InstruMentAR: Auto-Generation of Augmented Reality Tutorials for Operating Digital Instruments Through Recording Embodied Demonstration

Ziyi Liu*
liu1362@purdue.edu
Purdue University

Zhengzhe Zhu*
zhu714@purdue.edu
Purdue University

Enze Jiang
jiang708@purdue.edu
Purdue University

Feichi Huang
huan1540@purdue.edu
Purdue University

Ana Villanueva
villana@purdue.edu
Purdue University

Tianyi Wang
wang3259@purdue.edu
Purdue University

Xun Qian
qian85@purdue.edu
Purdue University

Karthik Ramani
ramani@purdue.edu
Purdue University

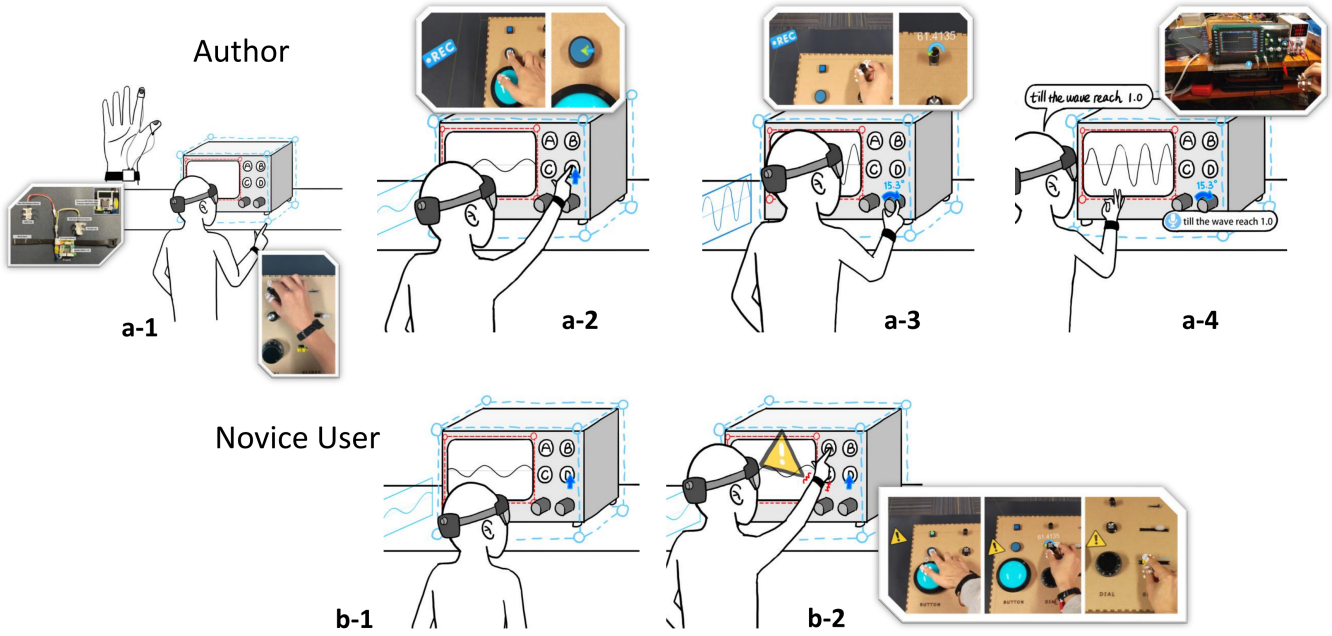

Figure 1: InstruMentAR can automatically generate AR tutorials by directly detecting and recording the user's operations on the physical UI elements. (a) The authoring workflow: (a-1) The author with the force sensing hand wearable specifies the interaction area and image capture area. (a-2) The author presses a button on the instrument (discrete operation). (a-3) The author turns a knob on the instrument (continuous operation). (a-4) The author records the voice instruction, while a screenshot is captured automatically. (b-1) The novice user follows the tutorial, which will advance automatically once the user has completed the correct operation. (b-2) The novice user is warned by preemptive feedback which can prevent the user from making mistakes.

ABSTRACT

Augmented Reality tutorials, which provide necessary context by directly superimposing visual guidance on the physical referent,

*Both authors contributed equally to this research.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581442

represent an effective way of scaffolding complex instrument operations. However, current AR tutorial authoring processes are not seamless as they require users to continuously alternate between operating instruments and interacting with virtual elements. We present InstruMentAR, a system that automatically generates AR tutorials through recording user demonstrations. We design a multimodal approach that fuses gestural information and hand-worn pressure sensor data to detect and register the user's step-by-step manipulations on the control panel. With this information, the system autonomously generates virtual cues with designated scales to respective locations for each step. Voice recognition and background capture are employed to automate the creation of text and images as AR content. For novice users receiving the authored AR tutorials, we facilitate immediate feedback through haptic modules. We compared InstruMentAR with traditional systems in the user study.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; *Interactive systems and tools*; User interface toolkits.

KEYWORDS

Augmented Reality, Embodied Demonstration, Immersive Authoring, Tangible Interaction, wearable Devices, Haptic Feedback

ACM Reference Format:

Ziyi Liu, Zhengzhe Zhu, Enze Jiang, Feichi Huang, Ana Villanueva, Tianyi Wang, Xun Qian, and Karthik Ramani. 2023. InstruMentAR: Auto-Generation of Augmented Reality Tutorials for Operating Digital Instruments Through Recording Embodied Demonstration. 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, 17 pages. https://doi.org/10.1145/3544548.3581442

1 INTRODUCTION

Digital instruments, such as home appliances, office and laboratory equipment, and recreational devices, are now interwoven into the fabric of our society [51, 71]. Most of these instruments feature a control panel populated with physical UI elements (e.g., buttons, knobs), which serve as the gateway for the user's operations [34, 35, 71]. Traditionally, image or video tutorials are created to guide users through an operation [25, 63]. Recently, Augmented Reality (AR) tutorials have emerged as a preferable alternative to traditional image and video tutorials, where visual guidance is displayed directly on the associated object and thus is always within the user's line of sight [42, 78]. This reduces the user's cognitive load by removing the need to switch the context and attention between the instrument and external information [51, 78, 84]. Different types of tasks require different formats of AR content [22]. In this paper, we focus on *arrows* with explicit scale and direction. In addition, we attached the *arrows* with supplementary text and images. All these forms of visual guidance have been shown to be effective in conveying instructions for physical user interfaces on control panels [3, 22, 46, 59].

The diversity of digital instruments and their associated operations have motivated researchers to empower end-users to author sequential AR tutorials on demand and in-situ [51]. To this end, they adopted the immersive programming paradigm [17, 54, 95, 104] to replace traditional 2D programming, in which users simultaneously operate the instrument and create AR guidance by direct manipulation with virtual objects (e.g., menus, widgets, toolset) [3, 27, 65]. For instance, a user drags an arrow from the virtual library and rotates it to point down at a button. The challenge of this authoring process is that for each step of the tutorial, the placement and scale of the AR visualizations must be properly adjusted by the users. Otherwise, the misplaced (e.g., an arrow pointing to the wrong button) and improperly sized (e.g., an arrow not being long enough to indicate where the slider should be pushed to) visualizations would undermine the effectiveness of the tutorials. This extra workload may cause strain on users who have to switch their attention back-and-forth between physical instrument operations and virtual object interactions [51, 97]. This issue is exacerbated when an operational task includes multiple steps (e.g., using an oscilloscope to measure current).

To address this issue, we seek to eliminate the process of manual generation of virtual guidance by recording each step of the user's demonstrated operations. As a user operates on the instrument, the system detects and records their manipulations with the UI element (e.g., button, knob) and automatically translates them into specific AR visualizations. For example, if a user presses a button, the system will render a straight arrow pointing downward to the button's location. Likewise, if a user rotates a knob, the system will generate a circular arrow with its starting and ending point matching the user's operation. This automated process is often referred to as authoring by embodied demonstration [27, 97]. It relieves users from continuous interaction with virtual objects, which could be both time-consuming and mentally demanding [27, 58, 97, 103].

The key to achieving authoring by demonstration is finding the proper technique to reliably track the manipulated objects to transfer them into the virtual world. To this end, many works which focus on assembly utilized overhead cameras to track the movement of manipulated objects and then generated their trajectories accordingly as visual guidance [37, 41, 76, 97, 99]. However, physical UI elements on the control panel are much smaller compared to the objects involved in the assembly tasks, which makes the tracking more susceptible to hand occlusion during manipulation [52, 93]. To address this issue, we propose to track the user's hand trajectories instead of the manipulated objects. Our approach can be summarized into two steps: 1) detecting *when* does the user's hand make contact with and manipulate the UI elements, and 2) recognizing *how* does the user's hand manipulate them. For the first step, we employ pressure sensors that are attached to the user's fingertips, which allow us to differentiate subtle differences in handling operations. For example, by comparing sensor readings, we can tell whether users are actually pressing a button or merely placing their fingers on top of it. For the second step, we resort to gesture tracking considering its maturity and wide adoption in modern AR devices. Based on the starting and ending point of manipulation detected in step one, we extract the gestural information in between. For instance, the position of the index finger informs the system where the button is pressed, so that the virtual arrow can be placed accordingly.

We present InstruMentAR, a system that can automatically generate AR tutorials by recording the author's embodied demonstrations.

The system includes two parts: a custom-designed hand wearable embedded with pressure sensors and haptic modules, and an authoring interface built on a pair of optical see-through AR glasses with hand tracking capability (i.e., Hololens 2 [5]). In the initial setup, the author defines the interaction area and the area for image capture (Figure 1 a-1). Then the author can proceed to operate the instrument step-by-step. In the backend, the system records each operation (e.g., press a button, or turn a knob) and generates AR visualizations accordingly. After each step of the operation, the system automatically captures an image of the previously specified area (Figure 1 a-4). In the meantime, the author can also record explanations that would later be transcribed to the text displayed in AR (Figure 1 a-4). During the access mode where the AR tutorial is displayed to a novice user, the system will automatically proceed to the next step once it detects that the user has performed the right operation (Figure 1 b-1). On the other hand, if the system detects the novice user is performing or about to perform an incorrect operation, the system will immediately issue a visual-haptic warning (Figure 1 b-2).

Following is a list of our contributions:

- An automated authoring workflow for end-users to create sequential AR tutorials for digital instruments by intuitive embodied demonstration.
- A multimodal approach that combines finger pressure and gesture tracking to translate the author's operations into AR visualizations.
- An automatic feedback mechanism that can determine whether the system should proceed to the next step or issue visual-haptic warning based on the novice user's real-time operation.
- A user study that measures the efficiency and usability of InstruMentAR by comparing it with a non-automated immersive authoring system.

2 RELATED WORK

2.1 AR Tutorials

Traditionally, image- and video-based tutorials were primarily employed to guide users through complex user interfaces and complicated tasks [25, 53, 63, 89, 90]. With the advent of Augmented Reality (AR) technology, we now have the opportunity to embed tutorials into the user's physical interaction space [21, 27, 42, 45, 69, 73, 92]. AR superimposes visual guidance directly on the associated object, which ensures that guidance and objects are concurrently within the user's lines of sight. As a result, it provides sufficient contextual information which helps users easily establish mental connections between the guidance and its referred component [22, 58, 78]. Prior studies have shown that users who receive AR tutorials tend to make fewer errors compared to those who receive tutorials from an external display (e.g., monitor, paper) [20, 81]. AR tutorials can be classified as either synchronous or asynchronous [83]. In the first category, an instructor must be present to create and deliver tutorials to novice users in real-time [59, 61, 64, 85]. For this paper, we focus on asynchronous AR tutorials [22, 83] that are pre-recorded by instructors, in which novice users can follow these tutorials at their own pace and at any time.

