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

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

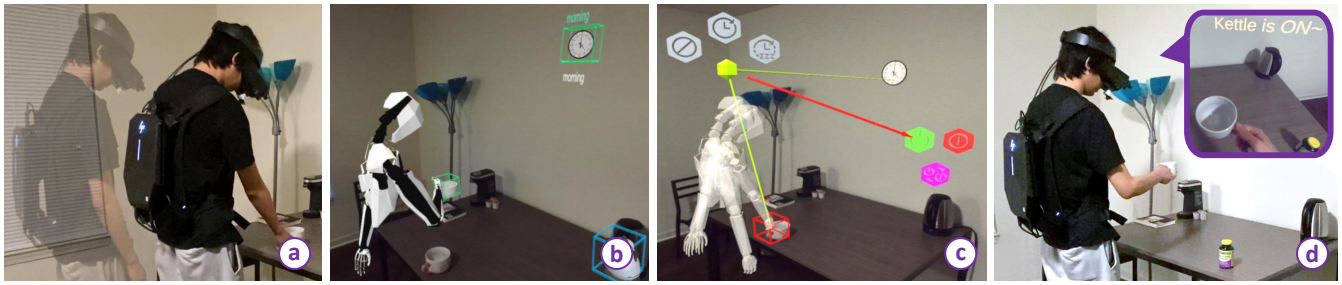

Figure 1. An overview of CAPturAR workflow. (a) A user conducts daily-life activities wearing a customized augmented reality head mounted device (AR-HMD). (b) The user visualizes his previous actions with context information in AR environment. (c) The user builds an *event* with a *human action* and *context attributes*, and connect the *event* to an IoT function to author a context-aware application. (d) When the *event* is detected, the IoT function is 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.

Author Keywords

Context-aware application, Augmented Reality,End-user programming tool, Ubiquitous computing, Embodied authoring, In-situ Authoring

UIST'20, October 20–23, 2020, Minneapolis, MN, USA

ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

CCS Concepts

•**Human-centered computing** → **Human computer interaction (HCI);** *Haptic devices;* User studies; Please use the 2012 Classifiers and see this link to embed them in the text: https://dl.acm.org/ccs/ccs_flat.cfm

INTRODUCTION

The concept of ubiquitous computing [73] has been gradually substantiated by the rapid growth of the Internet of Things (IoT) products [52]. 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 [57]. 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 [80], acoustic [40], and vision-based sensing [35, 55]. 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. Moreover, 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 accurate 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 [23, 82, 46]. 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[19, 8]. As an always-on wearable in the future, AR-HMDs have strong potentials for end-users to record their intentional and unintentional activities [56, 51] 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, 47]. An AR authoring interface allows users to intuitively view their own 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 [30, 32, 21]. 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 create human-involved context models using end-users' realistic daily activities and author customized context-aware applications in AR.
- An **integrated AR-HMD platform** composed of multiple camera systems which allows for 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 context-aware applications 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 [18] in a clear and intuitive manner. Towards this goal, an "*if...then...*" rule-based approach, namely trigger-action programming, is widely adopted. And many end-user programming interfaces are proposed to support fluent authoring of rule-based CAP. In a typical IFTTT flow [33, 72], users directly fill in a "*if* <*this*> *then* <*that*>" sentence with IoT related events to create a trigger-action pair. Alternatively, iCap [20] provided a GUI for users to define their trigger and action events through sketch and description. To make the authoring process easy to understand, visual programming interfaces are implemented by substantiating events to visual representations such as icons and arrows [17, 62, 3], magnets [71], and jigsaws [31, 16]. Further, to provide an in-situ and spatial aware authoring experience, researchers proposed tangible interfaces [7, 15, 43, 45]. 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 because of the lack of abilities to detect and visualize human actions. CAPturAR provides an always-on activity recording and detecting and enables end-users to customize sophisticated context models. We also expand the scope of human interactions with specialized IoT devices [44, 79, 64] to daily ordinary objects. Further, as an AR authoring tool,

CAPturAR supports users to visually program the rules in-situ by spatially connecting the context action and an IoT function.

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 [38, 42, 78], instant creation of stories and animations [26, 29, 63], and generating realistic tutorials [10, 14, 27]. Recently, GhostAR [11] proposed a workflow where end-users program human-robot collaboration tasks using their demonstrations as space-time reference.

Further, embodied demonstrations have been leveraged to customize gestures or action detection algorithms. Lv et al. [49, 50] enabled end-users to design multi-touch gestures on tablets. ACAPpella [19] allowed users to interact with multiple sensors. Exemplar [28] and M.Gesture [37] 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 human actions in daily life which include both unintentional and intentional patterns. Instead of acting the demonstrations one by one, CAPturAR provides fast browsing and selection of desired actions from cluttered and lengthy recordings. Moreover, we assist users to identify similar patterns by applying a pattern recognition algorithm to the entire recording. Then users can refine the action recognition algorithm by simply labeling the false positive and the true positive ones.

