Ubi Edge: Authoring Edge-Based Opportunistic Tangible User Interfaces in Augmented Reality

Fengming He*
he418@purdue.edu
School of Electrical & Computer Engineering, Purdue University
West Lafayette, IN, USA

Xiyun Hu*
hu690@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Jingyu Shi
shi537@purdue.edu
School of Electrical & Computer Engineering, Purdue University
West Lafayette, IN, USA

Xun Qian
qian85@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Tianyi Wang
wang3259@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

Karthik Ramani
ramani@purdue.edu
School of Mechanical Engineering, Purdue University
West Lafayette, IN, USA

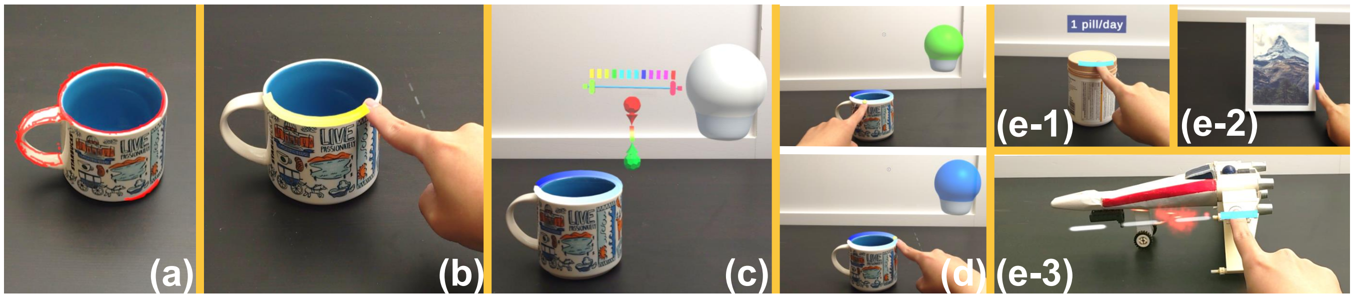


Figure 1: An overview of Ubi Edge system. (a)-(c) The user wants to author a TUI to control the color of the AR light bulb using the rim of the cup. (a) Detected edge feature points are overlaid on the real edges. (b) A user slides along the top rim of the cup to segment an edge as the input. (c) The user then connects the 'hue' behavior of the virtual light bulb with the selected edge to author an edge-based TUI. (d) Back in the real world, the user uses the finger to slide on the cup rim to gradually change the virtual bulb's color. (e) Three more TUIs authored by Ubi Edge: (e-1) An AR medication reminder appears after the user clicks the edge of a pill bottle's cap. (e-2) The digital photo frame displays a new photo after the user slides down on the right side of the photo frame. (e-3) An AR weapon shooting animation starts to play next to a toy aircraft after the user clicks an edge on the wing.

ABSTRACT

Edges are one of the most ubiquitous geometric features of physical objects. They provide accurate haptic feedback and easy-to-track features for camera systems, making them an ideal basis for Tangible User Interfaces (TUI) in Augmented Reality (AR). We introduce Ubi Edge, an AR authoring tool that allows end-users to customize edges on daily objects as TUI inputs to control varied digital functions. We develop an integrated AR-device and an integrated vision-based detection pipeline that can track 3D edges and detect the touch interaction between fingers and edges. Leveraging the spatial-awareness of AR, users can simply select an edge by sliding fingers along it and then make the edge interactive by connecting it to various digital functions. We demonstrate four use cases including multi-function controllers, smart homes, games, and TUI-based tutorials. We also evaluated and proved our system's usability through a two-session user study, where qualitative and quantitative results are positive.

*Both authors contributed equally to this research.

CHI '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3580704

KEYWORDS

Tangible User Interface, Augmented Reality, immersive authoring

ACM Reference Format:
Fengming He, Xiyun Hu, Jingyu Shi, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. Ubi Edge: Authoring Edge-Based Opportunistic Tangible User Interfaces in Augmented Reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3544548.3580704

1 INTRODUCTION

Tangible User Interface (TUI) [40] has become one of the essential ways of rendering haptic feedback in Augmented Reality (AR). End-users can pervasively execute digital services by physically interacting with surrounding objects [25]. Further, opportunistic TUI [28], which tightly maps the affordance of the physical objects (e.g., corner, cylinder, and surface) with the functions of the virtual widgets (e.g., buttons, knobs, and touch pads), provides a more intuitive connection between the user inputs and digital services. However, such strict mappings [30, 106] limit the generalizability of opportunistic TUI. It is hard for end users to find an everyday object that perfectly matches the target digital functions both geometrically and semantically.

Such a challenge inspires us of leveraging local geometric features, instead of the entire objects, in the context of customizing opportunistic TUI. Specifically, we focus on edges and sides, especially the linear and circular ones, that exist ubiquitously on every object such as tables, books, mugs, and monitors. Edges provide sharp tactile feedback [24, 68, 80] that even allows users to provide accurate inputs without careful attention [41, 88]. Meanwhile, the affordances involved in assorted types of edges, i.e., short edges and corners (intersections of edges) as buttons, linear edges as sliders, and circular edges as knobs, can fulfill the diversified user needs to control different virtual widgets [70]. Additionally, edges provide strong geometric features for computer vision algorithms, which allows for instant, marker-free and non-intrusive tracking compared with marker-based systems like [13, 25, 102].

Traditional predefined TUIs require predesignated pairs of tangible interfaces and digital functions. Therefore, the versatility of TUIs is limited, and it is hard to adapt the predefined TUIs to any user or task that is beyond the original design [5, 23]. To address this issue, the concept of end-user authoring [55] was proposed to equip end-users with customization tools so that users can follow their preferences and build interactions based on the local context. Multiple prior works [82, 83, 98] have demonstrated such a possibility by leveraging in-situ AR/VR visualization, embodied interactions and program by demonstration technique. Yet, the interacting system for building pervasive and customized tangible AR interfaces still remains to be explored.

Following this thread, we propose Ubi Edge, an authoring system that empowers end-users to author edge-based opportunistic TUIs in AR. We build an integrated head-mounted device (HMD) by mounting a lidar camera [39] onto an advanced AR headset [35] to enable pervasive and accurate detection of all available edges on physical objects via a point-cloud-based edge detection algorithm (Figure 1a). While wearing the AR-HMD, an end-user can simply use the finger to slide along a physical edge to select an interactive segment as the TUI input (Figure 1b). Using the AR interface, the user then builds a connection between the selected edge and the desired digital functions (Figure 1c). After the authoring process, the user can enjoy the edge-based opportunistic TUI in daily life (Figure 1d,e), while the back-end system keeps tracking the touch and slide interactions performed on the authored edge.

We develop an integrated vision-based pipeline for the customized AR-HMD so that Ubi Edge can accurately and efficiently track geometric edges as well as user interactions with edges while minimizing user intervention. Broadly speaking, our vision-based pipeline consists of: 1) a robust RGB-D edge detection algorithm adapted from vanilla Canny Edge detector [10]; our algorithm can detect salient geometric edges while suppressing image texture-based edges and redundant edges, 2) a lightweight neural network detector to separate foreground from background, prune detected edges, quickly eliminate large portions of background feature points, and thereby save computation, 3) an off-the-shelf pose estimation sub-system that can accurately track 6 degrees of freedom (DoF) of an object, and 4) an edge-based iterative closest point (ICP) to refine and finalize the alignment and correspondence of object edges.

In summary, we highlight our contributions as follows:

- An end-to-end authoring workflow that allows end-users to author edge-based opportunistic TUIs that execute digital functions when the users interact with customized segments of physical edges on everyday objects.
- An integrated algorithm for reconstructing, detecting, and tracking 3D edges on everyday objects as well as interactions between fingers and the 3D edges.
- An immersive AR authoring interface that supports end-users to segment edges through in-situ interactions while referring to the physical object, and defining the corresponding digital functions through visual programming.

2 RELATED WORK

2.1 Opportunistic Tangible User Interface

Opportunistic Tangible User Interface (TUI) [28] utilizes geometry shapes (e.g., cylinder, bulge, and cube) of everyday objects as passive haptics and spatial references to facilitate end-users to accurately and pervasively control digital functions in Augmented Reality (AR). Opportunistic Controls [28] enables users to manipulate AR objects by touching ambient surfaces. Given a virtual content, Annexing Reality [30] turns everyday objects into tangible proxies by looking for the physical object with a similar shape as the virtual content in the real environment and overlaying the virtual content on top of the physical object. Similarly, Gripmarks [106] repurposes handheld objects into interactive surfaces by rendering corresponding AR contents above the physical objects based on different grasps and object shapes. While most prior works target leveraging the affordance of the entire object as haptics and semantic references, they hamper the scalability of the opportunistic TUI, i.e., users need to find a physical object that has high semantic similarity with the target digital function, and carefully perform the interaction to achieve the desired control.

An edge of a physical object, on the other hand, is particularly suitable to be used as opportunistic TUI input owing to its ubiquitous existence and accurate haptic feedback. Prior works prove that edges can provide crucial geometric information and strong tactile feedback to users [24, 68, 80]. Users can also directly ascertain the edge location through the sharp tactile of edges without focusing on the objects [34, 41, 88]. Unifone [34] lets users touch the sides of one-hand-held devices without causing occlusion on screens. Haptic Edge Displays [41] attaches pins on the side of smartphones so that users can receive sharp tactile feelings even when phones are in pockets. Recent studies also show that users have better performance and more preference for edges than 2D surfaces when finishing haptic-related tasks [44, 69]. Specifically, Joshi et al.[44] proves that users perform comparable or even better on table ridges than on top surfaces when performing haptic-related tasks. Following the metaphor of opportunistic TUIs, we would like to explore how to leverage edges as general inputs for opportunistic TUI applications.

2.2 Author Tangible User Interface in AR

TUI authoring, which defines the mapping between input physical interactions and output digital functions, extends the TUI's functionalities and at the same time fulfills specific user/task needs [5, 23]. Specifically, by introducing end-users into the design process, customized TUI authoring systems [5, 23, 32, 47, 89] enable couplings of arbitrary tangible inputs (e.g., gestures, objects) to various digital functions and thereby generate different combinations of interactions.

