CAPturAR: An Augmented Reality Tool for Authoring Human-Involved Context-Aware Applications

Tianyi Wang[1]*, **Xun Qian**[1]*, **Fengming He**[2], **Xiyun Hu**[1], **Ke Huo**[1], **Yuanzhi Cao**[1], **Karthik Ramani**[1,2]
[1]School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907 USA
[2] School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
[wang3259, qian85, he418, hu690, khuo, cao158, ramani]@purdue.edu

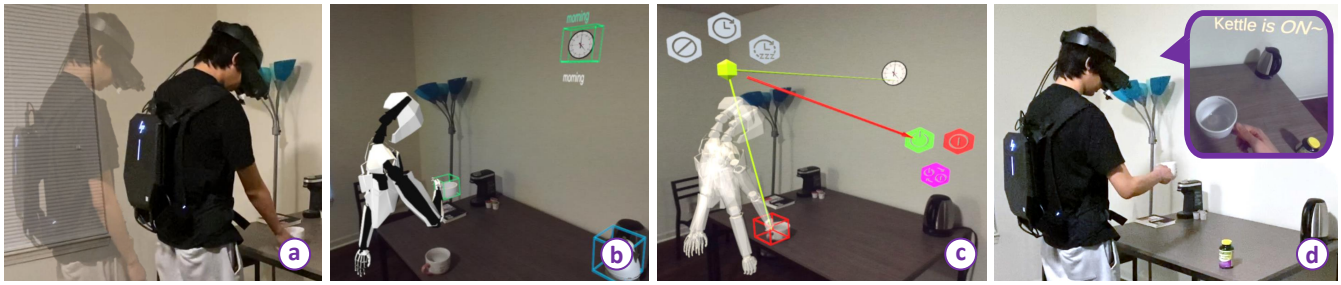

Figure 1. An overview of CAPturAR workflow. (a) A user conducts daily-life activities wearing a customized augmented reality head mounted device (AR-HMD). (b) The user visualizes his/her previous actions with contextual information in AR. The user can select certain context to search for relevant actions. (c) The user defines an *event* with a *human action* and several *context attributes*, and connects it to an IoT function to author a context-aware application (CAP). (d) When the *event* is detected in daily life, the IoT function is automatically triggered.

ABSTRACT

Recognition of human behavior plays an important role in context-aware applications. However, it is still a challenge for end-users to build personalized applications that accurately recognize their own activities. Therefore, we present CAPturAR, an in-situ programming tool that supports users to rapidly author context-aware applications by referring to their previous activities. We customize an AR head-mounted device with multiple camera systems that allow for non-intrusive capturing of user's daily activities. During authoring, we reconstruct the captured data in AR with an animated avatar and use virtual icons to represent the surrounding environment. With our visual programming interface, users create human-centered rules for the applications and experience them instantly in AR. We further demonstrate four use cases enabled by CAPturAR. Also, we verify the effectiveness of the AR-HMD and the authoring workflow with a system evaluation using our prototype. Moreover, we conduct a remote user study in an AR simulator to evaluate the usability.

*Tianyi Wang and Xun Qian contributed equally to this paper.

UIST '20, October 20–23, 2020, Virtual Event, USA.

ACM ISBN 978-1-4503-7514-6/20/10 ...$15.00.
http://dx.doi.org/10.1145/3379337.3415815

Author Keywords

Context-aware Application, Augmented Reality, End-user Programming Tool, Ubiquitous Computing, Embodied Authoring, In-situ Authoring

INTRODUCTION

The concept of ubiquitous computing [77] has been gradually substantiated by the rapid growth of the Internet of Things (IoT) products [53]. One of the critical differentiators between the emerging IoT and the classic telecontrol system is the intelligence introduced by IoT's context-awareness. Understanding the context of users and environments empowers the smart things to deliver timely and appropriate service without explicit interference from users [59]. With the IoTs acting as perception units, inferring environmental contexts, such as room temperature, lighting, moisture, etc., can be easily achieved. Although accurately inferring activity is an essential component of an advanced context-aware application (CAP), it remains challenging.

Firstly, human actions are **pervasive and spatial**. A meaningful action may happen anywhere, such as drinking coffee in a living room, doing yoga in a bedroom. Secondly, human actions can be **delicate and complex**. An action may involve the movement of the human body and both hands, and sometimes with objects. Thirdly, human actions are **ambiguous and subtle**. The intention of an action usually depends on relevant context information such as objects, location and time. For instance, picking up a cup in the morning and in the evening could suggest different intentions, i.e., drinking coffee and drinking milk.

One way of enabling pervasive human action detection is by embedding more advanced sensors into our surroundings, such as RFID [9], electric field [84], acoustic [41], and vision-based sensing [36, 57]. However, these sensors are embedded into the environment or the objects, which implies the scalability of CAPs will be greatly hampered. As an essential complementary component in IoT, wearable devices provide a promising approach to address the pervasiveness of human actions due to its always-on and always-with-user nature. Also, the CAPs built with wearable platforms are less dependent on the external infrastructures, as their perception capabilities are intrinsic.

Research has shown multiple non-vision approaches for action detection [13], but they often suffer from coarse granularity. And these methods are usually dedicated to human action detection, which may fail in cases of human-object interactions. Computer vision, on the other hand, is more accessible as a general human activity detection method. Moreover, to incorporate the challenge of pervasiveness, we need a wearable platform for the sensors. In particular, the emerging AR head mounted devices (AR-HMD) offer rich environmental sensing capabilities, including 6 degrees of freedom (DOF) tracking and an egocentric vision system that provides high-quality data for accurately inferring the delicate human-object interactions.

With a large amount of data, the computer vision community has made great progress on human action detection with pre-trained models [24, 86, 47]. But, when it comes to human-object interactions, users' spontaneous and diverse activities which are highly context sensitive, can easily invalidate a general model. Besides, the end-users can better disambiguate and interpret the contexts from their recorded actions[20, 8]. As an always-on wearable in the future, AR-HMDs have strong potentials for end-users to record their intentional and unintentional activities [58, 52] in a human-centered way.

Furthermore, compared to a traditional GUI, users directly experience the advantages of in-situ visualization of human activities through virtual human avatars and replicas of the objects in AR [11, 48]. An AR authoring interface allows users to intuitively view their previous actions and precisely label the desired motions for training. Moreover, users can freely walk around in the authoring AR scene and perform spatial interactions with the replicas of ordinary objects and IoTs [31, 33, 22]. This way, users can easily associate the motions with relevant context information from the environment and IoTs.

To this end, we propose CAPturAR, an AR authoring workflow, which allows users to record their daily activities, revisit the recorded scenarios, create and improve their personal context models, then build and deploy their own customized CAPs onto AR-HMD platforms. We demonstrate our workflow with a video-see-through AR-HMD modified with an additional downward-looking fisheye camera (Figure 1). Together with the front depth camera, we are able to reconstruct the 3D pose of the user's upper body and detect hand-object interactions in real-time. Wearing this device while conducting daily activities, users' moving trajectories, body actions, and hand interactions will be captured and associated with the map of the environment. In the authoring stage, the recorded scenarios are represented by an avatar and virtual replicas of the objects in AR. We design the interface of CAPturAR to allow fast navigation through the timeline and precise selection of the activity clips from the cluttered recordings. Then, based on users' understanding of their past behaviors, they interpret the selected demonstration clips as contexts and generate detection models with the motion data. Users can also designate necessary contextual information (e.g., time, location, objects) to disambiguate the activities. Further, users test human action detection performance and refine the context models through iterations. After users are satisfied with the detection performance, they can design the rules of the CAPs using our spatial programming interface in AR. After authoring, users can experience the functions of the CAPs instantly. In summary, we highlight our contributions as follows.

- An **all-in-one workflow** for creating human-involved context models using end-users' realistic daily activities, and authoring customized CAPs in AR.
- An **integrated AR-HMD platform** composed of multiple camera systems supporting non-intrusive recording of end-users' activities and context detecting while running CAPs.
- An **AR authoring interface** for browsing, selecting and editing previous activities, and creating flexible CAPs through spatial interaction and visual programming.

RELATED WORK

End-user Programming for Context-aware Applications

We focus on allowing end-users to define the contextual information and the desired task-relevant service [19] in a clear and intuitive manner. Towards this goal, an "*if...then...*" rule-based approach, namely trigger-action programming, is widely adopted. Many end-user programming interfaces are proposed to support fluent authoring of rule-based CAP. In a typical IFTTT flow [34, 76], users directly fill in a "*if <this> then <that>*" sentence with IoT related events to create a trigger-action pair. Alternatively, iCap [21] provided a GUI for end-users to define the trigger-action events through sketches and descriptions. To make the authoring process easy to understand, visual programming interfaces are implemented by substantiating events to visual representations such as icons and arrows [18, 65, 3], magnets [75], and jigsaws [32, 17]. Further, to provide an in-situ and spatial-aware authoring experience, researchers proposed tangible interfaces [7, 16, 44, 46]. By simply interacting with the IoTs, users can record their actions and create CAP based on the records.

