TutorialLens: Authoring Interactive Augmented Reality Tutorials Through Narration and Demonstration

Junhan Kong
The Information School,
University of Washington
Seattle, WA, USA
junhank@uw.edu

Dena Sabha
Human-Centered Design and Engineering, University of Washington
Seattle, WA, USA
dsabha19@uw.edu

Jeffrey P. Bigham
Human-Computer Interaction Institute, School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA
jbigham@cs.cmu.edu

Amy Pavel
Human-Computer Interaction Institute, School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA
apavel@cs.cmu.edu

Anhong Guo
Computer Science and Engineering,
University of Michigan
Ann Arbor, MI, USA
anhong@umich.edu

ABSTRACT

Exploring unfamiliar devices and interfaces through trial and error can be challenging and frustrating. Existing video tutorials require frequent context switching between the device showing the tutorial and the device being used. While augmented reality (AR) has been adopted to create user manuals, many are inflexible for diverse tasks, and usually require programming and AR development experience. We present TutorialLens, a system for authoring interactive AR tutorials through narration and demonstration. To use TutorialLens, authors demonstrate tasks step-by-step while verbally explaining what they are doing. TutorialLens automatically detects and records 3D finger positions and guides authors to capture important changes of the device. Using the created tutorials, TutorialLens then provides AR visual guidance and feedback for novice device users to complete the demonstrated tasks. TutorialLens is automated, friendly to users without AR development experience, and applicable to a variety of devices and tasks.

CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**.

KEYWORDS

Augmented reality, tutorials, dynamic interfaces, authoring tools, narration, demonstration, computer vision.

ACM Reference Format:
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 (SUI '21), November 9–10, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485279.3485289

1 INTRODUCTION

Using unfamiliar devices and interfaces has been a pervasive challenge – purchasing tickets at a subway station in an unfamiliar city; trying to print a document at a new workplace; parking or sending packages with self-service kiosks. In these situations, especially with complicated tasks that might involve many possible user interactions, trial and error can be challenging, frustrating and time-consuming.

To help users interact with these interfaces, various tutorial systems have been created, but most are not sufficient to meet diverse user needs and scenarios. Traditional user manuals are usually too information-heavy and require high cognitive load; On-device instructions usually lack context on which part of the devices to look at and interact with, such as where to place physical objects (like inserting paper checks to ATM machines). AR and video tutorials have also been created to provide more context to users, yet video tutorials require frequent context switching between the device playing the video and the device a user is trying to interact with, and give little feedback on whether users correctly followed the instructions. Moreover, existing AR tutorials are often designed for a single interface (sometimes by the manufacturers) thus hard to generalize to a variety of tasks. Additionally, authoring AR tutorials often requires expertise such as programming and 3D modeling. Thus, creating interactive, contextual, and easy-to-use AR tutorials for a variety of tasks remains a challenge.

To identify the key challenges and needs in authoring user tutorials, we first conducted a two-part formative study. For the first part, we selected 12 tutorial videos on a variety of devices, iteratively created a code book focusing on the hierarchy of and relationship between user actions and device feedback, and coded the videos using the code book. However, since many of these videos were created by expert content creators and involved heavy post-hoc editing, they were not representative of a regular user's workflow when creating tutorials to share with their friends and family. To

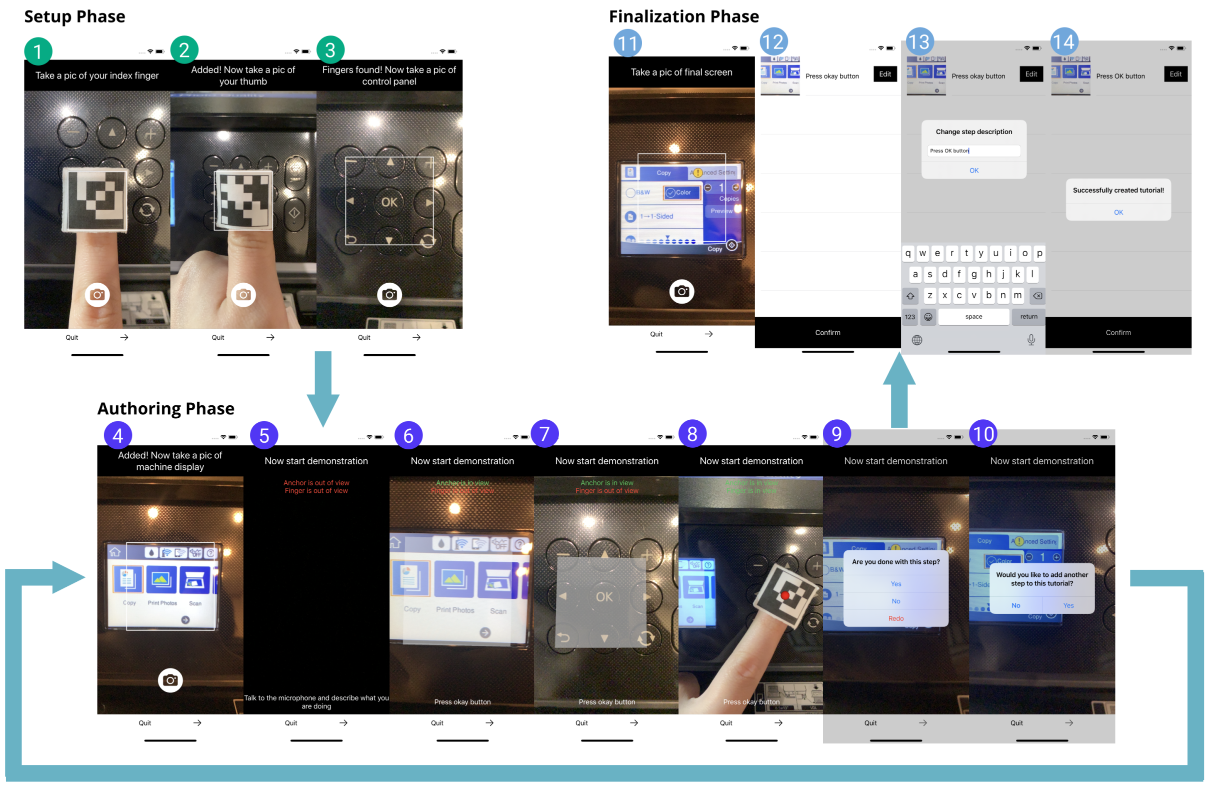
Setup Phase
1
Take a pic of your index finger
2
Added! Now take a pic of your thumb
3
Fingers found! Now take a pic of control panel
Finalization Phase
11
Take a pic of final screen
12
Press okay button
Edit
Confirm
13
Change step description
Press OK button
OK
14
Press OK button
Successfully created tutorial!
OK
Authoring Phase
4
Added! Now take a pic of machine display
5
Now start demonstration
Anchor is out of view
Finger is out of view
Talk to the microphone and describe what you are doing
6
Press okay button
7
Anchor is in view
Finger is out of view
8
9
Are you done with this step?
Yes
No
Redo
10
Would you like to add another step to this tutorial?
No
Yes
Quit

Figure 1: Example screens of an author creating an interactive AR tutorial using TutorialLens: first configuring finger markers for better tracking, then taking a picture of the device control panel (screens 1-3). After that, the author repeats the process of taking a picture of the current display screen state, verbally describing and demonstrating the current step, and confirming step completion or redoing it (screens 4-10). In the end, the author takes a picture of the final display screen state, reviews the steps and edits their descriptions (screens 11-14).

better investigate the process and challenges when creating a video tutorial in-situ, we then conducted a user study with 10 participants. From the video coding and user study findings, we extracted key insights, including allowing hands-free interactions, providing sufficient context to novice device users, being clear and concise with instructions, enabling easy editing of tutorials, and showing potential errors and failures. Some of these key insights informed our design decisions of TutorialLens.

