SmartRecorder: An IMU-based Video Tutorial Creation by Demonstration System for Smartphone Interaction Tasks

Xiaozhu Hu
xhubk@connect.ust.hk
Department of Computer Science and Technology, Tsinghua University & Division of Integrative Systems and Design, Hong Kong University of Science and Technology
Sai Kung, Hong Kong, China

Yanwen Huang
hyw0058@gmail.com
Department of Computer Science and Technology, Tsinghua University
Haidian, Beijing, China

Bo Liu
boliu97@uw.edu
Global Innovation Exchange, University of Washington
Bellevue, WA, USA

Ruolan Wu
wurl21@mails.tsinghua.edu.c
Department of Computer Science and Technology, Tsinghua University
Haidian, Beijing, China

Yongquan Hu
yongquan.hu@unsw.edu.au
Department of Computer Science and Engineering, University of New South Wales
Kensington NSW, Sydney, Australia

Aaron Quigley
hos_reverse_these_cse@unsw.edu.au
Department of Computer Science and Engineering, University of New South Wales
Kensington NSW, Sydney, Australia

Mingming Fan
mingmingfan@ust.hk
Computational Media and Arts Thrust, Hong Kong University of Science and Technology (Guangzhou) & Division of Integrative Systems and Design, Hong Kong University of Science and Technology
Guangzhou, Guangdong, China

Chun Yu*
chunyu@tsinghua.edu.cn
Department of Computer Science and Technology, Tsinghua University
Haidian, Beijing, China

Yuanchun Shi
shiyc@tsinghua.edu.cn
Department of Computer Science and Technology, Tsinghua University & Qinghai University
Haidian, Beijing, China

ABSTRACT

This work focuses on an active topic in the HCI community, namely tutorial creation by demonstration. We present a novel tool named SmartRecorder that facilitates people, without video editing skills, creating video tutorials for smartphone interaction tasks. As automatic interaction trace extraction is a key component to tutorial generation, we seek to tackle the challenges of automatically extracting user interaction traces on smartphones from screencasts. Uniquely, with respect to prior research in this field, we combine computer vision techniques with IMU-based sensing algorithms, and the technical evaluation results show the importance of smartphone IMU data in improving system performance. With the extracted key information of each step, SmartRecorder generates instructional content initially and provides tutorial creators with a tutorial refinement editor designed based on a high recall (99.38%) of key steps to revise the initial instructional content. Finally, SmartRecorder generates video tutorials based on refined instructional content. The results of the user study demonstrate that SmartRecorder allows non-experts to create smartphone usage video tutorials with less time and higher satisfaction from recipients.

CCS CONCEPTS

• **Human-centered computing** → **User interface toolkits**.

KEYWORDS

Tutorial Creation by Demonstration, IMU-based Interaction Trace Extraction, Non-experts, Smartphone Usage Video Tutorial

*Corresponding authors

IUI '23, March 27–31, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0106-1/23/03…$15.00
https://doi.org/10.1145/3581641.3584069

ACM Reference Format:
Xiaozhu Hu, Yanwen Huang, Bo Liu, Ruolan Wu, Yongquan Hu, Aaron Quigley, Mingming Fan, Chun Yu, and Yuanchun Shi. 2023. SmartRecorder: An IMU-based Video Tutorial Creation by Demonstration System for Smartphone Interaction Tasks. In *28th International Conference on Intelligent User Interfaces (IUI '23), March 27–31, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3581641.3584069

1 INTRODUCTION

Video tutorials for smartphone interaction tasks, abundant on websites like Youtube, are essential to help users tackle the challenges that multiple smartphone functions bring. However, creating video tutorials requires video editing skills, which challenges non-experts [7]. Furthermore, making tutorials manually is also time-consuming because of the variety of smartphone interfaces and interaction tasks. We seek to design an automatic system with minimal human intervention to lower the human effort when creating video tutorials about smartphone interaction tasks so that people without video editing skills can still make smartphone usage video tutorials easily and efficiently.