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 the objects, such as RFID[58, 44], capacitive sensing [53, 64], electric filed sensing [80], vibration and sound detection [81, 40]. Recently, Sozu [79] also presented activities detection by harvesting energy flow. Nonetheless, these methods require users to be close to the sensors or interacting 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 [41, 34, 2], GPS [60, 4] and biosensors [76, 65]. However, these wearable sensors are not suitable for detecting complex and delicate motions. Also, in some cases, wearable sensor outputs are often fused with other sources of information to achieve accurate detection.

Meanwhile, vision-based approaches are proposed by install cameras in the environment [77]. Further, researchers use egocentric wearable cameras to detect human actions [59, 69, 22, 36, 48]. 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. [75] and Tome et al. [70] used a downward-looking fisheye camera that covers all limbs. Thus the 3D human pose 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 context-aware human action detection.

CAPTURAR

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 in Figure 2 (a). The stereo camera is responsible for providing video-see-through AR experiences. Also, it is equipped with object detection algorithm and tracks the 3D positions of surrounding objects. The fisheye camera covers the user's limbs within its field of view (Figure 2 (b)), supporting always-on reconstructions of the human 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 results of human pose and object detection. Additionally, we obtain the spatial trajectory of the user from the 6-DOF tracking supported by the VR headset.

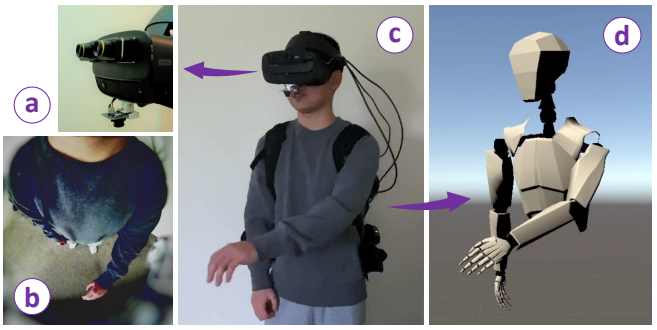

Figure 2. System hardware setup. (a) The customized AR-HMD with a stereo camera and a fisheye camera. (b) Fisheye camera view. (c) The AR-HMD is connected to a backpack computer. (d) Reconstructed virtual avatar.

CAPturAR system walk-through

We demonstrate the workflow of CAPturAR with a rule-based CAPs example scenario in the teaser figure. After getting up in the morning, the user would routinely pick up the cup and go to the kettle to make some hot tea. With CAPturAR, the user can create a CAP that automatically turns on the kettle and starts preparing the hot water as soon as the user picks up the cup. To author a CAP with the CAPturAR, the system first record the user's daily activities (Figure 1 (a)), and then these activities can be visualized in-situ by the user

as ***avatar cursor*** for human motions, and ***context attribute*** for smart thing interactions as well as other abstract spatial and temporal context information (Figure 1 (b)). The user can click on the object of interest to view all its past human interactions (Figure 1 (b)). Then the user selects the segment of picking-up action using the *avatar cursor* and links it to the cup object. Now the user has authored an ***event*** with a picking-up ***human action*** and a *context attribute* which is the cup. The *event* enables CAPturAR to infer the user's intention and do a specific thing if that event happens. The user wonders if he has authored this *event* correctly and CAPturAR can properly detect it. CAPturAR offers a function to improve the model by labeling ***similar events*** in the recording. *Similar events* denote the situations that CAPturAR would infer the same intention as the *event* authored by the user. By browsing the *Similar events*, the user finds out that CAPturAR mistakenly detects holding a cup as picking-up it. Thus he notes the holding cup as a negative example to avoid false positive detection. The user also realizes that picking-up a cup in the evening is also considered as similar, so he adds a third *context attribute* to the *event*, which is *time* as morning. At this step, the user is satisfied with his definition of the *event* and moves on to complete this CAP by connecting this *event* with an IoT function, the kettle's switch, as illustrated in Figure 1 (c). Next time when the user picks up the cup in the morning, the kettle will be automatically turned on, as shown in Figure 1 (d). To summarize, a user first finds a meaningful *human action*, then attach values of *context attributes* to make an *event*. After that, the user verifies the *event* by browsing the *similar events* and completes the CAP by linking the event to an IoT function.

Framework of CAPturAR

The framework of our system is illustrated in Figure 3. CAPturAR borrows the metaphor from object-oriented programming [74] and substantiates the abstract context information as ***human action***, ***context attribute*** and ***events***. The authoring workflow guides the user to build *events* with *human actions* and *context attributes*, and then create CAPs by connecting *events* with IoT functions.