Next, we present TutorialLens, a system for authoring interactive AR tutorials through voice narration and user demonstration. TutorialLens is designed for users who want to author device-oriented AR tutorials, where 1) the device is operated via touchscreen or physical controls, e.g. buttons, knobs, and sliders, 2) changes are visible via screen updates, and 3) the operations are initiated using hand/fingers. TutorialLens has two major modes: the authoring mode which allows authors to create AR user tutorials through narration and demonstration, and the access mode which guides novice device users to access unfamiliar interfaces with the created tutorials. In the authoring mode, after a few setup steps to configure the device control panel and finger markers, tutorial authors are first guided to demonstrate tasks step-by-step while verbally explaining what they are doing. During the demonstration, their hand movements and gestures are captured and recorded through 3D finger location tracking. Their narration are being transcribed in real time using automatic speech recognition. Then at the end of each step, they are asked to capture an update on the interface (e.g., changes on the display). After demonstrating and capturing all the steps, authors can review and edit descriptions of each step in a summary table, which concludes the authoring process. In the access mode, novice users are guided by TutorialLens to complete the demonstrated tasks. To start the guidance, novice users first point their phone cameras to the display screen of the device. Once the current step is identified by TutorialLens, users receive AR visual guidance of 3D finger movements overlaid on the device in the AR scene to complete the step. TutorialLens recognizes the current task progress by matching and identifying the previously captured interface changes within the camera's field of view, and retrieves the corresponding finger movements during demonstration. With the AR guidance, semantic text instructions, and audio feedback, TutorialLens guides a novice user to complete the demonstrated tasks step-by-step.

We conducted a user study to understand the effectiveness of TutorialLens, and showed that TutorialLens can effectively guide authors without AR development experiences to create usable tutorials to help novice device users complete tasks on a variety of devices. TutorialLens is friendly to authors without AR development experience, allows easy editing of tutorials, and is applicable to a wide variety of devices and tasks.

2 RELATED WORK

2.1 Video and Step-by-Step Tutorial Systems

Prior work has utilized various techniques and strategies in authoring image- and video-based tutorials to automate the authoring process and improve user experience. Pause-and-Play proposed by Pongnumkul et al. [29] reduces the burden of users having to manually pause and play tutorials by detecting important events in videos and linking them to the target application. DemoCut proposed by Chi et al. [7] minimizes video creators' post-processing by automatically organizing videos into meaningful segments and applying appropriate video editing effects. QuickCut by Truong et al. [34] allows creators to efficiently edit narrated videos by aligning transcripts with the raw footage and applying dynamic programming techniques to choose the ideal cut point. While such systems make video tutorials easier for novices to create, they still require frequent context switching between the device playing the video and the device a user is trying to interact with. Instead, our work lets novices create AR tutorials that are anchored to task steps and physical locations to show relevant instructions in context.

Step-by-step tutorial systems have been shown to be effective for complex user interfaces and complicated tasks, especially for users who do not prefer trial and error [22]. Research has been done in creating step-by-step tutorial systems, such as MixT by Chi et al. [6] which segments screen capture videos, applies video compositing techniques and highlights interactions through mouse trails. AniCode by Wang et al. [35] generates video tutorials in consumers' visual content based on authors' prior encoding of video animation. StateLens by Guo et al. [15] uses a state diagram to represent a user interface and to guide visually-impaired users to interact with touchscreens step-by-step, and their work inspired the modeling of user interactions into action sequences in this work. However, these systems are purely screen-based and provide instructions in 2D. In contrast, our work provides rich contextual information in the 3D space and supports a variety of user interactions such as pressing buttons, swiping or sliding, turning knobs, and potentially grabbing physical objects and more complicated gestures.

2.2 AR Tutorials

In recent years, AR has been widely adopted to scaffold interactions with interfaces and assist users with completing certain tasks. Blattgerste et al.'s work [3] compared conventional and AR instructions for manual assembly tasks and found that compared to using paper instructions, users made fewer errors with AR assistance. As AR tutorials provide more context compared to conventional instructions, AR tools have been developed to assist with a variety of tasks, including machine maintenance and repairing [17, 18], assembly tasks [16, 19, 32], health and medical assistance [9, 10], art and cultural experiences [1, 25, 30], etc. In these problem domains, AR is shown to be very effective as these tasks usually require lots of contextual information in a 3D space. This justifies our design decision of choosing AR over other types of guidance when helping users interact with devices with 3D components.

While most of these tools were shown to be effective in assisting with the specific tasks they were designed for, they each focused on a single application area, so developing such tools for a variety of purposes can take substantial time and effort. In response, Mohr et al.'s work [26] provided a more generalizable solution by tracking the tools people use and extracting tool paths and mapping to objects; however, it still required tools such as pens, markers, brushes to enable tracking on the tips of these tools. Chidambaram et al. [8] presented ProcessAR that creates in-situ procedural 2D/3D AR instructions through capturing subject-matter experts' environmental object interactions using an AR headset and controllers. Kong et al. [20] demonstrated AR tutorials to support older adults in using unfamiliar interfaces and used AR visual guidance overlaying on top of devices, which inspired the visual guidance in this work. In contrast to these prior work, our finger tracking approach does not require additional hardware other than a smartphone, and supports many hand-operated tasks. Our work supports AR tutorial creation for an increasingly common class of devices: those with a display screen that gives visual cues on device status change, e.g., printers, vending machines, and microwaves.

