EagleSense: Tracking People and Devices in Interactive Spaces using Real-Time Top-View Depth-Sensing

Chi-Jui Wu[1], Steven Houben[2], Nicolai Marquardt[1]
[1] University College London, UCL Interaction Centre, Gower Street, London, UK
[2] Lancaster University, Lancaster, UK
{ chi-jui.wu.15, n.marquardt } @ucl.ac.uk, s.houben@lancaster.ac.uk

ABSTRACT
Real-time tracking of people's location, orientation and activities is increasingly important for designing novel ubiquitous computing applications. Top-view camera-based tracking avoids occlusion when tracking people while collaborating, but often requires complex tracking systems and advanced computer vision algorithms. To facilitate the prototyping of ubiquitous computing applications for interactive spaces, we developed *EagleSense*, a real-time human posture and activity recognition system with a single top-view depth-sensing camera. We contribute our novel algorithm and processing pipeline, including details for calculating silhouette-extremities features and applying gradient tree boosting classifiers for activity recognition optimized for top-view depth sensing. *EagleSense* provides easy access to the real-time tracking data and includes tools for facilitating the integration into custom applications. We report the results of a technical evaluation with 12 participants and demonstrate the capabilities of *EagleSense* with application case studies.

Author Keywords
Depth-infrared sensing; real-time top-view tracking; posture and activity recognition; phone and tablet recognition

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

INTRODUCTION
Applications in ubiquitous computing ecologies increasingly leverage information about the location of people and devices for the design of novel interaction techniques (e.g., [41,52]). Such detailed information about the relation between people, devices and objects can enable new interactions, such as large displays that react to the presence of people approaching them, mobile phones that connect to each other when in close proximity to allow easy transfer of information, or systems that can support ad hoc interactions with multiple devices as part of small group collaboration (e.g., [34,41]). In particular, previous work demonstrates that leveraging the nuances of proximity, distance, orientation and social relations between people, devices and other innate objects in the environment can lead to better context-aware applications and systems [14]. There is also a need to support interactions in multi-device interactive spaces with diverse sensing modalities and mobile devices [15]. To build such applications, systems need a detailed understanding of the physical space and activities of users, but also the physical location of devices and objects in use by those users. Moreover, to enable broad adaptation, developers need supporting infrastructures [9] that provide computational representations of spaces (e.g., relations between devices and people) to enable new types of walk-up and use applications [33].

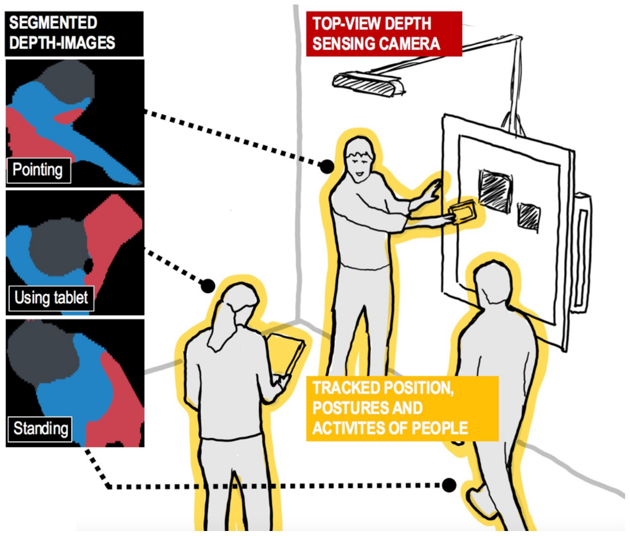

SEGMENTED DEPTH-IMAGES
Pointing
Using tablet
Standing
TOP-VIEW DEPTH SENSING CAMERA
TRACKED POSITION, POSTURES AND ACTIVITES OF PEOPLE

Figure 1. *EagleSense* real time tracking of people's location, postures and activities in interactive spaces.

Building a model of the space around the user requires sophisticated tracking hardware to allow systems to seamlessly blend devices into one shared interaction space. Building spatial tracking tools is often done using expensive and invasive tracking mechanisms that require specialized equipment, on-body markers, or even radios in devices [33,52]. Recently, depth-sensing cameras have been proposed as a non-invasive tracking technology for interactive systems

[8,18] and toolkits [33,38]. Many new interaction techniques leverage depth-sensing cameras to detect users' positions and gestures. Conventional use of depth cameras (such as [8,33,51]) rely on a front-facing setup that suffers from occlusion if multiple users interact with each other simultaneously. Recent work [18,34] showed that *top-view* (bird's-eye view) cameras can produce larger tracking areas, which enable tracking systems to support proxemics and cross-device interactions in interactive spaces, and most importantly while at the same time avoiding occlusion problems of front-facing cameras. However, beyond demonstrator systems, this approach is not well established or evaluated.

In this paper, we propose a new approach to top-down tracking of people and devices during collocated interaction. *EagleSense* (Figure 1) is a real-time tracking infrastructure using top-view depth-sensors capable of detecting users' position and orientation, as well as recognizing postures and activities, such as standing, sitting, pointing, using devices or reading paper documents. *EagleSense* provides a high performance, robust and precise activity recognition algorithm that recognizes both human activities, as well as device configurations, enabling a new range of interactive applications. It is a novel enabling technology designed to support building, designing and studying multi-device interactive spaces, body-centric spatial interactions and cross-device group interactions. To facilitate rapid prototyping of such applications, *EagleSense* provides a high-level web-based interface to access and use the spatial models in new applications.

We contribute a detailed description of our tracking and recognition pipeline, informed by the state-of-the-art computer vision tracking techniques, as well as a rigorous testing procedure going beyond earlier top-view tracking approaches [19,30] and systems [18,34]. Our method achieves 90.55% cross-subject activity recognition accuracy on a new dataset of 12 subjects. To demonstrate the capabilities and use of *EagleSense,* we present three use-case applications based on the RESTful tracking API. The *EagleSense* code, algorithms, dataset and tools are available as open source at https://github.com/cjw-charleswu/eaglesense.

RELATED WORK

Our work is informed by (i) computer vision algorithms for human detection and tracking, (ii) machine learning models for human activity recognition, and (iii) interactive systems using top-view depth-sensing cameras.

Human detection and tracking

HCI researchers used real-time human pose estimation (localization of skeleton joints) from single depth images [45] via commodity depth sensors such as the Microsoft Kinect [35] or Intel RealSense [22]. Li et al. [28] proposed a novel RFID-depth hybrid sensing approach for tracking both the identity and location of multiple people in groups. However, current commodity depth-sensing software (such as for the Microsoft Kinect [35]) does not provide the human skeleton joint positions when the camera is mounted at a top-down angle (used to minimize occlusions [34]).

Early research with top-view cameras focused on techniques for tracking people's position [20,42,47,50]. To address the lack of support for top-view human pose estimation for interactive tabletops, Haubner et al. [16] suggested to adopt Kinect's skeleton training pipeline [45] with a new dataset, where the subjects would wear color suits that have a distinct color-coding for each body part. Migniot et al. [36] developed a top-view multi-person skeleton tracking system using a particle filter. Recent approaches to human pose estimation with deep convolutional networks [21,48,49] show significant improvement on recognition in color images.

Human activity recognition

The availability of depth-sensing cameras has driven the research of human activity recognition using depth data. The use of depth over color images mitigates issues around lighting conditions, cluttered backgrounds, shadows, and occlusions [2]. However, conventional machine learning features used with RGB images for human activity recognition, such as Histogram of Oriented Gradients [7] and Space-Time Interest Points [27], are not discriminative in depth maps. Intuitively, the structure of skeleton joints can support identification of a range of human actions and activities. Therefore, recent work on human activity recognition [40,53,58,60] proposed view-invariant features based on the human skeleton (see [45]), including one or more of the following: pairwise joints differences, joints orientations, surface normals, depth occupancy patterns, and histograms composed of these features. Aggarwal et al. [1] provide a comparison of features from 3D depth data for human activity recognition.

This prior research contributed various datasets for human activity recognition, such as depth map sequences of game actions [29], color and depth images, along with skeleton joint positions, of daily activities and person-object interactions [53] (in multiple scenes [39]), and person-person interactions [57]. The number of participants in those datasets ranges from 10 to 30. Recent advances in deep learning and large-scale datasets (e.g., millions of samples and hundreds of classes) also improved the state-of-the-art human activity recognition of sports and actions from videos [24,46]. As those previous datasets are all captured by a front- or side-view camera, we contribute a dataset for top-view tracking of people and devices, facilitating further development of tracking systems as well as easier evaluation and comparison.