Most of the proposed end-user programming interfaces are device-centered and limited to IoT-only interactions. Human actions, however, are not well supported in such interfaces mainly due to the lack of capabilities to detect and visualize human actions. CAPturAR supports always-on activity detection and enables end-users to customize sophisticated context models. We also expand the scope of human interactions with specialized IoT devices [45, 83, 67] to daily ordinary objects. Further, as an AR authoring tool, CAPturAR supports users to visually program the rules in-situ by spatially connecting the contexts with IoT functions.

Authoring through Embodied Demonstration

The embodied demonstration allows users to use shape, positioning, and kinematics of their bodies as spatial references and create complex and dynamic content intuitively. Researchers have applied embodied demonstrations in interactive 3D modeling [39, 43, 82], instant creation of stories and animations [27, 30, 66, 54], and generating realistic tutorials [10, 15, 28]. Recently, GhostAR [11] proposed a workflow where end-users program human-robot collaboration tasks using their demonstrations as space-time reference while Porfirio et al. [62] used human-human interactions as the demonstration to program human-robot interactions.

Further, embodied demonstrations have been leveraged to customize gestures or action detection algorithms. Lv et al. [50, 51] enabled end-users to design multi-touch gestures on tablets. ACAPpella [20] allowed users to interact with multiple sensors. Exemplar [29] and M.Gesture [38] supported rapid iteration and fine-tune of the gestures after the demonstration, while MAGIC [5] enabled users to build a classification algorithm by acting multiple gestures. Most of such works employed a typical workflow of *demonstrate-edit-test*. Namely, users first demonstrate the action, edit or label the captured data using a GUI, then perform the actions again to test the classifier. And users may have to go through the steps repeatedly to generate more demonstrations and improve the performance. Thus, such workflow works best for intentional and short gestures. CAPturAR focuses on capturing arbitrarily long-lasting human actions in daily life which include both unintentional and intentional patterns. Instead of acting the demonstrations one by one, CAPturAR provides rapid browsing and selection of desired actions from the cluttered and lengthy recordings. Moreover, we assist users to identify similar patterns by applying a pattern recognition algorithm to the entire record. Then, users can refine the action recognition algorithm by simply labeling the false positive and the true positive samples.

Human Action Detection in Smart Environment

As an enabling technology to achieve context-aware computing, human action detection has been extensively studied. Various types of sensors are developed to detect human action following either an environmental or wearable approach. Researches have experimented with multiple technologies for sensors deployed in the surroundings or on the objects, such as fiducial markers [14], RFID [60, 45], capacitive sensing [55, 67], electric filed sensing [84], vibration and sound detection [85, 41]. Recently, Sozu [83] also presented detecting activities by harvesting energy flow. Nonetheless, these methods require users to be close to the sensors or to interact with them. And the cost of deploying and maintaining the environmental sensors is usually high. On the other hand, wearable sensors can monitor human action continuously in an unobtrusive way. Typical wearable sensors includes accelerometers [42, 35, 2], GPS [63, 4] and biosensors [80, 68]. However, these sensors are not suitable for detecting complex and delicate motions. Also, in some cases, wearable sensor outputs are fused with other sources of information to achieve accurate detection.

Meanwhile, vision-based approaches are proposed by installing cameras in the environment [81]. Further, researchers use egocentric wearable cameras to detect human actions [61, 73, 23, 37, 49]. By sharing the same view with users, egocentric cameras capture the user's hand movements and detect the involved objects, which allows for accurate detection of complex context. More importantly, instead of using a front-looking camera, Xu et al. [79] and Tome et al. [74] used a downward-looking fisheye camera that covers humans' limbs. Thus the 3D human poses can be retrieved without carrying extra hardware, which allows users to freely conduct daily activities. Inspired by these works, we build a customized AR-HMD with multiple camera systems to empower the context-aware human action detection.

CAPTURAR SYSTEM DESIGN

Integrated AR-HMD Platform

We customize an AR-HMD that enables human and context perception for CAPturAR. Our prototype is composed of a VR headset, a forward-facing stereo camera, and a downward-looking fisheye camera as shown in Figure 2 (a). The stereo camera is responsible for providing video-see-through AR experiences. Also, it is equipped with object detection algorithm that tracks the 3D positions of surrounding objects. The fisheye camera covers humans' limbs within its field of view (Figure 2 (b)), supporting always-on reconstruction of the upper-body skeleton in a non-intrusive hands-free manner. The reconstructed human skeleton is used for human action detection and visualization in the AR authoring interface (Figure 2 (c,d)). We further detect human-object interactions by combining the human skeleton and object detection results. Additionally, we obtain the spatial trajectory of the user from the 6-DOF tracking supported by the VR headset.

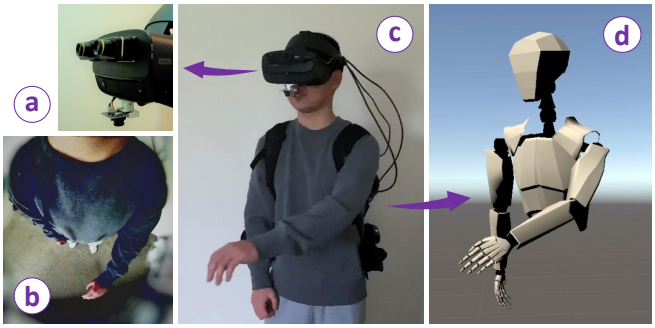

Figure 2. System hardware setup. (a) Customized AR-HMD with a stereo camera and a fisheye camera. (b) Fisheye camera view. (c) The AR-HMD is connected to a backpack computer. (d) Reconstructed upper-body skeleton.

System Walk-through

We demonstrate the workflow of CAPturAR with a rule-based CAP example in the teaser figure. After getting up in the morning, a user routinely picks up the cup and goes to the kettle to make a cup of hot tea. With CAPturAR, the user can create a CAP that automatically turns on the kettle to boil hot water as soon as the user picks up the cup in the morning. Before authoring a CAP, our system keeps recording the user's daily activities through the AR-HMD (Figure 1 (a)). During the authoring, these activities can be reconstructed and visualized in-situ using ***avatar cursor*** and ***context attributes*** (Figure 1 (b)). The user can select the *context attributes* of interest to

navigate the *avatar cursor* to the relevant past activities (Figure 1 (b)). Then the user trims a segment of picking-up action using the *avatar cursor* and includes another *context attribute*, the *cup* as an ***event***. The *event* enables CAPturAR to infer the user's intention and trigger an outcome if it happens. To improve the precision of the *event* definition, CAPturAR offers ***similar events*** function during the process. *Similar events* denote the situations where CAPturAR would infer the same intention as the authored *events*. For instance, after defining the *event* as a *picking-up* and a *cup*, by browsing the *similar events*, the user finds out that CAPturAR would mistakenly detects a holding-cup action as the same *event* due to the high similarity of the upper-body movement. Thus the user marks it as a negative example to avoid false positive detection. Additionally, the system would also recognize picking-up a cup in the evening as a *similar event*. It reminds the user to add another *context attribute* to the *event*, which is the *time* attribute, *morning*. As the user is satisfied with the *event* definition, he/she completes the CAP authoring by connecting it to an IoT function, turning-on the kettle, as illustrated in Figure 1 (c). Back to the user's life, when he/she picks up the cup in the morning again, the kettle will be automatically turned on, as shown in Figure 1 (d). To summarize the CAPturAR workflow, a user first finds and trims meaningful *human actions*, then attaches necessary *context attributes* to define *events*. After that, the user verifies the *events* by labeling the *similar events* and completes the CAPs by linking the *events* to IoT functions.

Framework of CAPturAR

The framework of our system is illustrated in Figure 3. CAPturAR borrows the metaphor from object-oriented programming [78] and substantiates the abstract context information as ***human actions***, ***context attributes*** and ***events***. The authoring workflow guides users to define *events* with *human actions* and *context attributes*, and create CAPs by connecting *events* with IoT functions.

A ***Human Action*** is a body movement that reveals the user's intention. In CAPturAR, we represent a *human action* by a sequence of ***human poses***. We let a user define a *human action* by trimming a segment of his/her past recorded actions.

Context Attributes are descriptors of the surrounding context. Dey et al. [21] made a summary of the categories to describe the context from end-users' perspectives: *activity, object, location, time, person, and state*. While the *activity* and *person* are usually used to describe users themselves, we design CAPturAR to perceive the following types of *Context attributes*.