2.3 AR Authoring Tools

To make AR authoring easier for non-expert users, research has been done in creating general-purpose AR authoring tools for users without programming and AR development experiences, such as Seichter et al.'s ComposAR [31] and Gimeno et al.'s work [13] on easy-to-use AR authoring tools for industrial applications. Lots of prior research focused on lowering the barrier of AR authoring and prototyping by providing a predefined set of 3D objects for authors to choose from [2, 11, 21, 24], while others further lowers this barrier by utilizing various detection and tracking methods to automatically identify user actions. AuthAR proposed by Whitlock et al. [36] achieves this by tracking materials and screwdrivers in assembly tasks. DuploTrack by Gupta et al. [16] uses depth information for automatic detection of the assembly process. Prior work has also leveraged visual and depth information or a mix of them to track points of interactions [6, 12, 14, 27, 28, 32]. In addition to creating tools that enable AR authoring, prior work has also evaluated the effect of various factors during AR tutorial authoring, and proposed guidelines for selecting appropriate techniques and information-presentation methods in AR tutorial creation [33]. Compared to prior tools that track specific objects and are thus application-specific, our work enables a large class of tasks by tracking finger movements and customized objects with dense feature points. Additionally, rather than solely focusing on visual guidance, our work allows the creation of tutorials through author narration and demonstration, thus providing multi-modal guidance for novice users to complete tasks.

3 FORMATIVE STUDY

To identify the key challenges and needs in authoring user tutorials, we conducted a formative study consisting of two parts: a video coding analysis of 12 tutorial videos on YouTube, and a remote user study with 10 participants for observing and understanding how they create tutorial videos.

3.1 Video Coding Analysis

The goal of our video coding analysis was to understand how existing tutorials are structured, and to summarize the types of feedback and guidance used to convey a step in the tutorial.

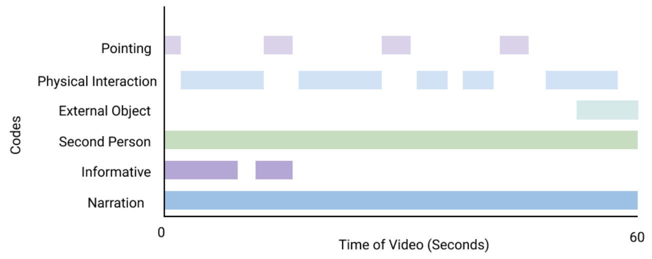

Figure 2: Visualization of an example coded video tutorial on how to use a parking meter kiosk. The pointing and physical interaction codes appear alternatively, the second person and narration codes appear from the beginning to the end, informative code appears at the beginning, and external object code appears towards the end.

We selected 12 tutorial videos by searching on YouTube, intentionally covering a wide variety of devices from public kiosks (e.g. parking meters, airport check-in) to home and workplace devices (e.g. coffee machines, smart fridges). We also chose videos long enough (average length ~5 mins) to have sufficient detail. Having a variety of devices and tutorials allowed us to have a comprehensive understanding of how a video tutorial might be made due to the complexity of the device. We marked the user interactions and device state transitions, as well as any additional editing done to the video. We also transcribed the videos and observed how authors demonstrated and described the process.

Through preliminary coding of the 12 videos as well as refining the codes, we iteratively created a code book summarizing the hierarchy of types of feedback on devices and given by creators of tutorials. The code book was organized into two sections: Analyzing what components of the *device* prompted a novice user to proceed to the next step, and analyzing what components the *creator* of the tutorial video did to guide a novice user in their tutorial video. We also identified seven major components in our code book that were shared between the device and creator parts: 3D object, text, audio, content, video type and interactions.

With the code book we created, we applied the codes to the video transcripts and observations, and then organized our findings into a few key insights included in Section 3.3. We have included our code book in the supplemental material, which contains more extensive information on the hierarchies we created for different types of feedback appearing in video tutorials. We have also included an example of a coded video using our code book in our supplemental materials, which we visualize in Figure 2.

3.2 User Study

Video coding analysis allowed us to identify how existing online tutorial videos are structured. However, we could not gain insights on the authors' experiences and perspectives on creating tutorial videos. For the second part of the formative study, we explored how people would create a "how-to" video tutorial from scratch, which gave us insights on opportunities to incorporate AR tools to be used in place of video tutorials.

We recruited a total of 10 participants through email solicitations and social media posts, and conducted the user study remotely via Zoom. During the study, we first asked participants to independently create a tutorial video on a device they had at home to walk an imagined novice user through a task. Once the tutorial video was created and uploaded, we used the code book to code the videos participants made. This allowed us to observe what participants did in their video, and to probe some questions about their process. After participants completed with creating their tutorial videos, we watched the tutorial videos with the participants and had them think aloud about their video. This allowed participants to review their video and answer questions about their experience while creating a tutorial as well as their needs in consuming tutorials. Each study session took about an hour and participant were compensated a 25 USD Amazon gift card.

3.3 Key Insights

We organized our video coding findings and responses from our user study using an affinity diagram, and extracted key insights below, which we used to inform the design of TutorialLens.

3.3.1 *Hands-Free Interactions.*

Participants mentioned that it was difficult to record with one hand. This was due to many factors such as trying to use a device requiring both hands with one hand, or making a full video in one take. To allow users to more easily interact with devices hands-free, a headset could be used so that users do not have to hold their phone with one hand. However, smartphones have their own advantages in that they are more commonly possessed. Additionally, we can easily incorporate multiple modalities of input and output in our design with smartphones, thus making it an overall better choice for the system.

To address the challenge of taking a video in one take, TutorialLens allows authors to create step-by-step instructions. Instead of playing a tutorial video from the beginning to the end, TutorialLens pauses between steps, so that in the end, novice device users can follow the guidance tutorial step-by-step.

3.3.2 *Providing Context.*

During our user study, participants felt the need to explain device features in depth when creating tutorials. However, during the interviews when participants discussed their needs in consuming tutorials, they mentioned that they often ignored such explanations, either because they felt they did not need it or the explanations did not seem quite relevant. Moreover, novice device users tended to care the most about efficiency and tried to go through the tutorials as quickly as possible, and even skipped parts to speed things up. Such a gap between the needs of tutorial authors and novice users inspired our step-by-step design of the system. In this way, authors are forced to create the tutorial step-by-step, only including information within the context of each step. Thus, created tutorials can provide relevant information to novice users through progressive disclosure.

3.3.3 *Being Clear and Concise with Instructions.*

Through our video coding analysis, we found that most of the video tutorials had verbal instructions. Comparing the two types of verbal instructions, we found that tutorial videos that were voiced over were more concise with instructions than the narrated tutorial videos. Meanwhile, participants mentioned that they wished they could go back and change their narration to make it more clear and concise. This inspired our design of incorporating speech recognition during authors' creation of tutorials, saving them as text descriptions, and then allowing authors to go back and edit the descriptions when they are done with the entire task.

3.3.4 Enabling Easy Editing. When finished with creating video tutorials, participants sometimes found their earlier demonstrations redundant, or contained mistakes that they wished to remove from the video tutorials they created. However, participants reported that they did not have sufficient video editing skills to efficiently remove the parts as they wished. Most tutorials did not have additional post-production editing and usually included the demonstrations for all the steps in one take. In our video coding analysis, tutorial videos for complex devices such as how to use a 3D printer also had a lot more components compared to a tutorial video for a parking meter or kiosk. To overcome this challenge, we designed TutorialLens to enable authors to easily edit tutorials on a smartphone. First, the system guides authors to demonstrate step-by-step, and allows them to confirm or redo at the end of each step; and second, when all steps are completed, authors are provided with a summary of steps in the created tutorial, and are allowed to further edit the description of each step. These designs allow authors to easily edit the created tutorials both during and after the creation process.