Different types of tasks demand different forms of visual guidance in the AR scene [22]. For assembly tasks, 3D animations of objects' virtual models are displayed to represent the assembly motion [27, 48, 101]. For tasks that require body coordination (e.g., dancing [16, 26], medical education [29], operating an industrial machine [46], and playing a musical instrument [87]), body avatars are employed in the tutorials to represent human motion. In this paper, we focus on operating physical UI elements on the instrument's control panel. It is impractical to create virtual models for all physical UI elements due to their diversity [51]. On the other hand, the display of a body avatar is not justified by the lack of body coordination required for control panel operations (e.g., rotating knobs, pressing buttons) [22]. In addition, due to the avatar's larger size relative to the control panel, there will inevitably be obstructions that could lead to confusion and distraction [22]. Based on these analyses, we adopt virtual arrows with explicit scale and direction, as well as supplementary text and images as visual guidance. They have been shown to effectively convey instructions for physical user interfaces [3, 22, 46, 59].

2.2 End-User Authoring of AR Tutorials

Empowering end-users to effortlessly author AR applications has long been the focus of HCI researchers [17, 56, 60, 70, 79]. For AR tutorial authoring, conventional AR authoring tools (e.g., Unity [12], Unreal [13], ARCore [1], ARkit [2]) decouple the programming and operation environment. Developers often struggle to project the status of the instrument after each operation while programming [27, 65]. In addition, the transition from the design on a screen to life-sized visualization in AR often comes with unexpected issues (e.g., visualizations that are not properly aligned) [67]. Recently, immersive programming has been adopted to blend the authoring process into the physical interaction space [54]. Authors can now refer to the status of instruments for context information while authoring. In addition, the WYSIWYG (What You See Is What You Get) metaphor [18] enables users to create virtual guidance (e.g., arrows, circles) through direct manipulation of virtual content. Nevertheless, developers still undertake the workload of manually selecting, as well as adjusting the position and size, and orientation of this virtual guidance. For every step, the developers have to switch their attention back-and-forth between operating on the instrument and interacting with virtual content, which could be mentally demanding and time-consuming [27, 58, 97], especially for those who are not proficient with virtual interactions [103].

To address this issue, we seek to eliminate the process of manual generation of virtual guidance by recording each step of the user's demonstrated operations. For example, TutoriVR [86] and Sketch-Sketch Revolution [30] record expert users' sketches in real-time and use them as sources to generate sketching tutorials. In particular, we are inspired by prior works about authoring AR tutorials for assembly tasks, which used overhead cameras [37, 41, 76, 97, 99] to record the movement of the assembled objects. Instrument operations are different from assembly tasks in that the user's hands occlude the majority of UI elements during the manipulation process. The resulting occlusion makes external vision-based tracking of physical UI elements unfeasible [52, 93]. To bypass the occlusion issue, Kong et al. [51] recorded users' finger trajectories instead of

the UI elements during instrument operations. In the access mode, each finger is represented by a colored ring to demonstrate to novice users how to move their fingers. The drawback of this approach is that the novice user must indirectly infer the intended movement of UI elements from overlaid 3D finger movement, which may lead to ambiguity and confusion. For instance, when viewing the replay, it can be challenging for novice users to tell whether a finger is simply touching a button or actually pressing it. Thus, we are highly motivated to explore ways to record the effect of the operation on the UI elements (e.g., where the slider is pushed to, the knob's rotated angle), so the system could generate *explicit* symbolic visual guidance (i.e., arrows) that matches those operations.

2.3 Tracking User Interactions with Physical Objects

Tracking users' interactions with physical objects could be either through tracking the status of the objects or the action of the user. For the first category, both vision-based [37, 76, 97] and hardware-based [62, 93] techniques have been proposed. However, both these approaches are not applicable to our scenario due to constant hand occlusions and difficulty in reconfiguring every instrument with extra hardware.

In the second category, some works have designed hand wearables that can accurately detect when and how a user is making contact with the tabletop [33, 49]. Similarly, some hand wearables have been designed to detect direct contact with different parts of the human body (e.g., face [24], palm [49], arm [39]). For physical objects other than a surface (e.g., food), Zhang et al. [102] designed a fork with embedded sensors that can detect and record the user's food consumption. Recently, with the maturity of hand tracking algorithms [95] and their wide adoption across mainstream AR devices (e.g., headsets [5, 8], phones [68]), gesture tracking has been utilized to detect physical interactions. However, gestural information alone is inadequate for this task because it can not pinpoint the beginning and ending points of the physical interaction. To address this issue, Wu et al. [100] combined gestural information with depth information from the Kinect camera [7] to detect the user's interactions with the kraft paper on the tabletop.

Inspired by the prior work, we designed a multimodal tracking approach that: 1) detects when an operation starts and ends through pressure sensor readings, and 2) detects the location and scale of the operation based on gesture information. The pressure data allows the system to differentiate subtle differences in operations (e.g., touch versus press) while the gesture information is used to derive the position, orientation, and scale of the visual guidance (i.e., arrows). Compared to conventional approaches that track manipulated objects, our multimodal tracking approach is more resilient to occlusions caused by hands as it tracks hands directly. Meanwhile, our approach does not require the pre-modeling or pre-training of the objects which significantly streamlines the setup process.

2.4 Feedback in AR Tutorials

Effective and personalized feedback is important for learning due to the different characteristics of users, as shown in the worlds by Gutierrez and Atkinson [38] and Bimba et al.[19]. Due to the absence of a human instructor during an asynchronous AR tutorial, it is crucial for the system itself to provide feedback to novice users [83]. Recently some works have started to integrate different forms of feedback (e.g., error detection, suggesting hints, task selection) into asynchronous AR tutorials [43, 46].

Meanwhile, haptic and visual feedback are two major feedback modalities [15, 47, 72, 75]. Prior works have sought to seamlessly combine these two to deliver realistic mixed reality experiences [77, 80, 91]. These two types of feedback are complementary to each other as visual feedback can convey richer information while haptic feedback is more natural and direct [32, 75]. Sigrist et al. [74] achieved enhancement in complex motor task learning through leveraging both feedback modalities. Inspired by these works, our system is designed to provide haptic feedback through a vibration module integrated into the wearable, as well as the visual feedback displayed in AR.

In InstruMentAR, the feedback is two-fold. On one hand, the AR tutorial will automatically proceed to the next step once the system detects the novice user has performed the correct operation. Pause-and-play [66] designed a similar mechanism that syncs the video playback with the application progress. On the other hand, the system will issue a warning once it detects a potential error from the novice user. Contrary to the conventional warning feedback which is only invoked after the error is made [43, 46, 96], we design a preemptive feedback mechanism in which the novice users are given a warning when they are performing or about to perform an incorrect operation. In other words, the system exploits half-touch interactions to trigger feedback. It is related to pre-touch interactions explored by prior works in the context of mobile phones [23, 44], as well as tabletop surfaces [88]. In InstruMentAR, as soon as the users place their fingers on the wrong button, the warning is immediately activated. The preemptive feedback is especially helpful in certain instrument operations where one incorrect action (e.g., pressing the wrong button) would result in a complete redo.

3 SYSTEM DESIGN

3.1 Implementation

We build our system on Hololens 2 [5] using Unity3D (2020.3.0.f1) [12]. The InstruMentAR user interface is implemented with support from the Microsoft Mixed Reality Toolkit (MRTK) [10]. The image cropping and perspective change functionality is implemented with OpenCV [11]. The voice-to-text functionality is implemented with Hololens built-in library.

3.2 Supported UI Elements and Operations

InstruMentAR supports 5 types of UI elements: touchscreen, switch, button, knob, and slider, which can cover the majority of instrument operations. Specifically, the touchscreen UI elements we focus on are digital buttons and digital sliders whose operation logic resembles their physical counterparts. They are prevalent in instruments that feature touchscreens (e.g., printers, and washing machines) and cover most of their touchscreen operations. More sophisticated touchscreen operations (e.g., multi-touch), which have less presence on these instruments, are not included. How these operations could be supported in the future is further discussed in section 6.4.

As a way to determine how a user would normally interact with these UI elements [98], we conducted a survey on 50 individuals

with prior experience with digital instruments to determine their preferred gesture for each operation (Figure 2). The survey subjects are college students who have participated in labs involving the operation of digital instruments with physical user interfaces (e.g., oscilloscope, CNC machine). All of them are also familiar with operating one or more digital instruments in their daily life (e.g., coffee machines, washing machines). For each UI element, survey subjects were presented with seven possible gestures and were asked to select up to three of their preferred gestures for each operation. The seven default gestures are generated by us based on our experience. The survey result is shown in Figure 2. The gestures circled in yellow are those that receive at least 30% of the total vote. The coverage rate indicates the proportion of the vote that these yellow-circled gestures represent in combine.

We incorporate all the commonly occurred results circled in yellow into the system (Figure 3 a), except for gestures for switch operation. We adopt the third most popular gesture in switch operation (i.e., the one with 30% of the vote). This pinch gesture which requires two fingers allows the system to determine the orientation of the generated virtual arrow (further explained in section 3.5.3).

As a first-generation prototype, InstruMentAR expects its users to perform respective operations using the gestures included in this taxonomy. We acknowledge this taxonomy is far from comprehensive, and it is likely that some users will be required to perform demonstrations using unfamiliar gestures. This limitation and its solutionwould be further discussed in section 6.4. We utilized the gesture tracking capability of HoloLens, which returns the real-time locations of every hand joint (Figure 3 b). All of the gestures for the operations require only the index finger, the thumb, or both. Meanwhile, the index finger and thumb are the two fingers that are least likely to be occluded by other fingers. Therefore, we conclude that extracting information from three joints (i.e., *IndexTip*, *IndexDistal*, *ThumTip*) on these two fingers is sufficient for tracking hand trajectories and generating corresponding visualizations.

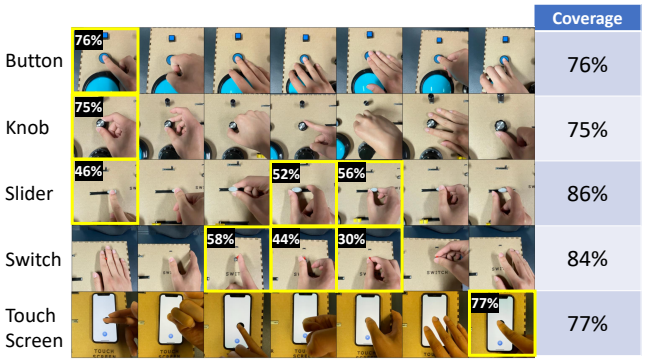

Figure 2: Questionnaire used for the elicitation survey and the result.

3.3 Hardware Design

InstruMentAR comes with a custom-designed hand wearable for detecting *when* the operation takes place and providing haptic feedback for an incorrect operation. The wearable has a minimalist

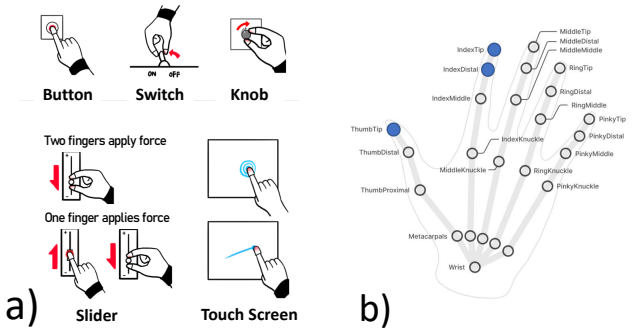

Figure 3: (a) Taxonomy for gestures associated with each operation (b) Tracked joints in Hololens

design with only three components — a wristband and two finger caps (Figure 4). Therefore, It hardly alters the appearance of the finger, which means it is less likely to interfere with the hand-tracking algorithm. Additionally, the reduced bulkiness would contribute to a more smooth operation. The finger caps are installed on the tip of the user's index finger and thumb, as they are the only two fingers involved in the operation. We adopt thin film pressure sensors that can gather pressure strength applied on the fingers. The wristband holds a microprocessor, a linear actuator haptic module, and a battery. To optimize hand tracking, these components are all hidden beneath the user's wrist when the wearable is put on (Figure 4). The microprocessor has WIFI capability which allows the wearable to communicate with the AR headset (i.e., Hololens 2).