- ***Object***, the object or IoT involved in the user's activities.
- ***Location***, the spatial property of the user.
- ***Time***, the time of the day.
- ***State***, the state of IoTs, e.g. light on/off.

The *human actions* and the *context attributes* are recognized by the AR-HMD and synchronously saved in a ***context database***, so that users can search for *human actions* by specifying the values of the *context attributes* and vice versa.

An ***Event*** comprises a meaningful *human action* and values of relevant *context attributes*. For instance, the user in the **CAPturAR system walk through** section creates an *event* with a *human action* of picking-up and two *context attributes*, the *object* attribute (*cup*) and the *time* attribute (*morning*).

A ***Similar Event*** of an authored *event* is a segment of the *context database* that holds the same *context attribute* values and similar *human actions* as the authored one. We introduce this function for two purposes. Firstly, *similar events* illustrate all situations where the authored *event* can be triggered, which helps reveal unnoticed constraints and improve the precision of the *event* definition. Secondly, we allow end-users to label the *similar events* to be negative or positive to improve the human action detection algorithm with more samples.

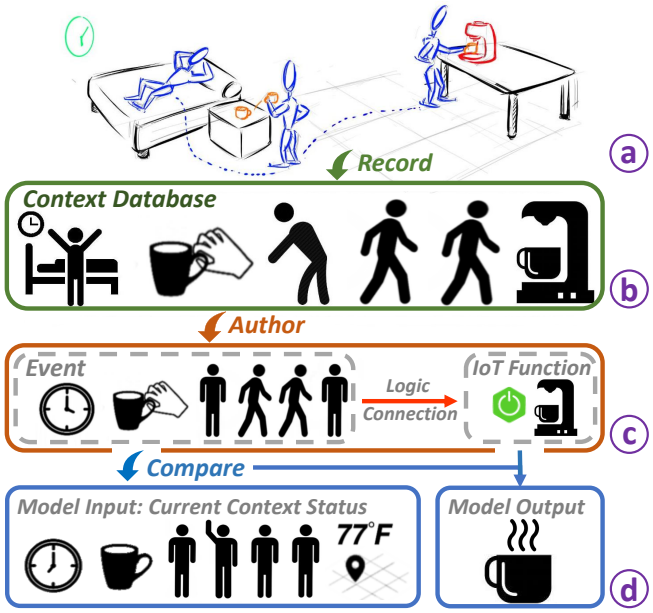

Figure 3. Framework of CAPturAR. (a) A user conducts daily activities. (b) The *human actions* and the status of *context attributes* are timestamped and recorded in *context database*. (c) The user creates an *event* by grouping the recorded activities and connect it with IoT functions to create a CAP. (d) CAPturAR compares the current context status with the *event*. If it happens, the IoT function will be triggered.

Event Detection and Function Triggering

As a rule-based authoring system, CAPturAR enables CAPs by detecting *events*. To detect an *event*, CAPturAR first acquires the current values of the *human action* and *context attributes* (*object*, *location*, etc.) and then compares them with the values saved in the *event* as shown in Figure 3 (c) and (d). The *event* is considered as happening if all these elements match.

To detect a *human action*, CAPturAR collects a sequence of *human poses* from the fisheye camera to form the ***current human action***, then calculates its Dynamic Time Warping (DTW) distance with the one saved in the *event*. CAPturAR assumes the *human action* of the *event* happens if the DTW distance is below a threshold. Further, if there are multiple *events*, CAPturAR uses the nearest neighbor algorithm to decide which one is actually happening.

For *context attributes*, CAPturAR considers an *object* attribute as being involved if it is spatially close to one of the user's two hands. Our system uses a fisheye camera to locate the

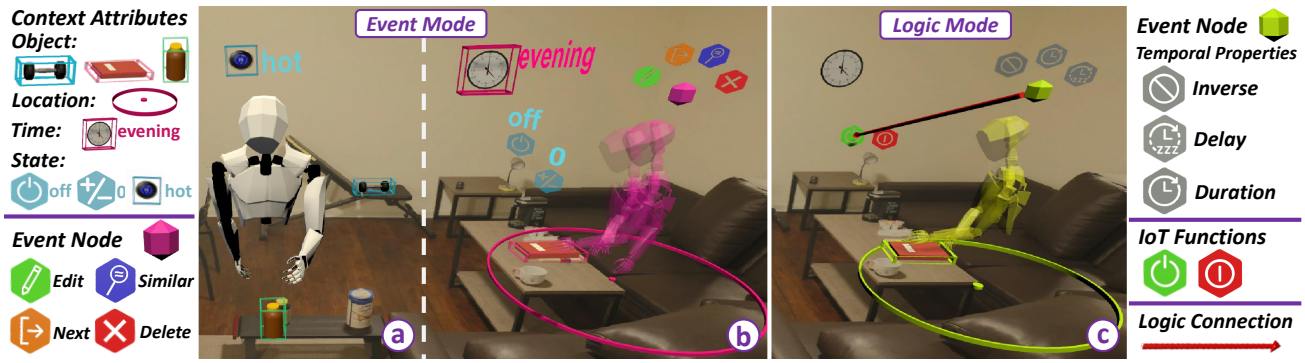

Figure 4. The authoring interface of CAPturAR. (a) The user can browse recorded activities in *Event Mode* by manipulating the *avatar cursor*. The pill bottle with green wireframe is an *object attribute* set to *suggestion* for rapid navigation of the *avatar cursor*. (b) An example *event* of *reading book on sofa in the evening* (magenta). It includes a *human action* (avatar clip) and three *context attributes* (evening, book, location circle). The user can toggle on an *event menu* from the *event node* to edit the *event*. (c) An example CAP of "*turn on the light when the event is detected*" authored in *Logic Mode*. The user connects the authored *event* from the *event node* to the *turn-on* icon of the table lamp with a red arrow. In *Logic Mode*, the *event menu* is replaced by the *temporal property menu* for toggling on/off *temporal properties* of the *event*.

hand positions and a front-facing stereo camera to obtain the object positions. The *location* attribute and the *time* attribute can be easily acquired from the 6-DOF spatial tracking and the internal clock of the AR-HMD. Meanwhile, the *state* attribute is acquired by communicating with the IoTs. Currently, CAPturAR uses simulated IoTs in AR while there is no difficulty scaling to real IoTs.

If a user has labeled *similar events*, CAPturAR compares the current status of the context with all *similar events* as well as the original *event* and uses a nearest neighbor algorithm. The *event* is detected if the *current human action* has a small DTW distance with any of the positive labeled *similar events*. In contrast, the *event* will not be considered happening if the *current human action* has a shortest DTW distance with a negative labeled *similar event*. Consequently, the false positive rate can be reduced while the true positive rate can be increased.

When an *event* is detected, usually the corresponding IoT functions are triggered immediately to deliver timely services. Additionally, we introduce three ***temporal properties*** of an *event*, namely ***inverse*** (triggering while it *does not* happen), ***delay*** (triggering after it happened for a while) and ***duration*** (triggering after it keeps happening for a while). Moreover, we allow logical connections among multiple *events*, namely ***sequential*** (triggering only if *event B* happens after *A*) and ***parallel*** (triggering if *A or B* happens). We embed these capabilities in CAPturAR authoring interface to meet the complicated requirements of CAPs.

In-situ Authoring in Augmented Reality

By leveraging the advantages of AR in in-situ visualization and spatial interaction, we are able to build an integrated authoring interface that combines low-level operations such as embodied demonstrations, with high-level visual programming that creates CAPs with flexible triggering logic. Users author CAPs through two *modes* sequentially: ***Event Mode*** and ***Logic Mode***. In *Event Mode*, users can browse the recorded activities by manipulating the ***avatar cursor*** and observing the status of *context attributes* (Figure 4 (a)). Users can set the *context attributes* as ***suggestions*** to rapidly navigate the *avatar cursor* within the long-lasting record. Then, users can define *events* by trimming a consecutive segment of the past actions with the *avatar cursor* and including other necessary *context attributes* (Figure 4 (b)). After creating the *events*, users can verify them by viewing the *similar events*. Further, users enter *Logic Mode* (Figure 4 (c)) to connect the *events* to IoT functions respectively to create CAPs. Users interact with the authoring interface through a VR hand-held controller. In the following paragraphs, we describe our UI designs in detail.