Creating video tutorials with professional software (e.g., Adobe Premiere, iMovie) is challenging for non-experts [6, 7, 21]. To lower the efforts for tutorial creation, some related works (e.g., MixT [10]) focus on extracting user steps from screencasts and generating instructional content automatically to liberate tutorial creators from editing instructional content. However, existing methods for extracting interaction traces have limitations that restrict their practical use on smartphones. For example, some works obtain touch events from touchscreen capacity signal [41] or AccesssibilityNodes of Android Application Framework [39] on a smartphone so that the explicit interaction events obtained from the operating system can assist screencast analysis and tutorial generation. However, obtaining capacity signals is impossible for off-the-shelf devices. Granting the rights for Accessibility API can take a few steps and brings extra trouble to tutorial creators. Additionally, Accessibility API allows the third party to inject actions to execute auto-clicks or other simulated gestures to control the users' device, which brings high privacy risks to users. We argue that using detailed data from the operating system framework is not the only way to accomplish automatic user step extraction. For vision-based methods towards extracting user actions from video recordings, existing approaches rely on tracking or detecting salient features in the video like mouse cursor [1, 10, 27], touch indicator [4, 5], and users' hands or fingers [11]. However, these features are not standard in smartphone screencasts. Using pure vision-based methods to deal with screencasts without salient features is challenging because it is hard to distinguish which screen animation is caused by users and which is caused by the operating system (e.g., pop-up advertisements and notifications).

To overcome the above challenges, we leverage smartphone IMU sensors, which are pervasive and accessible on off-the-shelf smartphones. This idea is enlightened by TapNet [18], which shows the potential of using smartphone IMU to accomplish off-screen taping input. In addition, obtaining signals from IMU sensors is a low-risk activity and requires no extra permission, which is safer and more convenient for users. We extend the method and scenario of TapNet [18]. In this work, we collect new dataset of IMU signals and build CNN models to recognize the touch position and multiple gesture types. Moreover, we combine the screencast of an interaction task with the data from the smartphone IMU sensors to improve the system performance of extracting interaction trace on the smartphone, which facilitates instructional content generation.

Based on this technique, we propose SmartRecorder, a novel tool that generates video tutorials of smartphone tasks with machine intelligence and minimal human involvement. It consists of two modules in the front end: Screencast Module and Tutorial Refinement Editor, and two in the back end: Input Step Analyzer and Tutorial Generator. The Screencast Module collects the screencast and the IMU data in the demonstration process. With computer vision and sensing algorithms, Input Step Analyzer analyzes the collected data to extract user actions automatically and calculates the tutorial metadata (keyframe, gesture, touch position, and instructional text) of each user input step. SmartRecorder also offers tutorial creators a simple editor on smartphones to refine the automatically generated tutorial metadata easily and efficiently. The tutorial Generator of SmartRecorder can generate the video tutorial of the target task according to the tutorial metadata. SmartRecorder reduces human work (by involving machine automation) and ensures the correctness of the tutorial content with the Tutorial Refinement Editor.

We evaluated the efficiency and effectiveness of SmartRecorder through a 2-phase user study. In phase 1, we compared the efficiency of creating video tutorials with and without using SmartRecorder. The results show that SmartRecorder facilitates tutorial creators to make a video tutorial efficiently. The time cost of human editing is reduced by 75.62%. Subjective feedback shows tutorial creators' satisfaction and willingness to use SmartRecorder. In phase 2, we recruited 18 participants to evaluate the guiding effectiveness of the tutorial generated by SmartRecorder. The quantitative results show that with the video tutorials created by SmartRecorder, participants had significantly higher task completion rates and shorter completion time when completing the tasks. Tutorials created by SmartRecorder also received higher user satisfaction than manually produced tutorials.

We make the following three contributions in the work. 1) We propose the IMU-based technique to extract user interaction traces from the screencast of smartphone interaction tasks, which addresses the challenges of existing methods. 2) We apply the above technique to real use and develop SmartRecorder, an IMU-based tutorial creation by demonstration system, to help non-experts create smartphone usage video tutorials with minimum human effort. 3) Our user studies show that SmartRecorder facilitates non-experts to create video tutorials easily and efficiently; moreover, these tutorials were able to help users complete tasks more effectively with higher user satisfaction compared to traditional methods.

2 RELATED WORKS

We discuss prior research from two aspects: extracting interaction traces from pre-recorded demonstrations and editing instructional content in video tutorials.