The profile (i.e., curvature) of the finger cap is designed to conform to the profile of the fingertip. We add conductive tapes on the fingertips to support touchscreen operations. Two finger bands that can be tightened by rubber bands hold the cap in its place. For each finger, three sizes (small, medium, large) of finger caps are provided. The pressure sensor is secured by a slot and a set of buckles.

With no intention to make the finger cap excessively cumbersome by attaching multiple pressure sensors, we only include one pressure sensor to the fingerpad which provides an effective sensing area of 9mm. However, this pressure sensor alone can only detect the press gesture while the poke gesture remains undetected. To address this issue, we design a force-transfer pad that can redirect the force applied elsewhere to the pressure sensor. With this design, the users can perform either poke or press to interact with the physical UI elements, which would suffice for the UI interactions we enlisted.

3.4 Operation Threshold

The relationship between the pressure sensor's analog reading and the pressure it receives is depicted in (Figure 6 a). To detect the physical interaction between the finger and the UI, we set 350 g as the minimal force change (*DF* in Figure 6 a), which implies that the user has to exert at least that amount of force onto the UI element before they could actually move the UI element. This value is determined after experimenting with the operation on various physical

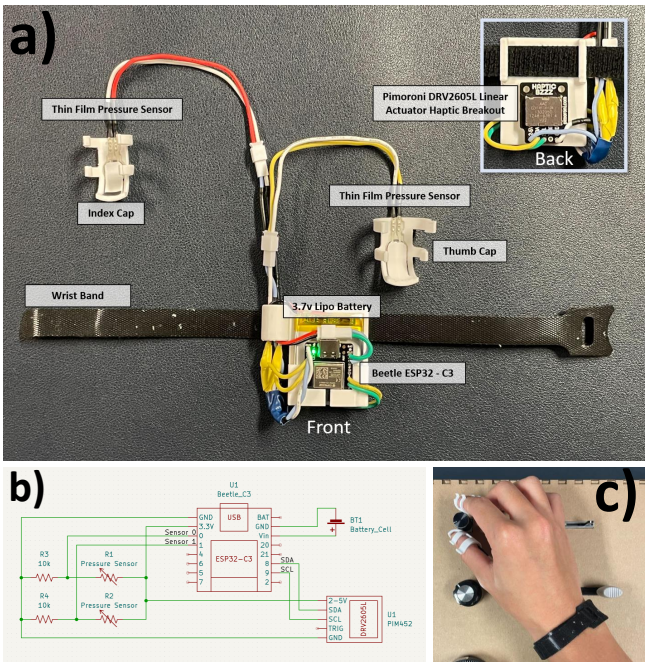

Figure 4: (a) The layout of the hand wearable. (b) The circuit diagram. (c) The user wears the hand wearable that is designed to minimize the occlusion for hand tracking.

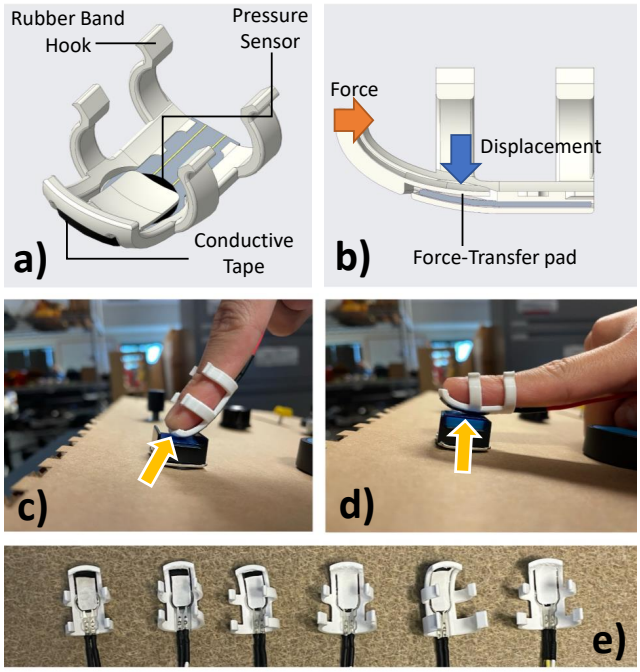

Figure 5: (a) CAD of the finger cap. (b c d) The force transfer pad allows the user to apply force through multiple angles. (e) The finger caps come in different sizes.

UI elements on common instruments. Currently, we adopt the minimum pressure required for operation as the universal threshold value.

Initial pressure applied to the pressure sensor (*baseline value* in Figure 6 a) varies between users and fluctuates over time, due to different wearing habits and the current pose of the finger. To derive the linear relationship between the analog readings of *Threshold Value* and *Baseline Value*, we collected 8 sets of data to perform data fitting. During the data collection, we applied the same force to the UI element (i.e., button) and measured the initial analog reading of the pressure sensor (*Baseline Value* as B) and the analog readings of the pressure sensor when the minimal force is exerted (*Threshold Value* as T). Based on the derived equation $T = -0.3923B + 1417.2$, we can dynamically calculate the (analog readings) of *operation threshold* from the baseline value, which is updated every time the user's hand enters the interaction area near the instrument. Once the pressure sensor's analog reading passes *operation threshold*, the system determines the operation has started. Likewise, if the reading decreases lower than the *operation threshold*, the system determines the operation has ended.

To enrich the functionality of the pressure sensor, we introduced another concept named half-press. The analog reading for the half-press takes half the value of the *operation threshold* so that it can detect operations that are about to happen. This value will be later utilized for the preemptive warning feature.

3.5 Author Mode

3.5.1 Setup Procedure. In the initial setup process, the user is required to draw a bounding box to specify the interaction area (Figure 7), which serves two purposes. Firstly, the system will only receive the pressure signal during the time when the hand is within the interaction area. For instance, when the user taps the table or rubs finger outside of the area, the system disregards the pressure signal. It ensures that only operations associated with the instrument will be registered. Secondly, every time the hand enters the interaction area, the *baseline value* for calculating the *press threshold* is reset to the current pressure sensor reading. Additionally, the user must specify the area for automatic image capture, which is typically where the instrument's screen is located(Figure 7). The rationale and purpose for this process will be further explained in the later section.

3.5.2 Operation Detection. As mentioned in the last section, the wearable can help the system determine *when* the instrument is operated. Then the system needs to determine *how* the instrument is operated. After an operation is completed, the system records the gestural information and the pressure sensor input associated with this operation. We develop a decision-tree-based algorithm (Figure 8) to classify UI operations based on this information. At the first level, the decision tree analyzes whether one or both pressure readings exceed the operation threshold. The *Gesture Pose Filter* on the second level determines whether the user performs a touch or a pinch gesture by analyzing the hand joint data. Meanwhile, the *Gesture Transformation Filter* on both the second and the third level

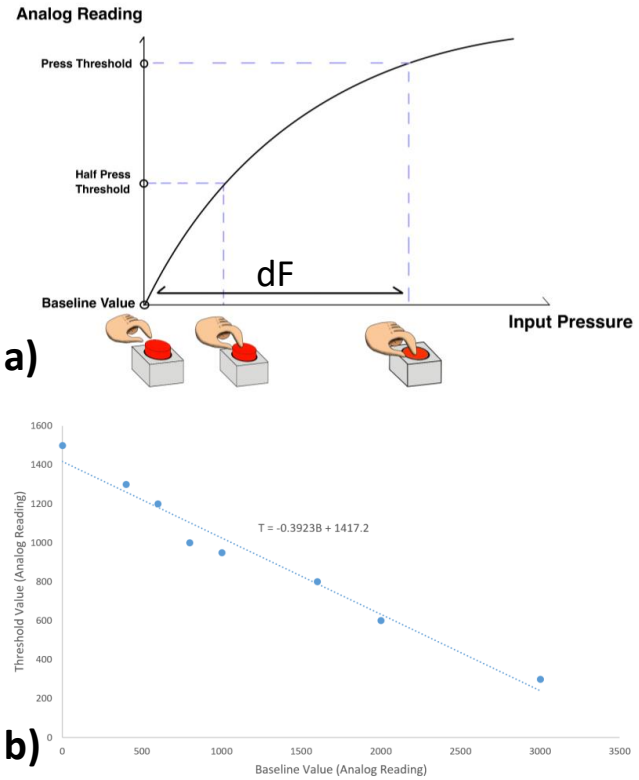

Figure 6: (a) Relationship between the pressure sensor's analog reading and the pressure it receives. (b) Relationship between the initial analog readings of the pressure sensor and the analog readings of the pressure sensor after the minimal force is applied.

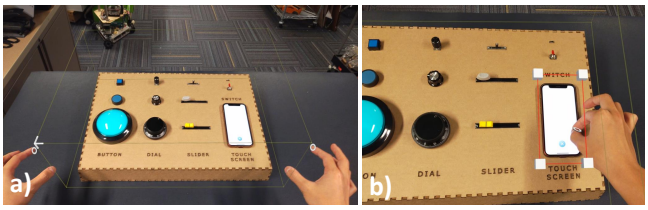

Figure 7: (a) Draw a bounding box around the interaction area. (b) Draw a red rectangle around the screen area.

determines whether the user's hand traverses a distance or rotates in place. The system determines whether the operation is on the touchscreen by determining whether it falls within the user-defined screen boundary.

3.5.3 Auto-Generation and Post-Editing of the Arrows. After the operation is detected, the system creates virtual arrows with different scales and orientations on top of the associated UI elements (Figure 11 a). For discrete operations (i.e., press a button/a touchscreen, turn a switch), the system retrieves the location where the operation takes place as well as the hand pose during operation. Based on

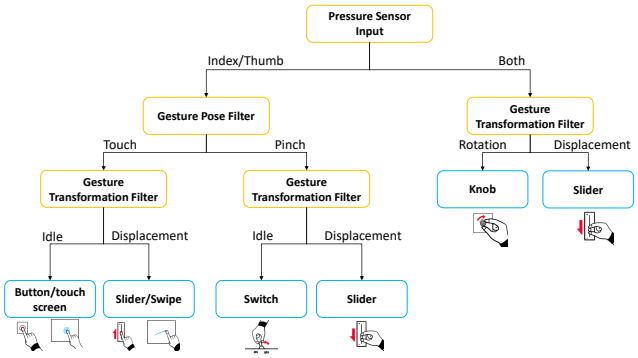

Figure 8: Decision Tree Algorithm.

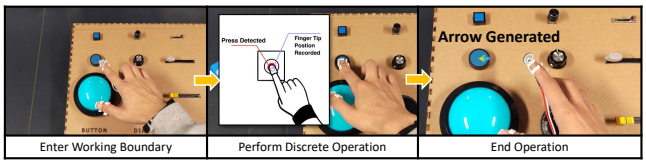

Figure 9: Workflow for discrete UI recording. (Left) The user's hand enters the working boundary and the pressure sensor baselines are reset, (Middle) the press is detected and the fingertip position is recorded, (Right) the user ends the operation, and the UI element is classified as a button.

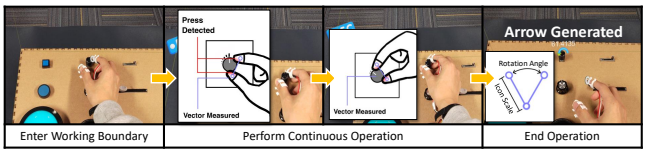

Figure 10: Workflow for continuous UI recording. (Left) The user's hand enters the working boundary and pressure sensor baselines are reset, (Middle) the press is detected and the vector between thumb and index tip is recorded, (Right) the user ends the operation, and the UI is classified as a knob and the rotation angle is recorded.

the information, the system generates an arrow that appears at the exact location where the finger is detected when starting the operation. The system also sets the orientation of the arrow based on the hand pose when starting the operation. For the button operation, the arrow points in the same direction as the index finger. For the switch operation, the arrow points from the finger which exerts force to the finger which does not.