We use spatially distributed ***AR icons*** to support interactions between end-users and the surrounding contexts. In *Event Mode* (Figure 4 (a,b)), we use wireframed virtual models as the *object attributes*, a clock model as the *time attribute*, circles drawn by users as the *location attributes*, and hexagonal icons located next to the *objects* as the *state attributes*. Each of the *context attributes* can be in one of the following three states, namely *idle state* (cyan) when a user has not interacted with the attribute, *suggestion state* (green) when a user sets it as a *suggestion* to see relevant activities, and *event state* (magenta) when a user includes it in an *event*. In *Logic Model*, we use hexagonal icons located next to the smart things to represent IoT functions.

In *Event Mode*, we introduce the ***avatar cursor*** for conveniently browsing, visualizing and editing recorded actions. The *avatar cursor* is an in-situ placed human-avatar (Figure 4 (a)) that represents the pose of the user at a specific point of time in the past. Just like the cursors in video editing software, users can move the *avatar cursor* forward and backward in time domain to replay the recorded actions, while the values of all *context attributes* are synchronously updated and displayed as the *AR icons*. Such synchronization allow us to introduce the ***suggestion*** function for rapid navigation within the long-lasting record of previous activities. Instead of searching for an action from the *context database*, users can directly locate the *avatar cursor* to the relevant actions by selecting some *context attributes* as *suggestions* (Figure 4 (a)).

To define a *human action* in *Event Mode*, users trim a consecutive part of recorded actions using the *avatar cursor* in a hold-and-drag manner as similar as selecting a part of the text on PC. The selected *human actions* are displayed as semi-

transparent human avatars (Figure 4 (b)). Once a *human action* is selected, an *event node* and an *event menu* will be displayed above the avatars. Users can select or deselect a *context attribute* (edit button), find *similar events* in the *context database* (similar button), author another event (next button) or delete this event (delete button). The *similar events* are visualized closely to *events*. The menu of *similar event* let users to label it as a negative (delete button) sample or keep it as a positive sample (next button).

In *Logic Mode*, we implement a ***visual programming interface*** (Figure 4) (c). The *context attributes* which do not belong to any *event* are hidden, while the *event* and the icons of the available IoT functions are displayed. Users can build logic connections between the *events* and the IoT functions by drawing red arrows in AR to create CAPs. With the arrow metaphor, users can either connect one *event node* with another to implement a *sequential* logic, or connect multiple *event nodes* to one IoT functions to implement a *parallel* logic. Additionally, the *event menu* is replaced by the *temporal property menu* where users can toggle the *temporal properties* (*inverse*, *delay*, *duration*) to specify temporal triggering logic. Utilizing the temporal and logical relations between *events* and IoT functions, CAPturAR supports rule-based CAPs as discussed in [21] as well as process-driven CAPs as shown in [65].

USE SCENARIOS

With CAPturAR, users can program their actions to create context-aware applications and build smart interfaces for their surrounding environment. Here we demonstrate four different CAPs in household scenarios.

Augment Everyday Objects

CAPturAR is aware of the user's interaction with everyday objects. Leveraging the AR interface, users can attach digital functions, which can be triggered by their actions, to everyday objects. Here, we augment a kettle, a pill bottle and a wipe bottle with a timer function, a counter function, and a reminder function respectively (Figure 5).

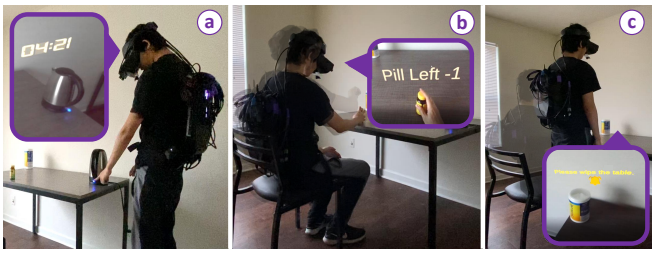

Figure 5. Augment everyday objects. (a) The time starts to count down when the user turns on the kettle. (b) The number of pills left in the pill bottle minus one when the user takes a pill. (c) The wipe bottle reminds the user to clean the table when the user stands up.

Workout Reminder

The object-oriented authoring interface of CAPturAR enables users to manipulate and connect multiple *events* and build CAPs based on a series of related activities. Here a user wants CAPturAR to remind him/her to do dumbbell-lifting every 30 minutes of reading. As shown in Figure 6, the user authors a reading-book *event* with a temporal property of *30 minutes duration*, and connects to a dumbbell-lifting *event* with an *inverse* property. The dumbbell-lifting *event* is further connected to a reminder function on the dumbbell. Thus, if the user has been reading for 30 minutes, CAPturAR will check if he/she has done any dumbbell-lifting. If not, a reminder will pop up above the dumbbell.

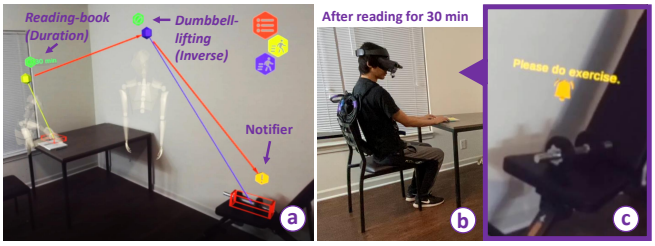

Figure 6. Workout reminder. (a) Two sequentially connected *events* with a *duration* and an *inverse temporal properties*. (b) The user has to do exercise every 30 minutes while reading. (c) The system reminds the user if no dumbbell-lifting was detected within the past 30 minutes.

Sequential Task Tutorial

Leveraging the realistic visualization of human actions in AR, CAPturAR can also create embodied and adaptive tutorials for sequential tasks. An instructor wants to demonstrate his/her routine task of repairing a bike, so he/she creates a CAP using CAPturAR with three sequentially connected *events*, shaking the lubricant, spreading it on the front wheel and then on the back wheel. A novice comes and follows the tutorial (Figure 7). CAPturAR detects once the novice completes a step and starts to play the demonstration of the next step.

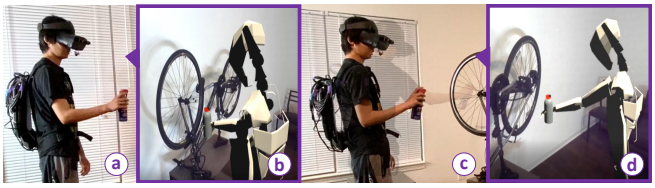

Figure 7. Sequential task tutorial. When the novice has finished the current steps (a, c), the next steps will be revealed automatically (b, d).

Tangible AR Game Creation

CAPturAR makes the user's surrounding environment a playground that can be interacted with through actions. Here a user used to throw cans into a rubbish bin, so he/she creates an AR game of basketball shooting as shown in Figure 8. Once he/she picks up an empty coke can, CAPturAR attaches a virtual basket and a virtual basketball to the rubbish bin and the coke can respectively.

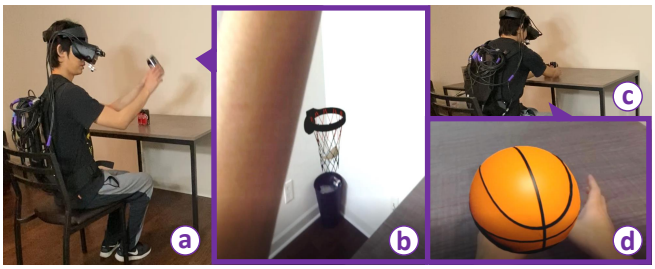

Figure 8. Tangible AR game creation. As the system detects a shooting action (a, c), the authored AR game is activated (b, d).

IMPLEMENTATION

System Hardware and Software Setup

We build our customized AR-HMD as shown in Figure 2 with 1) a VR headset (Oculus Rift S [1]) with SLAM embedded, 2) a front-facing stereo camera (ZED Dual 4MP Camera [72], 720p, 60fps) for video-see-through AR and object detection, and 3) a downward-looking fisheye camera (1080p, 60fps, 180 deg FOV) attached to the bottom of the VR headset for action recognition. The AR-HMD is connected to a backpack computer (HP VR Backpack G2, Intel Core i7-8850H, 2.6GHz CPU, 32GB RAM, NVIDIA RTX 2080 GPU). The CAPturAR authoring interface is developed using Unity3D(2019.2.12f1). To interact with the AR authoring interface, we use an Oculus Touch controller.