2.1 Extracting Interaction Traces from Pre-recorded Demonstrations

Manually extracting the step-by-step operating information from the demonstration and making a video tutorial requires high labor cost and time cost [22]. Therefore, technical assistance is needed to reduce the workforce. Extracting step-by-step information from pre-recorded demonstrations is the key to generating instructional content automatically and semi-automatically. In the domain of computer software, capturing demonstration workflow from software API is a classic method to extract step-by-step information

for tutorial generation [3, 10, 13, 15, 17, 25] and lead the interaction between learners and tutorial content [12, 16, 25, 31, 33, 36]. As obtaining the interaction log-data from multiple third-party mobile applications is impossible, such a method is not applicable on smartphones. There are also some computer vision (CV) techniques aiming at extracting user actions from demonstrative screencast [1, 4, 5, 10, 27, 42] or video recording of operation behaviors [11]. However, these techniques rely on tracking and detecting specific features in the recorded videos, such as mouse cursor [1, 10, 27, 42], touch indicator [4, 5] and users' fingers [11], while such features are not common in smartphone screencasts. Using pure vision-based methods to extract each step of user input from a smartphone screencast is quite challenging because it is difficult to identify which screen animation is caused by users and which is caused by the operating system. To tackle this challenge, some related works track the metadata of user input from touchscreen capacity signal [41], and Android debugging bridge (ADB) [40]. However, such methods require developer permissions and extra devices, which is impractical for regular users with off-the-shelf smartphones. Another approach is to leverage the Accessibility API provided by the smartphone OS [28, 39, 44]. However, Granting the rights for Accessibility API can take a few steps and brings extra trouble to tutorial creators. Even if the permissions are set, some user inputs still cannot be captured, such as interactions with unlabeled UI elements [40]. Additionally, Accessibility API allows the third party to inject actions to execute auto-clicks or other simulated gestures to control users' devices [5, 14], which brings high privacy risks to users. Because of this, the Accessibility API of iOS can only be used by built-in smartphone apps, and third-party apps have no rights to obtain user interaction traces from the Accessibility API. TapNet [18], which presents the potential of using smartphone IMU to detect different types of "Tap", inspires us to leverage smartphone IMU to detect the characteristics of user input. As built-in IMU is ubiquitous among off-the-shelf smartphones and only offers low-level signals with low privacy risk (no need for extra authentication in system settings), end-users and third parties can benefit from it for safety and convenience. In this work, we use the IMU channel to improve the performance of the CV channel for extracting user input steps from pre-recorded smartphone screencasts and generating instructional content accordingly. Our approach differs from TapNet [18] in two ways. First, TapNet [18] only focuses on tap gesture while ours recognizes six gestures. This is non-trivial as distinguishing two categories (tap, no tap) and six categories require training different AI models. Second, we quantified the detection performance of different gestures and present how IMU signals help to recognize different gestures.

2.2 Editing instructional content in video tutorials

Without artificial check and refinement, fully automated approaches for generating tutorials may often cause errors that are difficult to avoid [2, 10, 44]. Some related works involve authors' checking and labeling during the demonstration process to ensure the correctness of the generated instructional content in the interactive tutoring system [24, 41, 43]. However, such methods interrupt the demonstration, so they are less suitable for creating video tutorials that show an integral process of interaction task completion. In the domain of video editing, related works illustrate that informational video editing is time-consuming and challenging for non-experts [19]. Professional software (such as iMovie [20] or Adobe Premiere [37]) offers enough functions for producing various types of videos. At the same time, it brings barriers to non-experts due to complex work in timeline control, motion arrangement, and frame sequencing [19, 32]. Although simplified video editing tools have been designed for authoring certain types of videos (such as motion graphics [19] and marketing videos [9]), they are mainly used for style creation instead of instructional content editing. Some works propose advanced methods for generating video tutorials from a well-edited markdown-formatted tutorial [8]. Still, there is a lack of work on simplifying the editing of instructional content in tutorial videos. However, prior works also enlighten us to leverage markdown scripts for instructional content editing [8] and reduce the editing complexity by replacing timeline control with automatic key shot selection [19]. In this work, we lower the effort of tutorial creators by combining automatic instructional content generation with minimum post-refinement. By improving the recall of automatic user step extraction, the metadata of each step can be fully checked by creators. Creators only need to delete the redundant steps in candidate steps extracted automatically and refine the static instructional content (e.g., instructional text), which liberates the creators from complicated timeline operation and tool usage for video editing.