There is limited work on human activity recognition used with top-view cameras. Hu et al. [19] achieved moderate recognition accuracy on six different activities postures – standing, bending, sitting, pointing, stretching, and walking – with the mixtures-of-parts posture descriptor proposed by [62]. This setup requires a second depth-sensing camera to project the skeleton joint positions from the side-view coordinate system to the top-view coordinate system. Lin et al. [30] extended the top-view person detector from [50] to recognize six different activities.

Interactive systems using top-view cameras

Top-view tracking systems have previously been used to engage users in interactive games [4,26,31], to support interactions around tabletops [18,55,56] and to enable cross-devices interactions [34,41]. Wilson and Benko's LightSpace [56] slices 2D orthographic virtual camera planes through the unified 3D mesh (combining multiple top-view depth sensors) to facilitate recognition of human activities. DT-DT [18] is a top-view system that tracks people's position and a set of gestures, recognizing human activities within a 10-frames interval using 3D occupancy patterns of shape features [17]. They demonstrated the tracking system around a tabletop and a floor-projection interactive visualization, including a preliminary testing of the system with six users and four gestures. GroupTogether [34] is a top-view tracking system that recognizes the proxemics of devices *and* people, developed for cross-device content sharing (considering both micro-mobility and people's F-formations). Its hybrid tracking uses depth-data and radio signals trilateration for positioning people and devices. HuddleLamp [41] uses a RGBD-hybrid sensing approach for detecting stationary phones and tablets on a table from a lamp shade-sized depth-sensing camera, by applying standard contour finding and depth thresholding techniques and leveraging low infrared reflectance. More recently, DIRECT [59] proposes another novel depth-infrared sensing algorithm to improve finger and touch tracking on a flat table surface, consisting of infrared edge detection and iterative flood fills on the arm, hand, finger, and tips.

Our work builds upon these earlier approaches and integrates computer vision techniques into our novel processing pipeline to provide a real-time top-view human activity recognition system. *EagleSense* reliably recognizes people's position and orientation, as well as key interaction activities, at fast framerates of 30 Hz, allowing real-time interaction.

EAGLESENSE SYSTEM

EagleSense (Figure 1 and 2) is a top-view tracking system that combines depth and infrared data for human posture and activity recognition. It is an enabling tracking infrastructure that can be used, for example, by proxemic-aware [14] or cross-device systems [34,41] to detect and localize users and their activities, and recognize their uses of devices within an interactive space. In the following, we introduce the requirements for *EagleSense*, discuss the tracking system, and then guide through the key stages of the tracking pipeline.

Requirements

As outlined by Edwards et al. in the "Infrastructure Problem in HCI" [9], technical infrastructures are a core requirement in human interfaces that often directly influence and shape user experiences of technology and applications in user space. Therefore, to enable cross-device applications and interaction techniques to make use of a spatial tracking systems, the system needs to support five central requirements:

R1. **<u>Non-invasive tracking:</u>** To provide a "walk-up and use" experience, the tracking infrastructure needs to be non-invasive. This avoids the need for users to wear markers, tags or other equipment to be recognized by the system. Moreover, non-invasive tracking enables users to walk in and out of the space to opt in or out of the entire system.

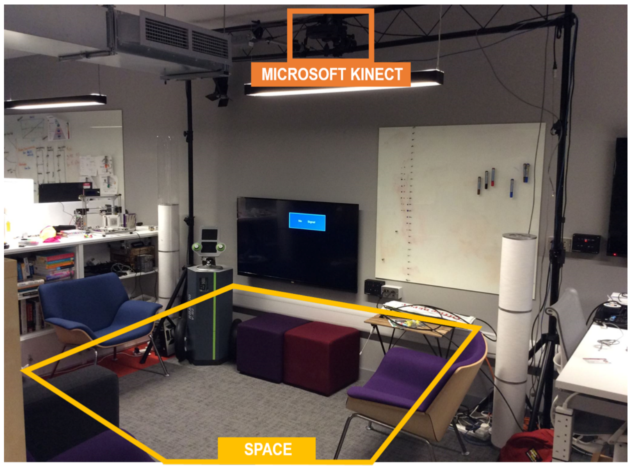


Figure 2. *EagleSense* physical setup.

R2. **<u>Off-the-shelf technology:</u>** By using off-the-shelf and affordable technology, the tracking infrastructure welcomes new developers, and integrates with existing toolchains, frameworks and design practices. Moreover, we envision that this technology could be further developed, refined and distributed to enable a broader audience to make use of the tracking infrastructure.

R3. **<u>Reliable real-time tracking</u>**: To provide a seamless interaction experience, the system needs to provide real-time tracking data. Delays in update rates for activity recognition and device detection should be minimized to maintain a fluent flow of tracking data. Such reliability in detection and timing consistency is required for operation consistency in user space.

R4. **<u>High-level access to spatial data</u>**: To enable cross-device system developers to design, deploy and test new systems, interaction techniques and applications – without the need for them to understand, use and configure low level tracking data – the tracking infrastructure should provide high-level access to real-time tracking data and update mechanisms.

R5. **<u>Configurability for activities</u>**: To allow for extendibility of the tracking infrastructure for scenarios and requirements beyond the standard set, it should allow for flexible definition, configuration and training of new activity recognition algorithms.

Tracking Overview and Technical Setup

EagleSense uses a ceiling-mounted down-facing depth camera at height of 264cm above the floor, allowing the tracking area of approximately 240cm by 170cm (Figure 2). The system detects people from the shape of contours in the depth frame after background subtraction, and creates a depth and infrared silhouette for each detected person (Figure 3AB). *EagleSense* segments the depth silhouette into three layers (Figure 3C), where silhouette-extremities features (Figure 3D-E) are extracted for posture and activity recognition. A gradient tree boosting method is used to classify the six different postures and activities. The *EagleSense* tracking pipeline consists of two phases: PHASE A for real-time segmentation of individual body layers, which is then used for PHASE B, the posture and activity recognition. The system was implemented in Visual C++ with three programming libraries: Microsoft Kinect SDK, OpenCV and XGBoost. The system runs at 30 frames per second on a 2.3GHz CPU and 8.0GB RAM computer. The machine learning model was trained on a single GTX660 2GB GPU. All the tests reported in this paper were done on this setup.

PHASE A: Top-View Depth-Sensing Tracking

The first phase of the *EagleSense* tracking pipeline is the real-time segmentation of a tracked person and their body parts. *EagleSense* localizes the head position and orientation, and produces a three-layers segmentation (Figure 3C).

Step 1 – Depth and infrared streams. Using the Kinect, the depth and infrared maps are acquired as 8-bit depth images of 512 by 424 pixels. The original 10-bit depth map is normalized to 8-bit, whereas the last 8 bits of the original 16-bit infrared map are removed. The latter procedure creates high contrast on the pixels of low infrared reflectance, which are used for the detection of mobile devices, as done in [41].

Step 2 – Human detection. *EagleSense* models the background from the first 120 depth frames using a Gaussian Mixture Models algorithm [64,65]. The learning rate of the background model is set to zero after the initial background frames are processed. When the scene is empty (i.e. no person detected) for 300 consecutive frames to ensure that the system does not mistakenly treat actual users as background, the background model is reset to the automatic learning rate until a detected person enters the field of view again. This simple mechanism allows the tracking system to run continuously without recalibration. To reduce depth-sensing noise from the camera and background subtraction, we smooth the foreground depth map using a 5 by 5 median filter, a common technique employed by previous work [44,50]. We keep the infrared map unfiltered to retain as much sparse infrared pixels on devices (which the system attempts to recognize) as possible, especially on small devices like mobile phones. In practice, the filtered depth map contains only actual users' contours. In rare occasions, the image may contain small artefacts, hence the system filters out small components, as done in [34], that have a size of less than the area of a circle with a radius of 30cm.