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

Context attributes are descriptors of the surrounding context. Dey et al. [20] 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 smart thing that is involved.
- ***Location***, the spatial property of the user.
- ***Time***, the time during the day.
- ***State***, the state of IoTs, e.g. light on/off.

The values of the *context attributes* are recorded synchronously with the *human action* of the user and saved in ***context database***. *Context database* enables users to search for *human actions* by specifying the values of *context attributes*.

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 which is *cup* and the *time* attribute which is *morning*.

Similar events are generated by comparing a user created *event* with all data in *context database*. Thus, *similar events* the same *context attribute* values with the user created *event* and have similar *human actions*. We design *similar events* for two purposes. Firstly, by browsing the *similar events*, users can see various situations when the *event* is triggered, which helps the users to debug their authoring and specify more detailed context information. Secondly, we let users label the *similar events* as negative or positive to train the human action detection algorithm with more examples.

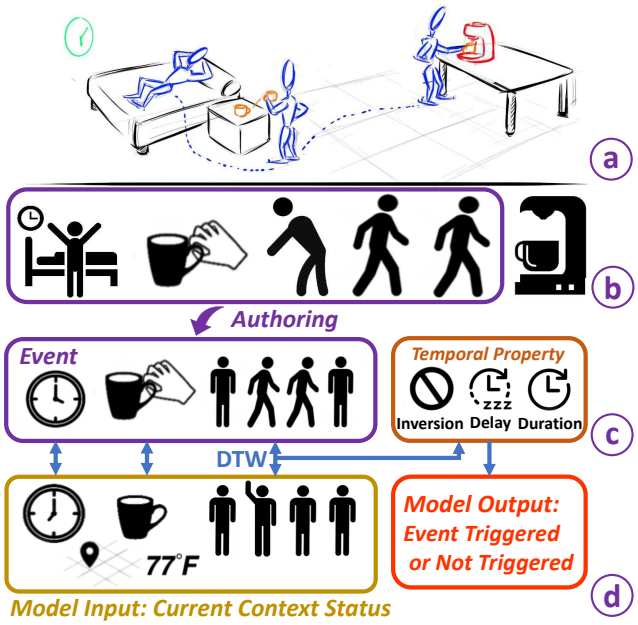

Figure 3. Framework of CAPturAR. (a) User conducts daily activities. (b) The *human actions* and the status of *context attributes* are timestamped and recorded in *context database*. (c) User creates events based on recorded actions. (d) CAPturAR detects *events* by comparing *human actions* and *context attributes*.

Detecting events

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 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* as currently involved if it is spatially close to one of the hands. Our system uses a fisheye camera to locate the hand positions and uses a front-facing stereo camera to obtain the object position. The values of *location* and *time* can be easily acquired given the 6-DOF tracking and internal clock of the AR-HMD. Meanwhile, the *state* attribute is acquired by communications with the IoTs. Currently, CAPturAR uses simulated IoTs in AR while there is no problem to scale to real IoTs.

If the user has labeled *similar events*, CAPturAR then 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*. But 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 service. On the other hand, we explore different temporal properties of *events*, namely *inverse* (triggers while actually not happening), *delay* (triggers after happening for a while) and *duration* (triggers after continuing for a while), as well as logical properties among multiple *events*, namely *sequential* (triggers only if *A* then *B*) and *parallel* (triggers if *A* or *B*). We embed these properties in CAPturAR authoring interface to allow for flexible rule-based CAPs.

In-situ authoring in Augmented Reality

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 generating embodied demonstrations, with high-level visual programming that creates CAPs with flexible triggering logic. Users author CAPs by going through two modes sequentially: ***Event Mode*** (Figure 4) and ***Logic Mode*** (Figure 5). In *Event Mode*, users can build *events* by selecting *context attributes* and *human actions*. In *Logic Mode*, users can connect the *events* to IoT functions to create CAPs. Users interact with the authoring interface through a VR hand-held controller. Particularly, we design the following features in AR authoring interface to enable a smooth authoring experience.

An ***avatar cursor*** for conveniently browsing, manipulating and selecting recorded actions. The *avatar cursor* is an in-situ placed human-avatar (Figure 4) in *Event Mode* that represents the pose of the user at a point of time in the past. Just like the cursors in video editing software, users can move the *avatar cursor* forward and backward using the directional buttons on the hand-held controller to browse the action, and look for the start and end point of an action. Further, users can select a part of recorded actions with *avatar cursor* in a similar hold-and-drag manner as selecting a part of text on a PC. Meanwhile, the values of all the other *context attributes* are temporally synchronized and update themselves correspondingly as the user moves the *avatar cursor*.