3 THE IMU-BASED INPUT STEP ANALYZER

We introduce our methods for extracting user input steps from a smartphone screencast of an interaction task and calculating the key elements of tutorial metadata. As this is a key contribution of this work, we present it in this separate section. This technique could generally be used in other integrative systems not only restrict to SmartRecorder. Fig 1 shows the pipeline of this technique. In this section, We firstly define the key elements of a user input step that constitute the metadata of a tutorial. Next, we present the sub-modules of the Input Step Analyzer. Then we show the system organization of combining video processing and IMU data processing to extract user input steps and calculate the tutorial metadata. Finally, we evaluate the performance of the proposed method by comparing the system performance with and without the assistance of IMU data. In the next section, we present how we construct the SmartRecorder based on this IMU-based Input Step Analyzer.

3.1 Metadata of Tutorial Content

By analyzing some existing step-by-step smartphone usage tutorials, we propose that each tutorial step should include four metadata to present instructional content. (1) The keyframe in the screencast when the user starts touching. The keyframe shows the recipient what the key interface looks like. (2) The screen-based gesture (Tap, Long tap, Scroll up, Scroll down, Scroll left, Scroll right, Typing). We take "Typing" as a gesture because "Typing" is a unit of a step, and there is no need to divide "Typing" into several Taps. Presenting gesture information helps tutorial recipient understand what to

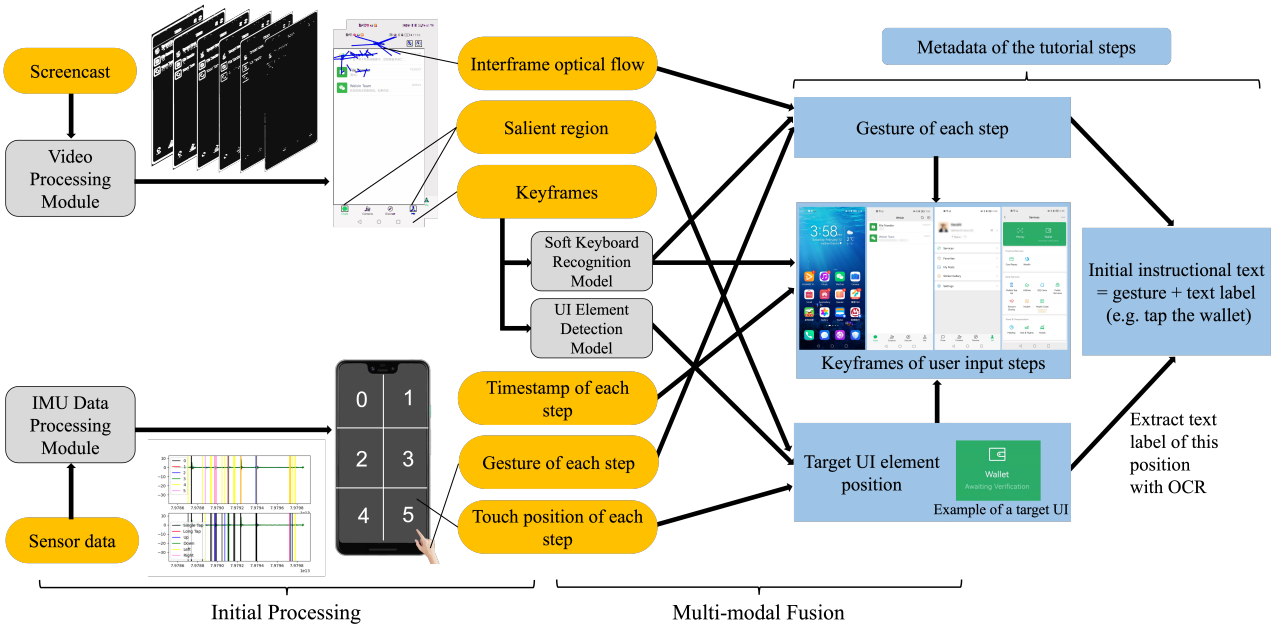


Figure 1: Pipeline of Input Step Analyzer. In the initial processing, the Video Processing Module and IMU Data Processing Module firstly process the screencast and smartphone IMU data respectively. The Video Processing Module outputs a set of keyframes where a user input could happen, as well as inter-frame salient regions and optical flow of each keyframe. The IMU Data Processing Module extracts the timestamp, gesture, and touch position of each user input from the IMU signals. Then the keyframes are processed by two assistive models: Soft Keyboard Recognition Model and UI Element Detection Model to filter the keyframes, recognize the typing gesture, and generate candidates of target UI element. These data are combined with other outputs of Video Processing Module and IMU Data Processing Module to finally generate the metadata of the tutorial steps.