Authoring TUI input requires the perception of human-object interaction. On the one hand, attaching hardware sensors [81, 99] and applying electrically conductive material [100] onto ordinary objects to detect human inputs such as touches and slides have been explored. Leveraging vision-based tracking algorithms, prior arts also propose to attach fiducial markers on objects [25, 47, 55, 102, 107] to detect interactions. Further, Light Widgets [21] uses a multi-camera system to detect users' skin color, thereby tracking users' hand interactions with everyday surfaces. Similarly, WorldKit [89] adopts an RGBD projector to detect customized touch interactions on surfaces of everyday objects. However, most of these authoring works lead to intrusive TUIs. The authoring process may impair the object's original functions since additional hardware and fiducial markers are attached. Some other works require users to deliberately spare a physical space for the interaction detection to work [13].

In contrast, opportunistic TUI grants a more unobtrusive and intuitive interaction since it leverages the affordance of physical objects. Gripmarks [106] enables users to appropriate interactions with physical objects into customized digital functions, while Annex Reality [30] and Funk et al. [23] leverage the 6DOF manipulation of physical objects as the inputs of opportunistic TUI. These authoring works target interactions that leverage the specific affordance of the entire physical objects (e.g., pulling the spray bottle handle and rotating a soda bottle). Yet, when a user attempts to control a specific digital widget, the user has to find a physical object with the desired affordance, which limits the generalizability of the opportunistic TUI. As addressed in the previous subsection, edges, which serve as a ubiquitous and scalable geometric feature, show promising potential in addressing this issue. Hence, we aim to develop a system that enables end-users to customize opportunistic TUI that leverages the edges of physical objects as inputs.

2.3 Edge Detection Methods

Geometric features like edges, surfaces, and shapes have been long been studied in the computer vision area [1, 15]. Vision-based edge detection has also been applied to a variety of applications including image segmentation [10, 76], pose estimation [14, 37], and object detection [7, 65]. Nowadays, researchers have a deeper understanding towards 3D edge detection thanks to the advances of depth cameras (e.g. Kinect [101], Intel®Realsense™ [48]).

A typical 3D edge detection is composed of two main steps: 1) Extracting edge feature points where surface normals suddenly change, and 2) fitting parametric feature curves (e.g. Bezier curves). Among the works of edge feature point extraction, Choi et al. [15] proposed a real-time method for detecting occluding edge points, occluded edge points, RGB edge points, and boundary edge points from a point cloud of everyday environments. Then, Bazazian et al. [4] investigates challenges for extracting edge feature points from the unorganized point cloud and introduces a new method for understanding geometric information in noisy real environments. Similarly, Ahmed et al. [1] extracts edge feature points and corners from the workplace and applied the approach for robotic welding. Later, learning-based methods that extract or classify edge feature points and corners from the point cloud are proposed [33, 85, 95, 96]. Meanwhile, prior arts [22, 67, 84, 103] has extensively explored the parameter estimation step. However, estimating parameters for all edge feature points is computationally expensive and the estimated edges may not even correspond to true geometric edges (e.g. an edge resulted from texture or illumination/shadow) [6, 64].

In this work, rather than first fitting and presenting edges to users and then letting users choose edges as TUI inputs, we first extract the edge feature points and then let users define their own tangible edge inputs by referencing the feature point results. Thereby, our system skips redundant computation, avoids inaccurate edge prediction results, and satisfies the customization needs.

2.4 Design Immersive Authoring Interface in AR

Augmented Reality (AR) enables users to interact with both virtual and physical environments and shows strong potential to provide in-situ authoring for TUIs. Specifically, compared with traditional desktop-based 2D programming, the immersive authoring capability [54] provided by AR enables users to view and manipulate 3D virtual contents directly in the physical environment. Following the immersive authoring metaphor, previous works [38, 51, 97] leverage real objects as spatial references and create 3D virtual elements whose spatial location coordinates with physical surroundings. Utilizing the characteristic of spatial awareness of AR, researchers facilitate fast 2D video prototyping by blending 3D manipulations into 2D videos [56, 58] and creation of 3D animation of virtual contents [11, 12, 57, 93], through in-situ authoring.

In addition, the visual programming capability provided by AR has been utilized to work together with the immersive interfaces [20, 31, 82, 98] so that users are able to map personal interactions with virtual content behaviors. For example, GesturAR [82] allows users to customize freehand interactions with virtual objects by connecting gestural inputs and corresponding reactions in AR, while CAPturAR [83] supports users to define smart object functions that are aware of activity-involved contextual events via spatial programming.

Following the immersive authoring metaphor and fully utilizing the visual programming enabled by AR, Ubi Edge endeavors to develop an AR authoring interface that allows end-users to prototype edge-based in-situ TUI authoring when spatially referring to the physical objects and immersively associating the edge inputs with assorted digital functions.

3 UBI EDGE SYSTEM DESIGN GOAL

Ubi Edge aims to allow end-users to easily create personalized edge-based AR TUIs in their own environment. This way, users can interact with them in daily life. To this end, the users should be provided with sufficient candidate edges in their surroundings. Meanwhile, they should be able to select any edge intuitively and effortlessly. After that, the users can choose a digital function (i.e. smart thing functions, interactive AR widgets, AR applications) and map it with the selected edge. Specifically, Ubi Edge mainly focus on the following design goals of the authoring experience:

- **Ubiquitous edge detection** (DG1). In order for users to use any nearby geometric edges for TUIs, our system is expected to detect most edges around users and users' interaction with the edges accurately in real time.
- **Intuitive edge selection** (DG2). To reduce the complexity of edge selection/manipulation during the authoring process, we aim to let users set an edge using the programming by demonstration technique[3, 12, 59, 82]. Users should be able to select a piece of edge by simply sliding or tapping it with their fingertips.
- **Flexible function design** (DG3). To let the users customize the edge function as they wish, Ubi Edge should include sufficient choices of digital functions and a straight-forward AR spatial programming interface. So that users can easily pair interactive edges with any possible digital functions and edit the function parameters as mentioned in Section 2.4.
- **In-situ visualization and feedback** (DG4). To enable end-users to understand which edges can be authored and to help users to effectively program their own TUIs, Ubi Edge needs to provide real-time visual feedback about interactable edges and corresponding authoring results. The benefits of adopting such immersive visualization have been discussed in Section 2.4.

4 UBI EDGE

We derived three main system features based on the discussion of design goals above: (1) an edge detection pipeline for users to scan object and register geometric edges (DG1), (2) an input-output model for end-users to efficiently prototype TUIs (DG2, 3), and (3) an authoring interface that provides real-time visual feedback about what the user has authored (DG3, 4). In this section, we first walk-through our workflow with a specific example. Then, we illustrate the core algorithms we adopted for edge detection. Next, we introduce the input-output model for designing the authoring system. Finally, we present our authoring interface.

4.1 System Walk-through

The overall system workflow is shown in Figure 1 (a-d): a user first scans the object and registers geometric edges into our system. Then with the support of our AR authoring interface, the user can use touch to select his/her desired edge and author the TUI following a trigger-action programming metaphor. Finally, the user can experience the authored TUI application. Here, we briefly illustrate the workflow of Ubi Edge using an example scenario where an end-user wants to author a TUI that controls the color of an AR lamp using a cup's top circular rim. The user first moves to a position where the cup's rim is visible, at the same time, our system captures the RGBD information of the object to extract geometric edge points on the object. Next, the user initiates the authoring process by sliding the finger along the desired edge to customize a segment as a ***tangible edge*** (Figure 1b), while Ubi Edge determines the starting and ending point of the edge by detecting the touch interaction with the cup's rim. Then the user sets the ***edge type*** for the selected edge to be *continuous*, which means our system will treat the input as a continuous value between 0 to 1. Next, the user specifies the available range of the AR lamp's color, and connects the *tangible edges* with the color range behavior via spatial programming (Figure 1c). Back in run-time usage, our system keeps detecting the touch event and the finger's relative position on the authored *tangible edge*. Now, the user is able to control the AR lamp's color by sliding the fingertip along the authored rim (Figure 1d).

4.2 Edge Detection

Guided by DG1, we customize an AR-HMD and develop an integrated vision-based pipeline to detect all geometric edges in the physical environment as well as users' interactions with these edges. In this subsection, we first describe the components of the customized AR-HMD and then explain how our system achieves accurate edge detection through the proposed vision-based pipeline.

4.2.1 Customized AR-HMD. We customize HoloLens2 AR-HMD [35] with a high-precision lidar camera (Intel®Realsense^{TM} Lidar Camera L515 [39]) so that Ubi Edge is capable of detecting edges and touches (Figure 2a). The Hololens headset is responsible for digital content visualization and tracking users' finger positions. Also, the built-in SLAM of HoloLens is utilized to obtain the global position of the edge so that we can locate objects as well as edges in physical space. The lidar camera connected to a backpack computer performs geometric edge detection and tracking as well as object detection and tracking. We detect the touch interaction by combining finger tracking results from HoloLens2 and edge detection results from the lidar camera.

4.2.2 An integrated edge detection pipeline. To enable users to use any geometric edge in the surroundings, Ubi Edge accurately detect geometric edges through two steps: ***edge registration*** and ***edge matching***. The *edge registration* is responsible for ascertaining geometric edge points of the object before authoring and *edge matching* is responsible for locating edges during both authoring and application use.

During the *edge registration*, given the RGBD images for the target physical object, our system utilizes a vision-based algorithm [15] to extract geometric edge points, i.e., high curvature edge points, where surface normals suddenly change. However, the edge detection provided by [15] automatically detects feature points for all geometric edges in the scene (Figure 2b-1). Since edges on the object are our major concern, we then trimmed edges on the object out of the background. Specifically, we obtain the bounding box

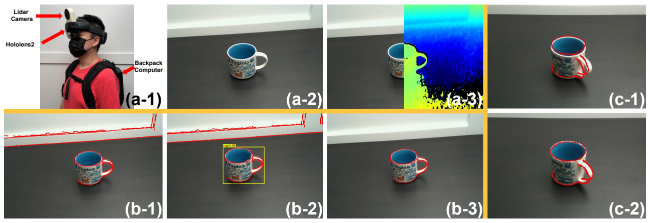


Figure 2: (a)Customized AR-HMD Setup: (a-1) The customized HoloLens2 with a Lidar camera. (a-2) Hololens2 camera view. (a-3) Lidar camera view with RGB+D information. (b)Edge Registration: (b-1) Detected edge feature points in the scene. (b-2) Object detection result for the cup. (b-3) Trimmed edge feature points of the target object. (c) ICP refinement: (c-1) Initial 6Dof prediction of edges. (c-2) 6DoF pose estimation of edges after ICP.

for the target physical object using a lightweight object detection method [72] (Figure 2b-2), project the bounding box back to 3D using the depth information, and remove the background feature points whose distance with the bounding box is beyond an empirical threshold of 2.5 cm and out of the bounding box (Figure 2b-3). Then we reconstruct the geometric outline of the object by merging all the extracted edge points captured from different viewpoints together. A post-processing filter [74] is utilized for outlier removal.