AR virtual contents that substantiate the *context attributes* and *events* for clear visualization and intuitive operations (Figure 4). *Object* attributes are displayed as virtual models superimposed onto the real objects. The positions of the *object* attributes are synchronized with the *avatar cursor* to illustrate human object interaction. The abstract context information such as IoT *state*, *time*, and *temperature* is described by spatially placed icons and words. Further, users can specify a *location* attribute by directly drawing a circle on the floor. *Events* are represented by ***event nodes***. By pointing at the *event node*, the corresponding *human action* plays repeatedly to illustrate the action. Through the menu above the *event node*, users can edit this *event*, go to another *event*, find *similar events*, or delete this event. While editing an *event*, users can add *context attributes* to it or add new segments of *human actions*. All elements belonging to the same *event* are displayed in the same color. If the user goes to another *event*, the *context attributes* will be reset to the default color (blue) for future editing.

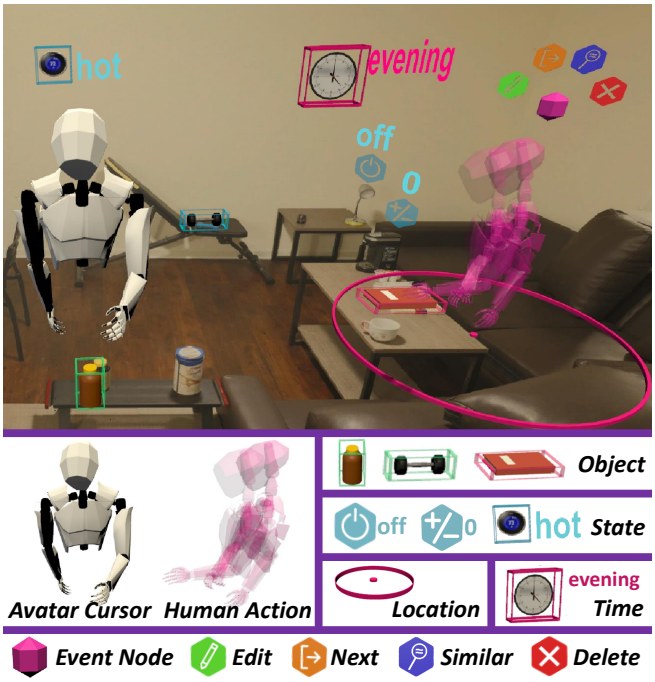

Figure 4. ***Event Mode*** **of CAPturAR user interface. Left, the** ***avatar cursor*****, which is navigated to a** ***pill bottle*** **related action by using the** ***pill bottle*** **as a** ***suggestion*****. To the right shows an event of** ***reading book close to sofa in the evening.***

A ***suggestion*** function in *Event Mode* for rapidly navigation among the long record of previous actions using *context attributes*. Users may have accumulated long records of their actions, which makes it difficult to search for an action. Since the *context attributes* status are timestamped, users can select some *context attributes* as *suggestions* to navigate the *avatar cursor* to the time points that are relevant to those *context attributes*. In Figure 4, the user selects the pill bottle as a *suggestion*, then the *avatar cursor* will navigate to the actions

involving the *pill bottle*. Users may also adjust the value of IoT *states* and *time* attributes while using them as *suggestions*. Further, users can also select multiple *suggestions*. For instance, a user can select *time* adjusted to *morning* as an additional *suggestion* to find the actions interacted with the *pill bottle* in the morning.

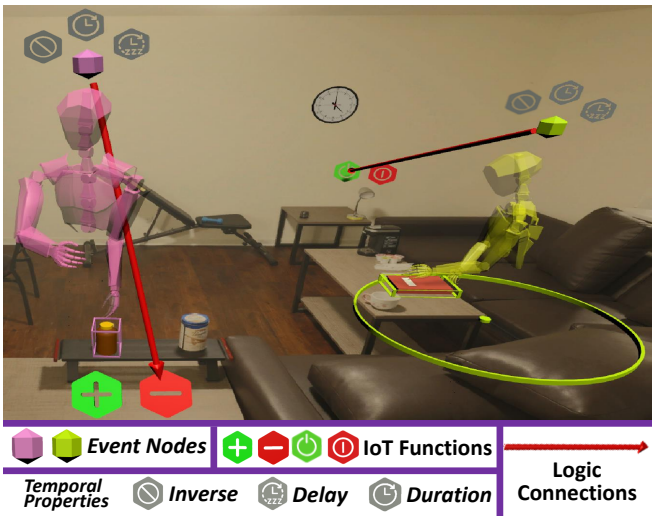

Figure 5. *Logic Mode* of CAPturAR user interface. Left, *If taking a pill from the pill bottle, the number of the pills minus one*. Right, *If reading book near by the sofa, turn on the light.*

A ***visual programming interface*** in *Logic Mode* (Figure 5). In *Logic Mode*, both events and IoT functions are spatially presented so that users can create rule-based CAPs by simply linking *event nodes* with IoT functions using arrows as *logic connections*. Further, users can connect one event node to another to implement a *sequential* logic, or connect multiple event nodes to one IoT functions to implement a *parallel* logic. Additionally, users can attach *temporal properties* (*inverse*, *delay*, *duration*) to *events* using the add-on menu above the *event node*.