While the scale of the arrows is the same for discrete operations, this is not the case for continuous operations (i.e., turn a knob, swipe a screen, or push a slider). At both the beginning and the end of the knob operation, the system retrieves the thumb-to-index-finger vectors and generates the arrows accordingly. Through vector subtraction, the system calculates the rotated angle of the knob which

determines the length of the arrow. Meanwhile, the distance between the index finger and thumb, which is related to the knob's size, determines the radius of the arrow. For the slider and swipe operation, the arrow's length and direction should match the index finger's trajectory throughout the operation.

In some cases, the auto-generated arrows may not live up to the user's standard due to detection inaccuracies. Therefore, the system allows users to post-edit the auto-generated arrows by adjusting their locations, sizes, and orientations. The adjustment process is performed through direct manipulation of the virtual arrows (Figure 11 b), which is similar to the traditional immersive programming approaches mentioned in the related work [17, 54].

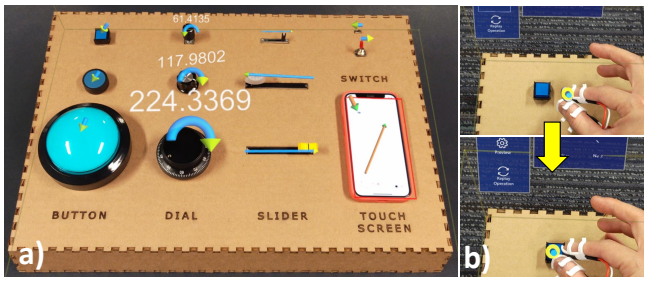

Figure 11: (a) The arrows for different UI operations that are automatically generated with our system. (b) Post adjustment.

3.5.4 Supplemental Materials. Although virtual arrows can be easily interpreted, they may not convey sufficient instructions to guide users through complex tasks. Therefore we allow users to record their voices and capture the image of a specific area to convey additional information. After each operation, the user can perform a *pinch* gesture outside the interaction area to record detailed instructions. The recording is transcribed to text which will be displayed on the side of the instrument (Figure 12).

Unlike voice recording, which must be performed manually, the system automatically captures images after each step. During the setup, the user manually specifies the area for image capture by drawing a red rectangle around it. The red rectangle, which constantly appears during the authoring process, serves two purposes. First, it constantly reminds users of its location to prevent them from turning their heads away. Then the red boundary displayed in the captured image can facilitate the automatic image processing. Specifically, the system segments the contour of the red rectangle by analyzing the image's RGB value. The retrieved contour helps the system determine the coordinates of the rectangle's four vertices. Based on these coordinates and the rectangle's original aspect ratio, the system can automatically crop and correct the image's perspective (Figure 12 b) with the help of built-in functions from OpenCV [4]. In other words, the final image that will be displayed alongside the instrument is cropped to what is contained within the red rectangle and the perspective is altered to provide users with a clearer view of the required information. To avoid occlusion of the screen, the system would wait until the hands no longer appear between the AR headset and the screen area to capture the image. This is feasible because the system knows the location of the hand, the screen area, and the AR headset.

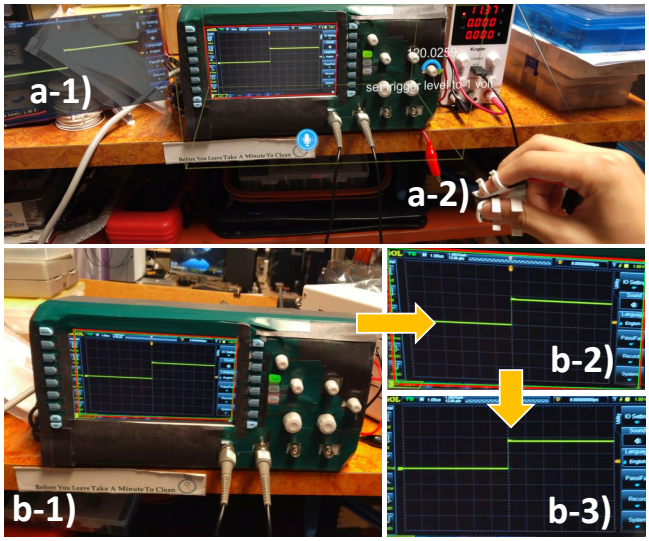

Figure 12: (a-1) The automatically captured screenshot, (a-2) The user performs the pinch gesture outside the working boundary to record voice, (b-1) the raw image from the Hololens, (b-2) cropped image based on red rectangle segmentation, (b-3) image after perspective transformation.

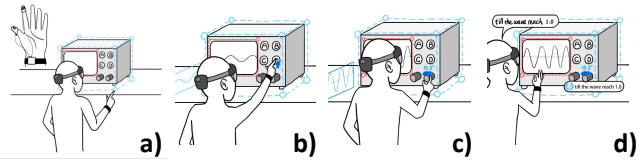

Figure 13: Overall workflow of the tutorial Authoring (a) The user specifies the interaction area and image capture area. (b) The user presses a button on the instrument (discrete operation). (c) The user turns a knob on the instrument (continuous operation). (d) The user records voice instructions.

3.6 Access Mode

3.6.1 Preemptive Feedback. We design a preemptive feedback mechanism so that the system will issue a warning prior to the occurrence of an error. The feedback includes a vibration from the haptic module as well as a warning sign shown in AR (Figure 14). This is especially helpful when users would have to start over from several steps back due to an incorrect operation, which is a frustrating experience. There are two classes of errors: locating the wrong UI elements (e.g., pressing the wrong button Figure 14 a) and operating in the wrong way (e.g., pushing the slider in the wrong direction Figure 14 b c). Based on the types of errors, the preemptive feedback is in three stage. Firstly, the system retrieves the finger's location

when it receives the *half press* signal indicating the finger is touching the UI element. If the position is not aligned with where the pre-authored operation takes place (the first class of error), a warning is immediately issued. Then, we predict the fingers' intended movement direction by comparing the pressure signals from *IndexTip* and *ThumTip* respectively, which is only feasible for slider operations. For knob and swipe operation, the warning is issued as soon as the user starts rotating/swipping in the wrong way. Only when the user removes his finger from the control panel or corrects the operation to the proper course (e.g., immediately turning the knob reversely), would the warning feedback disappear.

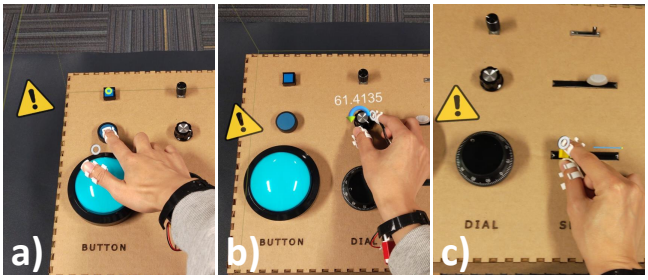

Figure 14: Preemptive warning when (a) pressing the wrong button (b) turning the knob in the wrong way (c) pushing the slider in the wrong way.

3.6.2 Error Handling and Automatic Playback. An error handling mechanism should be implemented in the event that the user performs the incorrect operation despite the warning. After each operation is performed, the system will detect the operation and compare it with the pre-authored operation to determine if an error has been made. Due to the offset during gesture tracking, we anticipate that continuous gestures (e.g. push a slider, turn a knob) tracked in real-time cannot precisely match the pre-recorded ones, even if the operation is performed correctly. Thus, we allow for a twenty percent margin of error when comparing the trajectories of continuous operations. The manual pausing and forwarding of a tutorial's playback are automated by our system. If an error has been detected, the AR tutorial will pause. Users may resume it manually if they believe that the incorrect operation does not have any negative consequences, or they may choose to restart the process if they believe otherwise. If no error is detected, the system will automatically advance to the subsequent step.

4 APPLICATION SCENARIO

The predominance of digital instruments is a hallmark of the modern world. The five types of physical UI elements can cover the majority of the common instrument operations. We present three application scenarios in which InstruMentAR's ability to generate AR tutorials automatically could be useful.

4.1 Generating Lab Manuals

Experiments conducted in laboratories frequently require the operation of one or more instruments, and previous research has demonstrated the need for instructors to create tutorials in a more

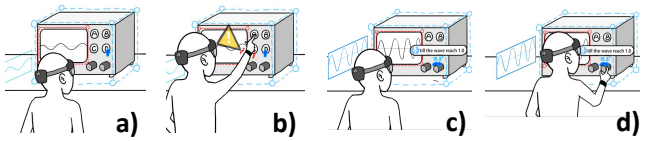

Figure 15: Overall workflow of the tutorial accessing (a) The user starts by importing the previously authored AR tutorials, which can be accomplished with a single click if the instrument remains stationary. If not, the user can utilize the fiducial marker to align the visualization. (b) The user places the hand on the wrong button and receives a preemptive warning. (c) The user presses the right button and the system automatically displays the instruction of the next step. (d) The user completes the operation.

effective and efficient manner [57]. In addition to the oscilloscope that is adopted in our user study, InstruMentAR could also be applied to a centrifuge from a biochemistry laboratory (Figure 16 a-1), or a laser cutter from a mechanical laboratory (Figure 16 a-2).

4.2 Workforce Training

Companies all over the world have been searching for more efficient means of training their employees. With InstruMentAR, companies can easily create AR training manuals to familiarize employees with the use of the instrument at work. For example, factories can train workers on how to operate new models of CNC machines (Figure 16 b-1), and airlines can train pilots on how to operate new models of aircraft (Figure 16 b-2).

4.3 Generating Operation Manuals for Everyday Instruments

The instruments used by common people on a daily basis could also benefit from easily generated AR tutorials. Examples of these instruments include coffee machines (Figure 16 c-1), printers (Figure 16 c-2), and music pads (Figure 16 c-3). These AR manuals can be generated not only by manufacturers but also by users themselves. While referring to the paper/video-based manuals, users perform operations on the instrument and the corresponding AR tutorials will be generated in the meantime. These AR tutorials are displayed directly on the instrument and thus always in a readily accessible manner. Therefore, users do not need to retrieve the tutorial from elsewhere whenever they forget about the operation, which is more convenient.

5 USER STUDY EVALUATION

We conducted a three-session user study to evaluate: a) the accuracy of our operation detection model, b) how InstruMentAR can support end-users to efficiently and effortlessly author AR tutorials, and c) what novice users feel about the automation features when receiving the tutorial.

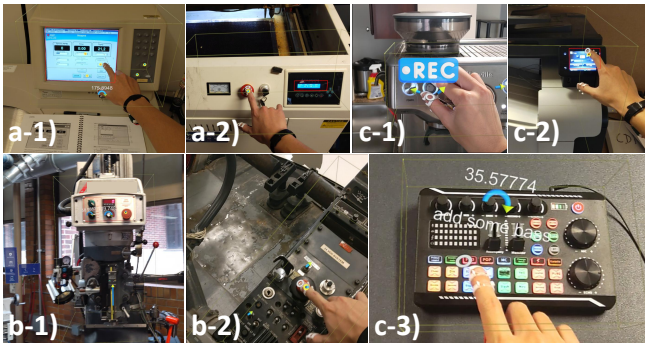

Figure 16: Possible application scenarios (a-1) Centrifuge (a-2) Laser cutter (b-1) CNC machine (b-2) Control panel of helicopter (c-1) Coffee machine (c-2) Printer (c-3) Music pad.

5.1 Session 1: Detection Model Evaluation

In this session, we aim to evaluate our operation detection model. We recruited 10 users for this user study. We built a mock-up interface populated with tested UI elements (Figure 17). Similar practices have been adopted by prior works for technical evaluation purposes [22, 46]. In the later sections, we evaluate the user experience based on more realistic tasks. The users needed to perform every supported operation associated with each UI element. For example, we tested two operations on the switch which use the thumb and the index finger to press the switch respectively. Each operation was required to be performed 30 times consecutively by each user. For the button, the knob, and the slider, the user performed 10 times on each of the three shapes of the UI element. An error was recorded if the spawned arrow did not meet our standard — position within the diameter of 2 centimeters, orientation and scale within 30% offset. We summarize the errors into 3 categories:

Error 1: The system cannot detect when the operation starts since the pressure sensor does not respond to finger contact. This is because users sometimes manipulate the UI element with less force than the threshold value. This type of error can be reduced by allowing users to define customized threshold values (further discussed in section 6.3).