Step 3 – Human tracking. A person's center of mass on the contour, or the *body center*, is continuously tracked as soon as they are detected. However, depth-sensing cameras usually fail to estimate depths close by the field of view peripheral. This results in unreliable tracking when the person is near the tracking area boundary. In such cases, it is difficult to determine the person's head center position and orientation; for example, when people walking in and out of the tracking area. Therefore, *EagleSense* tracks people based on their body center position. To provide accurate tracking data,

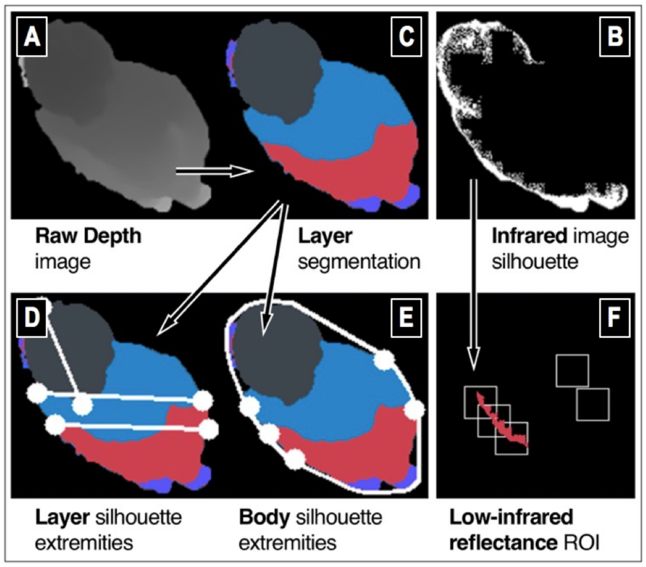

A
C
B
Raw Depth image
Layer segmentation
Infrared image silhouette
D
E
F
Layer silhouette extremities
Body silhouette extremities
Low-infrared reflectance ROI

Figure 3. Preprocessing of the depth (A) and infrared (B) silhouettes into three layers (C), layer (D) and body (E) extremities, and low infrared reflectance regions (F).

it defines an inner tracking area, or the *activity zone*, where it tracks other information about the users – their head position and orientation and their activity. In the current setup, the *activity zone* is defined as 15cm inward from the depth map dimensions, in which the person's head is mostly visible (the value is determined based on empirical observation of the evaluation dataset). A person is inside the *activity zone* if the highest point on the contour (minimum depth value) is within the *activity zone* bounding box.

Step 4 – Human depth and infrared silhouettes. Motivated by silhouette-based human activity recognition [5,29,61,63], which omits human pose estimation, we extend the use of silhouettes to human posture and activity recognition with a top-view depth-sensing camera. *EagleSense* extracts the person's depth and infrared silhouettes from the current frame using a bounding rectangle around their contour. We accentuate the dark pixels (with low infrared reflectance) on the infrared silhouette, by setting all white pixels (high infrared reflectance) to black then applying binary thresholding (Figure 3B and 3F). The detection of mobile devices from the infrared silhouette will be explained shortly. To enable real-time tracking performance, the depth and infrared silhouettes are downsampled to 32 by 32 pixels.

Step 5 – Body segmentation. Next, *EagleSense* segments the depth silhouette into three layers (Figure 3c and 4), informed by the methods of previous top-view tracking systems [18,30,34]. Related to our tracking approach, Lin et al. [30] proposed a model for six different activities. They threshold the person's depth silhouette at fixed intervals, and retain one contour from the first layer (head) and at most two contours from each of the other two layers (shoulder and body). The sequence of all contours centroid positions is aligned using Dynamic Time Warping, and then used as input into a Support Vector Machine classifier which achieved

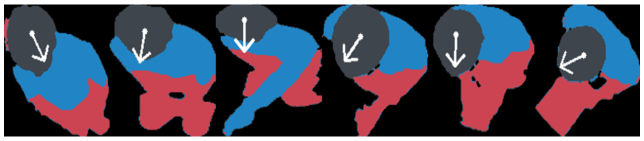


Figure 4. Body segmentation by depth for standing, sitting, pointing, using phone, using tablet, and reading paper (from left to right), and the estimated head orientation.

an average 98.15% accuracy on three random cross-subject tests (see technical evaluation). However, centroid positions are not ideal features for tracking and recognition algorithms, because some postures (e.g., standing and pointing) have similar centroids positions (Figure 4). Furthermore, their approach cannot distinguish between holding different devices and objects. We will address these limitations in Phase B.

In contrast to earlier work (e.g., [30]), *EagleSense* recognizes postures and activities from *single images* for achieving real-time performance, and we evaluate the methodology through a complete cross-subject test on a dataset consisting of more participants and samples. *EagleSense* improves upon earlier approaches on body segmentation for top-view depth maps. Instead of using fixed depth thresholds [16,30,34], we employed a K-means clustering algorithm to obtain the head and two body layers. The K-means clustering algorithm is the running time bottleneck of the *EagleSense* tracking pipeline, which we addressed in Step 4 by downsampling the input depth silhouette. In the implementation, K (number of clusters) and I (number of iterations) are both set to 5. We want to obtain three body parts and ignore the background (4 clusters). However, we found that the depth differences between the head and shoulder could be very small when tracking from above and after downsampling, especially when the person is far away from the depth-sensing camera. In such scenarios, the K-means clustering algorithm would produce undesirable results, for example, clustering the head and shoulder into one layer when the person is looking down at their device. Therefore, we set K to 5 and ignore the last cluster (usually the feet or the silhouette outline). We also join components other than the head with the second layer.

Step 6 – Head position and orientation. To estimate the head position and orientation, *EagleSense* fits an ellipse to the largest contour in the first segmented layer (head). Since the head contour in the downsampled silhouette is too small for a precise head orientation estimation, the head contour is upsampled with respect to the original depth silhouette dimensions. The head position is the center of the fitted ellipse, and the head orientation is the angle of the ellipse major axis (Figure 4), as done in [18,34]. In our implementation, we use the following method for tracking the human head orientation. We assume that when users enter the field of view, they are walking forward-facing into the scene tracked by the depth-sensing camera. This enables the system to resolve the initial head orientation from 180-degree ambiguity, and the head orientation in succeeding frames is subsequently updated. *EagleSense* captures and processes depth frames in real-time, so it sees any turning motion of a person's head. The orientation is transformed to a range between 0 and 360 degrees clockwise from the positive x-axis from the camera field of view.

PHASE B: Posture and Activity Recognition

In the second phase of our tracking pipeline, we describe how we recognize postures and activities from the three-layer body segmentation. Our results show that despite discarding the human pose estimation requirement (as studied by related work [21,45,48,49]), *EagleSense* can achieve high recognition accuracy with features based on extremities, or extreme points, on the depth and infrared silhouettes.

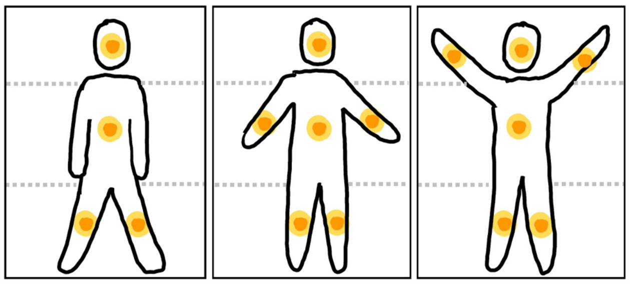


Figure 5. Human skeleton in three layers (front-view).

First, we provide a rationale for the chosen extremities from a simple human skeleton structure (Figure 5). The three layers should contain the head, upper-body, and lower-body, respectively. However, much of the human body is self-occluded in the top-down view to varying extent (e.g., see postures in Figure 4). Thus, we want to recognize the overall body or layer structure, rather than the local features [30]. Specifically, we extract patterns from the extremities which shape the corresponding posture or activity. We also avoid any assumptions about the relative positions of the shoulder, arms, and body [18,34]. Our current approach can recognize more postures and activities than those previously proposed, as captured by the dataset, including the use of devices (i.e. phones and tablets) and pointing gesture at different heights and orientations.