Retrieve Human Body Pose from Fisheye Camera View

We built and trained a deep neural network (DNN) to retrieve 3D body pose from the fisheye camera, as shown in Figure 9. Due to the limited field of view of the fisheye camera, we only focus on body poses of which hands are below the level of head. The DNN comprises two concatenated parts. For the first part we adopt the convolutional pose machine structure presented in OpenPose [12] with VGG19 [70] backbone to detect 2D locations and orientations of joints in fisheye images. For the second part, we use a customized convolutional neural network (CNN) to infer the 3D joints positions from 2D. The DNN runs on the backpack computer with Tensorflow 2.0 [26] at 24Hz. To further convert the joint positions into a realistic human avatar, we use FinalIK Unity3D plugin [25]. To train the DNN, we used a Kinect Azure [6] to collect ground truth data of the 3D joints position. Meanwhile, we took fisheye camera images as the training input. The ground truth of 2D joints locations and orientations were obtained by projecting the 3D joints positions onto the fisheye camera images [56]. In total, 170K images were collected from 35 volunteers. The two parts of the DNN are trained separately. The first part took 24 hours on an NVIDIA 2080Ti GPU, and the second took 12 hours on the same GPU.

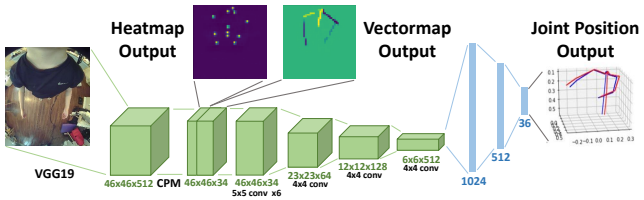

Figure 9. Upper-body tracking network structure.

Detect Interaction with Objects

To track 3D positions of objects, we first use the Yolo v3 [64] implemented in OpenCV for Unity [71] to find the 2D positions on the RGB image from the ZED stereo camera. Then we reproject the 2D position back to 3D using ZED's depth image. Any object which is out of the FoV is assumed to be stable. To detect user's interactions with small, movable objects, we measure the distance between the objects and the user's left or right hands and detect interaction if the distance is below 10cm. For large and fixed appliances such as lamps, we detect interaction whenever the user is close to the appliances. Note that the fisheye camera and the ZED have separated FoVs. To ensure the detection of both hands and objects, we slightly adjust the direction of the fisheye camera to cover the ZED's FoV. We trained the Yolo v3 to detect 15 daily objects including a cup, a mug, a kettle, a candy bottle, a pill bottle, a book, a tea can, a coke can, a wipes can, a pair of pliers, a dumbbell, a bowl, a bottle of hand sanitizer, a lamp, a rubbish bin, for study and demo purpose. For each object, we collected approximately 2000 images using the method mentioned in [40]. Specifically, we placed virtual 3D bounding boxes around the objects in AR and recorded the images from the ZED stereo camera. The 3D bounding boxes are projected onto images as 2D labels. The training took 12 hours on an NVIDIA 2080Ti GPU.

PRELIMINARY SYSTEM EVALUATION

The CAPturAR workflow relies on the capabilities of the integrated AR-HMD platform, namely upper-body pose tracking, human-object interaction, and *human action* recognition. To evaluate these capabilities, we conducted a 3-session preliminary system evaluation.

Accuracy of Upper-body Pose Tracking

Accurate tracking of the human body pose plays an essential role in human action detection and virtual avatar reconstruction. To test the tracking accuracy of our customized AR-HMD, we compared the 3D positions of the 12 upper-body joints acquired by the fisheye camera with the results from a Kinect Azure camera, which were used as ground truths. During the test, the tester performed 10 common movements such as *picking up*, *putting down*, and *reaching out with both hands* continuously while the data samples were collected at 4Hz automatically. Figure 10 (a) demonstrates four example sets of model inputs and outputs during the test. Three researchers participated in this test and collected 825 data samples in total. Then, we calculated the distances of all 12 joint positions between the ground truths and the model outputs, as shown in Figure 10 (b). The average position error was 4.34cm ($SD = 3.59cm$). Typically, pelvis and both hands produced larger errors since those joints were far from the pivot and had larger movements. The left hand produced the largest error, 8.56cm ($SD = 5.58cm$) which was still smaller than half of an adult's palm. Thus, the customized AR-HMD has a similar tracking ability as Kinect Azure and can satisfy the requirements of tracking human upper-body actions precisely.

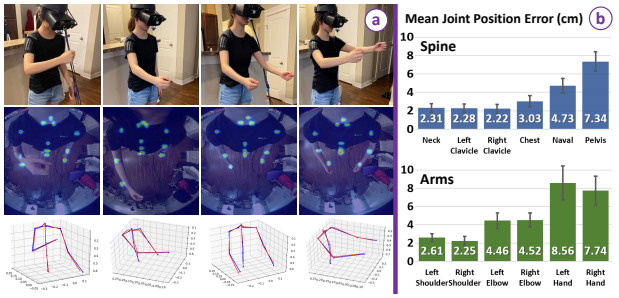

Figure 10. Test of upper-body pose tracking using a fisheye camera. (a) Examples of human poses, joint heatmaps overlayed on the fisheye images, and 3D reconstructions of the joints (red lines: predictions, blue lines: ground truths). (b) Mean joint position error of different joints.

Accuracy of Human-object Interaction Detection

We aimed to test the accuracy of detecting human-object interactions in temporal and spatial domains by measuring two sets of values, 1) start and end time of each interaction and 2) the average distance between the object and the hand during the interaction. We included six objects in the experiment. Figure 11 (a) shows the first-person view of the tester. During the test, the tester picked up an object, interacted with the object for approximately 10 seconds, then put the object back, and repeated the process with other objects. The test was performed three times by three researchers respectively. Regarding the ground truth of the interaction time, we recorded the egocentric view of the tester and retrieved the time from the video. Totally, 18 interactions (three interactions with each object) and 36 start and end times (six for each object) were recorded. We present the result in Figure 11 (b). The average interaction time error was 0.42s ($SD = 0.14s$) and average hand-object distance during the interaction was 4.68cm ($SD = 2.40cm$). The results indicate that CAPturAR can detect human-object interactions accurately both in time and space domains, which supports the realistic reconstruction and visualization of the interactions, i.e. an object moving with the avatar's hand.

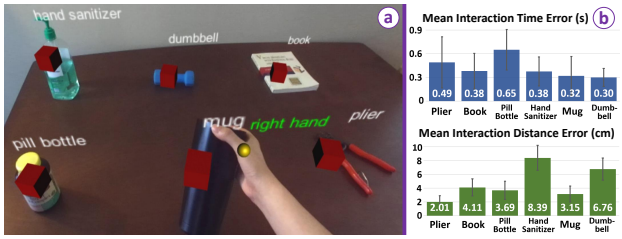

Figure 11. Test of human-object interaction detection. (a) Experiment setup. (b) Mean time error of interaction start and end time (top), and mean distance error between the hand and the objects (bottom).

Performance of Human Action Recognition

CAPturAR detects the happening of a *human action* by applying DTW distance and nearest neighbor algorithms to the real-time *human pose* sequence acquired by the fisheye camera. To quantify the performance of this method, we applied it to a classification task with 10 daily actions as listed in Figure 12. A tester repeated each action for 10 times. We equally divided the samples into two sets and applied a 2-fold cross-validation method to calculate the classification accuracy. We present the result as a confusion matrix in Figure 12. The overall classification accuracy was 89%, which validated the feasibility of our system in accurately recognizing *human actions* back in daily life. Additionally, using advanced skeleton-based action-detection algorithms such as [69] may further improve the robustness.

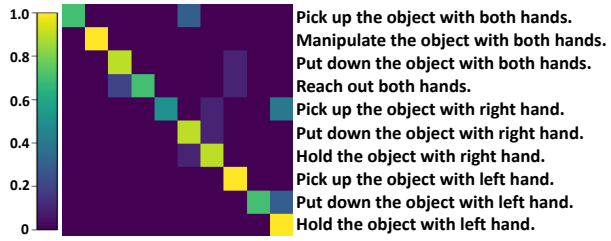

Figure 12. Confusion matrix of the *human action* classification accuracy.

REMOTE USER STUDY

Complying with the requirements of social distancing, we conducted a remote user study to evaluate the user experience of the CAPturAR authoring interface. Since the remote users had no access to the AR-HMD, we developed an AR simulator on PC where the view of the AR-HMD was fixed as shown in Figure 13. Also, we mapped the handheld controller operations to the mouse for the interactions by clicking the red buttons shown in Figure 13.

We invited 12 users (7 males and 5 females, whose ages range from 21 to 30) to participate in a two-session remote user study. 7 out of 12 users have AR/VR experience, and 8 out of 12 users own commercial IoT products such as smart light bulbs and smart speakers. None of the users had experience with our system before the user study. The study took a consecutive 1.5 hours and each user was paid 10 dollars for compensation. During the study, each user ran the software on his/her computer, shared the screen, and communicated with the researchers through online calls. The entire study processes were screen- and voice-recorded for post-study analysis. After every session, each user completed a survey with object Likert-type (scaled 1 to 5) questions, targeting on the level of agreement towards the using experience of the system features. After all sessions were finished, each user took a conversation-type interview to provide subjective feedback and finished a standard System Usability Scale (SUS) questionnaire (P=Participant).