USE CASE SCENARIO

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 non-smart objects

CAPturAR is aware of the user's interaction with non-smart objects. Leveraging the AR interface, users can attach digital functions, which can be triggered by their actions, to non-smart objects. Here, we augment a kettle, a pill bottle and a wipe can with a timer function, a counter function, and a reminder function respectively (Figure 6).

Healthy life 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 some dumbbell-liftings after 30 minutes of reading without any exercises. As shown in Figure 7, the user authors a reading-book *event* with a

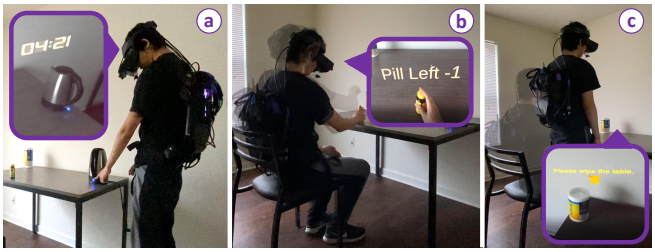

Figure 6. Augment non-smart objects. (a) The time starts to count down when the user turns on the kettle. (b) The number of pills in the pill bottle minus one when the user takes a pill. (c) The wipe bottle reminds the user to clean the table when the user stands up.

temporal property of *duration 30 minutes*, and connects to a dumbbell-lifting *event* with a *inverse* property. The dumbbell-lifting *event* is further connected to a reminder function on the dumbbell. Thus, if the user has read for 30 minutes, CAPturAR will check if he/she has done any dumbbell-lifting. If no dumbbell-lifting is performed, a reminder will pop up.

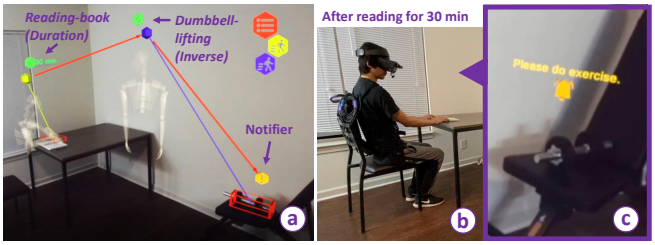

Figure 7. Healthy life reminder. (a) Two sequentially connected *events* with a *duration* and a *inverse* property respectively. (b) The user has to do exercise every 30 minutes of reading. (c) The system reminds the user if no dumbbell-lifting is detected within the past 30 minutes.

Sequential task tutorial

Leveraging the realistic visualization of human actions in AR, CAPturAR can also create embodied and adaptive tutorials for sequential tasks. A tutor 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 8). CAPturAR detects once the novice completes a step and starts to play the demonstration of the next step.

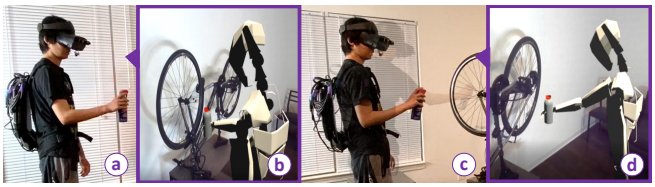

Figure 8. Sequential task tutorial. When the novice has finished the current step, next step automatically reveals as reference.

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 9. Once he/she picks up an empty coke can, CAPturAR detects the action and attaches a virtual basket and a virtual basketball to the rubbish bin and coke can respectively.

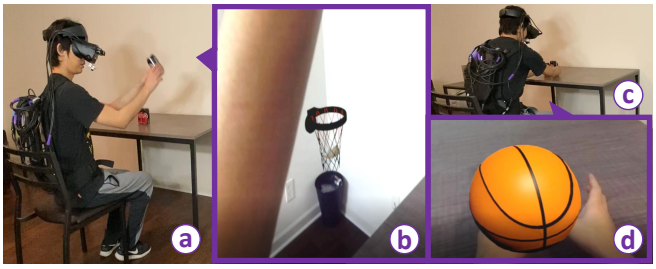

Figure 9. Tangible AR game creation. As the system detects a shooting action, the AR game is activated.