Error 2: The system misses when the operation ends as the pressure sensor fails to detect the decrease in applied force upon the UI element. The tension between the finger and the finger cap gets larger as the cap may slip away from the original position, therefore preventing the pressure value from decreasing below the threshold. This type of error can be reduced by improving the mounting mechanism which prevents the cap from slipping.

Error 3: Without the occurrence of the first two types of errors, the system sometimes still spawns the arrow with incorrect transformation (position, orientation, scale). This is due to the malfunction of hand-tracking algorithms which is beyond our control.

In Figure 18, we report on the result of the study by classifying the appeared errors into these 3 categories. All the data passed the Kolmogorov–Smirnov normality test so that we can use the ANOVA test to evaluate the data.

Regarding the performance of button pressing, screen touching, and screen swiping with the poke gesture and the press gesture respectively, we performed an ANOVA test which suggested that there was no significant difference between them ($p = 0.2586 > 0.01$, $p = 0.8154 > 0.01$, $p = 0.4804 > 0.01$). This result suggested that the force transfer pad design is effective in redirecting the force applied elsewhere to the pressure sensor. We also compared the detection performance between different switch operations. The ANOVA test suggested that there was no significant performance difference between using the thumb and using the index finger to turn the switch ($p = 0.0992 > 0.01$). We also compared the detection performance among different slider operations. The slider operation where the index finger presses on the slider has a significantly higher detection performance compared to the operations with a pinch gesture while only one pressure sensor on either the index or the thumb finger is triggered ($p = 0.01093 < 0.03$, $p = 0.0004 < 0.01$). This is due to the relatively low force required to move the sliders from sideways, which means the actual force during the latter two operations was close to the threshold we set. Therefore, they were more prone to error. Finally, we discovered that the majority of failures were caused by a hand-tracking malfunction. Meanwhile, continuous operations were more prone to the loss of hand tracking.

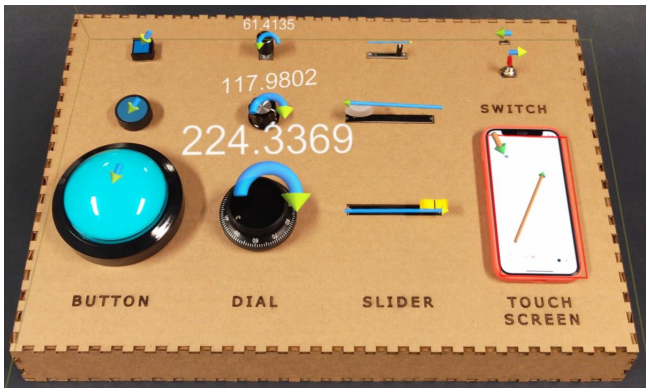

Figure 17: Mock-up machine for technical evaluation.

	Discrete UI Element						Continuous UI Element						
	Button(Poke)	Button(Press)	Switch(Thumb)	Switch(Index)	Touch Screen(Press)	Touch Screen(Poke)	Knob	Slider 1(Press)	Slider 2(Thumb)	Slider 2(Index)	Slider 3(Pinch)	Swipe Screen(Press)	Swipe Screen(Poke)
Pressure sensor fails to detect operation start	2.33%	1.67%	2.67%	1.67%	1.67%	2.33%	3.33%	1.67%	3.67%	4.00%	0.67%	3.00%	2.67%
Pressure sensor fails to detect operation end	1.00%	1.33%	1.33%	1.00%	2.00%	1.67%	2.00%	1.00%	1.67%	2.00%	0.67%	1.33%	1.33%
HoloLens fails to track hand precisely	8.00%	7.00%	6.00%	5.00%	6.00%	6.00%	11.00%	7.00%	10.00%	10.33%	10.00%	10.33%	12.00%
Error percentage due to lost traction	70.59%	70.00%	60.00%	65.22%	62.07%	60.00%	67.35%	72.41%	65.22%	63.27%	88.24%	70.45%	75.00%
Successful trial	88.67%	90.00%	90.00%	92.33%	90.33%	90.00%	83.67%	90.33%	84.67%	83.67%	88.67%	85.33%	84.00%
Successful trial standard deviation	2.21%	2.58%	3.65%	1.53%	2.33%	1.80%	4.00%	4.90%	3.06%	2.77%	3.33%	4.27%	3.59%

Figure 18: Results for Session 1: Performance of each operation detection.

5.2 Session 2: Authoring AR Tutorials

In this session, we aim to evaluate the author mode of InstruMentAR. As mentioned in the introduction, InstruMentAR aims to automate traditional immersive authoring systems for AR tutorials [3, 65]. Therefore, our baseline system is designed based on these systems where AR tutorials are created by manual manipulation of virtual content. Specifically, the user has to manually select, place, and adjust the arrows (Figure 20 a,b) while taking pictures (Figure 20 c)

once the user completes the current step, the user has to manually forward to the next step by pressing a button. The voice-to-text functionality is also implemented in the baseline as it is a feature that has been adopted by many AR devices (e.g., Hololens [6]). Meanwhile, it is not part of the automation feature as users still need to manually start and stop the recording (further discussed in section 6.1).

The two tasks we choose for this session are common tasks from the circuitry lab (Figure 19 a), which is using an oscilloscope to observe and measure two different signal outputs from a microprocessor board. Each task contains multiple steps of operations. Compared to the mock-up task in session one, this realistic task provides users with more context during operation. The oscilloscope does not have a switch, a slider, or a touchscreen, and the associated operations are not included. Nevertheless, the operations on switches, sliders, and touchscreens were extensively evaluated in session one. Meanwhile, their tutorial authoring workflows are similar to that for buttons and knobs. 10 users (8 males and 2 females, aging from 19-25) were recruited for this session. They were all college engineering students who have taken circuitry labs before. Therefore, they understood the operation logic behind the task and were considered as target users of InstruMentAR in this scenario. Each user was required to complete the authoring for both tasks through referencing the paper-based manual we provided. To counter-balance the familiarity effects, we separated the users into two groups randomly. Specifically, 5 users used the baseline system for the first task and then InstruMentAR for the second task. In contrast, the other 5 users reverse the order by using InstruMentAR first and then the baseline system. Due to the diversity of users' proficiency with manipulating virtual objects, we expect a high disparity in the quality of manually authored tutorials. Therefore, we designed an intervention mechanism in the baseline system. Specifically, if the position, orientation, or scale of the authored virtual arrows deviates more than the maximum value we set, the system does not allow the user to proceed to the next step. For InstruMentAR, we set the same maximum value that the arrows can deviate. During the user study, thanks to the high accuracy of our detection model (demonstrated in section 5.1) and post-edition from users, all users with InstruMentAR passed the minimal requirement themselves without any intervention.

5.2.1 Evaluation Metrics & Results. We report the average time it took for each user to author the tutorial for each step of the task (Figure 21). Specifically, we group the interactions into two categories. The first category is the virtual interactions that happen in the virtual space aided by the AR device. In this category, we measured the virtual arrow creation time in the baseline system which is the amount of time it took for a user to select an arrow from a library, place it in a particular location, and adjust its size and orientation. For InstruMentAR, arrow creation time only includes the amount of time it took for the user to perform an operation as the arrow is generated automatically. For both systems, we recorded the time for creating supplemental materials. For InstruMentAR, we recorded the occurrence and the time it took for post-editing. The second category is the physical interactions that happen entirely in the physical space. After virtual manipulation of the arrow, the baseline system users needed to physically operate

Task	Step		UI Element
Measure Signal of LED 1	1	Turn on the oscilloscope	6
	2	Adjust vertical scale	9
	3	Adjust trigger level	11
	4	Adjust horizontal scale	10
	5	Measure signal frequency	2
	6	Measure on-signal width	3
	7	Change measurement menu	5
	8	Measure signal duty cycle	1
Measure Signal of LED 2	1	Turn on the oscilloscope	6
	2	Turn on input signal 2	7
	3	Adjust vertical scale	9
	4	Adjust vertical offset	8
	5	Change measurement menu	4
	6	Measure & compare signal	2
	7	Measure and compare on-signal	3
	8	Change measurement menu	5
a)	9	Measure and compare signal duty cycle	1

Figure 19: (a) The description of each task. (b) The oscilloscope with its physical UI elements marked for better reference in the description.

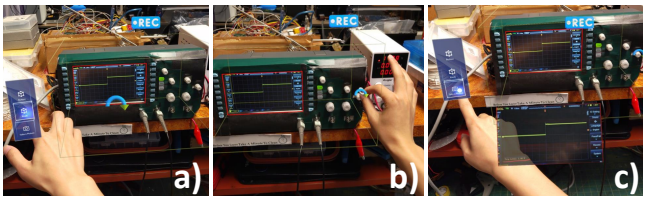

Figure 20: The Baseline system where the user, (a) creates an arrow (b) adjusts the position and orientation, and (c) takes a picture of the screen.

on the UI element to match its status as required in the next step (shown as physical operation time). This procedure is omitted in InstruMentAR as it integrates the physical operation into the arrow creation process. Meanwhile, users with both systems needed to refer to manuals for each operation. The occurrence and the time it took for manual referencing were both recorded. We recorded

how many times the user switched between a physical interaction and a virtual interaction. Finally, we compile the total average time taken for a single step and for a single task.

As hypothesized, the time it took for manual authoring is significantly longer than for automated authoring (346.16s vs 160.2s). The occurrence of context switch in the baseline system was significantly higher than that in InstruMentAR. This is attributed to the fact that users using the baseline system had to switch constantly between operating on the instrument and interacting with virtual elements. Specifically for authoring arrows during continuous operations (i.e., rotating a knob), we observed that most users (8 out of 10) performed multiple context switches during a single operation. For instance, they first positioned a virtual arrow at the desired location, then physically rotated the knob, and finally turned back to adjust the virtual arrow's scale and orientation. Meanwhile, for the baseline system users, it took them more time to reference manuals. Based on the user feedback, they often *"forget (P4)"* what they read from the manual after virtual operations that *"take too long (P6)"*. Therefore, they needed to go back to the manual to *"revive their memory (P6)"*. Notably, individuals with prior experience with Hololens required much less time for creating arrows during the manual authoring (Figure 22). In comparison, such distinction was not obvious for automated authoring which is reflected by the low standard deviation in the arrow generation time.

The Likert-type question ratings regarding specific features in InstruMentAR are shown in (Figure 23). In general, users found the workflow easy to follow (Q1: avg=4.4, sd=0.9165). *"The whole process is not much different than what I would normally operate on an oscilloscope (P3)"*. *"I don't need to memorize any operation, not like in the former (manual system) system (P5)"*. The setup procedure which involves drawing the bounding box around the instrument, specifying the display plane & the image capture area was considered to be straightforward (Q7: avg=3.9, sd=1.3). Most of them acknowledged that the post-edit option which allows them to further adjust the auto-generated AR guidance is necessary (Q3: avg=4.2, sd=1.249). *"I suppose no system is perfect, so it would always be nice to have a backup (P7)"*. Also, users believed it is necessary to have images and text as supplementary material in addition to the virtual arrow (Q2: avg=4.5, sd=0.5). *"I cannot imagine how people could get the waveform right without comparing their current screen to the image I captured (P9)"*. Meanwhile, users appreciated the fact that the visual guidance (i.e., arrow) for knobs has a dynamic shape and scale to match actual operations (Q5: avg=4.5, sd=0.922). *"It makes sense to have arrows of different lengths to represent 180-degree turn and 10-degree turn differently, even with the image as the reference (P5)"*. Most of the participants did not object to using specified gestures for operation (Q6: avg=4.2, sd=0.7483). In the meantime, they did not feel the hand wearable is too cumbersome to impede their operations (Q4: avg=1.9, sd=0.7). *"It is not ideal to wear a glove for sure, but I don't think it is a big deal considering the time and trouble it saved (P10) "*. Nevertheless, some complaints were received regarding the last two aspects, which will be further discussed and analyzed in the next session. Before the user study, we informed our users of the limitations of the gesture tracking speed and asked them to control their pace to avoid loss of tracking. Nevertheless, the loss of tracking still occasionally occurred and was reported as causing a *"frustrating experience (P5)"* as users had to *"carefully adjust my hand (P10)"* to get it re-tracked. The users did not feel the negative effect of field of view on video capture in this task because the screen of the oscilloscope is close to the main interaction area. The standard SUS survey results for the study received 81 out of 100 with a standard deviation of 15.12, which indicated high usability for the authoring mode of InstruMentAR.