3.3.5 Showing Potential Errors and Failures. Participants also mentioned that they wanted to explain or show potential errors that might happen when using the device, for example, how to set up a camera to shoot an image with a blurred background, or how a device might react if things were measured or loaded incorrectly. Participants also wanted to show what would happen to the display screen if different settings were selected. Although we did not address this finding in the TutorialLens design, we will discuss potential future directions of merging different paths and supporting branching of tasks in our discussions in Section 6.4.

4 TUTORIALLENS

TutorialLens has two modes, the *authoring* mode and the *access* mode. In the *authoring* mode, a *tutorial author* is guided to add references that help the system understand the device interface, and then asked to specify a set of tasks and demonstrate each task step-by-step, for example, copying a document using a printer. TutorialLens generates a sequence of user actions for a task by processing the captured narration and demonstration in real-time and asking for the author's verification. Then in the *access* mode, a *novice user* selects which task they want to perform using the interface. TutorialLens then guides the user to interact with the interface step-by-step, using a combination of AR visual guidance as well as text and audio instructions.

4.1 Authoring Mode

In the authoring mode, the tutorial author is first guided through the setup phase as shown in Figure 1, screen 1 to 3. The author first configures the finger markers by taking a picture of each of their finger wearing the markers. When finished with finger marker configuration, the author takes a picture of the control panel of the device, to be used as a *device marker*, a concept we will revisit below. After that, the author takes a picture of the initial display screen.

Then the author enters the iterative authoring phase. From this point, the phone microphone starts to capture what the author says. When at least one of the author's fingers starts to appear in the camera view, TutorialLens starts to record the user interaction – movement of the author's detected fingers. It also records the screen to capture a video clip of the demonstration. When all fingers disappear from the camera view and do not appear within about 2 to 3 seconds, TutorialLens prompts to ask the author if they have completed this step and that the device display screen has been updated. The author can confirm step completion, keep demonstrating, or redo the current demonstration in case they have made some mistakes or are not satisfied with their demonstration. When step completion is confirmed, TutorialLens saves the recorded interaction, video clips, captured display screen status, and transcription of author narration of this step. After that, TutorialLens asks the author if they want to add another step to the tutorial, and repeats the process until the author responds "no", indicating that they have reached the end of the tutorial.

Then the author enters the finalization phase. The author is asked to take a picture of the final display screen, and then shown a summary table including the captured display screen image and transcription of the author's narration of each step. The author can edit the text transcriptions and watch the video clip of the captured demonstration of each step, or confirm to save the tutorial (including the image files, video clips, as well as a JSON file containing finger locations and transcription text of each step) locally. The author can also select tutorials in the main page to upload saved tutorials and share their tutorials with other people.

4.1.1 Modeling Tasks. TutorialLens models a task on a user interface as an *action sequence*. Each *action sequence* contains a sequence of *states* of the interfaces that can be uniquely identified by a corresponding *reference image*. A *reference image* is an image of part of an interface that changes while users interact with it, to give feedback on the current status of the device, for example, the digital display on a microwave that would change as a user presses a button. An *action* is required between two neighboring *states* in the sequence, and makes the transition from the previous state to the next. The *action* is usually certain user interaction with the interface, for example, pressing a button, opening or closing parts of the machine, etc.

4.1.2 Detecting User Actions. The user action detection mechanism of TutorialLens is based on the Apple ARKit framework, and specifically utilizes the image tracking functionality to locate authors' fingers in the 3D space. In order for ARKit to track fingers more accurately, authors need to wear *finger markers* (Figure 1 (1) and (2)). The system also uses speech-to-text in the Apple Speech framework to transcribe author narration, and uses the Apple ReplayKit framework to record the screen while capturing user actions.

TutorialLens is able to track all five fingers on one hand of an author. For simplicity in our demonstration, we will focus on using the index finger and the thumb. Whenever any of the author's configured finger markers starts to appear in the camera view (meaning that they starts with the user step demonstration), the system starts to record the 3D finger location of any finger in view. When all fingers disappear from the camera view, the system prompts to confirm with the author if this step has been completed.

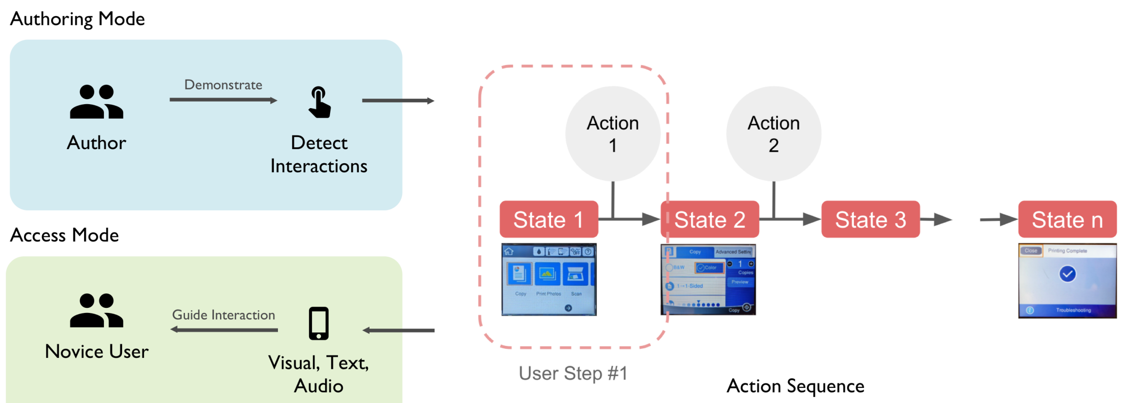

Figure 3: System Design: In the *authoring* mode, an *author* is guided to demonstrate tasks step-by-step. The system captures the author's narration and demonstration for each step, and generates an action sequence containing the status of device display screen, user actions, and text description of that step. In the access mode, a novice user follows the AR guidance and text and audio instructions to access the interface and complete the task.

TutorialLens also applies smoothing to user action detection in order to increase its robustness against temporary loss of tracking. When no finger is found in the camera view, the system does not immediately prompt to ask if the step is completed, but instead waits for 150 frames (about 2 to 3 seconds). When no finger is found throughout the 150 frames, the system considers it as step completion.

4.1.3 *Using Device Markers for Better Detection and Tracking.*

As ARKit is mostly designed for room scale tracking using feature points on large surfaces, it does not work well when detecting the 3D finger locations in our use case. To make user action detection more accurate and robust, we use relative 3D finger locations to a *device marker* instead of using absolute coordinates. We define a *device marker* to be a static location on the interface, for example, the machine logo, the control panel, etc. A device marker is expected to be within the same camera view as the demonstrated user actions, such as pressing a button; it also needs to be trackable, which means it has enough feature points and is large enough for ARKit to recognize. Before an author demonstrates a task, TutorialLens allows them to add customized device markers, so that the system can record the 3D finger locations relative to the device markers during demonstration.