We asked the users to author CAPs using pre-recorded activities generated by one of the researchers performing daily activities while wearing the AR-HMD (Figure 13 (b)). Totally, 14 daily activities were included in the recorded actions, such as reading books, having meals and drinking coffee. All the activities happened in a 6 meters by 6 meters household environment. The total length of the record is around 20 minutes. Some of the activities were repeated multiple times. To evaluate the authoring correctness, we prepared test records happened in the same environment. In each test record, some activities satisfied the authored *events* while some did not. After the user completed a CAP, we loaded the corresponding record to test whether the CAP was properly activated. To quantify the authoring accuracy, we counted the numbers of true positive detection (TP), false positive detection (FP) and false negative detection (FN) during the tests and calculated the F_1 score ($2TP/(2TP+FP+FN)$) (true negative (TN) was not available in our case).

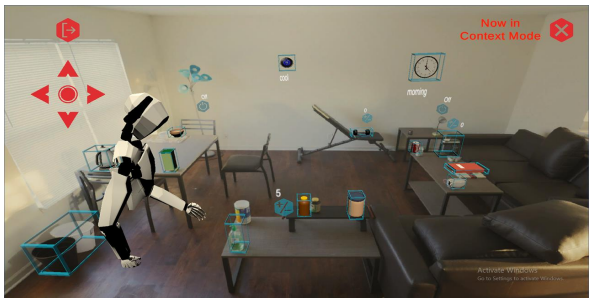

Figure 13. The semi-AR user interface for remote user study.

Table 1. Descriptions of the user study tasks.

CAP	*Events*	Logic Connections and Functions
1	Taking pill.	Pill count of the pill bottle minus 1.
2	Having meal.	Turn on the music player.
3	(a) Drinking coke. (b) Throwing the coke can.	(a) →Coke count plus 1. (b) →Show me a reusable icon above the trash can.
4	(a) Reading a book on the dining table. (b) Reading a book on the sofa.	(a) →Turn on the floor lamp. (b) →Turn on the table lamp.
5	(a) Having meal (with Delay 30 minutes). (b) Taking pill (with Inversion).	(a) →(b) (b) →Push a notification on the pill bottle.
6	(a) Drinking coffee. (b) Coffee count on the coffee machine = 2. (c) Lifting the dumbbell (with Inversion).	(a) →Coffee count on the coffee machine plus 1. (b) →(c) →Push a notification on the dumbbell.

Session 1: Event Definition Precision

In this session, we tested the usability of CAPturAR in precisely defining human involved *events* with two of our core features, *similar event*, and adding *context attributes*. The users were asked to create two *events* and author two simple CAPs respectively (CAP 1 and 2 in Table 1). For each *event*, users created it in three progressively detailed ways, 1) only selecting a *human action* as the *event* (Motion-Only-No-Verification), 2) after creating the *event*, labeling the *similar events* to be positive or negative (Motion-Only-With-Verification), and 3) after labeling, adding the relevant *context attributes* to the *events* (Motion-Object-With-Verification). After each trial, we loaded the corresponding test record and counted the numbers of TP, FP and FN. To challenge our authoring system, we deliberately recorded highly comparable movements, such as *taking-pill* versus *drinking-water* or *reading-book* versus *having-meal*, in the same test record.

Result and discussion. All 12 users completed the authoring processes. The result is illustrated in Figure 14. By labeling the *similar events*, the F_1 scores of the two tasks greatly increased (T1: from 50.50% (SD=0.13) to 83.33% (SD=0.08), T2: from 75.63% (SD=0.12) to 92.22% (SD=0.12)), mainly because of the significant decrease of the FP detection. The result implied that the *similar event* feature enabled the users to better characterize the *human actions* and help improve the algorithm. Moreover, after adding *object* attributes to the *events*, the precision of the two tasks increased (T1: from 83.33% to 97.22% (SD=0.10), T2 from 92.22% to 98.33% (SD=0.06)), which revealed the importance of associating *human actions* with other context information. Generally, the users were able to precisely define *events* using the *similar event* and the *context attribute* features we designed.

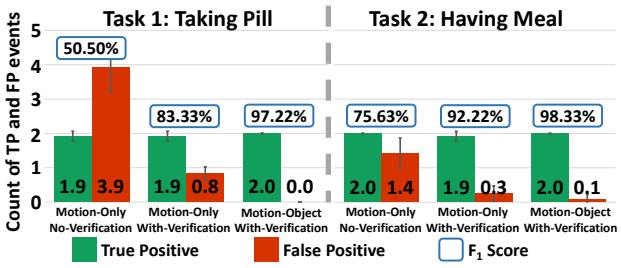

Figure 14. Event definition precision result

Session 2: Overall Usability

In this session, we aimed to test the overall usability of CAPturAR authoring interface with more complex CAPs (CAP 3-6 in Table 1). Each CAP contained 2-3 *events* and 1-2 IoT functions. Specifically, CAP 3 had two *events* with same *object* attribute (*coke*) but different *human actions* (*drink* and *throw*). CAP 4 had two *events* sharing one single activity (*reading book*) but happening at different locations (*at the table* versus *on the sofa*). CAP 5 required *temporal properties* (*inverse* and *delay*). CAP 6 was a comprehensive task with three *events*.

Result and discussion. All 12 participants successfully completed the authoring tasks. The average detection precision of the four tasks were 94.44% (SD=0.13), 97.22% (SD=0.10), 91.67% (SD=0.29) and 97.22% (SD=0.10), which indicated that after a short training process, most users were able to successfully author CAPs using our system.

The system feature related Likert-type ratings collected from the 2-session study are shown in Figure 15. In general, after the tutorial, the participants were confident to author *human action* dependent CAPs using our system and agreed with the intuitiveness and smoothness of our system workflow (Q8: AVG=4.42, SD=0.90). *"The event mode and the logic mode are closely integrated. I can easily define an event, and connect it to a smart function. (P7)"* Meanwhile, the majority of the users appreciated the clear view during the authoring process (Q10: AVG=4.42, SD=0.90). *"I like the design of replaying the avatar when I hover on the Event icon but displaying it when I start authoring. It represents the elements clearly and won't distract me. (P1)"*

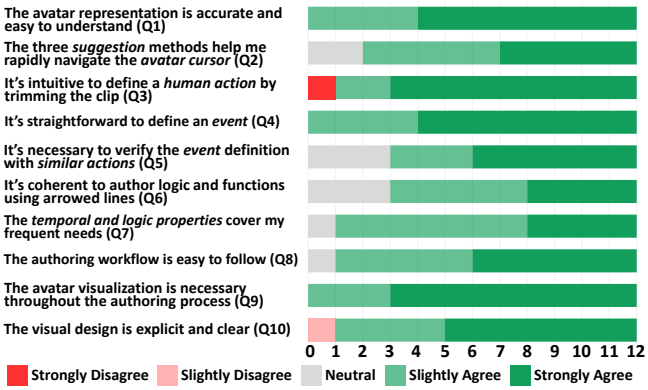

Figure 15. Likert-type result after the two-session user study.

We received positive comments about the visual representations of the authoring interface. Representing the human movements using the humanoid avatar was receptive by the users (Q1: AVG=4.67, SD=0.49). *"It's easy to understand what the avatar is doing. I think showing me a digital model of myself can remind me of what I did in the past. (P12)"* And the utilization of the avatar during the authoring process was highly accepted by the users (Q9: AVG=4.75, SD=0.45). *"When I can see what the avatar did in front of me, it becomes much easier and more straightforward to define a motion I want. (P8)"* Additionally, defining a human motion by trimming the avatar representation was accepted by the participants (Q3: AVG=4.50, SD=1.17). *"I like the idea of trimming avatar to define an action. And sometimes, I thought I didn't trim it precisely, but the system still successfully detected it. (P2)"*

We also asked the users about the features of CAPturAR authoring interface. The *suggestion* feature for rapidly browsing history records was welcomed as a decent feature (Q2: AVG=4.25, SD=0.75) and most users frequently used *suggestions* to find actions while creating events. *"That suggestion*

feature lets me quickly find what I want from the long history. Actually, you remind me of using location to find the time point as well. (P10)" Meanwhile, the participants felt confident of precisely defining an *event* using *context attributes* (Q4: AVG=4.67, SD=0.49). *"In real life, I do need different levels of constraints as triggers. Some can be easily triggered, and some are very constrained. I appreciate the idea of including different attributes in the events. (P5)"* Additionally, the *similar activity* feature received complimentary remarks (Q5: AVG=4.25, SD=0.87). *"I'm very satisfied with the similar activity feature. When I can help improve the back-end algorithm of action detection, I feel much more confident about what I just defined. We all know the current techniques are still not intelligent enough. (P11)"*

Regarding the experience of authoring CAPs through logic connections in *Logic Mode*, the survey result showed positive feedback (Q6: AVG=4.08, SD=0.79). *"I think using arrowed lines to build logic and IoT functions is very easy to follow. (P6)"* And the logic options and the *temporal properties* mostly covered the requirements in daily life (Q7: AVG=4.25, SD=0.62). *"I am not sure I would use all of the logic connections in one function, but each of them is definitely necessary in my daily life. Also, I really like the idea of the inverse logic. Because I don't want the system to bother me when I already have that thing in mind. (P3)"* Last but not least, the standard SUS survey result is 80.33 out of 100 with a standard deviation of 12.24, illustrating the high usability of our system.