When the *edge registration* is completed, we have the geometric edge feature points of a physical object. However, when the object is moved to a new position, the edge points' 6DoFs change. Therefore, our system needs to establish correspondence between original edge points and current edge points so that our system can detect whether a user is interacting with an edge and decide which edge is being touched. To achieve this goal, we perform the *edge matching*. Specifically, we detect the object as we do in *edge registration* and track the object's 6Dof using an off-the-shelf network-based keypoint descriptor and spatial-temporal non-local memory [86] that is robust against hand occlusion. However, the 6Dof prediction result of the object does not exactly correspond to the pose of edges as there might be rotation errors or translation offsets (Figure2c-1). Therefore, we further refine the prediction by matching current detected edges with the original edges in *edge registration* through the edge-based ICP method [15] (Figure2c-2). During the application use, for each authored edge, Ubi Edge renders corresponding TUI when touch interaction takes place.

The edges detected and their descriptors so far are merely a set of 3D points without higher-order geometric structure (e.g. our system has not defined which subsets of 3D points constitute a line segment and has not estimated the parameters for describing the line segment) and cannot be directly used to map digital functions. To this end, our system lets users define the line segment by touch.

4.3 Input-output Model for Edge-based Opportunistic TUI Authoring

To achieve DG3, we provide a programming tool that adopts input-output model as the programming modality. The input-output model has been widely used in authoring systems to assist end-users

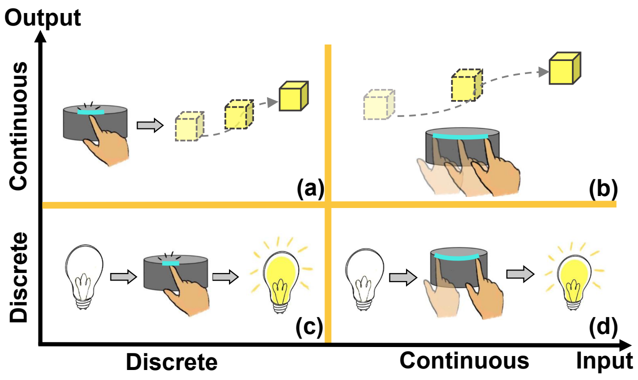


Figure 3: Input-output taxonomy: (a) a discrete touch input triggers a continuous behavior, (b) a continuous touch controls continuous output (c) a discrete touch input triggers a discrete output, and (d) a continuous touch triggers a discrete output.

in efficient customization [2, 27, 46, 53, 82]. Within the input-output model, an *input* is initiated by a subject and an *output* is generated in response to a trigger of the *input*. In our work, the *input* of the TUI interaction refers to touch interaction with a *tangible edge* and the *output* is the behavior of relevant digital content.

To achieve intuitive edge selection/manipulation (DG2), Ubi allows users to use body actions to demonstrate. Based on our review of previous tangible interaction works [13, 26, 55, 90], the single finger interaction causes the least occlusion while providing intuitiveness. Therefore, we adopt index finger touch-based interaction to enable intuitive and precise manipulation on edges. We summarize *input* along with the *output* into the following categories:

- *Discrete input*, which implies if a touch interaction takes place.
- *Continuous input*, which represents the status of the finger consisting of touch positions, touch manipulation path lengths, and finger moving directions.
- *Discrete output*, which indicates the state of digital content (e.g., transitions, rotations, scales) changes temporally after the touch interaction.
- *Continuous output*, which represents a series of digital content's state transitions after the touch interaction.

Furthermore, we would like to adopt a trigger-action programming metaphor to empower users with effortless authoring. Specifically, users can connect the *trigger*, i.e., *input*, with their target *action*, i.e, *output* and create combinations of different types of *inputs* with *outputs* along with the resulting TUI interactions.

Discrete + Discrete (Figure 3c): This category represents the digital content status changes from one to another after a touch interaction with a *tangible edge* is detected. The most common scenario is that an on/off function is triggered when the user touches the edge. An example is shown in Figure 1e-1. After the user touches the edge segment of the pill bottle cap, a virtual AR medication reminder indicating pills to take appears.

Discrete + Continuous (Figure 3a): This group means that a series of changes of digital content status occurs after a touch interaction is detected. A typical interaction is that an AR animation starts when the user touches the edge. We show an example in Figure 1e-3, where an AR weapon animation starts to play after the user touches the edge of the airplane model's wing.

Continuous + Discrete(Figure 3b): This category shows that the state of digital contents adjusts corresponding to the alteration of finger status during the touch interaction. Figure 1e-2 shows an example that the augmented album changes to the next page when the user moves the finger from top to bottom of the album edge.

Continuous + Continuous: This class demonstrates TUI interactions where the digital content responds concurrently to the finger status update. An intuitive scenario to come up with is that the edge serves as a slider. We provide an example in Figure 1d where the cup's rim is repurposed into a controller for changing the light color of the AR bulb and the AR bulb's color gradually changes from green to blue when users move their fingers from left to right.

Furthermore, to support users in defining *input* and *output* directly, Ubi Edge introduces two types of ***edge types*** and two types of ***behaviors*** of digital functions for users to choose: (1) *Discrete edge*, which is linked with the *Discrete input* of the input-output model and serves as a button activation. (2) *Continuous edge*, where a continuous touch interaction with touch position, finger moving direction, and length of current touch path will be monitored. (3) *Discrete behavior*, which represents limited or temporal status of digital contents (e.g. the on/off status, current AR content's position, etc.). And (4) *Continuous behavior*, which corresponds to the *Continuous output* such as 3D animations. With the support of trigger-action metaphor, users are able to connect one *input* to multiple *outputs* so that multiple *behaviors* can be activated at the time. Or they can associate multiple *inputs* to the same *output* and either *input* can activate the *output*.

4.4 Authoring Interface

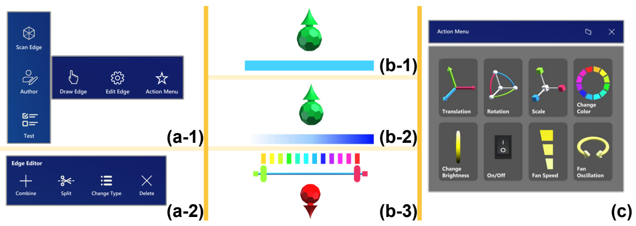


Figure 4: The AR interface of Ubi Edge. (a-1) Left-hand menu (upper). The left column is the main menu for three modes. The right column is the sub-menu corresponding to *Author Mode*. (a-2)Edge Editor Menu(bottom). (b-1) *Discrete edge*. (b-2) *Continuous edge*. (b-3) An example *Continuous behavior* icon with a slider where users can define the start and end of the behavior. (c) The *Action Menu* holds all *behaviors* for a digital object.

Leveraging the advantages of immersive experiences and affordances of spatial awareness provided by AR, we introduce the authoring interface that enables in-situ digital content manipulation, visualized embodied demonstration, and effortless trigger-action programming (Guided by DG4). The authoring interface of Ubi Edge is composed of three modes: (1) ***Scan Mode*** for users to scan all edges on the physical object, (2) ***Author Mode*** to design and edit TUIs for *tangible edges*, and (3) ***Play Mode*** to test and experience authored TUI applications. An AR main menu floating next to users' left hands is used for switching between different modes and toggling on/off sub-menus for current mode (Figure 4a-1).

Users start the TUI creation by first entering *Scan Mode* and registering edges within their interested region into Ubi Edge. During the *Author Mode*, users first select the *tangible edge*, i.e., the line segment that the user is going to interact with. After users click the button for *tangible edge* selection, an AR replica of detected edge point results is automatically aligned with the physical object (Figure 2b-3). Now users can select their desired *tangible edge* by sliding their index fingers along the physical edge. Ubi Edge starts/stops detecting users' interaction with the physical edge when users send a speech command which is provided by Microsoft Mixed Reality Toolkit (MRTK). At the same time, our system selects the points near users' index fingers and renders a smooth line segment for users through a line smoothing algorithm [29]. For each *tangible edge*, our system stores the inherent information including length, contained points, and path for later use.

When a *tangible edge* is available, users can utilize the 'Edge Editor' (Figure 4a-2) to author *input* mentioned in Section 4.3 and perform other modifications for the *tangible edge*. For each *tangible edge*, users can customize the *edge type* mentioned in Section 4.3 which specifies the touch interaction type. The visualization of edge types changes correspondingly as users modify the *edge type* (Figure 4b-1 and 4b-2). Ubi Edge supports users to map a set of *discrete edges* to multiple *behaviors* respectively. To this end, we provide the 'Split' function (Figure 4a-2) to users and users can split the edge into multiple *tangible edges* which share the same *edge type* property as the original *tangible edge*. Specifically, users can simply touch the current *tangible edge* and the *tangible edge* will be automatically segmented at the touch point. Users can also utilize the 'Split' function for *continuous edge* to take full advantage of an edge with moderate length and ease their authoring load. Divided edges can also be merged into their original status through the 'Combine' function using a similar manner. Additionally, to avoid the situation where users might mistakenly touch the nearby edge, the segmented edges are automatically divided by an empty interval with an empirical length of 1.1 cm. If the current authored edge overlaps with previously authored edges, our system automatically segments the current authored edge at the intersection point.