Step 7 – Extracting 3D interest points and 2D planes. Next, we extract several 3D interest points and two-dimensional planes from the three segmented body layers:

- **Layer centers and extremities (3 centers and 6 extremities).** *EagleSense* finds the mass center and the furthest two points (Figure 6, row 1) in each layer; these positions reveal the overall layer structure, position and orientation.
- **Layer contours (6 contours).** Each segmented layer consists of one or more contours (disconnected components) because of self-occlusion. To connect the body structure, we find three largest contours (with more than 5 points) in the 2nd and 3rd layer.
- **Body extremities (5 extremities).** Lastly, it finds the convex hull and convexity defects of the depth silhouette contour (Figure 6, row 2). Since we are most interested in locations where people hold mobile devices or

other objects (the arm is usually extended from the top-view), we sample at most five convexity defects (informed by the star-figure human silhouette model [5,63]). First, we remove points that are within the head layer. Then, we iteratively remove a point from pairs of convexity defects with the smallest Euclidean distance, until only five convexity defects remain. The convexity defects may fall outside of the actual silhouette contour (as returned by the convex hull algorithm), hence we correct their position to the nearest pixel on the contour within a 5 by 5 window. They are the final body extremities (Figure 6, row 3).

Step 8 – Calculating Feature Vector. Our algorithm considers the following 72 values in the feature vector:

- **Layer areas (3 features).** The ratio of nonzero pixels in each layer with respect to the total silhouette area.
- **Layer contour counts (2 features).** The number of contours in the 2nd and 3rd layer.
- **Layer maximal distances and areas (15 features).** The maximal 2D Euclidean distance in each layer. Also, the maximal 2D Euclidean distance and area in each of the three largest contours in the 2nd and 3rd layer.
- **Intra-layer positions (27 features).** The relative 3D positions between the center and the two extremities in each layer. Also, the relative 3D position between the two extremities in each layer.
- **Inter-layer positions (18 features).** For each of three largest contours in the 2nd layer, the relative 3D positions between the contour center and the 1st and 3rd layer center.
- **Body extremities (1 feature).** The number of body extremities.
- **Infrared of devices (6 features).** To detect mobile devices from the infrared silhouette, we calculate the area of the largest contour inside a 16 by 16 (half of the silhouette size) region of interest centered at each body extremity. In addition, we also calculate the area of the largest contour in all regions of interests combined (using all ROIs as a single mask).

Step 9 – Machine learning algorithm. In general, the task of human activity recognition is about recognizing dynamic actions, or a sequence of actions occurred within a time interval, for example the motion of standing up as opposed to the standing posture. Some work modelled the temporal dynamics of human actions using graphical models such as Gaussian Mixture Model, Dynamic Time Warping, and discrete Hidden Markov Model [29,30,58]. Some classified human activity sequences with Naïve Bayes Nearest-Neighbor and Support Vector Machine [53,60]. Deep learning community used Convolutional Neural Network to classify human activity videos at large-scale [24,46]. *EagleSense* does not model dynamic actions, instead, it recognizes postures and activities from single images (e.g., whether the person is currently using a tablet), because it is a supporting tracking infrastructure for ad-hoc interactions that provides real-time

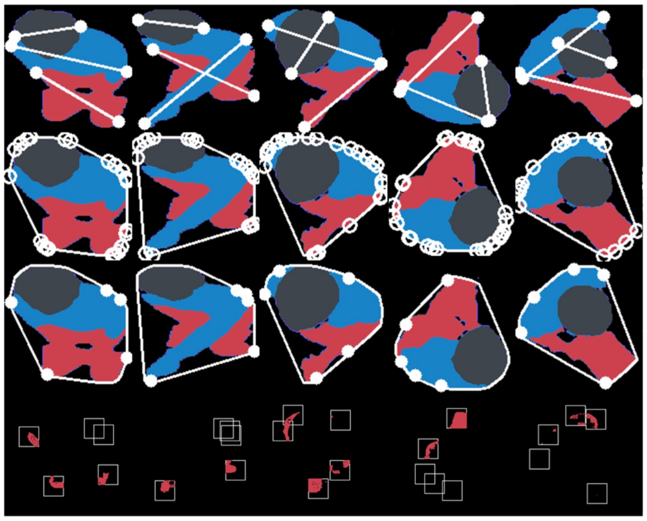


Figure 6. Preprocessing of the silhouettes for sitting, pointing, using phone, using tablet, reading paper (from left to right). First rows are layers extremities, second and third rows are unfiltered and filtered body extermities (convexity defects), and last rows are infrared ROIs.

information about people's position, orientation, their posture and use of mobile devices. After a comparison of machine learning algorithms, we chose the tree boosting system XGBoost [6] for posture and activity classification, because it provides the best performance and speed during both training and testing, and it leverages our proposed weak features.

For each incoming frame from the depth-sensing camera in real-time, *EagleSense* classifies each person into a posture or activity category, by using the features as inputs into the gradient boosting tree classifier (trained on our entire dataset). Although training was done in XGBoost's Python package, the C++ tracking application can use the classifier via the Python C API.

Step 10 – Physical setup considerations. For our setup, the average processing time on each image in the dataset (considering only images with a person tracked inside the *activity zone*) is 11.16msec (std=1.40msec), including an average processing time of 2.12msec (std=0.41msec) per person (from step 3 to the step 8). Thus, the *EagleSense* system scales linearly with the number of people tracked. The average processing time required to predict a posture or activity using our classifier is 0.24msec (std=1.79msec); further speedup is possible with parallel threads. The system can reliably track up to four people (and more in seated positions) with enough room for interactions at 30 fps in our physically constrained testing environment.

The Kinect v2 camera can sense depth from a maximum distance of 8 meters, although the data starts to degrade at 4.5 meters [35]. For top-view tracking we recommend to place the camera at least 2.5-3.0m above the ground floor to allow sufficient effective tracking area (better at 3.0-3.5m). We expect our method to degrade gradually as the height of the sensor increases, because lower reliability of the depth-sensing data will disrupt body segmentation in Step 5 as well as the detection of device infrareds in Step 8. On the other hand, mounting the camera higher increases the tracking area, hence the visibility of people and devices. We expect the system to operate well in environments suitable for time-of-flight (TOF) cameras (e.g., usually indoor environments). Previous research (e.g., [10,11,37]) suggests algorithms for resolving the multipath interference problem (from multiple reflections) in TOF cameras including Kinect v2 sensors.

Our current tracking and recognition pipeline assumes that the camera is placed at a top-down position, where the camera is closer to people's head and to the feet, allowing the body to be segmented by depth. This design decision was made to best minimize occlusions such as one obstructing another in the camera's field of view. In addition, the system assumes that people hold objects at some distance from the body; it would potentially fail to detect mobile devices if they are held at closer to a person's body.

EVALUATION

To evaluate the performance and reliability of our tracking algorithm, we conducted a technical evaluation of the *EagleSense* real-time tracking system.

Study Design and Tracking Dataset

We recruited 12 participants (5 male and 7 female university graduate students and staff, between 150 to 186 cm tall and 21 to 34 years old) to perform six different postures and activities: standing, sitting, pointing, using phone, using tablet, and reading paper. These specific postures and activities reflect frequent occurrences in everyday tasks [3] including multi-device uses [23]. Participants used their own mobile phones (with displays ranging from 2.4 to 5.0 inches) while performing the "using phone" activity (11 of the devices were smart phones, one was a feature phone). The participants used the same 9.7-inch tablet.

Moving beyond previous tracking evaluations, where participants were asked to perform repeated, short sequences of activities [29,30,39,40,53], we captured these activities in one complete recording session for each participant. Each activity lasted one minute (one session ~5 minutes), allowing us to better understand the system limitations in realistic scenarios. First, the participant walked randomly in any direction, occasionally walking in and out of the tracking area. Second, the participant picked up and read a paper about the current work. Third, the participants used their own mobile phones freely, and fourth, they played a game on the tablet. They could stand at any position and orientation during these activities (and they moved frequently in between different positions). Fifth, the participants solved a quiz by pointing gestures at a display while seating in front of a large screen. Last, they watched a short video, also in seated position.

A total of 84992 depth and infrared frames (512 by 424 pixels) were collected and labeled by the researchers (representing ~45 minutes of tracking recordings), excluding the initial background frames. An image was labelled as empty if the person's head is more than half-occluded or if they are entirely out of sight (93.84% were non-empty images). We used 77024 images (90.61%) in our evaluation, which consist of non-emptied images where the detected person is within the *activity zone.* To label and annotate the dataset for the testing of machine learning algorithms, we developed a web-based ground-truth image labeling application (which is part of our *EagleSense* open-source release and can be used by other researchers for further labeling of training data). The tool enables fast annotation of a large dataset by facilitating the tagging of the same label to multiple images at once; our entire dataset was labelled in less than an hour.