	Virtual Interaction								Physical Interaction							
	Arrow Creation (s)		Supplemental Materials Creation (s)		Post Editing Occurrence (time)		Post Editing Time Per Editing (s)		Physical Operation (s)		Lab Manual Referencing (s)		Context Switch Occurrence (time)		Per Step Creation (s)	
Button	M	SD	M	SD	M	SD	M	SD	M	SD	M	SD	M	SD	M	SD
Baseline	11.60	3.31	8.26	0.99	N/A		N/A		2.16	0.17	13.94	3.01	2.62	0.43	34.41	7.96
InstrumentAR	2.21	0.38	6.20	1.10	1.25	0.72	4.08	2.50	N/A		8.13	0.91	1.26	0.16	17.22	1.67
Knob	M	SD	M	SD	M	SD	M	SD	M	SD	M	SD	M	SD	M	SD
Baseline	24.69	7.93	9.90	0.97	N/A		N/A		3.56	0.29	17.68	1.57	4.22	0.39	55.83	9.26
InstrumentAR	2.65	0.57	7.29	0.90	0.50	0.50	2.69	3.18	N/A		11.74	0.96	2.29	0.35	22.76	2.15

Total Time Per Task (s)		
	M	SD
Baseline	346.16	54.99
InstrumentAR	160.21	12.41

Figure 21: Results for Session 2: Authoring completion time for the baseline system and InstruMentAR.

		Average Time (s)	Standard Deviation (s)
Button Arrow Creation	Experienced User	8.28	1.09
	Novice User	13.82	2.27
Knob Arrow Creation	Experienced User	18.65	2.28
	Novice User	28.72	7.80

Figure 22: Difference in arrow creation time between experienced users and novice users.

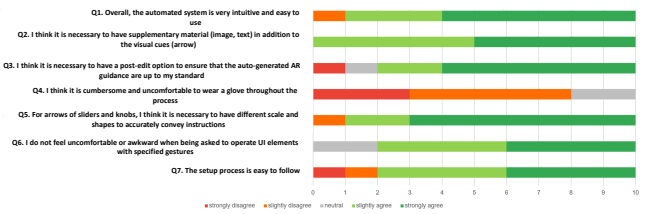

Figure 23: Results for Session 2: Likert-type questionnaire results.

5.3 Session 3: Receiving AR tutorials

In this session, we recruited 10 users (8 male, 2 female, aging from 20-25). 2 of the users had experienced AR applications from phones. None of them had used Hololens before. Similar to the prior session, each user was a college engineering student. The study lasted around 1 hour and each user was compensated with 20 dollars. We first asked our users to walk through the Hololens 2 official tutorial to learn how to navigate the user interface with basic hand gestures. After completing the task, the user completed surveys with Likert-type (scaled 1-5) questions regarding the user experience of specific system features and a standard System Usability Scale (SUS) questionnaire on InstruMentAR. We then conducted an open-ended interview to get subjective feedback on our system. In this session, we aim to evaluate the automation features (i.e., automatic playback and feedback) in the access mode. As discussed in section 2.1, AR tutorials have found widespread adoption [3, 9], due to their many advantages over traditional paper/video-based tutorials [22, 58, 78]. Meanwhile, the comparison between paper/video-based tutorials

and AR tutorials has been thoroughly explored in the past [20, 81]. Therefore, we focus on comparing InstruMentAR with existing AR tutorials that cannot track users' interactions. In this session, the baseline system delivers the same AR tutorial with the same visual cues (i.e., arrows, text, and images) as InstruMentAR. However, the baseline system lacks the automation features in InstruMentAR as it cannot track user interactions. Specifically, users with the baseline system must manually forward the tutorial every time they complete a step and they cannot receive any warning feedback. The AR tutorial used in both the baseline and InstruMentAR was authored by us so we can control its quality and ensure that it is consistent for all users. This is important as the evaluation target is the novel features in the access mode instead of the quality of AR tutorials. We adopted the same oscilloscope tasks described in the previous session. Like in the previous session, we separated the users into two groups randomly. 5 users used the baseline system for the first task and then InstruMentAR for the second task. The other 5 users reversed the order by using InstruMentAR first and then the baseline system.

5.3.1 Evaluation Metrics & Results. We report the average time it took for each user to complete the operation of the task following the tutorial (Figure 24). We recorded how much time it took for baseline users to manually click on the next step to forward the tutorial. This operation is omitted in InstruMentAR. Overall, it took less time for the users to follow the AR tutorials in InstruMentAR (57.06s vs 107.7s). The time difference (50.04s) was attributed to the omitted procedure of forwarding steps (20.65s) and the warning feedback. According to the feedback from our users, the real-time feedback helped them feel confident that they had completed each step correctly, so they did not need to constantly reference tutorials. Four users preferred the InstruMentAR to the baseline system. It was in contrast to the authoring session where InstruMentAR was the unanimous favorite. Most of the complaints came from the fact that they had to use specified gestures. *"It is already challenging to understand each operation, I don't want the devote more effort on remembering gestures (P1)"*(Q3: avg=3.4, sd=1.1135). Also, the feelings towards the hand wearable were mixed. *"Sometimes I worry I might accidentally press the wrong button with the extra hardware on my fingertip (P6)"*(Q2: avg=2.9, sd=0.9165). Nevertheless, most users appreciated the preemptive warning feature (Q4: avg=4.9, sd=0.9434). *"I don't have to worry about pressing the wrong button and starting all over again. Now I have more confidence when making the next move (P2)"*. The feelings towards the automatic playback feature were also generally positive (Q1: avg=3.7, sd=0.9). *"It feels like magic that the tutorial can move forward on itself"*. The SUS survey results for the study received 68 out of 100 with a standard deviation of 17.49.

6 DISCUSSION, LIMITATIONS, AND FUTURE WORK

In this section, we discuss the primary results of the study, the system limitations, and their implications in detail. We also provide insights and recommendations for the future design of automated AR authoring systems for instrument operations.

	Click Next Step (s)		Physical Operation (s)		Total Time Per Step (s)	
Button	M	SD	M	SD	M	SD
Baseline	2.42	0.56	3.25	0.47	8.72	1.56
InstrumentAR	N/A		1.79	0.40	3.23	0.60
Knob	M	SD	M	SD	M	SD
Baseline	2.45	0.44	5.36	0.50	12.95	1.27
InstrumentAR	N/A		3.52	0.66	5.80	0.75

Total Click Next(s)		
	M	SD
Baseline	20.65	3.59
InstrumentAR	NA	N/A

Total Time Per Task (s)		
	M	SD
Baseline	107.70	11.08
InstrumentAR	57.06	4.48

Figure 24: Results for Session 3: Tutorial consumption time for the baseline system and InstruMentAR.

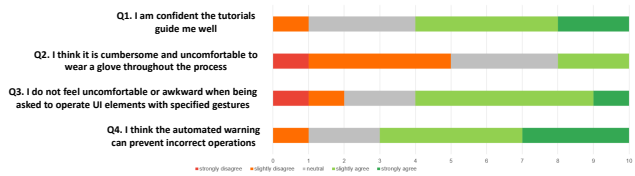

Figure 25: Results for Session 3: Likert-type questionnaire results.

6.1 Automated Authoring

As hypothesized, the automated authoring workflow is appreciated by all users and is less dependent on the user's familiarity with virtual interaction. It conforms to the low-friction design principle [14, 28] which asserts that users should be able to accomplish their tasks as easily and directly as possible. We believe that some aspects of the workflow could be further automated in the future. For example, the user does not need to manually define the interaction area by drawing a bounding box if the system can automatically determine this area by scanning the geometry of the instrument. However, the resolution of current AR devices' built-in depth camera is too low to accurately scan the geometry of the instrument. In addition, the "voice recording" has been identified as the most time-consuming process in the current authoring workflow as users have to manually *"decide when to start and when to end (P2, session one)"*. With the advancement in natural language processing (NLP) technologies and its promising applications in text summarization [31], it is worth investigating the feasibility of recording users' ongoing narrations in the background while automatically summarizing them into bulletin points displayed in AR.

6.2 Multimodal Tracking Approach

In essence, our multimodal tracking approach utilizes the users' gestural information during the operation to derive the locations

and orientations of the object. It is more resistant to hand occlusions compared to conventional approaches that use external cameras to track manipulated objects directly. Besides, approaches relying on external cameras require careful calibration to a specific environment, thus limiting the mobility of interaction [40, 94]. In contrast, our approach employs hand tracking modules already integrated into the AR device as well as a plug-and-play wearable, which fosters spontaneous interactions. Also, our approach does not require pre-modeling or pre-training of the objects as opposed to other camera-based approaches [37, 99]. Therefore it can scale more easily to a variety of objects and tasks. For example, the user can utilize this approach to empower toys with AR animations (Figure 26 a). The squeeze upon the cannon can be detected by the pressure sensor while the cannon's rotation can be tracked based on the gestural information. Similarly, it can enable AR sketching on a hard surface (e.g., a wall, or a table) by recording the finger's trajectory during the surface contact (Figure 26 b).

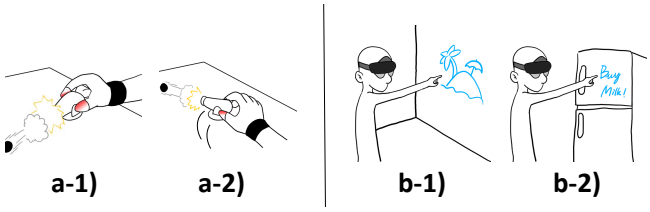

Figure 26: (a) The user plays with an AR-empowered toy. (a-1) The user squeezes the cannon to fire a virtual ball. (a-2) The user rotates the cannon to fire in different directions. (b) Sketching on a hard surface. (b-1) The user draws a virtual picture on the wall. (b-2) The user writes a sticker note on the refrigerator.

6.3 Supporting User-Defined Operation Threshold

As previously mentioned, we experimented with the operation of different physical UI elements on common instruments. We finally adopted the minimum pressure required as the universal threshold value. Due to the variety of the UI element's physical properties (e.g., button's string stiffness), this universal threshold value is a limiting factor for the system's scalability. It is unfeasible for developers to test every physical UI element available and recalibrate the threshold on a case-by-case basis. Therefore, we envision a workflow where end-users can set this value adaptively in real-time.

In the setup process, we introduce an additional step where the user manually records the pressure value when the UI element is being operated. This step is done by explicitly informing the system when the operation (e.g., pressing the button) takes place. In this way, the system can register the value in the back-end and apply it for future operations on this UI element. This step needs to be repeated on each UI element (e.g., button, knobs) of the instrument. This improved workflow allows the system to set a unique threshold value for each UI element.

6.4 Supporting More Diverse Gestures

The system currently only supports a fixed number of gestures for each operation, which does not take into account each user's personalized gestural patterns while performing an operation. It is a major source of the complaints we received. For example, a user (P2, session two) who preferred to use only the index finger to turn the knob complained that *"I wish I had more freedom in choosing how to (use my own gestures to) operate"*. To support more gestures for operation, we can extend the Gesture Filter (Figure 8) in the decision tree to recognize more gestures. This can be achieved by incorporating more finger joints as input while developing more sophisticated gesture tracking algorithms. The extension of the Gesture Filter can also help support more interaction types such as the multi-touch operation.