After users finish the authoring process for the *tangible edge*, users can specify the *behavior*, connects the *tangible edge* with *behavior* through trigger-action programming metaphor and finishes the authoring process. In detail, for each digital content, users drag a virtual behavior from the 'Action' menu (Figure 4c) which contains all available behaviors for the current digital object. Ubi Edge provides both universal behaviors for virtual contents (e.g. translations, rotations, scales, and animations) and pre-defined behaviors based on the characteristics of virtual contents. Users can decide whether the universal behavior change happens with respect to the users' coordinate or the global coordinate using a toggle button. For a *continuous behavior*, users need to specify the start and end values between where the edge will be mapped to (Figure 4 (b-3)). Then users can connect the authored *tangible edge* with target *behavior* through drag-and-drop interaction. Specifically, users pinch the green sphere above the *tangible edge*, drag a line out, approach the target *behavior* icon and release the pinch on the red sphere to enable the line connecting the *tangible edge* and target *behavior*. Now users can test the authored TUI application. In the *Play Mode*, Ubi Edge keeps tracking users' fingers and the edges' 6DoF as mentioned in Section 4.2 . Once the touch interaction is detected, both of the *tangible edge* and the connected *behavior* icon will be highlighted and the connected *behavior* will be triggered. Additionally, Ubi Edge will automatically hide all the lines and *behavior* icons and only the *tangible edge* is visualized.

5 APPLICATION SCENARIOS

With Ubi Edge, end-users are allowed to utilize the edges on nearby everyday objects to control virtual contents and IoTs. Here we demonstrate four different TUI applications enabled by Ubi Edge.

5.1 Repurpose Object into Multi-functional Controller

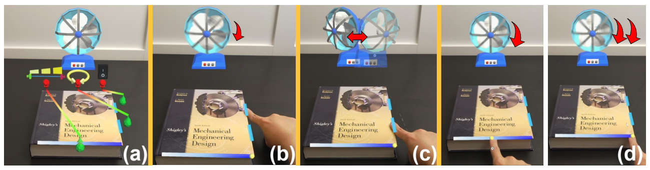


Figure 5: An AR fan controller: (a) The 'bottom' continuous of the book is connected to a fan speed behavior; the 'upper right' discrete edge is connected to an 'on/off' behavior; the 'lower right' discrete edge is connected to an 'oscillating' behavior. (b) The user can touch the 'upper right' edge to turn on the fan, (c) touch the 'lower right' edge to let the fan oscillate, and (d) slide his finger on the 'bottom' edge of the book to adjust the fan's speed.

Multi-functional controllers have been widely used for AR content manipulation due to the complexity of virtual contents [49, 94]. With the support of Ubi Edge, users can instantly repurpose a daily object into a multi-functional controller by choosing multiple edges on the object and connecting selected edges with different behaviors respectively. As shown in Figure 5a, a user connects the 'upper right' discrete edge of a book with an AR stand fan's 'on/off' behavior, the 'lower right' discrete edge of the book with the fan's 'oscillating' behavior, and the 'bottom' continuous edge with the 'fan speed' behavior. Thus, if the user touches the 'upper right' edge, the AR fan starts/stops working (Figure 5b). If the 'lower right' edge is touched, the AR fan starts to oscillate (Figure 5c), and then the user can change the fan speed by sliding on the 'bottom' edge of the book (Figure 5d).

5.2 Ubiquitous Control Over Smart Objects

Recently, nascent and consumer-oriented IoTs (e.g. smart light bulbs, thermostats, and speakers) which integrate physical objects with digital sensors, are rapidly increasing in people's working and living

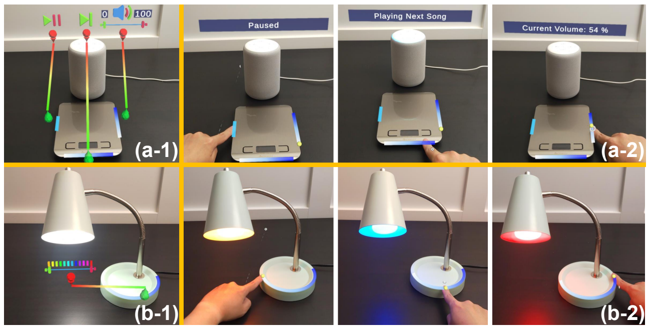


Figure 6: Audio player controller: (a-1) The 'left' discrete edge of the kitchen scale is connected to a 'pause/play' behavior; the 'bottom' continuous edge is connected to 'next/previous' behavior, and the 'right' continuous edge is connected to a 'volume' behavior. (a-2) The user can touch the 'left' edge to play music, swipe the bottom edge to jump to the next song and touch the 'right' edge to adjust the volume. Smart bulb controller: (b-1) The 'bottom' continuous edge of the lamp station is connected to 'hue' behavior of the IoT bulb. (b-2) The user can change the color from yellow, to blue and to red when moving his finger from the left of the edge to the middle, and to right.

environments [50, 52]. Currently, the most common way to control IoTs is to use a smartphone. Yet, our system empowers users to utilize the nearby edges to manipulate IoT functions. In Figure 6, we showcase a user trying to control IoTs using his table-around items. The user maps the three edge segments (i.e., 'left' discrete edge, 'bottom' continuous edge, 'right' continuous edge) of digital scale with an audio player's 'play/pause', 'previous/next song', and 'volume dial' respectively (Figure 6a-1). During the use, he can start to enjoy music when he touches the scale's 'left' edge. Now he could change to the next rhythm by swiping right on the 'bottom' edge. If he swipes down along the 'right' edge, then the audio player's sound will be dialed down (Figure 6a-2). Additionally, the user connects the continuous edge of the lamp's station with the light bulb hue (Figure 6b-1) and he can touch the station's edge to modify the light bulb's hue (Figure 6b-2).

5.3 Interactive Tangible AR Game

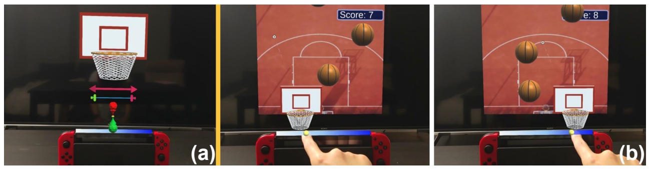


Figure 7: Gaming Controller: (a) The 'upper' continuous edge is connected to the virtual basket's 'horizon translation' behavior. (b) Then the user can slide his finger on the edge to let the basket catch falling basketballs.

Besides utilizing ambient edges to improve the quality of life, users are also able to create AR game controllers for entertainment with the support of Ubi Edge. Here we demonstrate an AR example in Figure 7. The user wants to utilize a handheld game console's edge to control the AR basket in the horizontal direction so that the basket can catch falling basketballs. Thus, the user connects the 'upper' edge with the AR's basket's 'horizon translation' behavior (Figure 7a). During the play, when the user slides his finger in the right direction, the AR basket correspondingly moves in the same direction (Figure 7b).

5.4 Ubiquitous Tangible AR Tutorial

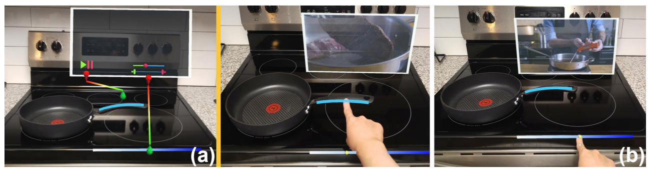


Figure 8: Virtual video tutorial controller: (a) The continuous electric range edge is connected to a progress bar and the discrete pan handle edge is connected to a 'play/pause' behavior. (b) The user can click on the handle of the pan to start/pause the video and utilize the range edge to jump to the next part.

Ubi Edge can also facilitate task performance for users. For example, users can author a tutorial-based TUI by attaching a video display behavior to a frying pan's edge for cooking study. Specifically, the user attaches 'play/pause' to the discrete edge on the pan handle, and 'playing progress' behavior to the continuous electric range edge (Figure 8a). Then the user can start watching the cooking video when his finger touches the pan handle edge and he can jump to his favorite parts by sliding on the range edge (Figure 8b).

6 SOFTWARE AND HARDWARE SETUP

As mentioned in Section 4.2.1, we build our customized AR-HMD as shown in Figure 2(a), which consists of a HoloLens2 AR-HMD with built-in SLAM and an Intel®Realsense™ lidar camera L515 [39]) for edge tracking. The lidar camera and HoloLens camera are calibrated ahead and a bidirectional data transfer between AR-HMD and the lidar camera is enabled by the Transmission Control Protocol (TCP) [71] connection in the local network. For finger tracking, instead of tracking certain gestures, the HoloLens 2 provides hand skeleton detection and can track all joint positions of both hands. We choose the index tip position mentioned in Section 4.2 and Section 4.3. During the edge detection pipeline, we adopt a resolution of 1280×720 for the RGB color image and a resolution of 1024×768 for the depth image. The color and depth frames are aligned with the same frame rate of 30 fps. In an indoor setting, the lidar camera is designed to capture depth from 0.25 m to 9 m, with reflectivity affecting the depth. Additionally, the lidar camera is connected to a backpack computer (HP VR Backpack [36], Intel Core i7-8850H, 2.6GHz, 32GB RAM, NVIDIA 2080 GPU), which provides computation power for edge detection and tracking.

As mentioned in Section 4.2, we applied object detection [72] method for background edge point removal and further object tracking. For object detection, we trained 13 daily objects including a photo frame, table, cup, book, toy, headphone, frying pan, desk, bowl, lamp, pill bottle, and monitor for study and demonstration use. For each object, we collected 2000 images. The detection model was pre-trained on ImageNet [17]. For the object detection result, the overall precision is 96.5% and mAP is 91.7%. For object tracking and edge tracking, we adopted the default settings in original algorithms respectively [15, 86]. The AR system is developed with Unity3D (2020.3.31f1) and the AR user interface was supported by Microsoft Mixed Reality Toolkit (MRTK)[1]. While the training and tracking experiments are conducted on the backpack computer mentioned above, the AR development is finished on a local PC (Intel Core i7-9700K, 3.6GHz CPU, 32GB RAM, NVIDIA RTX2080 GPU).