do with the target UI element in each step. (3) The target UI element's position shows where to perform the corresponding gesture. We treat the entire soft keyboard area as the target UI element of "Typing." We also take arbitrary position in the interface where performing "Scrolls" can lead to screen scrolling as the target position of "Scrolls." (4) The instructional text which describes what to do in each step is a summary of the other metadata. The metadata is the source of the tutorial instructional content. Regardless of the visual design, the instructional content of each step should include these four elements of tutorial metadata.

3.2 Sub-modules of the Input Step Analyzer

The Input Step Analyzer's function is to calculate each step's tutorial metadata from the task completion screencast and its corresponding IMU data. It consists of two primary sub-modules: Video Processing Module (Extracting the keyframes from the screencast when the user starts to touch the screen for input, detecting the regions that display salient animation within frames, and calculating inter-frame optical flow to indicate the screen movement pattern), and IMU Data Processing Module (Extracting the timestamp of each user input, recognizing the gesture of each user input and predicting the finger touch position of each user input, from the IMU Data). The Input Step Analyzer also contains two assistive models for further extracting the gesture and target UI element of each step: the Soft Keyboard Recognition Model (Recognizing the typing gesture

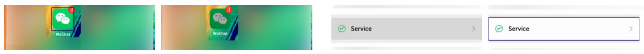


Figure 2: Examples of salient regions within inter-frames

according to the existence of a soft keyboard) and the UI Element Detection Model (Detecting the positions of all UI elements from a keyframe for predicting a possible UI element that the user tap).

3.2.1 *Video Processing Module.*

Video Splitter. To extract user input steps from a screencast video, we segment the screencast into several short pieces according to inter-frame difference since a user input can cause a screen animation. We calculate the inter-frame difference of the screencast thoroughly, filter out tiny and disconnected components and extract the video pieces during which the salient regions exist within inter-frames. Fig 2 shows the salient regions within inter-frames, which often contain the target UI elements because the user input often causes the visual animation of UI elements. However, the salient regions and target UI elements are usually not one-to-one mappings, so we stored all salient regions of a keyframe for further target UI element prediction. Such segmentation recalls all of the user input steps but includes some non-user-caused animation which is usually caused by message pop-up, advertisements, and page loading.

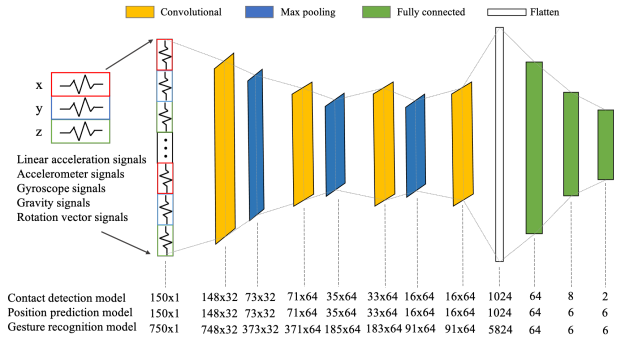


Figure 3: Structure of IMU Processing Module. It includes three models: contact detection model, position prediction model and gesture recognition model. They share the same model structure but have different shape of each layer. Five types of sensor signals concatenate together as the input.

Inter-frame Optical Flow Calculator. By calculating the inter-frame optical flow, we extract the movement pattern of feature points in the extracted short video pieces to predict the direction of the scroll gesture from the displacement vector of feature points. We use the Lucas-Kanade algorithm [29] to calculate the movement of feature points and compute the sum of the displacement vector, which contributes to gesture labeling.

Keyframe Filter. As the Video Splitter segments the screencast into several short video pieces, we take the start frame of each video pieces as a keyframe candidate for a user input. Therefore, we need to filter out redundant frames from this candidate set. We use optical character recognition (OCR) to extract the text of keyframe candidates and filter out the loading page by detecting keywords like "loading" and the pages with no words. We also detect the word variation among frames for further filtration to improve the precision of keyframe extraction. For example, if the words extracted from the start frame were the same as that of the end frame in a short video piece, we also filter out such frames because there is no semantic change in this video piece. However, there are still some redundant frames that we need to filter out, which is one reason for using the IMU channel.