For the i-th frame in the user interaction, we define

$$F_i = (F_{finger_1,i}, F_{finger_2,i}, \ldots)$$

and for each finger

$$F_{finger_j,i} = (x_{finger_j,i}, y_{finger_j,i}, z_{finger_j,i})$$

to be the 3D coordinates of user fingers relative to the device marker.

$$D_i = (x_{device,i}, y_{device,i}, z_{device,i})$$

is the 3D coordinates of the device marker in this frame. Note that for each F_i, a maximum of five fingers can be recorded, while not all finger locations are required. As long as one finger appears in the view, this finger location is recorded. By default, ARKit captures the 3D coordinates of fingers in the absolute coordinate system from camera view, say

$$F'_{finger_j,i} = (x'_{finger_j,i}, y'_{finger_j,i}, z'_{finger_j,i})$$

Thus, the system computes the relative locations of user fingers to the device markers as

$$F_{finger_j,i} = F'_{finger_j,i} - D_i$$

The system can then reproduce the finger locations in the *access* mode by reversing the computation. Note that in order to record the relative 3D finger locations, there must be at least one device marker available in the camera view. This means if all device markers are blocked or out of view, the finger locations would not be recorded.

TutorialLens also uses reference images added by users as additional device markers, as they have "semi-static" locations on the device within the time frame of an individual user step. Thus, multiple device markers would be available during user demonstration. This allows the system to choose whichever device marker available when calculating finger positions. This is especially helpful when some device marker is either out of view or blocked by users' hands during demonstration. Later in our design iterations (Section 4.3.5), we demonstrate that using device markers while recording finger locations and having multiple device markers to choose from substantially increase the accuracy of user action detection and reproduction.

4.2 Access Mode

In the access mode, a *novice user* can select from a list of available tutorials on the main page. TutorialLens starts by asking the novice user to point their camera to the display screen of the device, in order to determine the current step and retrieve the corresponding guidance. This process will be discussed in Section 4.2.1 in more detail. Whenever it recognizes the current user step, it informs the novice user of successful recognition, retrieves the 3D finger movements, and displays the AR visual guidance. TutorialLens also retrieves the description text of the current step, displays it as text instruction on the top of the screen, and announces it aloud to the novice user.

When the novice user completes a step, the display screen of the device updates. When this change is captured through the camera view, TutorialLens knows that the user has completed this step and automatically proceeds to the next step. However, if TutorialLens

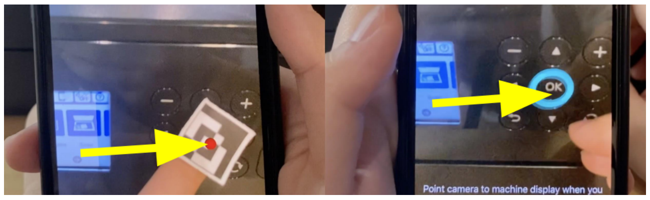

Figure 4: TutorialLens records the 3D coordinates of the finger locations relative to the device marker in the authoring mode, and reproduces the finger location corresponding to the current device marker in the access mode.

fails to automatically recognize display screen updates, the novice user can always force it to proceed to the next step by clicking on the right arrow at the bottom of the page. This process is repeated when the final state of the display screen is recognized.

4.2.1 Determining Current User Step. TutorialLens identifies the current user step by searching in existing reference images saved for this task. As the reference images have 1-to-1 mappings with all the steps of the sequence of actions, there should always exist at most one reference image in the camera view. For example, whenever the system recognized the home screen of the digital display of the printer, it knows that the user is in the first step of a task. Thus, the system knows which step a novice user is on, and provides corresponding guidance for the novice user to complete the step.

As a novice user completes a step, the display screen updates and a new reference image appears in the camera view. In this way, TutorialLens keeps proceeding to the next state of the action sequence until it reaches the final state.

4.2.2 Providing Guidance. After the system determines which step a novice user is currently on, it retrieves the corresponding sequence of finger locations of this step, and displays each finger as a colored ring in the AR scene to demonstrate to the novice user on how to move their fingers.

To do this, TutorialLens takes the recorded sequence of finger movements, and looks for the device marker used for each frame. If this device marker does not appear in the camera view, it displays an image of the part of the device to point to, and asks the novice user to move their phone camera to this part of the device. Once that device marker is detected, the system reproduces the absolute finger location in the current camera view using the finger coordinates relative to the device marker, by computing

$$F'_{finger_j,i} = D'_i + F_{finger_j,i}$$

where D'_i is the 3D coordinates of the device marker in the current camera view, and $F_{finger_j,i}$ is the stored relative finger location in the sequence.

TutorialLens might also play the video clip of the step demonstration as a secondary option when AR visual guidance of finger movements is not available. This happens when no finger movement has been recorded, usually when device markers cannot be found during the author's demonstration. While displaying the AR visual guidance or playing the video demonstration, TutorialLens also retrieves the saved text transcription of the author's description of the step, and displays it as text instructions to the novice user and announces the text using text to speech.

4.3 Design Iterations

4.3.1 End-User AR Tutorial with Pre-Specified Action Sequence. In the early stages of the project, we brainstormed possible uses of AR in assisting users with understanding and interacting with unfamiliar interfaces. We categorized user interaction with interfaces into three different major types: buttons and toggles, touchscreen gestures, and interactions involving real physical objects such as inserting bills or swiping credit cards. We designed different visual indicators for these different types of interactions, and created an example AR tutorial for a printer interface with pre-specified action sequence: select the copy option from home page, opening the top cover, placing the document to copy, closing the top cover and press the start button, and going back to the home page once the printing job is completed. Each of these steps was uniquely identified by a reference image of the digital display or some other part of the printer, and the device location was identified by a pre-specified device marker – the printer logo on the control panel.

4.3.2 Preliminary User Study. With our end-user AR tutorial with a pre-specified action sequence, we ran a preliminary user study with 3 users who had never used the printer interface before. We asked each user to complete the task of copying a document, by following the AR visual indicators as well as text and audio instructions. During the process, we asked the users to "think aloud" and verbalize their thought process.

From our preliminary user study, we found that users got confused by some of the visual indicators. More specifically, since some of the steps involved opening and closing parts of the printer, our prototype displayed a 3D simulation of the expected movement of these parts in the AR scene, for example, when the paper tray needed to be pulled out, our prototype displayed an animated 3D box simulating the tray moving out from the printer. However, users indicated that they did not know which exact part of the printer to interact with.

4.3.3 Using Simulated Finger Movements as Visual Indicators. Although our initial prototype received overall positive feedback, it was developed based on a fixed task with the action sequence, reference images and corresponding visual indicators inputted manually. The prototype was also developed specifically for the printer interface, thus the visual indicators in the prototype could hardly be authored in a generalizable way.