We performed three different tests to find out the recognition accuracy following the *EagleSense* tracking pipeline, using the same evaluation strategy employed by previous work [29,40,53]. These include (i) 1/3 samples test, (ii) 2/3 samples test, and (iii) cross-subject test. The 1/3 and 2/3 samples tests use one and two thirds of the dataset, respectively, for training and the rest for testing. Under these two tests, the training set is stratified-sampled from each activity category from each subject. The cross-subject test uses all the samples from half of the subjects in training, and the other half in testing. Our initial cross-subject test used samples from odd-numbered subjects in training and even-numbered subjects in testing. The average height of the subjects in the training and testing sets is 169.17cm and 168.5cm, respectively. The parameters of the tree boosting classifier were selected using 5-fold cross-validation. For example, the cross-subject test classifier consisted of 79 boosting trees with a maximum depth of 6 per tree. We conducted a complete cross-subject test that evaluated all possible combinations of cross-subject splits, as done in [53]. In our case, there are 12 (number of participants) choose 6 combinations of cross-subject tests (924 tests in total).

Results

Our method achieved 98.47% accuracy on the 1/3 samples test, 98.97% accuracy on the 2/3 samples test, 90.55% accuracy on the initial cross-subject test (Figure 7 left), and 89.48% accuracy on the complete cross-subject test (Figure 7 right). The result shows that the *EagleSense* tracking and recognition pipeline is generally robust across different subjects (>90% accuracy) in all posture and activity categories except "using phone" (63.82% accuracy). In this category, the classifier achieved only 43.92% on Subject 10 and poorly on Subject 2, 6, and 12 (58.60%, 59.85%, and 54.94%). It mostly misclassified "using phone" as "standing".

The gradient tree boosting classifier considers the infrared and area features (*infrared of devices* and *layer areas*) to be most discriminative (based on feature importance in the classifier). The number of nonzero pixels on the largest contour in all infrared ROIs combined help distinguish between holding different objects – phone, tablet, or paper. Tablets generally have large undetectable depth values (low infrared reflectance) on their surfaces, as discussed in [41], which corresponds to the values this feature extracts. In comparison, phones have less amount of low infrared reflectance, and paper almost none. The layer area ratio is also a good indicator

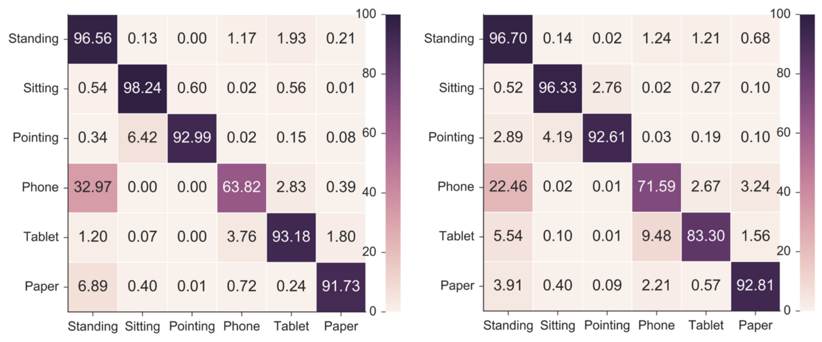


Figure 7. Activity recognition accuracy on the initial (left) and complete (right) cross-subject tests. X-axis: predicted. Y-axis: real.

of the current posture. Since the depth silhouettes are normalized to a fixed size, when a person is pointing, the relative size of the head becomes much smaller compared to the rest of the body. When a person is using a device, the second layer is usually the shoulder (smaller in size), instead of the front body (larger in size) when the person is standing.

We expected infrared-based features to have a strong influence on the recognition accuracy of activities related to holding an object: "using phone", "using tablet", and "reading paper". An experiment of the initial cross-subject test using all but infrared features (*infrared of devices*) showed that infrared features contributed considerably to the recognition of tablets (+33.12%) but only marginally to the recognition of phones (+10.27%), while the recognition accuracy of all other activities remained roughly the same (Figure 8).

Discussion of Evaluation Results

The *EagleSense* tracking and recognition pipeline achieved very high accuracy on both the 1/3 and 2/3 samples tests, as do previous work that employed the same evaluation procedure [29,40,53]. This is because the classifier was trained on almost all variations of the same classes that were sampled

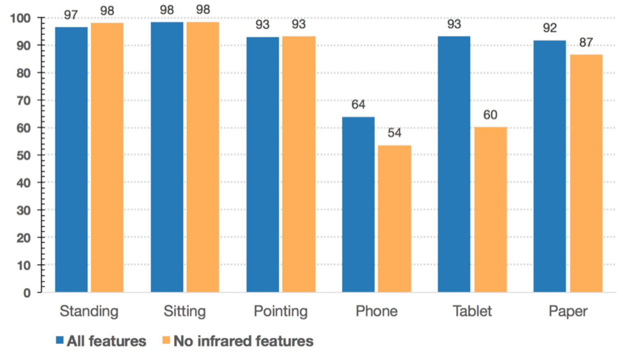


Figure 8. Activity recognition accuracy on the initial cross subject test, with and without using infrared features.

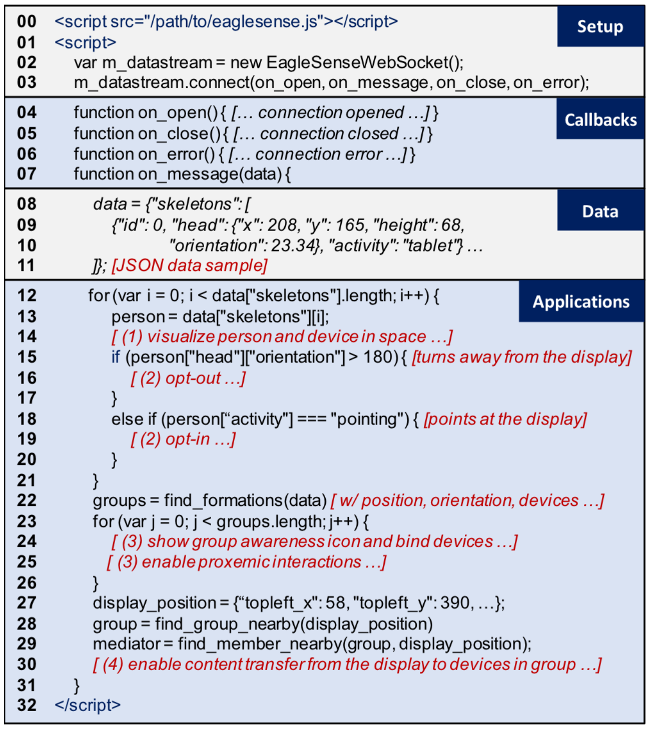
from all subjects. However, these results are not indicative of the out-of-sample performance, because they do not account for samples from unseen subjects, which would occur more frequently for in-the-wild deployment. Cross-subject tests mitigate this issue. The results between the initial and complete cross-subject tests very similar (Figure 7), except in the recognition of "using mobile phone" and "using tablet", which can be attributed largely to the differences between how people use devices. Although the method of using low infrared reflectance pixels to recognize these two objects in free motion is rudimentary, on average, the system still has good recognition results (mobile phone: 71.59%, tablet: 83.30%; Figure 7 right).

Compared to earlier work (e.g., HuddleLamp [41], which recognizes the position of phones and tablets on a flat surface), *EagleSense* attempts to recognize devices in a larger space where more different orientations are possible. However, mobile phones are relatively small to be segmented precisely from downsampled silhouettes. In our dataset, participants held their mobile phones with either one or two hands. Some held their phones closer to their body, while some held their phones further away. Overall, the dataset contains various snapshots of mobile phone usage at multiple locations and orientations. Furthermore, each subject used a different mobile phone with different display and frame sizes, causing different amounts of infrareds to be reflected and captured by the depth-sensing system. Some subjects' clothing or hair also reflected only small amounts of infrared light, hence they interfered with the detection of low infrared pixels and weakened the recognition of mobile devices. In summary, it is difficult to detect the presence of mobile phones simply by learning a pattern about the number of low infrared reflectance pixels (in top-view interactive spaces), as evidenced by the only marginal increase in the recognition accuracy of mobile phones after infrared features are included (Figure 8). However, this technique works well for recognizing tablets in 3D space, as large surfaces of low infrared reflectance are still possible to recognize after downsampling.