7 PRELIMINARY SYSTEM EVALUATION FOR EDGE DETECTION

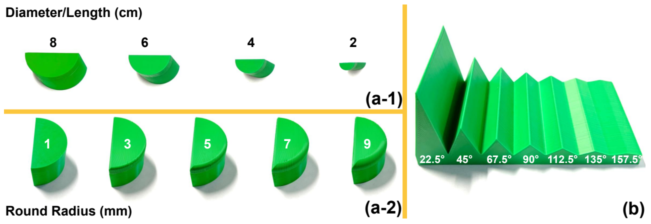


Figure 9: Preliminary evaluation setup: 3D printed half cylinders with different (a-1) diameters and (a-2) round radii. (b) Wedges with different dihedral angles.

Following DG1, we develop a vision-based edge detection pipeline in Section 4.2 to detect edges in the physical environment. In real-world scenarios, the edge detection capability is affected by edge properties including the curvature, the length, and the distance between edges and the lidar camera. Thus, in this section, we evaluate the effectiveness of edge detection with varying geometric edge parameters.

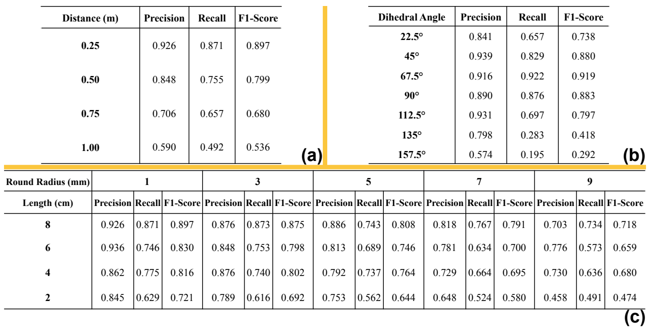
Setup. In the evaluation, we mainly consider four edge parameters: the rounded edge's radius [73], dihedral angles (Figure 9b), the length of edges, and the distance between edge and the camera which may infuence the performance of the edge detection algorithm. Specifically, the round radius and dihedral angle represent the curvature/sharpness of edges. In our life, the edge of actual objects is usually not a straightly sharp edge but a high-curvature surface, like a thin cylinder [75]. The round radius refers to the cylinder radius and the edge looks blunter as the round radius increases (Figure 9a-2). Previous works [1, 4] show that the sharpness also depends on the dihedral angle and we adopt the same parameter settings in these works. For the edge length which is represented by the diameter length of the half cylinder in Figure 9a-1, the minimum length of 2 cm is chosen to ensure the edge is long enough for fingers to interact. Moreover, the distance parameter is decided based on the specification of the lidar camera [39].

Procedure. We 3D print half cylinders with a diameter ranging from 2 cm to 8 cm (Figure 9a-1). For each size, we choose 5 round

[1]https://github.com/microsoft/MixedRealityToolkit-Unity

radii (Figure 9a-2). In total, we use 4 (length) $\times$ 5 (round radii) = 20 half cylinders to evaluate the impact of edge curvature and length. When measuring the effect of curvature and edge length, the distance is set at the same of 0.25m. To test the edge detection accuracy, we compare the results of detected edge feature points with the manually labeled ground truth. We adopted the same ground truth setting and threshold setting as the previous work [4]. For evaluation, we calculated three metrics: Precision, Recall, and F_1 score [79]: $F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$, as shown in Figure 10.

Result and Discussion. The result (Figure 10c) shows that edge detection is accurate when round radius $\leq$ 7mm. Our system can detect the short edge (diameter = 2 cm) for the small round radius. For the dihedral angle, the results are more accurate between 22.5° and 112.5° (Figure 10b). The optimal distance between the camera and the edges would be equal to or less than 0.5 m (Figure 10a). We will discuss the limited detection range later in Section 9. Overall, our results show state-of-the-art performance compared with the result of [1, 4].



Distance (m)	Precision	Recall	F1-Score
0.25	0.926	0.871	0.897
0.50	0.848	0.755	0.799
0.75	0.706	0.657	0.680
1.00	0.590	0.492	0.536

(a)

Dihedral Angle	Precision	Recall	F1-Score
22.5°	0.841	0.657	0.738
45°	0.939	0.829	0.880
67.5°	0.916	0.922	0.919
90°	0.890	0.876	0.883
112.5°	0.931	0.697	0.797
135°	0.798	0.283	0.418
157.5°	0.574	0.195	0.292

(b)

Round Radius (mm)	1			3			5			7			9		
Length (cm)	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
8	0.926	0.871	0.897	0.876	0.873	0.875	0.886	0.743	0.808	0.818	0.767	0.791	0.703	0.734	0.718
6	0.936	0.746	0.830	0.848	0.753	0.798	0.813	0.689	0.746	0.781	0.634	0.700	0.776	0.573	0.659
4	0.862	0.775	0.816	0.876	0.740	0.802	0.792	0.737	0.764	0.729	0.664	0.695	0.730	0.636	0.680
2	0.845	0.629	0.721	0.789	0.616	0.692	0.753	0.562	0.644	0.648	0.524	0.580	0.458	0.491	0.474

(c)

Figure 10: Preliminary evaluation of edge detection results for different (a) camera distances (diameter = 8 cm, round radius = 1mm), (b) dihedral angles (camera distance = 0.25 m), (c) edge lengths and round radius (camera distance = 0.25 m).

8 USER STUDY

We conducted a two-session user study to evaluate the accuracy of the touch interaction with edges, TUI authoring feasibility, and overall system usability. We recruited 14 users (11 males and 3 females, ages ranging from 21 to 30). 12 out of 14 had used AR or VR applications on smartphones, tablets, or head-mounted devices while the rest two had heard about AR and VR. 3 out of 14 had used TUI before and the others had a basic understanding of TUI concepts. Since Ubi Edge is designed to provide prototyping experiences for non-expert end-users, all the recruited users have no AR application programming experiences. None of them had used our system before the user study. The entire study took around 1.5 hours and each user and each user was paid a $20 e-gift card. The study was taken in a 5m by 5 m indoor area and screen-recorded for post-analysis. After a user arrived, we first introduced the background of our system and let users understand the entire system workflow and system UIs. Before the user study officially started, users first experienced the HoloLens2 built-in tutorial to learn basic usage of the HMD. After two sessions, each user completed a 5-scale Likert-type questionnaire and a standard System Usability Scale (SUS) questionnaire. Finally, we conducted a conversation-style interview with the users to obtain their subjective feedback about the system.

8.1 Quantitative Evaluation of the Touch Interaction

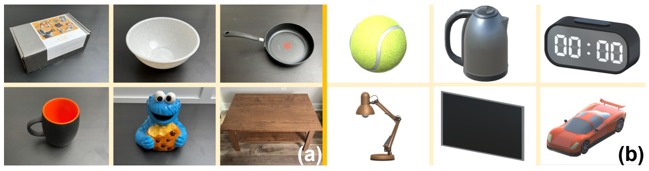


Figure 11: User study setup: (a) The six objects used in Session 1 and Session 2: a box, a bowl, a frying pan, a cup, a toy, and a coffee table. (b) The six virtual objects used in Session 2: a tennis ball, a kettle, a timer, an AR lamp, a video player, and a car.

8.1.1 Procedure. In this session, we evaluated the performance of the edge detection result, checked whether touch interaction with the edge can be correctly detected, and measured whether users can accurately manipulate both types of *tangible edges*. We provide 6 daily objects which vary in size, edge numbers, and edge curvatures for users: a box, a bowl, a frying pan, a cup, a toy, and a coffee table (Figure 11a). For each object, users first completed the scanning process as mentioned in Section 4.2. Then during authoring, users could choose any two tangible edges they want. One authored edge was set to *discrete* and connected with a 'counter' behavior, which visualized the number of detected touches. The other authored edge was set to *continuous* and connected with a 'slider' behavior, which showed the ratio of current touch position to start with respect to the edge length. During the test, users were asked to 1) perform 5 touches for each *discrete edge*, and 2) slide along the *continuous edge* and tries to position their fingers on 0%, 25%, 50%, 75% and 100% of the edge length. When touching the *discrete edges*, users were encouraged to touch different places on the edge. Following previous works [16, 43, 44], We recorded whether the system detected touch interaction for *discrete edges* and the target acquisition time of trials for *continuous edge*. We collected a total of 14 (Participants) $\times$ (6 objects) $\times$ 5 = 420 trials for both *discrete edge* and *continuous edge* respectively. Specifically, the target acquisition time refers to the total completion time for users to slide along the *continuous edge* and traverse the specific points, i.e, 0%, 25%, 50%, 75% and 100% of the edge length.

8.1.2 Result and Discussion. The evaluation result is shown in Figure 12. For the *discrete edges*, our system could accurately detect the touches performed by users with an average accuracy of 93.57% (SD=3.66) across six objects. Typically, the performance of the toy was not as good as the other objects. This was partially caused by that there are many concave edges detected on the object and it would cause the fingertip occlusion problem which we discuss later in Section 9. The frying pan was also challenging and many

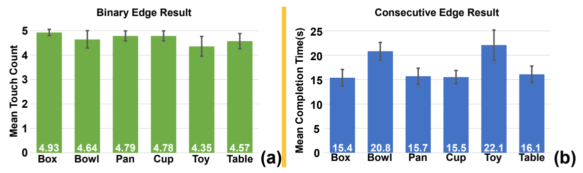


Figure 12: Results of User Study Session 1

short edges were detected due to the abrasive material on the pan's bottom area, which cluttered the visualization and the users could not accurately select their desired edges.

For the *continuous edge*, all users were able to successfully slide to the target positions. The results show that users could accurately manipulate the authored *tangible edges*. The average completion time to slide to all target positions for each edge was 17.59 seconds (SD = 4.32). This result is comparable with other state-of-art tangible sliders [16, 43]. It takes more time for users to complete tasks on the bowl (AVG = 20.82s, SD = 3.59) because users preferred to draw longer edges on the bowl. We also observed that users took more time on the toy (AVG = 22.07s, SD = 6.19) and this may be attributed to the toy's ceramic material and the edges do not provide as sharp a tactile feel as other objects.