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 embedded SLAM, 2) a front-facing stereo camera (ZED Dual 4MP Camera [68], 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. 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 AR 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 10. The DNN comprises two concatenated parts. For the first part we adopt the convolutional pose machine structure presented in OpenPose [12] with VGG19 [66] 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 [25] at 24Hz. To further convert the joint positions into a realistic human avatar, we use FinalIK Unity3D plugin [24]. 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 [54]. In total, 170K images were collected from 35 volunteers. The two parts of the DNN are trained separately. The first part took 24h on an NVIDIA 2080Ti GPU, and the second took 12h on the same GPU.

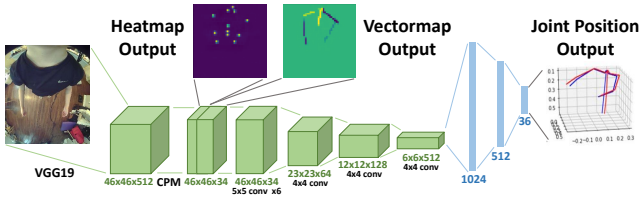

Figure 10. Upperbody Tracking Network Structure.

Detect interaction with objects

To track 3D positions of objects, we first use the Yolo v3 [61] implemented in OpenCV for Unity [67] 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 field of view 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. We trained the Yolo v3 to detect 15 daily objects including cup, kettle, pill bottle, book, coke can, wipes can, etc. for study and demo purpose. For each object, we collected approximately 2000 images using the method mentioned in [39]. The training took 12h on an NVIDIA 2080Ti GPU.

PRELIMINARY SYSTEM EVALUATION

The CAPturAR workflow relies on the capabilities of the integrated AR-HMD platform, namely 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 body pose plays an important 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 results from a Kinect Azure camera, which were used as ground truth. During the test, the tester performed 10 common movements such as *picking up*, *putting down*, *lifting with both hands* continuously while the data points were collected at 4Hz automatically. Figure 11 (a) demonstrates four poses during the test. Three researchers participated in this test and collected 825 data points in total. We then calculated the distances between joints positions from Kinect Azure and fisheye camera and present the results in Figure 11 (b). The average error is $4.34cm$ ($SD = 3.59cm$). Typically, the joints of the pelvis and both hands produced larger errors. Possibly because those joints are far from the pivot and have larger movements. The left hand produced the largest error ($8.56cm$ ($SD = 5.58cm$)) which was still smaller than the length of half palm. Thus, the customized AR-HMD has a similar tracking ability as Kinect Azure and can satisfy the requirement of tracking human actions precisely.

Accuracy of detecting human-object interaction

We aimed to test the accuracy of detecting human-object interaction in temporal and spatial domains. Specifically, we measured two values, 1) start and end time of the interaction and 2) the distance between the object and the hand during the interaction. We included 6 objects in the experiment. Figure 12 (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 3 times by 3 researchers respectively. For the ground truth of the interaction time, we recorded the egocentric view of the tester and retrieved the time from the video. In total, 18 interactions (3 interactions with each object) and 36 start and end times (6 for each object) were recorded. We present the

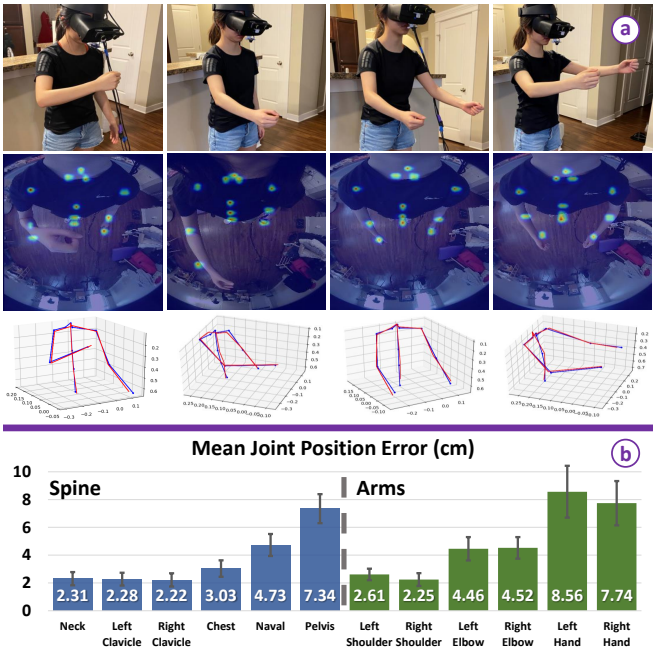

Figure 11. Tracking upper body using fisheye camera. (a) Example poses, joints heatmap,and 3D positions of joints. (b) Joint position error of different joints.

result in Figure 12 (b). The average interaction time error is $0.42s$ ($SD = 0.14s$) and average hand-object distance during the interaction is $4.68cm$ ($SD = 2.40cm$). The result indicates that CAPturAR can detect human-object interaction accurately. Further, realistic visualization of human action with objects, i.e. object follows the avatar's hand, can be constructed.