Nevertheless, it is still difficult for developers to anticipate every user's gesture preferences and incorporate them into the system in advance. Therefore, we believe that the most effective solution to this limitation is to enable end users to define their personalized gestures in real-time. Recently, GesturAR [95] achieved a similar outcome by allowing end users to define their own gestures which are used to animate AR content. The one-shot learning technique [50] they adopted allows the system to detect hand gestures using only one demonstration example by comparing the real-time hand data with it. In the future, we can develop a similar methodology to allow each user to demonstrate and register their own gesture into the system.

6.5 Incorporating Error Handling During Authoring

Right now, InstruMentAR only supports authoring single-thread, sequential tutorials that novice users follow step by step. If the novice makes an error during operation (e.g., presses the wrong button), the system will pause its automatic display. The novice user has to determine if this operation alters the status of the instrument. If not, users can manually resume the AR tutorials. Otherwise, they need to start from the first step by resetting the tutorial, which is described as *"discouraging (P3 session two)"* and *"frustrating (P2 session two)"*.

One way to address this problem is by incorporating error handling at the authoring stage. Inspired by prior works [55, 90], we envision adopting the branch-based AR authoring that allows authors to account for potential errors. The author can not only use the main branch to demonstrate the correct steps of operation but also use the child branch to direct novice users back to the closest checkpoint so they can restart from there. The multi-branched AR tutorials facilitate easier recovery from unintended errors for novice users.

6.6 Modeling the Instrument's Status

As previously mentioned, InstruMentAR generates sequential AR tutorials along a single linear path while tracking only user interactions as input to forward the steps. Meanwhile, tracking the instrument's own status through analyzing screen displays can be incorporated to enable further functionalities. For example, the current system only displays the screen images as additional visual

cues. During the user study, several users reflected that the diversity of visual guidance needs to be further expanded. *"It took me a while to find the voltage number on the picture that the instruction is referring to (P5 session two)"*. If the system were able to model and extract the voltage number from the screen capture, it would only display the number on the side. The number could also change dynamically as users gradually turn the knob if the system can understand the relationship between input and output. In addition, instrument status tracking can complement the existing user interaction tracking to determine the completion of the operation. For example, the "Complete" sign shown on the screen would inform the system to forward to the next step. Similarly, more extensive feedback could be facilitated by analyzing the warning signs on the screen.

Specifically, several recent works have successfully modeled information on the screen by adopting computer vision techniques and crowdsourcing [34–36, 51]. In light of their accomplishments, we plan to draw inspiration from their methodologies and incorporate instrument status modeling to achieve the aforementioned functionalities.

6.7 Improving the Hand Wearable Design

As the first generation of the prototype, our hand wearable has some flaws in terms of ergonomics, resulting in some user discomfort. For instance, a user argued that the vibration module should be placed on the fingertips alongside the pressure sensor since it is more close to *"where the error takes place (P8 session one)"*. In addition, another user complained that the pressure sensor attached to his fingertip altered his touch sensation as the user can no longer feel the texture of real objects, which results in an *"unrealistic and non-intuitive (P9 session one)"* operation. This observation aligns with the finding from prior work [82]. To address this issue, we will explore to find new methods to detect object contact (e.g., IMU strapped around the back of the finger [33]) without obstructing touch. In addition, the *operation threshold* for the pressure sensor reading is set arbitrarily based on our test on the different UI elements. In the future, it could be fine-tuned dynamically based on users' individual force profiles.

7 CONCLUSION

In this work, we present InstruMentAR, a system that can automatically generate AR tutorials by recording the author's embodied demonstrations. We start by eliciting 5 common UI elements and their associated operational gestures. Then, we designed the hand wearable which can precisely detect when the UI operation takes place. Furthermore, we develop the authoring interface allowing users to create AR tutorials through step-by-step demonstrations of the instrument. We also implement preemptive feedback and automatic feedback in access mode to aid novice users in consuming tutorials more efficiently. In the user study, we evaluated the efficiency and usability of InstruMentAR by comparing it against two baseline systems for authoring mode and access mode respectively. We hope that this paper creates new opportunities for automating and democratizing AR tutorial authoring for a wider audience.

ACKNOWLEDGMENTS

We wish to give a special thanks to the reviewers for their invaluable feedback. This work is partially supported by the NSF under the Future of Work at the Human Technology Frontier (FW-HTF) 1839971. We also acknowledge the Feddersen Distinguished Professorship Funds. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency. We sincerely thank Raja Pashin Farsak for his help in building the mock-up machine for the user study. We also thank Yikuan Liu for drawing the sketches.

REFERENCES

[1] 2022. ARCore Build the Future. https://developers.google.com/ar.
[2] 2022. Dive into the world of augmented reality. https://developer.apple.com/augmented-reality/.
[3] 2022. Dynamics 365 Remote Assist. https://dynamics.microsoft.com/en-us/mixed-reality/remote-assist/.
[4] 2022. getPerspectiveTransform(): Calculates a perspective transform from four pairs of the corresponding points. https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#ga20f62aa3235d869c9956436c870893ae.
[5] 2022. Hololens 2. https://www.microsoft.com/en-us/hololens/.
[6] 2022. Hololens: Voice input. https://learn.microsoft.com/en-us/windows/mixed-reality/design/voice-input.
[7] 2022. Kinect Azure. https://azure.microsoft.com/en-us/services/kinect-dk/.
[8] 2022. MagicLeap. https://ml1-developer.magicleap.com/en-us/learn/guides/lumin-sdk-handtracking.
[9] 2022. Modernize training and work instructions with Vuforia Studio. https://www.ptc.com/en/products/vuforia/vuforia-studio.
[10] 2022. MRTK. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05.
[11] 2022. OpenCV. https://opencv.org/.
[12] 2022. Unity Real-Time Development Platform. https://unity.com/.
[13] 2022. Unreal Engine: The most powerful real-time 3D creation tool. https://www.unrealengine.com/en-US.
[14] 2022. Victoria Young. 2015. Strategic UX: The Art of Reducing Friction. https://informationdesign.org/strategic-ux-the-art-of-reducing-friction/.
[15] Harini Alagarai Sampath, Bipin Indurkhya, Eunhwa Lee, and Yudong Bae. 2015. Towards multimodal affective feedback: Interaction between visual and haptic modalities. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2043–2052.
[16] Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2013. YouMove: enhancing movement training with an augmented reality mirror. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 311–320.
[17] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K Chilana. 2020. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.
[18] Mark Billinghurst, Raphael Grasset, and Julian Looser. 2005. Designing augmented reality interfaces. *ACM Siggraph Computer Graphics* 39, 1 (2005), 17–22.
[19] Andrew Thomas Bimba, Norisma Idris, Ahmed Al-Hunaiyyan, Rohana Binti Mahmud, and Nor Liyana Bt Mohd Shuib. 2017. Adaptive feedback in computer-based learning environments: a review. *Adaptive Behavior* 25, 5 (2017), 217–234.
[20] Jonas Blattgerste, Benjamin Strenge, Patrick Renner, Thies Pfeiffer, and Kai Essig. 2017. Comparing conventional and augmented reality instructions for manual assembly tasks. In *Proceedings of the 10th international conference on pervasive technologies related to assistive environments*. 75–82.
[21] Yuanzhi Cao, Anna Fuste, and Valentin Heun. 2022. MobileTutAR: a Lightweight Augmented Reality Tutorial System using Spatially Situated Human Segmentation Videos. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–8.
[22] 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.
[23] Xiang'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E Hudson. 2014. Air+ touch: interweaving touch & in-air gestures. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 519–525.
[24] Yu-Peng Chen, Chen Bai, Adam Wolach, Mamoun Mardini, and Lisa Anthony. 2021. Detecting Face Touching with Dynamic Time Warping on Smartwatches: A Preliminary Study. In *Companion Publication of the 2021 International Conference on Multimodal Interaction*. 19–24.
[25] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 93–102.

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

[27] Subramanian Chidambaram, Hank Huang, Fengming He, Xun Qian, Ana M Villanueva, Thomas S Redick, Wolfgang Stuerzlinger, and Karthik Ramani. 2021. Processar: An augmented reality-based tool to create in-situ procedural 2d/3d ar instructions. In *Designing Interactive Systems Conference 2021*. 234–249.

[28] Anna L Cox, Sandy JJ Gould, Marta E Cecchinato, Ioanna Iacovides, and Ian Renfree. 2016. Design frictions for mindful interactions: The case for microboundaries. In *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems*. 1389–1397.

[29] Daniel Eckhoff, Christian Sandor, Christian Lins, Ulrich Eck, Denis Kalkofen, and Andreas Hein. 2018. TutAR: augmented reality tutorials for hands-only procedures. In *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*. 1–3.

[30] Jennifer Fernquist, Tovi Grossman, and George Fitzmaurice. 2011. Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 373–382.

[31] Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47, 1 (2017), 1–66.

[32] Janet K Gibbs, Marco Gillies, and Xueni Pan. 2022. A comparison of the effects of haptic and visual feedback on presence in virtual reality. *International Journal of Human-Computer Studies* 157 (2022), 102717.

[33] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and low-latency sensing of touch contact on any surface with finger-worn IMU sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 1059–1070.

[34] Anhong Guo, Xiang'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P Bigham. 2016. Vizlens: A robust and interactive screen reader for interfaces in the real world. In *Proceedings of the 29th annual symposium on user interface software and technology*. 651–664.

[35] Anhong Guo, Jeeeun Kim, Xiang'Anthony' Chen, Tom Yeh, Scott E Hudson, Jennifer Mankoff, and Jeffrey P Bigham. 2017. Facade: Auto-generating tactile interfaces to appliances. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 5826–5838.

[36] Anhong Guo, Junhan Kong, Michael Rivera, Frank F Xu, and Jeffrey P Bigham. 2019. Statelens: A reverse engineering solution for making existing dynamic touchscreens accessible. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 371–385.

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

[38] Fernando Gutierrez and John Atkinson. 2011. Adaptive feedback selection for intelligent tutoring systems. *Expert Systems with Applications* 38, 5 (2011), 6146–6152.

[39] Teng Han, Khalad Hasan, Keisuke Nakamura, Randy Gomez, and Pourang Irani. 2017. Soundcraft: Enabling spatial interactions on smartwatches using hand generated acoustics. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 579–591.

[40] Jeremy Hartmann, Yen-Ting Yeh, and Daniel Vogel. 2020. AAR: Augmenting a wearable augmented reality display with an actuated head-mounted projector. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 445–458.

[41] Julie Heiser, Doantam Phan, Maneesh Agrawala, Barbara Tversky, and Pat Hanrahan. 2004. Identification and validation of cognitive design principles for automated generation of assembly instructions. In *Proceedings of the working conference on Advanced Visual Interfaces*. 311–319.

[42] Steven J Henderson and Steven K Feiner. 2011. Augmented reality in the psychomotor phase of a procedural task. In *2011 10th IEEE international symposium on mixed and augmented reality*. IEEE, 191–200.

[43] Bradley Herbert, Barrett Ens, Amali Weerasinghe, Mark Billinghurst, and Grant Wigley. 2018. Design considerations for combining augmented reality with intelligent tutors. *Computers & Graphics* 77 (2018), 166–182.

[44] Ken Hinckley, Seongkook Heo, Michel Pahud, Christian Holz, Hrvoje Benko, Abigail Sellen, Richard Banks, Kenton O'Hara, Gavin Smyth, and William Buxton. 2016. Pre-touch sensing for mobile interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2869–2881.

[45] Clemens Hoffmann, Sebastian Büttner, Michael Prilla, and Kai Wundram. 2020. Impact of augmented reality guidance for car repairs on novice users of AR: a field experiment on familiar and unfamiliar tasks. In *Proceedings of the Conference on Mensch und Computer*. 279–289.