Because of this, we needed to re-design the visual indicators so that they could be automatically generated given user input of the actions. As we hoped our authoring tool would work with both physical interfaces and touchscreen interfaces, as well as both flat interfaces and 3D devices, designing visual indicators based on device parts seemed quite challenging as their shape and position vary a lot across different devices.

However, one thing in common for these different types of interactions is that all of them require a user's hand in the process. Thus, instead of displaying the movement of parts of the device, we decided to record the finger movements of an experienced user interacting with the device, and then replay the finger movements to guide a novice user to use the device.

4.3.4 Prototyping Finger Tracking. We experimented with multiple different finger tracking methods, such as detecting convex hull

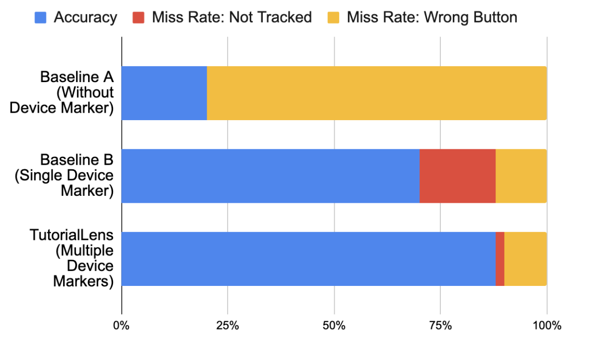

Figure 5: Accuracies and miss rates for the three approaches: (1) Baseline A (without device marker), (2) Baseline B (single device marker), and (3) TutorialLens (multiple device markers). Two types of errors occurred: (1) no device marker available thus finger is not tracked, and (2) AR visual guidance appearing on the wrong button. Note that error type (1) does not apply to Baseline A, as it does not use device markers.

using OpenCV [4], hand detection using OpenPose [5] and MediaPipe [23], or transfer learning on top of trained object detection models. However, these methods had critical drawbacks that made them not ideal for our use case, including (1) only providing 2D coordinates by processing the 2D camera view; (2) requiring the entire hand to be in the camera view to detect the finger junctions; (3) relying heavily on colors or shapes of fingers and having lots of false positives.

We then experimented with taking pictures of bare fingers, fingers with nail polish, and finger markers using QR codes and QR code mask patterns. Finally, we decided to go with 6×6 QR code mask patterns as finger markers that could provides enough feature points for image tracking, and meanwhile could be clearly seen by a phone camera during demonstration.

4.3.5 Using Multiple Device Markers. While experimenting with a variety of devices using our prototype, we found that a user's fingers would sometimes block the single device marker during demonstration. In this situation, the system could not accurately track the user's fingers and display the correct visual indicators. To see how much the system performance was improved with our multiple-device-marker approach, we created a mock-up user interface (containing a QR-code mask pattern that simulated the display screen, and a 4×4 grid of fake buttons that simulated the device control panel) and displayed it on a laptop display screen to make it of similar size as most home devices we tested during the user study (Figure 6). We then tested TutorialLens with no device marker, with a single device marker, and with multiple device markers. The multi-device-marker approach turned out to be very effective, achieving an 88% accuracy while reproducing the detected "button presses", compared to the 20% accuracy with no device marker and the 70% accuracy with a single device marker (Figure 5).

5 USER EVALUATION

We conducted a user study to evaluate (1) how well TutorialLens works in enabling experienced users to create AR tutorials, and (2) how usable the created AR tutorials are in helping novice users interact with unfamiliar interfaces.

5.1 Apparatus and Participants

The TutorialLens app ran on iOS with versions higher than 13.0, and was distributed to participants through TestFlight invitation. Participants were asked to either print out or draw out the finger markers for tracking and wear them on their thumb and index fingertips. Participants also used laptops or tablets of their choice to connect with the study coordinator remotely via Zoom meetings.

We recruited 7 pairs of (a total of 14) participants (separately from our formative user study) through email solicitations and social media posts, with each pair of participants living in the same household. Participants of our study had a variety in levels of experience with AR from little or no experience to lots of experience, and the majority of our participants had little or no experience with AR development. Participant demographics are presented in Table 1.

Due to COVID-19, it was not safe to ask participants to use public devices or come to lab space. Thus, the devices used in our studies were selected based on availability in participants' households. Each pair of participants were asked to choose a device in their household that had a display screen and a control interface. The device choices of each pair of participants were also included in Table 1. This restricts the types of devices we could use and thus the complexity of tasks was beyond our control. This makes our user study extra challenging because our app had to work in the wild.

5.2 Procedure

All study sessions were conducted remotely through Zoom and upon IRB approval of our institution. Each pair of participants in the same household were connected with the study coordinator in a Zoom meeting, during which participants pointed their webcam to the device they used for the tasks and followed instructions of the study coordinator. At the beginning of each study session, demographic information was collected and participants were asked to choose a device that had a display screen and a control interface, and to choose a task on the device that included as many steps as possible and ideally with interactions other than button presses. Participants also decided their roles as the tutorial *author* and the *novice user* based on their familiarity with the device of their choice.

Each study session consisted of two parts. In the first part, the *author* first completed a training session of creating a one-step tutorial for a simple task (such as pressing a button) on the device they chose. Then the *author* was asked to create a longer tutorial on a multi-step task on the same device. In the second part, the *novice user* was asked to follow the AR tutorial created in the first part of the study to complete the same multi-step task on the device.

All study sessions were recorded via Zoom upon participants' consent. We recorded the time of completion for creating the tutorial, and for finishing the task using the tutorial. We also asked participants to give ratings on a few statements on a Likert scale from 1 to 7 (1 being strongly disagree and 7 being strongly agree) regarding their overall experience and on each design component. Each study session took about 1.5 hours, and participants were compensated a 25 USD Amazon gift card.

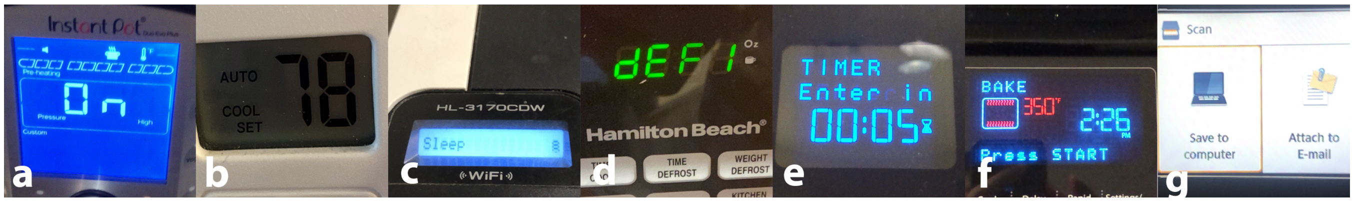

Figure 6: Example images of different device display screens taken by users during user studies: (a) instant pot, (b) AC, (c) printer, (d) microwave, (e) microwave, (f) oven, (g) printer.

Table 1: Participant demographics and devices used during user study.