8.2 Authoring Feasibility Evaluation

TUI	Edge (type)	Behavior (type)
1	Box's length, height, width (*continuous*)	A tennis ball's movement in x, y, z axis respectively (*continuous*)
2	Bowl's rim (*continuous*)	Virtual kettle's rotation in y axis (*continuous*)
3	Handle of frying pan (*continuous*)	Virtual timer's 'Add/Delete 30s' (*discrete*)
4	(a) Rim of cup (*continuous*) (b) Handle of cup (*continuous*)	(a) Virtual lamp's color (*continuous*) (b) Virtual lamp's illuminance (*continuous*)
5	(a) Toy's edge on cookie (*discrete*) (b) Toy's edge on mouth (*continuous*)	(a) Video Player's 'play/pause' (*discrete*) (b) Video Player's progress bar (*continuous*)
6	Coffee table's two edges (*discrete*)	Animations of (a) virtual car and (b) virtual tennis (*continuous*)

(a)

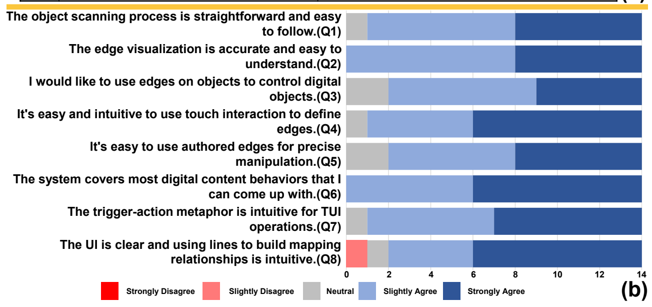


Figure 13: User study 2. (a) Six authoring tasks with different combinations of *input* and *output*. (b) Likert-type questionnaire results.

8.2.1 Procedure. In this session, we evaluated the performance of Ubi Edge authoring interface and let the users experience four combinations of *inputs* and *outputs* mentioned in Section 4.3. Users were asked to finish 6 edge-based opportunistic TUI authoring tasks (Figure 13a) by utilizing same physical objects as the first session (Figrue 11a) and 6 virtual objects (Figure 11b): 1) Box's length, height, width (continuous) of the box to control tennis ball's translation (continuous) in x, y, z directions respectively, 2) rim of bowl (continuous) to control virtual kettle's rotation (continuous) around y axis, 3) a continuous edge on frying pan to control a timer's 'Add/Delete 30 seconds' behavior (discrete), 4) rim of cup (continuous) to control virtual lamp's color (continuous) and the handle of cup (continuous) to control virtual lamp's brightness (continuous), 5) a discrete edge on toy's cookie to control video player's 'play/pause' (discrete) and a continuous edge on toy's mouth to control the video player's progress bar (continuous), and 6) coffee table's edges (discrete) to control animations of a virtual car and a tennis ball (continuous). We recorded whether users successfully interacted with the TUI during *Play Mode.*

8.2.2 Result and Discussion. All 14 users completed the six authoring tasks and tested the authored TUI applications. The average completion time to finish authoring tasks for each user was 38.65 minutes (SD=4.35). The Likert-type results collected from this session are shown in Figure 13b.

In general, users showed a high preference for utilizing geometric edges on everyday objects for tangible inputs and agreed with the intuitiveness of re-purposing tangible edges into controllers (Q3: AVG=4.57, SD=0.51). *"I think it would bring high convenience to my life. With the edge controller, I can turn everything nearby into a controller such as the arm of a chair without looking for the controller all around the house. (P1)"* As a design foundation of our system, the trigger-action programming metaphor received compliments from users (Q7:AVG=4.43,SD=0.51). *"The procedure of combining different inputs with outputs is straightforward. And I can create different controllers with the combinations. (P13)"* The embodied demonstration through touch was also welcomed by users (Q4: AVG=4,28, SD=0.72). *"Using touch is pretty simple. The system immediately shows an edge as I draw it and exactly renders the edge I want. (P7)"* Regarding the object scanning process, most users were satisfied with the scanning pipeline (Q1, AVG=4.36, SD=0.93). *"It's necessary and convenient to scan the objects which I want to turn into controllers because I only need to scan once. And I do not need to place a QR code on objects all the time. (P3)"* Yet, one user raised that *"Although the system provides everything I need from scratch, I prefer that the scanning process can be directly skipped. (P11)"* We will discuss this concern in Section 9. Meanwhile, the majority of the users are satisfied with the edges' in-situ visualization results in quality and quantity(Q2: AVG=4.43, SD=0.65). *"I am surprised that all the edges I drew are precisely aligned with the physical objects. It's cool that I use both curves and straight lines as controllers. (P4)"* Meanwhile, most participants felt confident in utilizing authored edges for precise manipulations (Q5: AVG=4.5,SD=0.65). *"I am pretty satisfied with the edge controller that I created. The simple touch trigger makes it perform just like a real button. When it functions as a slider, it's much like using a touchpad in the physical environment and the virtual objects behave as I expected. (P15)"* Users also acknowledged the coverage of digital contents (Q6:AVG=4.21, SD=0.70). *"The system comes up with almost all interactions for virtual contents. Initially, I can only imagine using it as a simple button. After using your system, I'm surprised that I can utilize it to create and control a 3D animation. (P8)"* Additionally, users complimented the UIs and the intuitive operations provided by immersive authoring (Q8:AVG=4.36, SD=0.63). *"Using a line to represent the logic connection is really simple and easy to follow. (P11)"* The standard SUS survey result is 87.86 out of

100 with a standard deviation of 9.08, showing the high usability of the whole system.

9 LIMITATIONS AND FUTURE WORK

Difficulties in edge detection. The in-situ edge visualization was proven to be accurate and covered most edges that users needed in the user study. Some users raised a concern that *"For some objects like the frying pan and the toy, there are much more edges than I expected. Although it does not affect selecting edges, it seems a little messy for me. (P10)"* For the toy, we noticed that many concave edges are presented. Since the edge detection algorithm does not distinguish concave edges from convex edges in 3D, the only way to differentiate the two types of edges is based on the users' view angle. For the frying pan, there are many unexpected edges because the bottom of the pan is made of an abrasive material so many tiny edges are detected. Similarly, the quality of edge detection greatly reduces when meeting transparent materials like glass. This problem has been addressed in [14] and we envision more robust edge detection algorithms could improve the performance of Ubi Edge in the future. Currently, as symmetric objects (in shape and visual appearance) are rotation-invariant to the Lidar camera, we are not able to track the rotation of these kinds of objects. This problem is also addressed in [23]. However, we can record the relative rotation of authored edge with respect to the user's location and renders an authored edge that keeps fixed relative rotation with users. Additionally, as users can only take RGBD images for their interested area, they may only take a partial edge segment for a long edge on large objects such as the table ridge. In this situation, there might be numerous similar edge segments on the long edge, and our system currently is not able to distinguish these similar edges. Similar to the rotation problem, we can also render the edge based on users' locations so that the edge is always within the user's reach. The partial edge segment problem can be expanded to a more general difficulty when the object falls out of the field of view (FoV) or is occluded by other objects/hands. As the 6DoF tracking method we used involves the feature point detection and tracking [86], the edge localization easily fails when most feature points are not in the FoV. Or in a clustered environment, the feature points of objects are more likely to be occluded by other items and cannot be detected. To deal with this issue, recent works [19, 104, 105] proposed learning-based methods which are robust to heavy occlusions. On the other hand, one benefit of the haptic feedback of the edge-based interfaces is that it enables an eyes-free experience. However, by adopting a vision-based edge detection and vision-based hand detection workflow, the current UbiEdge is not yet an eyes-free interface. Adding sensors on users' fingers to detect the touches with edges [63, 77] may be a solution to resolve the problem.

Difficulties in fingertip detection. The edge detection and tracking supported by Ubi Edge focus on geometric edges with sharp haptic feedback (e.g. ridges of desks, or rims of books). And the touch interaction with the sharp edges was proven to be adequate and accurate in the user study. However, some users mentioned that some concave edges are not rendered, and sometimes touch interactions with detected concave edges could not be detected. A user commented *"When I want to draw edges at the inside of the toy's mouth, the system does not correctly detect my finger moving. (P12)"* The situation also happens to concave corners. We noticed that when users want to touch the concave corner which connects the cup's handle with the cup's body, the finger detection capability provided by HoloLens2 is reduced because a significant part of the fingertip is occluded by the object. Introducing extra hardware devices onto fingers [9, 18, 66, 91] could be one solution. However, it usually requires extra setup and is cumbersome for users. Another possible way to address this problem is to utilize novel occlusion-based finger tracking algorithms [42, 62]. Or we can provide an evaluation score for each selected edge and thereby inform users of suitable edge candidates for TUIs.

Expansion of demonstrated gesture. Currently, Ubi Edge only supports index finger tracking and detection for two main reasons: 1) everyday edges are usually slim and more fingers would cause a severe occlusion problem, and 2) using the index finger is one of the most intuitive and simplest ways to interact with objects [26, 55, 90]. The index finger interaction was also highly accepted in our user study. Although Ubi Edge only focuses on the index finger's interaction with edges, HoloLens2 supports multi-finger tracking for both hands. Therefore, the system can be easily expanded to multi-finger/multi-hand interaction. Moreover, , some users suggested that *"It would be amazing if I can use more gestures or patterns like swiping back and forth to trigger a button (P13)."* In order to solve this problem, previous works have proposed to integrate gesture recognition devices with nearby smart devices or physical objects and thereby tackle the problem of the complex configuration of hands [52, 82, 87].

Distinction between object's original functions and TUI applications. While we enable always-on detection of touch interaction, users can mistakenly trigger the TUI when using an object's originally designed function. Thus, when and where to turn on the TUI becomes a problem. ICon [13] suggests that everyday objects can function as controllers only when they are on a desktop. Another method to adapt to different applications is to observe the way that users interact with objects through the learning-based method. For example, users may perform a fixed pattern of gestures when interacting with the object as a controller.

More complicated edge-based opportunistic TUIs. Ubi Edge provides TUIs with different combinations of *inputs* and *outputs* based on the trigger-action programming metaphor. However, some complex logic connections are currently not supported. *"I hope there will be a way to author a sequential connection. I mean, it's like a digital content behavior can happen only after another behavior is triggered. For example, I can let the coffee machine work only after my kitchen light is turned on and when I am entering the kitchen. (P4)"* This condition-based connection requires more spatial information about users and ambient physical surroundings. One possible solution is to bring a context-aware programming interface like Ivy [20], CAPturAR [83] and ProGesAR [92]. Additionally, for an interface that contains multiple edges, users currently need to specify each edge input and corresponding digital function respectively. For example, for a machine interface containing multiple buttons and knobs, the repetition of the authoring process might be tedious. The problem can be solved by reusing the customized elements [98] onto a new TUI through a copy-paste metaphor.