Output of Video Processing Module. The Video Processing Module outputs a set of keyframes which shows the status of the interface when a screen animation begins. The Video Processing Module's proposals of keyframes include all keyframes when the user start a touch input, but also include some keyframes where the animation is not caused by user input as we have stated before. Together with each keyframe, the the Video Processing Module also outputs inter-frame salient regions and optical flow.

3.2.2 IMU Data Processing Module. The IMU Data Processing Module consists of three separate convolutional neural networks (CNN), one for contact detection, one for touch position prediction, and another for gesture recognition. We leverage 5 types of sensor signals on the smartphone (signals from 3-axis accelerometer, 3-axis gyroscope, 3-axis gravity, 3-axis linear acceleration, and 3-axis rotation vector) as the input of these models. Each data point is recorded with a unique timestamp. We use the contact detection model to extract the timestamp of each input step. For each step, the other two models will be applied to predict the position of contact and recognize the type of gesture. We present our work on dataset building and model development.

Dataset. We developed a mobile application to collect data for model training and recruited 11 participants (5 females, 6 females; aged from 20 to 24, M = 22.1, SD = 1.50) from the campus. The participants were asked to use their own smartphones to participate in this data collection experiment. The experiment consists of seven tasks: Tap, Long tap, Scroll up, Scroll down, Scroll left, Scroll right, and motions without touching the screen. Participants were asked to hold the smartphone in the left hand and perform screen-based gestures with the right hand in a sitting position. Considering the diversity in real use, we reminded participants to perform gestures with different properties by changing the description on the screen. For Tap and Long Tap, we collected gentle, normal, and strong ones. For Scroll gestures, we collected normal, short, long, fast, and slow ones. Since Tap occurs more frequently than other gestures in daily use, we collected more Tap data (about three times compared to others). For position prediction, we divided the screen into 3x2 grids and asked participants to tap the same number of times in each grid. Motions without touching the screen were also collected as negative samples, including holding the phone, touching the back and sides of the phone, shaking and flipping the phone, and so on. To avoid fatigue, participants could take a break every ten minutes. The whole experiment lasted for half an hour.

We collected 5 types of sensor signals on the mobile phone (signals from 3-axis accelerometer, 3-axis gyroscope, 3-axis gravity, 3-axis linear acceleration, and 3-axis rotation vector). Since the shortest sampling period for some of the signals mentioned above is 10ms, we unified the sampling rate to 100Hz. Besides, the start timestamp, the end timestamp, and the contact position of each gesture were also read from the capacitive screen for labeling. The final dataset contains 9702 positive samples of the six gestures.

Models. The structure of different models is similar, which is inspired by TapNet[18]. We use a convolutional neural network with multiple one-channel convolutional layers. The shapes of input and output tensors differ between models. Fig 3 shows the structure of our models and the parameters of each layer.

Contact Detection Model. Regarding model development, it is important to decide the length of the time window. Data collected in previous user experiments show that signal variation caused by contact can be covered by a time window of 10 frames (100ms), which is chosen for the contact detection model. We define the frame when the gesture starts as t_0, and use IMU data in [t_0 - 5, t_0 + 5) as positive samples. Data collected from the motions without touching the screen is randomly split into negative samples. In order to avoid early reports, we also use data in the time window [t_0 - 10, t_0) as negative samples.

Position Prediction Model. The contact position should be available as soon as the user touches the screen, similar to the contact detection task. So we also use the time window of 10 frames for position prediction and split positive and negative samples in the same way.

Gesture Recognition Model. For gesture recognition, there remains a trade-off. The longer the time window is, the more complex the model is and the longer it takes to run the algorithm. On the other hand, if the time window is too short, it won't be easy to distinguish between different gestures. By analyzing the samples collected, we summarize the duration for each screen-based gesture: 40ms to 120ms for Tap, 400ms to 500ms for Long Tap, and 50ms to 500ms for the Scroll gestures. Based on such discovery, we use a time window of 50 frames (500ms) for gesture recognition. Specifically, [t_0 - 10, t_0 + 40) is chosen for positive samples and [t_0 - 15, t_0 + 35) for negative ones. Considering that 500ms is a relatively long time, but some gestures only last for a short time, we apply various padding operations to them, including zero padding (append zero to the end of the original gesture data sequence), random noise padding (append values which can be calculated by adding random noise to a fixed value such as the value when the original gesture ends) and no padding.