DISCUSSION AND FUTURE WORK

Assistance with defining *events*. Through the user study, all users were able to master the CAPturAR workflow of defining *events*. Some users expected more assistance to make this process smoother. *P4* hoped to get some help while defining the *human actions "When I trimmed a human action, I was expecting the system to give me some hints or help me decide, especially when the movement is common"*. Further, *P2* mentioned that the system could automatically propose the necessary *context attributes* after she had labeled several *similar events*. *"I wonder after I make several decisions with this feature, your system may already know what is my intention and give me some feedback"*. These comments reveal that while CAPturAR emphasizes authoring unique and personalized actions, it should hold some levels of intelligence to help define *events* more precisely and rapidly. For future improvements, we propose exploring human action detection and pattern recognition algorithms to reduce end-users' load and further increase the authoring precision.

Authoring by real-time demonstration. CAPturAR supports users to select activities from the past records. Yet, some users were curious about whether they could author an *event* by directly acting it out *"I think it would also be necessary to define an event by demonstrating real-time (P5)"*. It reminds us to leverage the human action detection capability to include users' real-time actions in the *events* or edit the recorded *human actions* by embodied demonstrations.

Detection performance and efficiency. CAPturAR showed acceptable performance when detecting the authored *events*. Yet, the DTW algorithm we currently adopt relies on the users' consistency of repeating the actions. Moreover, the time complexity of DTW algorithm is $O(nm)$, which makes it time-consuming to search for *similar events* when long *human actions* are selected. Additionally, recording long-lasting and high-fidelity *human actions* (24 FPS with approximately 200 bytes in each frame) may accumulate to high volume. Thus, we are motivated to explore advanced DNN encoders that compress the high-fidelity records for faster and more accurate action detection, as well as decoders for realistic avatar reconstruction in AR.

Creating CAPs with multiple users. Technically, CAPturAR supports multiple users to share their activities and author collaborative CAPs in the same environment. For instance, an IoT function may be triggered if one user is performing an activity while another user is doing something else. Or the collaboration of the users may trigger the IoT functions that are different from the ones triggered by single user activities. However, privacy may become a problem in such cases. Will it be offensive to visualize one user's activities to another user or create CAPs that depends someone else? Studies on how to balance between privacy and multi-user collaboration could be one of the research directions of future CAP authoring tools.

Bulky hardware setup. In our work, we applied an AR-HMD, and a backpack computer to handle the context capture and action recognition computation. However the mobility of end-users is limited due to the size of the devices. We envision lightweight HMDs and advanced cloud service in the future to remove the bulky setup.

CONCLUSION

In this paper, we presented CAPturAR, an all-in-one system that allows end-users to create human-involved context-aware applications. We discussed the CAPturAR framework of modeling human involved CAPs and adopted a workflow of creating CAPs by referring to a user's previous recorded daily activities. To achieve this goal, we proposed an integrated AR-HMD platform for always-on, non-intrusive activity recording and context sensing. Further, we developed an AR authoring interface for creating CAPs through in-situ visual programming. We have demonstrated four different use cases for smart home environments featuring augmenting everyday objects, a healthy life application, a sequential task tutorial, and a tangible AR game. We proved the performance of the AR-HMD platform in terms of context-aware human action detection with a system evaluation. Within the remote user study, we received complimentary feedback on the user experience of our authoring interface. Therefore, we believe that CAPturAR opens up a new perspective of incorporating human action into the context-aware application system and inspires advanced smart environment construction.

ACKNOWLEDGEMENTS

This work is partially supported by the NSF under grants Future of Work at the Human Technology Frontier (FW-HTF) 1839971 and Convergence Accelerator FW-HTF 1937036. We also acknowledge the Feddersen Chair Funds.

REFERENCES

[1] 2019. Oculus. (2019). https://www.oculus.com/.

[2] Mohammad Abu Alsheikh, Ahmed Selim, Dusit Niyato, Linda Doyle, Shaowei Lin, and Hwee-Pink Tan. 2016. Deep activity recognition models with triaxial accelerometers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

[3] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-Action-Circuits: Leveraging Generative Design to Enable Novices to Design and Build Circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 331–342.

[4] Daniel Ashbrook and Thad Starner. 2003. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous computing* 7, 5 (2003), 275–286.

[5] Daniel Ashbrook and Thad Starner. 2010. MAGIC: a motion gesture design tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2159–2168.

[6] Microsoft Azure. 2020. Azure Kinect DK. (2020). Retrieved April 20, 2020 from https://azure.microsoft.com/en-us/services/kinect-dk/.

[7] Chris Beckmann and Anind Dey. 2003. Siteview: Tangibly programming active environments with predictive visualization. In *adjunct Proceedings of UbiComp*. 167–168.

[8] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. 2017. Exploring end user programming needs in home automation. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 2 (2017), 1–35.

[9] Michael Buettner, Richa Prasad, Matthai Philipose, and David Wetherall. 2009. Recognizing daily activities with RFID-based sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*. 51–60.

[10] Yuanzhi Cao, Xun Qian, Tianyi Wang, Rachel Lee, Ke Huo, and Karthik Ramani. 2020. An Exploratory Study of Augmented Reality Presence for Tutoring Machine Tasks. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[11] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A Time-space Editor for Embodied Authoring of Human-Robot Collaborative Task with Augmented Reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 521–534.

[12] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[13] Liming Chen, Jesse Hoey, Chris D Nugent, Diane J Cook, and Zhiwen Yu. 2012. Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 790–808.

[14] Kai-Yin Cheng, Rong-Hao Liang, Bing-Yu Chen, Rung-Huei Laing, and Sy-Yen Kuo. 2010. iCon: utilizing everyday objects as additional, auxiliary and instant tabletop controllers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1155–1164.

[15] Pei-Yu Chi, Daniel Vogel, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2016. Authoring illustrations of human movements by iterative physical demonstration. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 809–820.

[16] Jeannette Shiaw-Yuan Chin, Victor Callaghan, and Graham Clarke. 2006. An End-User Programming Paradigm for Pervasive Computing Applications.. In *ICPS*, Vol. 6. 325–328.

[17] Jose Danado and Fabio Paternò. 2014. Puzzle: A mobile application development environment using a jigsaw metaphor. *Journal of Visual Languages & Computing* 25, 4 (2014), 297–315.

[18] Luigi De Russis and Fulvio Corno. 2015. Homerules: A tangible end-user programming interface for smart homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 2109–2114.

[19] Anind K Dey, Gregory D Abowd, and others. 2000. The context toolkit: Aiding the development of context-aware applications. In *Workshop on Software Engineering for wearable and pervasive computing*. 431–441.

[20] Anind K Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 33–40.

[21] Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive prototyping of context-aware applications. In *International Conference on Pervasive Computing*. Springer, 254–271.

[22] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, and George Fitzmaurice. 2017. Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In *Proceedings of the 43rd Graphics Interface Conference*. 156–162.

[23] Alireza Fathi and James M Rehg. 2013. Modeling actions through state changes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2579–2586.

[24] Basura Fernando, Sareh Shirazi, and Stephen Gould. 2017. Unsupervised human action detection by action matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1–9.

[25] FinalIK. 2019. FinalIK. (2019). Retrieved September 1, 2019 from `https://assetstore.unity.com/packages/tools/animation/final-ik-14290`.

[26] Google. 2020. Tensorflow. (2020). Retrieved April 20, 2020 from `https://www.tensorflow.org/`.

[27] Ankit Gupta, Maneesh Agrawala, Brian Curless, and Michael Cohen. 2014. Motionmontage: A system to annotate and combine motion takes for 3d animations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2017–2026.

[28] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. 2012. DuploTrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 389–402.

[29] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 145–154.

[30] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D puppetry: a kinect-based interface for 3D animation.. In *UIST*. Citeseer, 423–434.