APPLICATIONS USING EAGLESENSE API

EagleSense provides a web-based API for easy access to the tracking data, facilitating the development of ubiquitous computing applications running on distributed devices. The RESTful API for *EagleSense* can send the tracking data to an IP address as a JSON string via HTTP POST. This allows the easy integration of the tracking data into new web-based ubicomp applications. The JSON data package (Figure 9, lines 8-11) includes an ID number of the tracked person, X/Y coordinates (registered to a zero calibration point in the tracking space), height (the vertical distance from the depth camera), head orientation angle, and the currently detected activity. This live streaming JSON data enables developers to design and implement new applications and systems without any need to access low level tracking code.

To implement a new application using the live tracking data, developers need to include only a few lines of code in their


Setup
Callbacks
Data
Applications

```
00  <script src="/path/to/eaglesense.js"></script>
01  <script>
02      var m_datastream = new EagleSenseWebSocket();
03      m_datastream.connect(on_open, on_message, on_close, on_error);
04      function on_open() { [… connection opened …] }
05      function on_close() { [… connection closed …] }
06      function on_error() { [… connection error …] }
07      function on_message(data) {
08          data = {"skeletons": [
09            {"id": 0, "head": {"x": 208, "y": 165, "height": 68,
10                "orientation": 23.34}, "activity": "tablet"} …
11          ]}; [JSON data sample]
12          for (var i = 0; i < data["skeletons"].length; i++) {
13              person = data["skeletons"][i];
14              [ (1) visualize person and device in space …]
15              if (person["head"]["orientation"] > 180) { [turns away from the display]
16                  [ (2) opt-out …]
17              }
18              else if (person["activity"] === "pointing") { [points at the display]
19                  [ (2) opt-in …]
20              }
21          }
22          groups = find_formations(data) [ w/ position, orientation, devices …]
23          for (var j = 0; j < groups.length; j++) {
24              [ (3) show group awareness icon and bind devices …]
25              [ (3) enable proxemic interactions …]
26          }
27          display_position = {"topleft_x": 58, "topleft_y": 390, …};
28          group = find_group_nearby(display_position)
29          mediator = find_member_nearby(group, display_position);
30          [ (4) enable content transfer from the display to devices in group …]
31      }
32  </script>
```

Figure 9. Partial source code for prototype applications, implemented with the web-based *EagleSense* API.

JavaScript web applications (we include an example RESTful server and a JavaScript library in our open-source package to facilitate prototyping applications). First, the application needs to establish a WebSocket connection with the server (Figure 9, lines 0-3). The *EagleSense* server then streams the real-time tracking data to the front-end applications. The application handles the data stream through four custom callback functions (Figure 9, lines 4-7).

To demonstrate the tracking capability of the *EagleSense* system and the application of the API, we show partial code snippets for three small example applications (Figure 9, lines 12-32):

1. **Visualizing people's walking trajectories:** For our first example application, we created a visualization of people's walking trajectory in the environment. Such a visualization can support quantitative analysis of interactive systems (e.g., large public displays) by revealing people's movement patterns and multi-device use cases when interacting with the system. To implement this visualization application, the JavaScript code simply needs to iterate over the data structure containing the currently tracked people (Fig. 9, line 12) and can then use the X and Y coordinates of each person to visualize them on screen (our example uses a decaying algorithm over the last 1000 frames). The trajectories can then be visualized on the JavaScript canvas (shown in Figure 10, in two different hexagon grid sizes). A possible extension of this application could visualize other characteristics of people's interaction, such as people's orientation and/or use of devices.

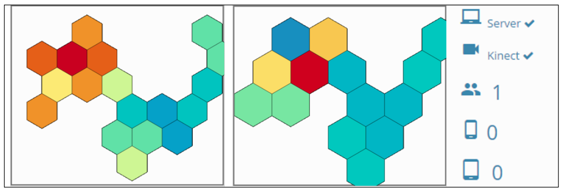


Figure 10. Visualising walking trajectories.

2. **Opting-in and out of interactions with a large display:** In our second application, we developed an example illustrating techniques for opt-in and opt-out gestures allowing a person to begin/end the interaction with an interactive game shown on a large display. After performing the pointing gesture at the display (Figure 11), the game begins and a person can control the game through body movements. If the user turns away from the display (based on *EagleSense*'s estimation of their head and body orientation), the game is paused and the user is implicitly opting-out of the interaction. This is implemented by monitoring the orientation of all currently tracked people (Fig. 9, lines 15-21).

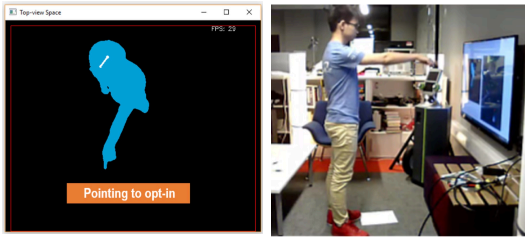


Figure 11. Using implicit and explicit gestures for opting-in and opting-out from interaction with wall display.

3. **Cross-device interactions and interactions based on people's formations:** our last example is informed by Greenberg et al.'s notion of proxemics interactions [13], and applications of cross-device interactions based on people's proximity and orientation [34]. It interprets the location of both currently tracked people and devices (Fig 9, lines 22-26) to identify which person is using their mobile device or tablet in close proximity to the large screen (Figure 12) in order to facilitate content transfer between mobiles and the large screen (Fig. 9, lines 27-30).

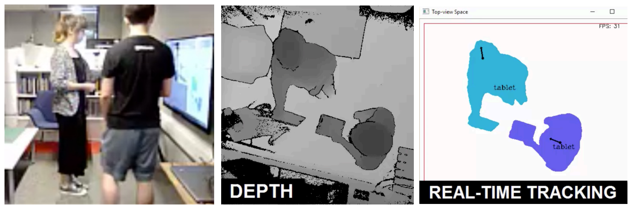


Figure 12. Recognizing two people and their devices when in front of large screen (left to right: space, depth image, real-time tracking of EagleSense).

While these are intended as starting points for applications, the JavaScript code snippets illustrate how the web-based API of *EagleSense* makes the real-time tracking data easily accessible for ubicomp applications, proxemic interactions or cross-device interactions.

EagleSense RESTful API focuses on providing real-time tracking data of devices, people, and activities, to enable developers and researchers to build new applications. However, inherent to the availability of REST clients, the infrastructure can easily be integrated with any tool, application or programming environment. For example, it enriches the space of depth-sensing systems for cross-device applications including XDKinect [38], HuddleLamp [41], GroupTogether [34]. By combining application models like Webstrates [25] or Connichiwa [43] with *EagleSense*, developers can quickly start crafting new spatially-aware multi-device interfaces. In future work, we aim to integrate a basic information and coordination mechanism into the core infrastructure to enable RESTful communication exchange between detected devices. Similar strategies, such as the QR code pairing mechanism used in HuddleLamp [41], can be used to pair recognized devices to the room-based information system. Further, we want to leverage multi-device support to provide proxemics and cross-device interactions over large, extended tracking spaces.

DISCUSSION

EagleSense contributes to the development of enabling infrastructures for ad-hoc, multi-device interactions in ubicomp ecologies. *EagleSense* supports recognition of different mobile devices (i.e. phones and tablets), extending previous work on proxemics [33] and cross-device interactions [34,41], and the platform provides applications and tools to extend the infrastructure's tracking capabilities. We now discuss strategies to further advance the real-time tracking provided through the *EagleSense* platform.

Dataset

In our dataset, we only captured a small subset of possible scenarios of the six postures and activities (for example, the subjects only performed the pointing gesture in seated position). There are fewer random actions in the dataset than to be expected in real-world deployments, but during training we included images that did not fit into any posture or activity category, such as the bending down/standing up motion, scratching, and attending to other objects (e.g., watch). Most subjects stood still while holding various devices and objects, which is also often the case in real-life scenarios, but this limits the range of body orientations captured during activities incorporating the use of these form factors. To ensure that any incremental improvement to the *EagleSense* infrastructure is also validated, future work will also build towards a heterogeneous testing dataset.

EagleSense System

Earlier in Step 10 we summarize the considerations for our setup. Although more validations are needed to assess the accuracy and effectiveness of the system in multiple environments in the wild, we expect, based on our initial dataset, that the methodology would generalize to new situations. In future work, we propose enriching the model by gathering more in situ data with depth-sensing cameras mounted at different positions and heights. We envision an iterative development cycle for *EagleSense* where infrastructure improvement would progress with interaction techniques research using this system. We provide tools to enable future collection and labelling of datasets, and for future work we are interested in providing an interactive interface for training machine learning models for the *EagleSense* infrastructure.