Bulky hardware setup. We have shown the intuitiveness and smoothness of the object scan process in the user study. Yet, as mentioned before, P11 would like the object scan prepared by developers before authoring. An alternative solution is to utilize the mesh model provided by the manufacturer, for example, IKEA dataset[60] and extract all edge points on the mesh model first [45, 78], and then compare with edges on physical objects during play. While the backpack computer provides the capability of real-time edge and object tracking, the bulky setup still limits users' mobility. We envision lightweight HMD's with compelling computation capability or a low-latency cloud service would solve the problem in the future. Moreover, the depth accuracy provided by the chosen lidar camera ($\sim$ 5 mm to $\sim$ 14 mm through 9 m^2 [39]) limits the edge detection accuracy. The preliminary system evaluation results (Section 7) show that the edge detection is more inaccurate when the distance between edges and the camera gets larger (>0.75 m). It is mainly due to the increased depth inaccuracy of the lidar camera at a further distance. The depth inaccuracy also leads to edge detection imprecision for a narrow interface such as keyboards and calculators with closely arranged buttons. Previous work that maps a virtual keyboard onto a physical keyboard requires additional trackers and is limited to specific keyboard models [8, 61]. We also envision that lightweight lidar cameras with high precision would solve the depth accuracy problem and ease the hardware setup in the future.

10 CONCLUSION

In this work, we present Ubi Edge, an AR authoring system for end-users to create customized opportunistic TUI applications on geometric edges. Ubi Edge enables users to interact with geometric edges on physical objects, which provides sharp haptic feedback, and repurposes edges as personalized tangible controllers through simple touches. We first discuss the benefits of opportunistic TUIs and the necessity of re-purposing geometric edges for TUIs. Then we present the overall system workflow and briefly walk-through of our system which contains three main steps: scanning edges of physical objects, authoring a personalized opportunistic TUI using scanned edges, and playing with the authored opportunistic TUI application. We also illustrate our integrated edge detection pipeline. Next, we explain the interaction taxonomy with edges and introduce an authoring model that maps different types of touch inputs to target behaviors of digital content. Guided by the authoring model, we design our authoring interface which enables users to prototype various interactions in situ by using the trigger-action programming metaphor. Further, to explore our system's scalability, we demonstrate four application scenarios supported by Ubi Edge: multi-function controller, smart home, interactive game, and TUI-based tutorial. In the user study, we first quantitatively evaluated the touch interaction results and demonstrated that Ubi Edge is able to accurately detect the touch interaction input that users authored. Then, we tested the usability of Ubi Edge and obtained positive feedback from users. To sum up, we believe that Ubi Edge opportunistically presents a novel perspective of exploiting geometric edges as a tangible input to create personalized TUI applications. We also envision our edge-based opportunistic TUI inspiring local feature-based TUI development in the future.

REFERENCES

[1] Syeda Mariam Ahmed, Yan Zhi Tan, Chee Meng Chew, Abdullah Al Mamun, and Fook Seng Wong. 2018. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 7350–7355.

[2] 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.

[3] 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.

[4] Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. 2015. Fast and robust edge extraction in unorganized point clouds. In *2015 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 1–8.

[5] Vincent Becker, Sandro Kalbermatter, Simon Mayer, and Gábor Sörös. 2019. Tailored controls: Creating personalized tangible user interfaces from paper. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*. 289–301.

[6] Saket Bhardwaj and Ajay Mittal. 2012. A survey on various edge detector techniques. *Procedia Technology* 4 (2012), 220–226.

[7] Irving Biederman and Ginny Ju. 1988. Surface versus edge-based determinants of visual recognition. *Cognitive psychology* 20, 1 (1988), 38–64.

[8] Sidney Bovet, Aidan Kehoe, Katie Crowley, Noirin Curran, Mario Gutierrez, Mathieu Meisser, Damien O Sullivan, and Thomas Rouvinez. 2018. Using traditional keyboards in vr: Steamvr developer kit and pilot game user study. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE, 1–9.

[9] Volkert Buchmann, Stephen Violich, Mark Billinghurst, and Andy Cockburn. 2004. FingARtips: gesture based direct manipulation in Augmented Reality. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. 212–221.

[10] John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698.

[11] 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.

[12] 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.

[13] 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.

[14] Changhyun Choi and Henrik I Christensen. 2012. 3D textureless object detection and tracking: An edge-based approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3877–3884.

[15] Changhyun Choi, Alexander JB Trevor, and Henrik I Christensen. 2013. RGB-D edge detection and edge-based registration. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1568–1575.

[16] Maxime Cordeil, Benjamin Bach, Andrew Cunningham, Bastian Montoya, Ross T Smith, Bruce H Thomas, and Tim Dwyer. 2020. Embodied axes: Tangible, actuated interaction for 3d augmented reality data spaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[18] Klaus Dorfmuller-Ulhaas and Dieter Schmalstieg. 2001. Finger tracking for interaction in augmented environments. In *Proceedings IEEE and ACM international symposium on augmented reality*. IEEE, 55–64.

[19] José Guedes dos Santos Júnior, João Paulo Silva do Monte Lima, and Veronica Teichrieb. 2021. Occlusion-robust method for RGB-D 6-DOF object tracking with particle swarm optimization. *Expert Systems with Applications* 174 (2021), 114736.

[20] 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.

[21] Jerry Alan Fails and Dan Olsen Jr. 2002. Light widgets: interacting in everyday spaces. In *Proceedings of the 7th international conference on Intelligent user interfaces*. 63–69.

[22] Simon Flöry and Michael Hofer. 2008. Constrained curve fitting on manifolds. *Computer-Aided Design* 40, 1 (2008), 25–34.

[23] Markus Funk, Oliver Korn, and Albrecht Schmidt. 2014. An augmented workplace for enabling user-defined tangibles. In *CHI'14 Extended Abstracts on*

Human Factors in Computing Systems. 1285–1290.

[24] Gregory J Gerling and Geb W Thomas. 2005. The effect of fingertip microstructures on tactile edge perception. In *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics Conference.* IEEE, 63–72.

[25] Aakar Gupta, Bo Rui Lin, Siyi Ji, Arjav Patel, and Daniel Vogel. 2020. Replicate and reuse: Tangible interaction design for digitally-augmented physical media objects. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–12.

[26] Chris Harrison, Hrvoje Benko, and Andrew D Wilson. 2011. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology.* 441–450.

[27] 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.

[28] Steven J Henderson and Steven Feiner. 2008. Opportunistic controls: leveraging natural affordances as tangible user interfaces for augmented reality. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology.* 211–218.

[29] John Hershberger and Jack Snoeyink. 1994. An O (n log n) implementation of the Douglas-Peucker algorithm for line simplification. In *Proceedings of the tenth annual symposium on Computational geometry.* 383–384.

[30] Anuruddha Hettiarachchi and Daniel Wigdor. 2016. Annexing reality: Enabling opportunistic use of everyday objects as tangible proxies in augmented reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* 1957–1967.

[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] Valentin Heun, Shunichi Kasahara, and Pattie Maes. 2013. Smarter objects: using AR technology to program physical objects and their interactions. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems.* 961–966.

[33] Chems-Eddine Himeur, Thibault Lejemble, Thomas Pellegrini, Mathias Paulin, Loic Barthe, and Nicolas Mellado. 2021. PCEDNet: A Lightweight Neural Network for Fast and Interactive Edge Detection in 3D Point Clouds. *ACM Transactions on Graphics (TOG)* 41, 1 (2021), 1–21.

[34] David Holman, Andreas Hollatz, Amartya Banerjee, and Roel Vertegaal. 2013. Unifone: Designing for auxiliary finger input in one-handed mobile interactions. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction.* 177–184.

[35] Hololens 2 2022. Hololens 2: Mixed Reality Technology for Business. https://www.microsoft.com/en-us/hololens.

[36] HP Z VR Backpack Workstation 2022. HP Z VR Backpack Workstation: VR this powerful is no toy. https://www.hp.com/us-en/shop/mdp/hp-z-vr-backpack-g1-workstation–1.

[37] Kohsia S Huang and Mohan M Trivedi. 2004. Robust real-time detection, tracking, and pose estimation of faces in video streams. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 3. IEEE, 965–968.

[38] Ke Huo and Karthik Ramani. 2017. Window-shaping: 3d design ideation by creating on, borrowing from, and looking at the physical world. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction.* 37–45.

[39] Intel RealSense LiDAR Camera L515 2022. Intel® RealSense™ LiDAR Camera L515. https://www.intelrealsense.com/lidar-camera-l515/.

[40] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems.* 234–241.

[41] Sungjune Jang, Lawrence H Kim, Kesler Tanner, Hiroshi Ishii, and Sean Follmer. 2016. Haptic edge display for mobile tactile interaction. In *Proceedings of the 2016 CHI conference on human factors in computing systems.* 3706–3716.

[42] Youngkyoon Jang, Seung-Tak Noh, Hyung Jin Chang, Tae-Kyun Kim, and Woontack Woo. 2015. 3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *IEEE Transactions on Visualization and Computer Graphics* 21, 4 (2015), 501–510.

[43] Yvonne Jansen, Pierre Dragicevic, and Jean-Daniel Fekete. 2012. Tangible remote controllers for wall-size displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 2865–2874.

[44] Nikhita Joshi and Daniel Vogel. 2019. An evaluation of touch input at the edge of a table. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* 1–12.

[45] C-Y Kao, Soundar R. T. Kumara, and Rangachar Kasturi. 1995. Extraction of 3D object features from CAD boundary representation using the super relation graph method. *IEEE transactions on pattern analysis and machine intelligence* 17, 12 (1995), 1228–1233.

[46] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology.* 395–405.

[47] Annie Kelly, R Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. 2018. ARcadia: A rapid prototyping platform for real-time tangible interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* 1–8.

[48] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. 2017. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 1–10.