Participant Group	Device	Author Age	Author Gender	Author Experience with AR	Author Experience with AR Dev	Novice User Age	Novice User Gender	Novice User Experience with AR	Novice User Experience with AR Dev
1	instant pot	29	male	some	little or no	23	female	some	little or no
2	AC	23	female	little or no	little or no	23	male	little or no	little or no
3	printer	18	female	some	some	20	female	some	little or no
4	microwave	26	male	lots of	some	27	female	little or no	little or no
5	microwave	23	female	little or no	little or no	26	female	little or no	little or no
6	oven	27	male	little or no	little or no	24	female	lots of	some
7	printer	23	female	some	little or no	23	male	little or no	little or no

5.3 Results

In summary, all authors were able to successfully create interactive AR tutorials with the TutorialLens iOS app, and all novice users were able to complete the specified task with the guidance. Authors spent an average of 53.5 seconds ($SD = 29.72$) per step while creating tutorials. The longest one took 463 seconds (about 8 minutes) for a 4-step tutorial, while the shortest took 90 seconds for a 4-step tutorial. Novice users spent an average of 27.32 seconds ($SD = 26$) per step while using the created tutorials to complete tasks on the devices. The longest took 258 seconds for a 4-step tutorial, while the shortest took 23 seconds for a 4-step tutorial. Note that during the process, some novice users had to press the next button to proceed to the next step as the display screen update could not be recognized. This might be due to the varied lighting conditions in the study environments, which made image recognition challenging. Another cause of recognition failure was that the starting state of some devices included the clock time (for example, default screen display of microwaves or ovens), thus the starting state of display screen would be different from the saved reference image when a novice user was trying to complete the task.

For subjective ratings, authors found the TutorialLens iOS app easy to learn ($M = 5.43, SD = 1.62$), somewhat comfortable ($M = 4.86, SD = 1.68$), and accurately capturing their demonstration ($M = 5.57, SD = 2.15$). More specifically, authors found the finger tracking somewhat accurate ($M = 4.83, SD = 1.94$), and user step detection very responsive and accurate ($M = 6, SD = 1.15$). Participants reported that the discomfort was mainly due to wearing finger markers, and positioning the finger markers in specific angles to be tracked. Some participants also reported that they felt less need of TutorialLens for creating tutorials of simple tasks on home devices, while they believed that it could be quite useful when the tasks and devices became a lot more complicated.

On the other hand, novice users found the created tutorials useful ($M = 5.14, SD = 1.35$), very comfortable to use ($M = 6.29, SD = 0.76$), and would be very willing to use tutorials created with TutorialLens for unfamiliar devices and tasks in the future ($M = 6.14, SD = 0.38$). Due to the restrictions in recruiting participants, some novice users in our user study were not completely unfamiliar with the devices. However, participants believed that TutorialLens could be especially helpful if the tasks were very complicated or if they were really unfamiliar with the devices.

Overall, most authors very much liked the step-by-step design of TutorialLens, and thought it was "very helpful" and "pretty natural." Participants felt like the design "helps you to get organized" and "forces you to do step-by-step." Meanwhile, novice users overall liked the guidance given by TutorialLens, and were very positive about the potential use of TutorialLens. Participants found the created tutorials "pretty intuitive" and "really easy to use." They also liked the feedback given by TutorialLens upon detection of step completion when there was an update on the display screen, which gave them a feeling that "it kinda know when I'm done with my step."

Another interesting finding during our user study was that, although some authors did not feel much of a need of TutorialLens when they were first done creating the tutorials, they were surprised and expressed more confidence in TutorialLens when they learned about the very positive feedback from novice users on the tutorials they created.

6 DISCUSSION AND FUTURE WORK

6.1 Automating AR Authoring

Most authoring tools for AR and for tutorial systems require lots of manual user input, and sometimes require users to separate the steps of a task at their own discretion. TutorialLens automates the authoring process by tracking authors' finger locations, thus "guessing" the completion of a step. This minimizes the amount of manual user input, and makes the tutorial creation process closest to the natural interaction of some experienced user showing some novice user how to use a device – through narration and demonstration. The created AR tutorials are also automated in that they do not require novice users to click on next buttons to proceed, but instead recognize visual updates on devices and automatically retrieve the corresponding guidance to provide for novice users.

6.2 Generalizability

TutorialLens is generalizable to a variety of devices and tasks. As long as the device gives visual feedback to users during interactions, authors can capture these visual updates when creating tutorials. Thus, TutorialLens could potentially be applicable to a wide range of devices, from public kiosks such as subway ticket machines and gas station kiosks, to workplace and home appliances such as printers, microwaves, ovens, etc.

TutorialLens can also support a wide range of gestures during demonstration. As it tracks different fingers on one hand of the author, it captures the overall hand movements in 3D as well as the relative movements of the author's fingers to each other. Thus, a variety of gestures could be captured, including pressing (tapping) buttons, swiping or sliding, turning knobs, grabbing items, etc.

A limitation of our study is that we only evaluated TutorialLens on relatively simple tasks involving button pressing and knob turning (P1, instant pot). Future work might investigate extending TutorialLens for more complicated tasks, e.g., those that require complex hand movement and 3D understanding of the space. To visualize complex hand movement, TutorialLens could reconstruct a model of the hand to show more subtle and delicate movements. To adapt to complex spatially distributed tasks, leveraging additional types of markers and 3D vision techniques could help extend TutorialLens' ability of task and activity recognition.

6.3 Multi-Modal Guidance

We have received positive comments on the multi-modal feedback of TutorialLens – supporting AR visual guidance with text instructions and audio feedback in the access mode. As they provided auxiliary instructions to support the AR visual guidance, users found these feedback useful in helping them understand the action to take in each step. Multi-modal feedback especially helps when finger movements are too vague for a novice user to identify what action to take with the interface, or when an author's fingers move too far away from the device markers, and thus can no longer be tracked by the system.

6.4 Supporting Tasks with Multiple Branches

One key insight from our formative study that we have not addressed in the TutorialLens design is the need to show potential errors and failures to users. The current TutorialLens design follows a sequential modeling of tasks, and authors can add multiple tutorials for different tasks on the same device, while we notice that these tasks might overlap with each other in some steps. Future work might consider refining the sequential modeling of tasks to support branching of user steps to combine different tasks on a device, so that authors can create a single comprehensive tutorial for a device as well as including possible failure paths for novice users to look at.

6.5 Limitation

The current finger tracking method in our prototype requires paper labels to be attached to user fingers. As mentioned above in our user study results (Section 5.3), these labels can negatively affect the user experience, and make it not as natural as the user demonstrating the interactions with their bare fingers. However, we believe that this issue could be resolved as hand and finger tracking algorithms develop and more features are supported in AR development platforms in the future.

Another limitation of TutorialLens is that it requires the device markers to be relatively close to the points of interactions on every step. For example, when an author's fingers move too far away from the display screen and control panel during demonstration, most of the finger movements will not be captured by the camera. In our prototype, such limitation is partly made up by the multi-modal feedback in the access mode – the user can still follow the text and audio instructions when they do not see AR simulations of finger movements on the screen.