Recognition Pipeline

The *EagleSense* recognition pipeline can be improved by first performing an orientation normalization [53,54] on the human silhouette. The recognition algorithm currently does not process any color images, but the system could incorporate color data to enable more robust detection of mobile devices and other objects in the environment (e.g., interactive surfaces), for example using convolutional neural networks. Furthermore, the dataset only includes activities performed by single subjects. Although from these individual activities, group patterns and configurations can easily be detected in application space, we are interested in including tracking methods for automatically detecting interactions between multiple people or more nuanced cross-device interactions (e.g., group formations or micro-mobility [34]).

Tracking Space

The *EagleSense* infrastructure is currently only evaluated with a single depth-sensing camera. However, since it readily provides real-time tracking of users, devices, and activities, we expect it to become a part of a larger tracking ecosystem consisting of multiple cameras and sensors. For example, this could advance strategies such as those in the Gestures Everywhere framework [12] which integrates sensing across multiple spaces to provide both low- and high-level tracking information about the users and groups. A generic and ubiquitous infrastructure for ad-hoc interactions, by combining *EagleSense* and other space models (e.g., [32,52]), can enable visualizations of interactive spaces as well as opting-in and opting-out interaction techniques at scale.

CONCLUSION

We presented *EagleSense*, a novel real-time tracking infrastructure for interactive spaces leveraging top-view depth-sensing cameras. *EagleSense* is an enabling technology designed to support building, designing and studying interactive spaces and cross-device group interactions. We also provide a new public dataset that enables researchers to empirically compare and study tracking algorithms for ad hoc collocated interactions in interactive spaces. The EagleSense system, the RESTful API, the testing dataset and accompanying tools support the development of spatially-aware multi-device applications are available as open source: https://github.com/cjw-charleswu/eaglesense.

REFERENCES

1. J. K. Aggarwal and Lu Xia. 2014. Human activity recognition from 3D data: A review. *Pattern Recognition Letters* 48: 70–80. https://doi.org/10.1016/j.patrec.2014.04.011

2. J.K. Aggarwal and M.S. Ryoo. 2011. Human Activity Analysis: A Review. *ACM Comput. Surv.* 43, 3: 16:1–16:43. https://doi.org/10.1145/1922649.1922653
3. Ling Bao and Stephen S. Intille. 2004. Activity Recognition from User-Annotated Acceleration Data. In *Pervasive Computing* (Lecture Notes in Computer Science), Springer. 1–17. https://doi.org/10.1007/978-3-540-24646-6_1
4. Aaron F. Bobick, Stephen S. Intille, James W. Davis, Freedom Baird, Claudio S. Pinhanez, Lee W. Campbell, Yuri A. Ivanov, Arjan Schütte, and Andrew Wilson. 1999. The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment. *Presence: Teleoperators and Virtual Environments* 8, 4: 369–393. https://doi.org/10.1162/105474699566297
5. D. Y. Chen, S. W. Shih, and H. Y. M. Liao. 2007. Human Action Recognition Using 2-D Spatio-Temporal Templates. In *2007 IEEE International Conference on Multimedia and Expo* (Conf. on Multimedia and Expo), 667–670. https://doi.org/10.1109/ICME.2007.4284738
6. Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM. 785-794. https://doi.org/10.1145/2939672.2939785*
7. N. Dalal and B. Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 886–893, vol. 1. https://doi.org/10.1109/CVPR.2005.177
8. Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. 2014. SpiderEyes: Designing Attention- and Proximity-aware Collaborative Interfaces for Wall-sized Displays. In *Proceedings of the 19th International Conference on Intelligent User Interfaces* (IUI '14), ACM. 143–152. https://doi.org/10.1145/2557500.2557541
9. W. Keith Edwards, Mark W. Newman, and Erika Shehan Poole. 2010. The Infrastructure Problem in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10), ACM. 423–432. https://doi.org/10.1145/1753326.1753390
10. M. Feigin, A. Bhandari, S. Izadi, C. Rhemann, M. Schmidt, and R. Raskar. 2016. Resolving Multipath Interference in Kinect: An Inverse Problem Approach. *IEEE Sensors Journal* 16, 10: 3419–3427. https://doi.org/10.1109/JSEN.2015.2421360
11. Daniel Freedman, Yoni Smolin, Eyal Krupka, Ido Leichter, and Mirko Schmidt. 2014. SRA: Fast Removal of General Multipath for ToF Sensors. In *Computer Vision – ECCV 2014*, Springer. 234–249. https://doi.org/ 10.1007/978-3-319-10590-1_16
12. Nicholas Gillian, Sara Pfenninger, Spencer Russell, and Joseph A. Paradiso. 2014. Gestures Everywhere: A Multimodal Sensor Fusion and Analysis Framework for Pervasive Displays. In *Proceedings of The International Symposium on Pervasive Displays* (PerDis '14), ACM. 98:98–98:103. https://doi.org/10.1145/2611009.2611032
13. Saul Greenberg, Sebastian Boring, Jo Vermeulen, and Jakub Dostal. 2014. Dark Patterns in Proxemic Interactions: A Critical Perspective. In *Proceedings of the 2014 Conference on Designing Interactive Systems* (DIS '14), ACM. 523–532. https://doi.org/10.1145/2598510.2598541
14. Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. 2011. Proxemic Interactions: The New Ubicomp? *ACM Interactions* 18, 1: 42–50.
15. Jens Grubert, Matthias Kranz, and Aaron Quigley. 2016. Challenges in Mobile Multi-Device Ecosystems. *mUX: The Journal of Mobile User Experience* 5, 1. https://doi.org/10.1186/s13678-016-0007-y
16. Nadia Haubner, Ulrich Schwanecke, Ralf Dorner, Simon Lehmann, and Johannes Luderschmidt. 2012. Towards a Top-View Detection of Body Parts in an Interactive Tabletop Environment. In *ARCS Workshops*, 135–246.
17. G. Hu and Q. Gao. 2010. A non-parametric statistics based method for generic curve partition and classification. In *2010 IEEE International Conference on Image Processing* (ICIP '10), 3041–3044. https://doi.org/10.1109/ICIP.2010.5654096
18. Gang Hu, Derek Reilly, Mohammed Alnusayri, Ben Swinden, and Qigang Gao. 2014. DT-DT: Top-down Human Activity Analysis for Interactive Surface Applications. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14), ACM. 167–176. https://doi.org/10.1145/2669485.2669501
19. N. Hu, G. Englebienne, and B. Kröse. 2013. Posture recognition with a top-view camera. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS '13), 2152–2157. https://doi.org/10.1109/IROS.2013.6696657
20. S. Ikemura and H. Fujiyoshi. 2012. Human detection by Haar-like filtering using depth information. In *2012 21st International Conference on Pattern Recognition (ICPR '12)*, 813–816.
21. Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. 2016. DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model. *Proceedings of Computer Vision – ECCV 2016, Volume 9910, Lecture Notes in Computer Science, Springer. 34-50.* https://doi.org/ 10.1007/978-3-319-46466-4_3
22. Intel. *Intel RealSense SDK, https://software.intel.com/en-us/intel-realsense-sdk*. Retrieved from https://software.intel.com/en-us/intel-realsense-sdk