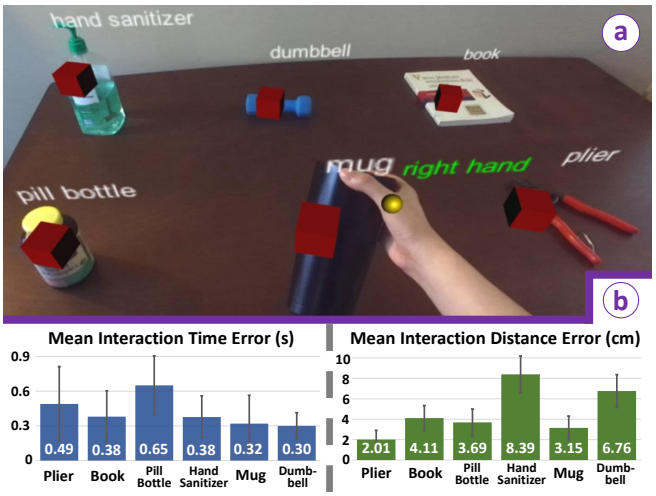

Figure 12. Test of human-object interaction detection. (a) Experiment setup. (b) The error of interaction start and ent time (left) and the distance between hand and object (left)

Performance of human action detection

CAPturAR detects human action by applying DTW distance and nearest neighbor algorithm *human pose* sequence acquired by the fisheye camera. To quantify the performance of this method, we applied the same approach to a classification task with 10 daily actions as listed in Figure 13. For each action, we collected 10 samples using the fisheye camera. All the samples were collected from one researcher. 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 13. The overall classification accuracy is 89%, which implies enough accuracy but still requires additional context information for more robust detection.

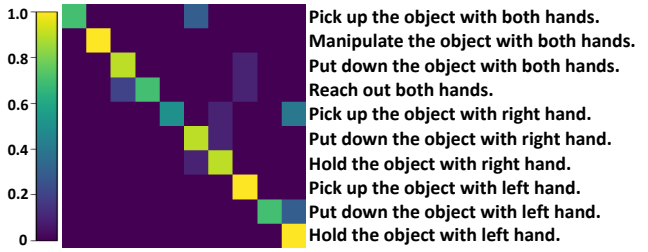

Figure 13. Confusion matrix of human action classification.

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 14. Also, we directly mapped the handheld controller operations to the mouse for interactions with the virtual contents by clicking. 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 researchers through online calls. The entire study processes were screen- and voice-recorded for post-study analysis. After each session, the users 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 14 (b)). Totally 14 daily activities were included in the recorded actions, including reading books, having meals, drinking coffee, etc. All the activities happened in a 6mx6m household environment. The total length of the record is around 20 minutes. Some of the activities were repeated multiple times. To evaluate the CAP authored by the users, we created a simulated environment that provides similar context information as a real environment and has virtual IoTs. For each task, We prepared test records generated using the same approach. After the user has completed a CAP, we loaded the corresponding record to

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.

the simulated environment to test whether the CAP is properly activated and delivers the correct service. To quantify the event detection accuracy, we counted the number of true positive detection (TP), false positive detection (FP) and false negative detection (FN) during the tests and calculated the F1 score ($2TP/(2TP+FP+FN)$) (true negative (TN) is not available in our case).

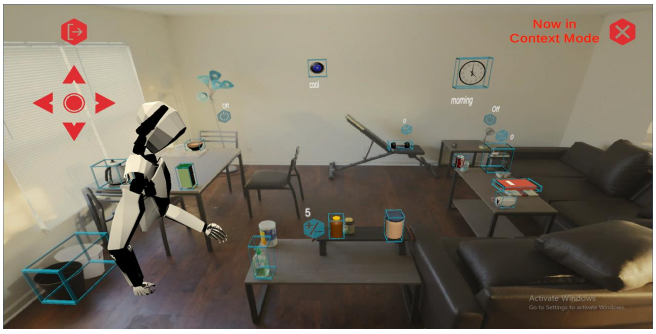

Figure 14. User Study Setup: Remote User Study Semi-AR Interface.

Session 1: Creating events

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

Result and discussion. All 12 users completed the authoring processes. The precision result is illustrated in Figure 15. By labeling the *similar events*, the F1 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 results implied that the *similar events* feature helps user better characterize the *human actions* and avoid false detection. After adding *object* attributes into 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 shows the importance of associating *human action* with context information. Generally, the users were able to precisely define *events* with the *similar events* and *context attributes*.