[31] Valentin Heun, James Hobin, and Pattie Maes. 2013. Reality editor: programming smarter objects. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 307–310.

[32] Jan Humble, Andy Crabtree, Terry Hemmings, Karl-Petter Åkesson, Boriana Koleva, Tom Rodden, and Pär Hansson. 2003. "Playing with the Bits" User-configuration of Ubiquitous Domestic Environments. In *International Conference on Ubiquitous Computing*. Springer, 256–263.

[33] Ke Huo, Yuanzhi Cao, Sang Ho Yoon, Zhuangying Xu, Guiming Chen, and Karthik Ramani. 2018. Scenariot: spatially mapping smart things within augmented reality scenes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[34] IFTTT. 2020. IFTTT. (2020). Retrieved April 1, 2020 from `https://ifttt.com`.

[35] Andrey Ignatov. 2018. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Applied Soft Computing* 62 (2018), 915–922.

[36] CN Joseph, S Kokulakumaran, K Srijeyanthan, A Thusyanthan, C Gunasekara, and CD Gamage. 2010. A framework for whole-body gesture recognition from video feeds. In *2010 5th International Conference on Industrial and Information Systems*. IEEE, 430–435.

[37] Georgios Kapidis, Ronald Poppe, Elsbeth van Dam, Lucas PJJ Noldus, and Remco C Veltkamp. 2019. Egocentric Hand Track and Object-based Human Action Recognition. *arXiv preprint arXiv:1905.00742* (2019).

[38] Ju-Whan Kim, Han-Jong Kim, and Tek-Jin Nam. 2016. M. gesture: an acceleration-based gesture authoring system on multiple handheld and wearable devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2307–2318.

[39] Yongkwan Kim and Seok-Hyung Bae. 2016. SketchingWithHands: 3D sketching handheld products with first-person hand posture. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 797–808.

[40] Michael Laielli, James Smith, Giscard Biamby, Trevor Darrell, and Bjoern Hartmann. 2019. LabelAR: A Spatial Guidance Interface for Fast Computer Vision Image Collection. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 987–998.

[41] Gierad Laput, Eric Brockmeyer, Scott E Hudson, and Chris Harrison. 2015. Acoustruments: Passive, acoustically-driven, interactive controls for handheld devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2161–2170.

[42] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 321–333.

[43] Bokyung Lee, Minjoo Cho, Joonhee Min, and Daniel Saakes. 2016. Posing and acting as input for personalizing furniture. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*. 1–10.

[44] Jisoo Lee, Luis Garduño, Erin Walker, and Winslow Burleson. 2013. A tangible programming tool for creation of context-aware applications. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 391–400.

[45] Hanchuan Li, Can Ye, and Alanson P Sample. 2015. IDSense: A human object interaction detection system based on passive UHF RFID. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2555–2564.

[46] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. 2017. Programming IoT devices by demonstration using mobile apps. In *International Symposium on End User Development*. Springer, 3–17.

[47] Yanghao Li, Cuiling Lan, Junliang Xing, Wenjun Zeng, Chunfeng Yuan, and Jiaying Liu. 2016. Online human action detection using joint classification-regression recurrent neural networks. In *European Conference on Computer Vision*. Springer, 203–220.

[48] David Lindlbauer and Andy D Wilson. 2018. Remixed reality: manipulating space and time in augmented reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[49] Yang Liu, Ping Wei, and Song-Chun Zhu. 2017. Jointly recognizing object fluents and tasks in egocentric videos. In *Proceedings of the IEEE International Conference on Computer Vision*. 2924–2932.

[50] Hao Lü and Yang Li. 2012. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2875–2884.

[51] Hao Lü and Yang Li. 2013. Gesture studio: authoring multi-touch interactions through demonstration and declaration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 257–266.

[52] Ana I Maqueda, Carlos R del Blanco, Fernando Jaureguizar, and Narciso García. 2015. Human-action recognition module for the new generation of augmented reality applications. In *2015 International Symposium on Consumer Electronics (ISCE)*. IEEE, 1–2.

[53] Panos Markopoulos. 2016. Ambient intelligence: vision, research, and life. *Journal of Ambient Intelligence and Smart Environments* 8, 5 (2016), 491–499.

[54] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. XRDirector: A Role-Based Collaborative Immersive Authoring System. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[55] Makoto Ono, Buntarou Shizuki, and Jiro Tanaka. 2013. Touch & activate: adding interactivity to existing objects using active acoustic sensing. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 31–40.

[56] OpenCV. 2020. Fisheye camera model. (2020). Retrieved April 20, 2020 from `https://docs.opencv.org/3.4/db/d58/group__calib3d__fisheye.html`.

[57] Manoranjan Paul, Shah ME Haque, and Subrata Chakraborty. 2013. Human detection in surveillance videos and its applications-a review. *EURASIP Journal on Advances in Signal Processing* 2013, 1 (2013), 176.

[58] Isabel Pedersen. 2009. Radiating centers: augmented reality and human-centric design. In *2009 IEEE International Symposium on Mixed and Augmented Reality-Arts, Media and Humanities*. IEEE, 11–16.

[59] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2013. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials* 16, 1 (2013), 414–454.

[60] Matthai Philipose. 2005. Large-scale human activity recognition using ultra-dense sensing. *The Bridge, National Academy of Engineering* 35, 4 (2005).

[61] Hamed Pirsiavash and Deva Ramanan. 2012. Detecting activities of daily living in first-person camera views. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2847–2854.

[62] David Porfirio, Evan Fisher, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2019. Bodystorming Human-Robot Interactions. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 479–491.

[63] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. 2010. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)* 6, 2 (2010), 1–27.

[64] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *arXiv* (2018).

[65] Michael Rietzler, Julia Greim, Marcel Walch, Florian Schaub, Björn Wiedersheim, and Michael Weber. 2013. homeBLOX: introducing process-driven home automation. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 801–808.

[66] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. Interactive body-driven graphics for augmented video performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[67] Munehiko Sato, Ivan Poupyrev, and Chris Harrison. 2012. Touché: enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 483–492.

[68] Lei Shi, Maryam Ashoori, Yunfeng Zhang, and Shiri Azenkot. 2018. Knock knock, what's there: converting passive objects into customizable smart controllers. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–13.

[69] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7912–7921.

[70] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[71] Enox Software. 2020. OpenCv For Unity. (2020). Retrieved April 20, 2020 from `https://enoxsoftware.com/opencvforunity/`.

[72] Stereolabs. 2019. ZED Mini Stereo Camera - Stereolabs. (2019). Retrieved September 1, 2019 from `https://www.stereolabs.com/zed-mini/`.

[73] Yu-Chuan Su and Kristen Grauman. 2016. Detecting engagement in egocentric video. In *European Conference on Computer Vision*. Springer, 454–471.

[74] Denis Tome, Patrick Peluse, Lourdes Agapito, and Hernan Badino. 2019. xR-EgoPose: Egocentric 3D Human Pose from an HMD Camera. In *Proceedings of the IEEE International Conference on Computer Vision*. 7728–7738.

[75] Khai N Truong, Elaine M Huang, and Gregory D Abowd. 2004. CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. In *International Conference on Ubiquitous Computing*. Springer, 143–160.

[76] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 803–812.

[77] Mark Weiser. 1993. Hot topics-ubiquitous computing. *Computer* 26, 10 (1993), 71–72.

[78] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-oriented drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4610–4621.

[79] Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. 2019. Mo 2 Cap 2: Real-time Mobile 3D Motion Capture with a Cap-mounted Fisheye Camera. *IEEE transactions on visualization and computer graphics* 25, 5 (2019), 2093–2101.

[80] Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. 2008. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering* 20, 8 (2008), 1082–1090.

[81] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. 2019. A comprehensive survey of vision-based human action recognition methods. *Sensors* 19, 5 (2019), 1005.

[82] Yupeng Zhang, Teng Han, Zhimin Ren, Nobuyuki Umetani, Xin Tong, Yang Liu, Takaaki Shiratori, and Xiang Cao. 2013. BodyAvatar: creating freeform 3D avatars using first-person body gestures. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 387–396.

[83] Yang Zhang, Yasha Iravantchi, Haojian Jin, Swarun Kumar, and Chris Harrison. 2019. Sozu: Self-Powered Radio Tags for Building-Scale Activity Sensing. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 973–985.

[84] Yang Zhang, Gierad Laput, and Chris Harrison. 2017. Electrick: Low-cost touch sensing using electric field tomography. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1–14.

[85] Yang Zhang, Gierad Laput, and Chris Harrison. 2018. Vibrosight: Long-Range Vibrometry for Smart Environment Sensing. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 225–236.

[86] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. 2017. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2914–2923.