23. Tero Jokela, Jarno Ojala, and Thomas Olsson. 2015. 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. 3903–3912. https://doi.org/10.1145/2702123.2702211
24. Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14). IEEE Computer Society, 1725-1732. http://dx.doi.org/10.1109/CVPR.2014.223
25. Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15), ACM. 280–290. https://doi.org/10.1145/2807442.2807446
26. Sami Laakso and Mikko Laakso. 2006. Design of a Body-driven Multiplayer Game System. *Comput. Entertain.* 4, 4. https://doi.org/10.1145/1178418.1178429
27. Ivan Laptev. On Space-Time Interest Points. *International Journal of Computer Vision* 64, 2–3: 107–123. https://doi.org/10.1007/s11263-005-1838-7
28. Hanchuan Li, Peijin Zhang, Samer Al Moubayed, Shwetak N. Patel, and Alanson P. Sample. 2016. ID-Match: A Hybrid Computer Vision and RFID System for Recognizing Individuals in Groups. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16), ACM. 4933–4944. https://doi.org/10.1145/2858036.2858209
29. W. Li, Z. Zhang, and Z. Liu. 2010. Action recognition based on a bag of 3D points. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops* (CVPRW '10), 9–14. https://doi.org/10.1109/CVPRW.2010.5543273
30. S. C. Lin, A. S. Liu, T. W. Hsu, and L. C. Fu. 2015. Representative Body Points on Top-View Depth Sequences for Daily Activity Recognition. In *2015 IEEE International Conference on Systems, Man, and Cybernetics* (SMC '15), 2968–2973. https://doi.org/10.1109/SMC.2015.516
31. Pattie Maes, Trevor Darrell, Bruce Blumberg, and Alex Pentland. 1994. ALIVE: Artificial life interactive video environment. In *AAAI Conference on Artificial Intelligence*. 1506.
32. Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. 2012. 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. 31–40. https://doi.org/10.1145/2396636.2396642
33. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. 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. 315–326. https://doi.org/10.1145/2047196.2047238
34. Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. 2012. 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. 13–22. https://doi.org/10.1145/2380116.2380121
35. Microsoft. *Microsoft Kinect SDK, https://developer.microsoft.com/en-us/windows/kinect*. Retrieved from https://developer.microsoft.com/en-us/windows/kinect
36. Cyrille Migniot and Fakhreddine Ababsa. 2014. Hybrid 3D–2D human tracking in a top view. *Journal of Real-Time Image Processing* 11, 4: 769–784. https://doi.org/10.1007/s11554-014-0429-7
37. Nikhil Naik, Achuta Kadambi, Christoph Rhemann, Shahram Izadi, Ramesh Raskar, and Sing Bing Kang. 2015. A Light Transport Model for Mitigating Multipath Interference in Time-of-Flight Sensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '15), 73–81. https://doi.org/10.1109/CVPR.2015.7298602
38. Michael Nebeling, Elena Teunissen, Maria Husmann, and Moira C. Norrie. 2014. XDKinect: Development Framework for Cross-device Interaction Using Kinect. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (EICS '14), ACM. 65–74. https://doi.org/10.1145/2607023.2607024
39. Bingbing Ni, Gang Wang, and Pierre Moulin. 2013. RGBD-HuDaAct: A Color-Depth Video Database for Human Daily Activity Recognition. In *Consumer Depth Cameras for Computer Vision*, Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren and Kurt Konolige (eds.). Springer, 193–208. https://doi.org/10.1007/978-1-4471-4640-7_10
40. Omar Oreifej and Zicheng Liu. 2013. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '13), 716–723. https://doi.org/10.1109/CVPR.2013.98
41. Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. 2014. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14), ACM. 45–54. https://doi.org/10.1145/2669485.2669500
42. Michael Rauter. 2013. Reliable Human Detection and Tracking in Top-View Depth Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '13), 529–534. https://doi.org/10.1109/CVPRW.2013.84

43. Mario Schreiner, Roman Rädle, Hans-Christian Jetter, and Harald Reiterer. 2015. Connichiwa: A Framework for Cross-Device Web Applications. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '15), ACM. 2163–2168. https://doi.org/10.1145/2702613.2732909
44. Loren Arthur Schwarz, Artashes Mkhitaryan, Diana Mateus, and Nassir Navab. 2012. Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing* 30, 3: 217–226. https://doi.org/10.1016/j.imavis.2011.12.001
45. Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. 2013. Real-time Human Pose Recognition in Parts from Single Depth Images. *Commun. ACM* 56, 1: 116–124. https://doi.org/10.1145/2398356.2398381
46. Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger (eds.). 568–576.
47. Shih-Wei Sun, Wen-Huang Cheng, Yan-Ching Lin, Wei-Chih Lin, Ya-Ting Chang, and Cheng-Wei Peng. 2013. Whac-a-mole: A head detection scheme by estimating the 3D envelope from depth image. In *2013 IEEE International Conference on Multimedia and Expo Workshops* (ICMEW '13), 1–4. https://doi.org/10.1109/ICMEW.2013.6618320
48. Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. 2014. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. In *Advances in Neural Information Processing Systems 27*. 1799–1807.
49. Alexander Toshev and Christian Szegedy. 2014. DeepPose: Human Pose Estimation via Deep Neural Networks. *In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '14), 1653–1660. https://doi.org/10.1109/CVPR.2014.214
50. T. E. Tseng, A. S. Liu, P. H. Hsiao, C. M. Huang, and L. C. Fu. 2014. Real-time people detection and tracking for indoor surveillance using multiple top-view depth cameras. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS '14), 4077–4082. https://doi.org/10.1109/IROS.2014.6943136
51. Jo Vermeulen, Kris Luyten, Karin Coninx, Nicolai Marquardt, and Jon Bird. 2015. Proxemic Flow: Dynamic Peripheral Floor Visualizations for Revealing and Mediating Large Surface Interactions. In Proceedings of *Human-Computer Interaction – INTERACT 2015*, Springer. 264–281. https://doi.org/10.1007/978-3-319-22723-8_22
52. D. Vogel and R. Balakrishnan. 2004. 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. 137–146. https://doi.org/10.1145/1029632.1029656
53. J. Wang, Z. Liu, Y. Wu, and J. Yuan. 2012. Mining actionlet ensemble for action recognition with depth cameras. In Proceedings of the *2012 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '12), 1290–1297. https://doi.org/10.1109/CVPR.2012.6247813
54. Jiang Wang, Zicheng Liu, and Ying Wu. 2014. Learning Actionlet Ensemble for 3D Human Action Recognition. In *Human Action Recognition with Depth Cameras*. Springer, 11–40. https://doi.org/10.1007/978-3-319-04561-0_2
55. Andrew D. Wilson. 2010. Using a Depth Camera As a Touch Sensor. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (ITS '10), ACM. 69–72. https://doi.org/10.1145/1936652.1936665
56. Andrew D. Wilson and Hrvoje Benko. 2010. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM Symposium on User Interface Software and Technology* (UIST '10), ACM. 273–282. https://doi.org/10.1145/1866029.1866073
57. Christian Wolf, Julien Mille, Eric Lombardi, Oya Celiktutan, Mingyuan Jiu, Moez Baccouche, Emmanuel Dellandréa, Charles-Edmond Bichot, Christophe Garcia, and Bülent Sankur. 2012. The liris human activities dataset and the icpr 2012 human activities recognition and localization competition. *LIRIS UMR 5205 CNRS/INSA*. Laboratoire d'Informatique en Images et Systmes d'Information.
58. L. Xia, C. C. Chen, and J. K. Aggarwal. 2012. View invariant human action recognition using histograms of 3D joints. In *2012 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW '12), 20–27. https://doi.org/10.1109/CVPRW.2012.6239233
59. Robert Xiao, Scott Hudson, and Chris Harrison. 2016. DIRECT: Making Touch Tracking on Ordinary Surfaces Practical with Hybrid Depth-Infrared Sensing. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces* (ISS '16), ACM. 85–94. https://doi.org/10.1145/2992154.2992173
60. X. Yang and Y. L. Tian. 2012. EigenJoints-based action recognition using Naive-Bayes-Nearest-Neighbor. In *2012 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW '12), 14–19. https://doi.org/10.1109/CVPRW.2012.6239232
61. Xiaodong Yang, Chenyang Zhang, and YingLi Tian. 2012. Recognizing Actions Using Depth Motion

Maps-based Histograms of Oriented Gradients. In *Proceedings of the 20th ACM International Conference on Multimedia* (MM '12), 1057–1060. https://doi.org/10.1145/2393347.2396382

62. Y. Yang and D. Ramanan. 2011. Articulated pose estimation with flexible mixtures-of-parts. In *2011 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '11), 1385–1392. https://doi.org/10.1109/CVPR.2011.5995741
63. E. Yu and J. K. Aggarwal. 2009. Human action recognition with extremities as semantic posture representation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 1–8. https://doi.org/10.1109/CVPRW.2009.5204242
64. Z. Zivkovic. 2004. Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition* (ICPR '04), 28–31 Vol.2. https://doi.org/10.1109/ICPR.2004.1333992
65. Zoran Zivkovic and Ferdinand van der Heijden. 2006. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters* 27, 7: 773–780. https://doi.org/10.1016/j.patrec.2005.11.005