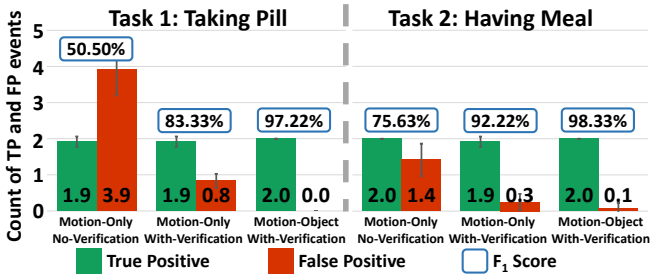

Figure 15. Event Definition Precision Result

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 contains 2-3 *events* and 1-2 IoT functions. Specifically, CAP 3 has 2 *events* with same *object* attribute (*coke*) but different *human actions* (*drink* and *throw*). CAP 4 has 2 *events* sharing same activities (*reading book*) but happens at different locations (*by the table* versus *by the sofa*). CAP 5 requires *temporal properties* (*inverse* and *delay*). CAP 6 is a comprehensive task with 3 *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, 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 16. In general, after the tutorial, the participants were confident to author a human action dependent context-aware application 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)"*

We received 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 welcomed by the participants (Q3:

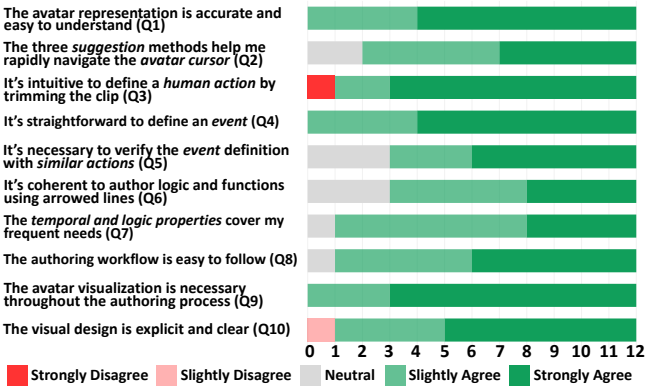

Figure 16. Likert-type result after the two-session user study.

AVG=4.50, SD=1.17). *"I like the idea of trimming that avatar to define an action. And sometimes, I thought I did not trim it precisely, but finally the system still successfully detected it. (P2)"*

We also let users tell us about the features of CAPturAR, such as *suggestion* and *similar event*. The Suggestion feature for browsing history records was welcomed as a decent feature (Q2: AVG=4.25, SD=0.75). *"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)"* The participants felt confident of precisely defining an Activity (Q4: AVG=4.67, SD=0.49). *"I think I do need a different number of constraints in different Events because sometimes I want the function to be triggered easier, and sometimes more strict. (P5)"* And 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 connections in *Logic Mode*, the survey result showed positive feedback (Q6: AVG=4.08, SD=0.79). *"I think using arrowed lines to connect logics and functions is very easy to follow. (P6)"* And the logic options provided 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 Inversion logic. Because I don't want the system to bother me when I already have that thing in mind. (P3)"* 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

Defining human action. All the users provided complimentary feedback on defining the *human action* during authoring, but some raised an interesting issue. *"When I trimmed a human action, I also wanted the system to give me some recommendations. (P4)"* This finding introduces an essential consideration when designing human involved CAPs: **How to balance between the customization (such as embodied demonstration) and the pre-definition of a human action**. One potential solution is to apply a pattern recognition algorithm to the recorded human actions and suggest the user with candidate segments. However, this may still be a research question because of the complexity and ambiguity of human activities.

Defining event. Through the user study, most of the users embraced the *similar event* feature while some users mentioned an inspiring question. *"I wonder after I make several decisions with this feature, your system may already know what is my intention and can give me some feedback. (P2)*. Here raises another question of **How to balance between the human inputs and the computational outcomes to improve the system performance**. Applying AI algorithms may serve as one potential solution to understand the user's need and automatically help the user precisely define an *event*. From this perspective, further researches can be conducted on developing a well-balanced AI system for providing customized experience based on human guidance.

Detection performance. Regarding the trigger detection, CAPturAR is proved with high performance when detecting the authored *event*s. Yet, the dynamic time warping algorithm we currently adopt relies on the user's consistency of repeating the actions. Advanced data comparison algorithms such as probabilistic methods and deep neural networks could be applied to improve the system's robustness further.

Bulky hardware setup. In our work, we applied an AR-HMD, and a backpack computer to handle the context sensing and action recognition computation. However the mobility of the users is limited due to the size of the devices. We envision the lightweight HMD and advanced cloud service in the future to remove the bulky setup. Moreover, by integrating more sensors such as Lidar and IR cameras into a lightweight HMD, the system would capture more complicated actions and grant more possibility to the authoring process.

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 non-smart 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.

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] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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).

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

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

[55] 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.

[56] 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.

[57] 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.

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

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

[64] 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.

[65] 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.

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

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

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

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

[70] 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.

[71] 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.

[72] 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.

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

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.