TutorialLens also requires a non-subtle update on the display screen of the device after each step, otherwise it would not be able to identify current steps and thus provide guidance for novice users. This might narrow the types of devices TutorialLens can be applicable to, or we might have to make the trade-off between the level of automation in authoring and the range of devices the system supports. Future work might investigate multi-modal recognition during authoring to expand the scope of device changes the system can capture. Aside from recognizing display screen updates of each step, we could potentially determine the current step a user is on by capturing audio or other forms of feedback from the device, such as a beeping sound after clicking a button.

7 CONCLUSION

We have presented TutorialLens, a system for authoring interactive AR tutorials through narration and demonstration. TutorialLens allows authors without experience with AR development to easily create interactive AR tutorials of devices step-by-step by narration and demonstration, and then provides AR visual guidance with supporting text and audio feedback to guide novice device users to complete tasks using the created tutorials. TutorialLens automatically captures authors' interactions and generates action sequences for the tutorials, and allows easy editing while creating the tutorials. It also automatically recognizes current user step and provides contextual AR visual guidance and multi-modal feedback for novice users to complete tasks on unfamiliar devices. Our formative study identified the key challenges and user needs which informed many design components of the TutorialLens system. Our user evaluation demonstrated that TutorialLens could effectively guide authors to create usable AR tutorials for novice users. TutorialLens is friendly to authors without AR development experience, allows easy editing, and can be applicable to a variety of devices and tasks.

REFERENCES

[1] Nagore Barrena, Andrés Navarro, Sara García, and David Oyarzun. 2016. CoolTour: VR and AR authoring tool to create cultural experiences. In *Intelligent Interactive Multimedia Systems and Services 2016*. Springer, 483–489.

[2] Martin Bauer, Bernd Bruegge, Gudrun Klinker, Asa MacWilliams, Thomas Reicher, Stefan Riss, Christian Sandor, and Martin Wagner. 2001. Design of a component-based augmented reality framework. In *Proceedings IEEE and ACM International Symposium on Augmented Reality*. IEEE, 45–54.

[3] 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.

[4] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008* (2018).

[6] 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.

[7] Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Bjoern Hartmann. 2013. Democut: generating concise instructional videos for physical demonstrations. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. 141–150.

[8] 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 *Proceedings of the Designing Interactive Systems Conference*.

[9] Tom Djajadiningrat, Pei-Yin Chao, SeYoung Kim, Marleen Van Leengoed, and Jeroen Raijmakers. 2016. Mime: An AR-based System Helping Patients to Test their Blood at Home. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 347–359.

[10] Tom Djajadiningrat, Patray Lui, Pei-Yin Chao, and Christian Richard. 2016. Virtual Trainer: A Low Cost AR Simulation of a Sudden Cardiac Arrest Emergency. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 607–618.

[11] Ralf Dörner, Christian Geiger, Michael Haller, and Volker Paelke. 2003. Authoring mixed reality—a component and framework-based approach. In *Entertainment Computing*. Springer, 405–413.

[12] Markus Funk, Thomas Kosch, and Albrecht Schmidt. 2016. Interactive worker assistance: comparing the effects of in-situ projection, head-mounted displays, tablet, and paper instructions. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 934–939.

[13] Jesús Gimeno, Pedro Morillo, Juan Manuel Orduña, and Marcos Fernández. 2013. An easy-to-use ar authoring tool for industrial applications. In *Computer Vision, Imaging and Computer Graphics. Theory and Application*. Springer, 17–32.

[14] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*. 143–152.

[15] 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.

[16] 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.

[17] Pavel Gurevich, Joel Lanir, Benjamin Cohen, and Ran Stone. 2012. TeleAdvisor: a versatile augmented reality tool for remote assistance. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 619–622.

[18] Steven Henderson and Steven Feiner. 2010. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics* 17, 10 (2010), 1355–1368.

[19] 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.

[20] Junhan Kong, Anhong Guo, and Jeffrey P Bigham. 2019. Supporting Older Adults in Using Complex User Interfaces with Augmented Reality. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*. 661–663.

[21] Taehee Lee and Tobias Hollerer. 2008. Hybrid feature tracking and user interaction for markerless augmented reality. In *2008 IEEE Virtual Reality Conference*. IEEE, 145–152.

[22] Rock Leung, Charlotte Tang, Shathel Haddad, Joanna Mcgrenere, Peter Graf, and Vilia Ingriany. 2012. How older adults learn to use mobile devices: Survey and field investigations. *ACM Transactions on Accessible Computing (TACCESS)* 4, 3 (2012), 1–33.

[23] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. 2019. MediaPipe: A Framework for Building Perception Pipelines. *arXiv preprint arXiv:1906.08172* (2019).

[24] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. 197–206.

[25] Nikoletta Mavrogeorgi, Stefanos Koutsoutos, Angelos Yannopoulos, Theodora Varvarigou, and George Kambourakis. 2009. Cultural heritage experience with virtual reality according to user preferences. In *2009 Second International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services*. IEEE, 13–18.

[26] 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 ACM CHI Conference on Human Factors in Computing Systems*. 6547–6558.

[27] Tam V Nguyen, Bilal Mirza, Dorothy Tan, and Jose Sepulveda. 2018. ASMIM: Augmented Reality Authoring System for Mobile Interactive Manuals. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*. 1–6.

[28] 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.

[29] 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.

[30] Dariusz Rumiński and Krzysztof Walczak. 2013. Creation of interactive ar content on mobile devices. In *International Conference on Business Information Systems*. Springer, 258–269.

[31] Hartmut Seichter, Julian Looser, and Mark Billinghurst. 2008. ComposAR: An intuitive tool for authoring AR applications. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, 177–178.

[32] Anna Syberfeldt, Oscar Danielsson, Magnus Holm, and Lihui Wang. 2015. Visual assembling guidance using augmented reality. *Procedia Manufacturing* 1 (2015), 98–109.

[33] Keishi Tainaka, Yuichiro Fujimoto, Masayuki Kanbara, Hirokazu Kato, Atsunori Moteki, Kensuke Kuraki, Kazuki Osamura, Toshiyuki Yoshitake, and Toshiyuki Fukuoka. 2020. Guideline and Tool for Designing an Assembly Task Support System Using Augmented Reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 486–497.

[34] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. Quickcut: An interactive tool for editing narrated video. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 497–507.

[35] Zeyu Wang, Shiyu Qiu, Qingyang Chen, Natallia Trayan, Alexander Ringlein, Julie Dorsey, and Holly Rushmeier. 2019. AniCode: authoring coded artifacts for network-free personalized animations. *The Visual Computer* 35, 6 (2019), 885–897.

[36] Matt Whitlock, George Fitzmaurice, Tovi Grossman, and Justin Matejka. 2019. AuthAR: Concurrent Authoring of Tutorials for AR Assembly Guidance. (2019).