[49] Pascal Knierim, Dimitri Hein, Albrecht Schmidt, and Thomas Kosch. 2021. The SmARtphone Controller. *i-com* 20, 1 (2021), 49–61.

[50] Thomas Kubitza and Albrecht Schmidt. 2017. meSchup: A platform for programming interconnected smart things. *Computer* 50, 11 (2017), 38–49.

[51] Tobias Langlotz, Stefan Mooslechner, Stefanie Zollmann, Claus Degendorfer, Gerhard Reitmayr, and Dieter Schmalstieg. 2012. Sketching up the world: in situ authoring for mobile augmented reality. *Personal and ubiquitous computing* 16, 6 (2012), 623–630.

[52] Gierad Laput and Chris Harrison. 2019. SurfaceSight: a new spin on touch, user, and object sensing for IoT experiences. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* 1–12.

[53] David Ledo, Jo Vermeulen, Sheelagh Carpendale, Saul Greenberg, Lora Oehlberg, and Sebastian Boring. 2019. Astral: Prototyping Mobile and Smart Object Interactive Behaviours Using Familiar Applications. In *Proceedings of the 2019 on Designing Interactive Systems Conference.* 711–724.

[54] Gun A Lee, Gerard J Kim, and Mark Billinghurst. 2005. Immersive authoring: What you experience is what you get (wyxiwyg). *Commun. ACM* 48, 7 (2005), 76–81.

[55] Gun A Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. 2004. Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality.* IEEE, 172–181.

[56] Germán Leiva and Michel Beaudouin-Lafon. 2018. Montage: a video prototyping system to reduce re-shooting and increase re-usability. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology.* 675–682.

[57] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2021. Rapido: Prototyping Interactive AR Experiences through Programming by Demonstration. In *The 34th Annual ACM Symposium on User Interface Software and Technology.* 626–637.

[58] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid augmented reality video prototyping using sketches and enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–13.

[59] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-user development: An emerging paradigm. In *End user development.* Springer, 1–8.

[60] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. 2013. Parsing ikea objects: Fine pose estimation. In *Proceedings of the IEEE international conference on computer vision.* 2992–2999.

[61] Mark McGill, Stephen Brewster, Daniel Pires De Sa Medeiros, Sidney Bovet, Mario Gutierrez, and Aidan Kehoe. 2022. Creating and Augmenting Keyboards for Extended Reality with the K eyboard A ugmentation T oolkit. *ACM Transactions on Computer-Human Interaction* 29, 2 (2022), 1–39.

[62] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2017. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of the IEEE International Conference on Computer Vision.* 1154–1163.

[63] Anh Nguyen and Amy Banic. 2015. *3DTouch: A wearable 3D input device for 3D applications.* IEEE.

[64] Jie Ni, Tim K Marks, Oncel Tuzel, and Fatih Porikli. 2014. Detecting 3D geometric boundaries of indoor scenes under varying lighting. In *IEEE Winter Conference on Applications of Computer Vision.* IEEE, 1–8.

[65] Clark F Olson and Daniel P Huttenlocher. 1997. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on image processing* 6, 1 (1997), 103–113.

[66] Claudio Pacchierotti, Gionata Salvietti, Irfan Hussain, Leonardo Meli, and Domenico Prattichizzo. 2016. The hRing: A wearable haptic device to avoid occlusions in hand tracking. In *2016 IEEE Haptics Symposium (HAPTICS).* IEEE, 134–139.

[67] Hyungjun Park and Joo-Haeng Lee. 2007. B-spline curve fitting based on adaptive curve refinement using dominant points. *Computer-Aided Design* 39, 6 (2007), 439–451.

[68] Jaeyoung Park, Andrew J Doxon, William R Provancher, David E Johnson, and Hong Z Tan. 2012. Haptic edge sharpness perception with a contact location display. *IEEE Transactions on Haptics* 5, 4 (2012), 323–331.

[69] Myrthe A Plaisier, Wouter M Bergmann Tiest, and Astrid ML Kappers. 2009. Salient features in 3-D haptic shape perception. *Attention, Perception, & Psychophysics* 71, 2 (2009), 421–430.

[70] Henning Pohl and Michael Rohs. 2014. Around-device devices: my coffee mug is a volume dial. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. 81–90.

[71] Jon Postel. 1981. *Transmission control protocol.* Technical Report.

[72] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[73] Paul L Rosin. 1999. Measuring corner properties. *Computer Vision and Image Understanding* 73, 2 (1999), 291–307.

[74] Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, and Michael Beetz. 2007. Towards 3D object maps for autonomous household robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3191–3198.

[75] T Saito, M Shinya, and T Takahashi. 1989. Highlighting rounded edges. In *New Advances in Computer Graphics*. Springer, 613–629.

[76] Jun Shen and Serge Castan. 1992. An optimal linear operator for step edge detection. *CVGIP: Graphical models and image processing* 54, 2 (1992), 112–133.

[77] Yilei Shi, Haimo Zhang, Jiashuo Cao, and Suranga Nanayakkara. 2020. VersaTouch: A Versatile Plug-and-Play System that Enables Touch Interactions on Everyday Passive Surfaces. In *Proceedings of the Augmented Humans International Conference*. 1–12.

[78] Jinyun Sun, Yong Jiang, Jing Jiang, and Xiaobao Bai. 2016. Triangular mesh construction based on point cloud matrix and edge feature extraction. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*. IEEE, 275–279.

[79] Abdel Aziz Taha and Allan Hanbury. 2015. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC medical imaging* 15, 1 (2015), 1–28.

[80] Bernard J Van der Horst and Astrid ML Kappers. 2008. Using curvature information in haptic shape perception of 3D objects. *Experimental brain research* 190, 3 (2008), 361–367.

[81] Tianyi Wang, Ke Huo, Pratik Chawla, Guiming Chen, Siddharth Banerjee, and Karthik Ramani. 2018. Plain2Fun: Augmenting Ordinary Objects with Interactive Functions by Auto-Fabricating Surface Painted Circuits.. In *Conference on Designing Interactive Systems*. 1095–1106.

[82] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 552–567.

[83] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPturAR: An augmented reality tool for authoring human-involved context-aware applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 328–341.

[84] Wenping Wang, Helmut Pottmann, and Yang Liu. 2006. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (ToG)* 25, 2 (2006), 214–238.

[85] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. 2020. Pie-net: Parametric inference of point cloud edges. *Advances in neural information processing systems* 33 (2020), 20167–20178.

[86] Bowen Wen and Kostas Bekris. 2021. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8067–8074.

[87] Anusha Withana, Roshan Peiris, Nipuna Samarasekara, and Suranga Nanayakkara. 2015. zSense: Enabling shallow depth gesture recognition for greater input expressivity on smart wearables. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3661–3670.

[88] Jacob O Wobbrock, Brad A Myers, Htet Htet Aung, and Edmund F LoPresti. 2003. Text entry from power wheelchairs: EdgeWrite for joysticks and touchpads. In *Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*. 110–117.

[89] Robert Xiao, Chris Harrison, and Scott E Hudson. 2013. WorldKit: rapid and easy creation of ad-hoc interactive applications on everyday surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 879–888.

[90] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D Wilson, and Hrvoje Benko. 2018. MRTouch: Adding touch input to head-mounted mixed reality. *IEEE transactions on visualization and computer graphics* 24, 4 (2018), 1653–1660.

[91] Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2012. Magic finger: always-available input through finger instrumentation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 147–156.

[92] Hui Ye and Hongbo Fu. 2022. ProGesAR: Mobile AR Prototyping for Proxemic and Gestural Interactions with Real-world IoT Enhanced Spaces. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[93] Hui Ye, Kin Chung Kwan, Wanchao Su, and Hongbo Fu. 2020. ARAnimator: in-situ character animation in mobile AR with user-defined motion gestures. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 83–1.

[94] HyeonBeom Yi, Jiwoo Hong, Hwan Kim, and Woohun Lee. 2019. Dexcontroller: Designing a vr controller with grasp-recognition for enriching natural game experience. In *25th ACM Symposium on Virtual Reality Software and Technology*. 1–11.

[95] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*. 386–402.

[96] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2790–2799.

[97] Ya-Ting Yue, Yong-Liang Yang, Gang Ren, and Wenping Wang. 2017. SceneCtrl: Mixed reality enhancement via efficient scene editing. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 427–436.

[98] Lei Zhang and Steve Oney. 2020. Flowmatic: An immersive authoring tool for creating interactive scenes in virtual reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 342–353.

[99] 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.

[100] 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.

[101] Zhengyou Zhang. 2012. Microsoft kinect sensor and its effect. *IEEE multimedia* 19, 2 (2012), 4–10.

[102] Clement Zheng, Peter Gyory, and Ellen Yi-Luen Do. 2020. Tangible interfaces with printed paper markers. In *Proceedings of the 2020 ACM designing interactive systems conference*. 909–923.

[103] Wenni Zheng, Pengbo Bo, Yang Liu, and Wenping Wang. 2012. Fast B-spline curve fitting by L-BFGS. *Computer Aided Geometric Design* 29, 7 (2012), 448–462.

[104] Leisheng Zhong, Yu Zhang, Hao Zhao, An Chang, Wenhao Xiang, Shunli Zhang, and Li Zhang. 2020. Seeing through the occluders: Robust monocular 6-DOF object pose tracking via model-guided video object segmentation. *IEEE Robotics and Automation Letters* 5, 4 (2020), 5159–5166.

[105] Leisheng Zhong, Xiaolin Zhao, Yu Zhang, Shunli Zhang, and Li Zhang. 2020. Occlusion-aware region-based 3D pose tracking of objects with temporally consistent polar-based local partitioning. *IEEE Transactions on Image Processing* 29 (2020), 5065–5078.

[106] Qian Zhou, Sarah Sykes, Sidney Fels, and Kenrick Kin. 2020. Gripmarks: Using Hand Grips to Transform In-Hand Objects into Mixed Reality Input. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

[107] Guillaume Zufferey, Patrick Jermann, Aurélien Lucchi, and Pierre Dillenbourg. 2009. TinkerSheets: using paper forms to control and visualize tangible simulations. In *Proceedings of the 3rd international Conference on Tangible and Embedded interaction*. 377–384.