[46] Gaoping Huang, Xun Qian, Tianyi Wang, Fagun Patel, Maitreya Sreeram, Yuanzhi Cao, Karthik Ramani, and Alexander J Quinn. 2021. Adaptutar: An adaptive tutoring system for machine tasks in augmented reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[47] M Gail Jones, Alexandra Bokinsky, Thomas Tretter, and Atsuko Negishi. 2005. A comparison of learning with haptic and visual modalities. (2005).

[48] Bui Minh Khuong, Kiyoshi Kiyokawa, Andrew Miller, Joseph J La Viola, Tomohiro Mashita, and Haruo Takemura. 2014. The effectiveness of an AR-based context-aware assembly support system in object assembly. In *2014 IEEE Virtual Reality (VR)*. IEEE, 57–62.

[49] Wolf Kienzle, Eric Whitmire, Chris Rittaler, and Hrvoje Benko. 2021. Electroring: Subtle pinch and touch detection with a ring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[50] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Vol. 2. Lille, 0.

[51] Junhan Kong, Dena Sabha, Jeffrey P Bigham, Amy Pavel, and Anhong Guo. 2021. TutorialLens: authoring Interactive augmented reality tutorials through narration and demonstration. In *Symposium on Spatial User Interaction*. 1–11.

[52] Suha Kwak, Woonhyun Nam, Bohyung Han, and Joon Hee Han. 2011. Learning occlusion with likelihoods for visual tracking. In *2011 International Conference on Computer Vision*. IEEE, 1551–1558.

[53] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community enhanced tutorials: improving tutorials with multiple demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1779–1788.

[54] 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.

[55] 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.

[56] Jian Liao, Adnan Karim, Shivesh Jadon, Rubaiat Habib Kazi, and Ryo Suzuki. 2022. RealityTalk: Real-Time Speech-Driven Augmented Presentation for AR Live Storytelling. *arXiv preprint arXiv:2208.06350* (2022).

[57] Shaofeng Lu, Ying Cheng, Xiaoyang Wang, Yang Du, and Eng Gee Lim. 2017. Exploring the effectiveness of student-generated video tutorials in electronic lab-based teaching. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–4.

[58] Peter Mohr, David Mandl, Markus Tatzgern, Eduardo Veas, Dieter Schmalstieg, and Denis Kalkofen. 2017. Retargeting video tutorials showing tools with surface contact to augmented reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 6547–6558.

[59] Peter Mohr, Shohei Mori, Tobias Langlotz, Bruce H Thomas, Dieter Schmalstieg, and Denis Kalkofen. 2020. Mixed reality light fields for interactive remote assistance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[60] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. Protoar: Rapid physical-digital prototyping of mobile augmented reality applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[61] Ohan Oda, Carmine Elvezio, Mengu Sukan, Steven Feiner, and Barbara Tversky. 2015. Virtual replicas for remote assistance in virtual and augmented reality. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 405–415.

[62] James Patten, Hiroshi Ishii, Jim Hines, and Gian Pangaro. 2001. Sensetable: a wireless object tracking platform for tangible user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 253–260.

[63] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2016. VidCrit: video-based asynchronous video review. In *Proceedings of the 29th annual symposium on user interface software and technology*. 517–528.

[64] Thammathip Piumsomboon, Gun A Lee, Jonathon D Hart, Barrett Ens, Robert W Lindeman, Bruce H Thomas, and Mark Billinghurst. 2018. Mini-me: An adaptive avatar for mixed reality remote collaboration. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

[65] Alexander Plopski, Varunyu Fuvattanasilp, Jarkko Poldi, Takafumi Taketomi, Christian Sandor, and Hirokazu Kato. 2018. Efficient in-situ creation of augmented reality tutorials. In *2018 Workshop on metrology for industry 4.0 and IoT*. IEEE, 7–11.

[66] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F Cohen. 2011. Pause-and-play: automatically linking screencast video tutorials with applications. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 135–144.

[67] Arnaud Prouzeau, Yuchen Wang, Barrett Ens, Wesley Willett, and Tim Dwyer. 2020. Corsican twin: Authoring in situ augmented reality visualisations in virtual reality. In *Proceedings of the International Conference on Advanced Visual Interfaces*. 1–9.

[68] Jing Qian, Jiaju Ma, Xiangyu Li, Benjamin Attal, Haoming Lai, James Tompkin, John F Hughes, and Jeff Huang. 2019. Portal-ble: Intuitive free-hand manipulation in unbounded smartphone-based augmented reality. In *Proceedings of*

the 32nd Annual ACM Symposium on User Interface Software and Technology. 133–145.

[69] Xun Qian, Fengming He, Xiyun Hu, Tianyi Wang, and Karthik Ramani. 2022. ARnnotate: An Augmented Reality Interface for Collecting Custom Dataset of 3D Hand-Object Interaction Pose Estimation. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[70] Shwetha Rajaram and Michael Nebeling. 2022. Paper Trail: An Immersive Authoring System for Augmented Reality Instructional Experiences. In *CHI Conference on Human Factors in Computing Systems*. 1–16.

[71] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. Retrofab: A design tool for retrofitting physical interfaces using actuators, sensors and 3d printing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 409–419.

[72] Tobias Schneider, Sabiha Ghellal, Steve Love, and Ansgar RS Gerlicher. 2021. Increasing the user experience in autonomous driving through different feedback modalities. In *26th International Conference on Intelligent User Interfaces*. 7–10.

[73] Eldon Schoop, Michelle Nguyen, Daniel Lim, Valkyrie Savage, Sean Follmer, and Björn Hartmann. 2016. Drill Sergeant: Supporting physical construction projects through an ecosystem of augmented tools. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 1607–1614.

[74] Roland Sigrist, Georg Rauter, Laura Marchal-Crespo, Robert Riener, and Peter Wolf. 2015. Sonification and haptic feedback in addition to visual feedback enhances complex motor task learning. *Experimental brain research* 233, 3 (2015), 909–925.

[75] Roland Sigrist, Georg Rauter, Robert Riener, and Peter Wolf. 2013. Augmented visual, auditory, haptic, and multimodal feedback in motor learning: a review. *Psychonomic bulletin & review* 20, 1 (2013), 21–53.

[76] Ana Stanescu, Peter Mohr, Dieter Schmalstieg, and Denis Kalkofen. 2022. Model-Free Authoring By Demonstration of Assembly Instructions in Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[77] Ryo Suzuki, Hooman Hedayati, Clement Zheng, James L Bohn, Daniel Szafir, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. 2020. Roomshift: Room-scale dynamic haptics for vr with furniture-moving swarm robots. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–11.

[78] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. 2022. Augmented Reality and Robotics: A Survey and Taxonomy for AR-enhanced Human-Robot Interaction and Robotic Interfaces. In *CHI Conference on Human Factors in Computing Systems*. 1–33.

[79] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. Realitysketch: Embedding responsive graphics and visualizations in AR through dynamic sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 166–181.

[80] Ryo Suzuki, Eyal Ofek, Mike Sinclair, Daniel Leithinger, and Mar Gonzalez-Franco. 2021. HapticBots: Distributed Encountered-type Haptics for VR with Multiple Shape-changing Mobile Robots. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 1269–1281.

[81] Arthur Tang, Charles Owen, Frank Biocca, and Weimin Mou. 2003. Comparative effectiveness of augmented reality in object assembly. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 73–80.

[82] Yujie Tao, Shan-Yuan Teng, and Pedro Lopes. 2021. Altering Perceived Softness of Real Rigid Objects by Restricting Fingerpad Deformation. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 985–996.

[83] Santawat Thanyadit, Parinya Punpongsanon, Thammathip Piumsomboon, and Ting-Chuen Pong. 2022. XR-LIVE: Enhancing Asynchronous Shared-Space Demonstrations with Spatial-temporal Assistive Toolsets for Effective Learning in Immersive Virtual Laboratories. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW1 (2022), 1–23.

[84] Michael Thees, Sebastian Kapp, Martin P Strzys, Fabian Beil, Paul Lukowicz, and Jochen Kuhn. 2020. Effects of augmented reality on learning and cognitive load in university physics laboratory courses. *Computers in Human Behavior* 108 (2020), 106316.

[85] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. 2019. Loki: Facilitating remote instruction of physical tasks using bi-directional mixed-reality telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 161–174.

[86] Balasaravanan Thoravi Kumaravel, Cuong Nguyen, Stephen DiVerdi, and Björn Hartmann. 2019. TutoriVR: A video-based tutorial system for design applications in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[87] Carlos Torres and Pablo Figueroa. 2018. Learning how to play a guitar with the HoloLens: A case study. In *2018 XLIV Latin American Computer Conference (CLEI)*. IEEE, 606–611.

[88] Edward Tse, Chia Shen, Saul Greenberg, and Clifton Forlines. 2006. Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In *Proceedings of the working conference on Advanced visual interfaces*. 336–343.

[89] Ana Villanueva, Ziyi Liu, Yoshimasa Kitaguchi, Zhengzhe Zhu, Kylie Peppler, Thomas Redick, and Karthik Ramani. 2021. Towards modeling of human skilling for electrical circuitry using augmented reality applications. *International Journal of Educational Technology in Higher Education* 18, 1 (2021), 1–23.

[90] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Kylie Peppler, Thomas Redick, and Karthik Ramani. 2020. Meta-AR-app: an authoring platform for collaborative augmented reality in STEM classrooms. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.

[91] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Feiyang Wang, Subramanian Chidambaram, and Karthik Ramani. 2022. Colabar: A toolkit for remote collaboration in tangible augmented reality laboratories. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW1 (2022), 1–22.

[92] Ana M Villanueva, Ziyi Liu, Zhengzhe Zhu, Xin Du, Joey Huang, Kylie A Peppler, and Karthik Ramani. 2021. Robotar: An augmented reality compatible teleconsulting robotics toolkit for augmented makerspace experiences. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

[93] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, and Haiyan Zhang. 2018. Project zanzibar: A portable and flexible tangible interaction platform. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[94] Chiu-Hsuan Wang, Seraphina Yong, Hsin-Yu Chen, Yuan-Syun Ye, and Liwei Chan. 2020. HMD light: Sharing In-VR experience via head-mounted projector for asymmetric interaction. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 472–486.

[95] 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.

[96] Giles Westerfield, Antonija Mitrovic, and Mark Billinghurst. 2013. Intelligent augmented reality training for assembly tasks. In *International conference on artificial intelligence in education*. Springer, 542–551.

[97] Matt Whitlock, George Fitzmaurice, Tovi Grossman, and Justin Matejka. 2019. AuthAR: concurrent authoring of tutorials for AR assembly guidance. (2019).

[98] Jacob O Wobbrock, Meredith Ringel Morris, and Andrew D Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1083–1092.

[99] Li-Chen Wu, I-Chen Lin, and Ming-Han Tsai. 2016. Augmented reality instruction for object assembly based on markerless tracking. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 95–102.

[100] Qin Wu, Jiayuan Wang, Sirui Wang, Tong Su, and Chenmei Yu. 2019. MagicPAPER: Tabletop interactive projection device based on tangible interaction. In *ACM SIGGRAPH 2019 Posters*. 1–2.

[101] Masahiro Yamaguchi, Shohei Mori, Peter Mohr, Markus Tatzgern, Ana Stanescu, Hideo Saito, and Denis Kalkofen. 2020. Video-annotated augmented reality assembly tutorials. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1010–1022.

[102] Zuoyi Zhang, Huizhe Zheng, Sawyer Rempel, Kenny Hong, Teng Han, Yumiko Sakamoto, and Pourang Irani. 2020. A smart utensil for detecting food pick-up gesture and amount while eating. In *Proceedings of the 11th Augmented Human International Conference*. 1–8.

[103] Fengyuan Zhu and Tovi Grossman. 2020. Bishare: Exploring bidirectional interactions between smartphones and head-mounted augmented reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[104] Zhengzhe Zhu, Ziyi Liu, Tianyi Wang, Youyou Zhang, Xun Qian, Pashin Farsak Raja, Ana Villanueva, and Karthik Ramani. 2022. MechARspace: An Authoring System Enabling Bidirectional Binding of Augmented Reality with Toys in Real-time. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.