back. While we try to deal with this by mirroring skeleton's information when suited, our approach is not yet perfect, resulting in right and left side swapping, including hands. This persists until a disambiguating pose is detected with certainty. These spikes are more recurring when tracked people's density increases, because sensors with non-optimal point-of-views are used to circumvent occlusions. When testing with more than four people in the same conditions, we noticed that *Creepy Tracker* sporadically lost the main subject for a brief moment. This originated a new identifier and invalidated the trial session.

### Discussion and Design Considerations

The obtained measurements are sufficient to delineate guidelines and design considerations for applications using the *Creepy Tracker*. Indeed, results of the latency task suggest that our tracker is adequate for context-aware scenarios and for more traditional input modalities, such as pointing techniques. Although, for virtual reality applications, latency is definitely just above the minimum threshold for virtual reality applications. Furthermore, results also show that for context and proximity-aware interactions the accuracy is satisfactory, despite the Kinect's relatively low resolution and noise. Except when using exclusively tracker data for selection tasks, targets should have radius of at least 15cm. Finally, as demonstrated by the results, increasing the density of people results in a increasingly more inaccurate tracking and sensor switching.

## LIMITATIONS

*Creepy Tracker* offers markerless full-body tracking of human and static surfaces in room-scale applications. However, certain limitations need to be taken into account when developing context-aware experiences. Currently, our toolkit models displays as planar rectangular static surfaces, but does not yet support handheld devices. Also, a person's orientation is not always consistent in crowded settings. This is because the skeleton provided by the toolkit will depend on the confidence values reported by each depth camera. As a user moves, different cameras become selected, creating unnatural transitions between frames. This also occurs when multiple people are inside the tracking space, which also leads to inconsistencies when occlusions between users are first detected.

## CONCLUSIONS AND FUTURE WORK

We developed the *Creepy Tracker* to facilitate researchers designing new interaction techniques for context-aware environments. Our toolkit offers real-time marker-less tracking of people, while providing the means to defining the exact position and orientation of interactive surfaces. A performance evaluation against a highly accurate marker-based system shows that our tracker is adequate to building and evaluating context-aware interactions. Yet, interaction designers need to accommodate tracking errors when considering scenarios that require high accuracy.

In the future we plan to employ multiple sensors to provide handheld device recognition for personal interaction and user identification, similarly to [43]. Moreover, we are also exploring different approaches to merging body tracking models to maximize both tracking consistency and accuracy.

## ACKNOWLEDGMENTS

This work was partially supported by: Fundação para a Ciência e a Tecnologia (FCT), through grants UID/CEC/50021/2013, IT-MEDEX PTDC/EEISII/6038/2014, and SFRH/BD/91372/2012; CAPES-Brasil Foundation, Ministry of Education of Brazil through the scholarship grant Ref. 9040/13-7; and the European Research Council under the project Ref. 336200.

## REFERENCES

1. Aggarwal, J. K., and Ryoo, M. S. Human activity analysis: A review. *ACM Computing Surveys (CSUR) 43*, 3 (2011), 16.
2. Annett, M., Grossman, T., Wigdor, D., and Fitzmaurice, G. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 337–346.
3. Antifakos, S., and Schiele, B. Beyond position awareness. *Personal and Ubiquitous Computing 6*, 5-6 (2002), 313–317.
4. Ark, W. S., and Selker, T. A look at human interaction with pervasive computers. *IBM systems journal 38*, 4 (1999), 504–507.
5. Augsten, T., Kaefer, K., Meusel, R., Fetzer, C., Kanitz, D., Stoff, T., Becker, T., Holz, C., and Baudisch, P. Multitoe: High-precision interaction with back-projected floors based on high-resolution multi-touch input. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST ’10, ACM (New York, NY, USA, 2010), 209–218.
6. Azimi, M. Skeletal joint smoothing white paper. Tech. rep., 2012. `http://msdn.microsoft.com/en-us/library/jj131429.aspx`.
7. Ballendat, T., Marquardt, N., and Greenberg, S. Proxemic interaction: Designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS ’10, ACM (New York, NY, USA, 2010), 121–130.
8. Beck, S., Kunert, A., Kulik, A., and Froehlich, B. Immersive group-to-group telepresence. *IEEE Transactions on Visualization and Computer Graphics 19*, 4 (2013), 616–625.
9. Bolt, R. A. Put-that-there: Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’80, ACM (New York, NY, USA, 1980), 262–270.
10. Dey, A. K., Abowd, G. D., and Salber, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction 16*, 2 (2001), 97–166.
11. Fails, J. A., and Olsen, J. D. Light widgets: interacting in every-day spaces. In *Proceedings of the 7th international conference on Intelligent user interfaces*, ACM (2002), 63–69.
12. Grønbæk, K., Iversen, O. S., Kortbek, K. J., Nielsen, K. R., and Aagaard, L. Igamefloor: A platform for co-located collaborative games. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE ’07, ACM (New York, NY, USA, 2007), 64–71.
13. Gugenheimer, J., Stemasov, E., Frommel, J., and Rukzio, E. Sharevr: Enabling co-located experiences for virtual reality between hmd and non-hmd users. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI ’17, ACM (New York, NY, USA, 2017), 4021–4033.
14. Hall, E. T. *The hidden dimension*. Doubleday & Co, 1966.
15. Harris, A., Rick, J., Bonnett, V., Yuill, N., Fleck, R., Marshall, P., and Rogers, Y. Around the table: Are multiple-touch surfaces better than single-touch for children’s collaborative interactions? In *Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1*, International Society of the Learning Sciences (2009), 335–344.
16. Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., and Schiele, B. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. *arXiv preprint arXiv:1605.03170* (2016).
17. Jokela, T., Ojala, J., and Olsson, T. A diary study on combining multiple information devices in everyday activities and tasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI ’15, ACM (New York, NY, USA, 2015), 3903–3912.
18. Jones, B., Sodhi, R., Murdock, M., Mehra, R., Benko, H., Wilson, A., Ofek, E., MacIntyre, B., Raghuvanshi, N., and Shapira, L. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST ’14, ACM (New York, NY, USA, 2014), 637–644.
19. 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 *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS ’12, ACM (New York, NY, USA, 2012), 31–40.
20. Marquardt, N., Diaz-Marino, R., Boring, S., and Greenberg, S. The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST ’11, ACM (New York, NY, USA, 2011), 315–326.
21. Marquardt, N., Hinckley, K., and Greenberg, S. Cross-device interaction via micro-mobility and f-formations. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST ’12, ACM (New York, NY, USA, 2012), 13–22.
22. Marquardt, N., Jota, R., Greenberg, S., and Jorge, J. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. *Human-Computer Interaction–INTERACT 2011* (2011), 461–476.

23. Marshall, P., Hornecker, E., Morris, R., Dalton, N. S., and Rogers, Y. When the fingers do the talking: A study of group participation with varying constraints to a tabletop interface. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, IEEE (2008), 33–40.

24. Matthews, T., Dey, A. K., Mankoff, J., Carter, S., and Rattenbury, T. A toolkit for managing user attention in peripheral displays. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, ACM (2004), 247–256.

25. Mendes, D., Relvas, F., Ferreira, A., and Jorge, J. The benefits of dof separation in mid-air 3d object manipulation. In *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, VRST '16, ACM (New York, NY, USA, 2016), 261–268.

26. Orts-Escolano, S., Rhemann, C., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., Kim, D., Davidson, P. L., Khamis, S., Dou, M., et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ACM (2016), 741–754.

27. Pejsa, T., Kantor, J., Benko, H., Ofek, E., and Wilson, A. Room2room: Enabling life-size telepresence in a projected augmented reality environment. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, ACM (2016), 1716–1725.

28. Pierce, J. S., Forsberg, A. S., Conway, M. J., Hong, S., Zeleznik, R. C., and Mine, M. R. Image plane interaction techniques in 3d immersive environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM (1997), 39–ff.

29. Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, ACM (New York, NY, USA, 1998), 179–188.

30. Rekimoto, J., and Nagao, K. The world through the computer: Computer augmented interaction with real world environments. In *Proceedings of the 8th annual ACM symposium on User interface and software technology*, ACM (1995), 29–36.

31. Schilit, B., Adams, N., and Want, R. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, IEEE (1994), 85–90.

32. Seyed, T., Azazi, A., Chan, E., Wang, Y., and Maurer, F. Sod-toolkit: A toolkit for interactively prototyping and developing multi-sensor, multi-device environments. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, ACM (New York, NY, USA, 2015), 171–180.

33. Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. Real-time human pose recognition in parts from single depth images. *Communications of the ACM 56*, 1 (2013), 116–124.

34. Slater, M., and Usoh, M. Body centred interaction in immersive virtual environments. *Artificial life and virtual reality 1*, 1994 (1994), 125–148.

35. Sousa, M., Mendes, D., Ferreira, A., Pereira, J. M., and Jorge, J. Eery space: facilitating virtual meetings through remote proxemics. In *Human-Computer Interaction*, Springer, Cham (2015), 622–629.

36. Tang, R., Alizadeh, H., Tang, A., Bateman, S., and Jorge, J. A. Physio@home: Design explorations to support movement guidance. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, ACM (New York, NY, USA, 2014), 1651–1656.

37. Vermeulen, J., Luyten, K., Coninx, K., Marquardt, N., and Bird, J. Proxemic flow: Dynamic peripheral floor visualizations for revealing and mediating large surface interactions. In *Human-Computer Interaction*, Springer (2015), 264–281.

38. Vogel, D., and Balakrishnan, R. Interactive public ambient displays: Transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, ACM (New York, NY, USA, 2004), 137–146.

39. Weiser, M. The computer for the 21st century. *Scientific american 265*, 3 (1991), 94–104.

40. Weiser, M. The computer for the 21st century. *IEEE pervasive computing 1*, 1 (2002), 19–25.

41. Wilson, A. D. Depth-sensing video cameras for 3d tangible tabletop interaction. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on*, IEEE (2007), 201–204.

42. Wilson, A. D., and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 273–282.

43. Wilson, A. D., and Benko, H. Crossmotion: Fusing device and image motion for user identification, tracking and device association. In *Proceedings of the 16th International Conference on Multimodal Interaction*, ICMI '14, ACM (New York, NY, USA, 2014), 216–223.

44. Wu, C.-J., Houben, S., and Marquardt, N. Eaglesense: Tracking people and devices in interactive spaces using real-time top-view depth-sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, ACM (New York, NY, USA, 2017), 3929–3942.