3.2.3 *Assistive Models.*

Soft Keyboard Recognition Model. Since we take multiple Taps during a Typing as one user input step, we trained a convolutional neural network (CNN) with the similar structure of AlexNet [26] to recognize if there is a soft keyboard in the keyframes and condense the multiple keyframes which correspond to Typing (with a soft keyboard) to one user input step. The gesture of such an input step is relabeled to Typing. Our CNN was trained on a dataset containing 1080 smartphone screenshots (480 with soft keyboard, 600 without soft keyboard) and achieved an accuracy of 94.82% on the test set (containing 139 smartphone screenshots with soft keyboard and 170 without soft keyboard). We adjusted the hyperparameters of each layer according to the requirement of our work.

UI Element Detection Model. With prior knowledge of UI elements' position, the prediction of the target UI element will be more accurate. Based on such a motivation to train a UI Element Detection Model for detecting the UI elements of a given smartphone interface, we collected a dataset of smartphone UI elements and trained a UI Element Detection Model. We built custom software to collect UI data for training this model. Our dataset contains 1006 GUI screens from the top 28 most popular WeChat Mini Programs. The creation of our dataset consists of two sessions: collection and annotation. During the collection session, our software collected the screenshot of each traversed screen and corresponding accessibility data about UI elements. Since the collected data is not completely correct, we recruited 4 label workers to annotate it. For each UI element, the label worker labeled its bounding box and assigned one of 23 UI types (Text, Image, Icon, ImageButton, TextButton, ToggleButton, CloseButton, EditText, RadioButton, Checkbox, TabBar, TabBar Item, UnderLine, Container, Split bar, Rating bar, Scroll bar, Data Picker, Spinner, Dialog, Map) based on its visual features. With such a dataset, we used the EasyDL Object Detection API [1] to train our UI detection model. This model finally achieved a precision of 81.18% and a recall of 82.78%.

[1]https://ai.baidu.com/easydl/app/model/objdct/models

3.3 System Organization of the Input Step Analyzer

This part deliver how the sub-modules work together, and how their outputs are combined to generate the tutorial metadata of each step, as Fig 1 shows. We first present how the primary sub-modules process the screencast and IMU data separately in the initial processing. Then we illustrate the process of multi-modal fusion with the help of assistive models. Finally, we display the output of the Input Step Analyzer.

3.3.1 *Initial Processing.*

In this phase, the Video Processing Module process the screencast video, and the IMU Data Processing Module extracts the user input characteristics from smartphone IMU sinals. They work independently.

Processing Screencast Video. The Video Processing Module firstly segment the screencast video into several short video pieces which contain the keyframes of operation steps. The inter-frame salient regions and optical flow are recorded along with each video piece. Then, the Video Processing Module filter out some redundant video pieces and keyframe candidates according to the word information of the frames, and output a set of keyframes along with inter-frame salient regions and optical flow.

Processing IMU Data. The IMU data is firstly processed by the contact detection model. For the data points that are reported as positive examples by the contact detection model, they will be transmitted to position prediction model and gesture recognition model respectively. Finally, the IMU Data Processing Module outputs a timestamp, a 6-category gesture classification result, and a 6-category position prediction result of each positive sensor data point. The Video Processing Module and IMU Data Processing Module work independently in this initial processing phase. Such an organization can be implemented in an asynchronized and synchronized workflow.

3.3.2 *Multi-modal Fusion.*

In this phase, we leverage two pre-trained assistive models to process the keyframes that we extract from the screencast and combine the output of the video channel with the IMU channel. Finally, we get the keyframe, gesture, target UI element, and instructional text for each step.

Keyframe Filtration. With the keyframes we segmented from the screencast, we firstly use the Soft Keyboard Recognition Model to condense the keyframes corresponding to "Typing" as one frame. After that, with the timeline of the user input step that we extract from IMU data, we filter out the redundant frames that cannot be aligned to the timeline. Considering that there may be some time difference between the IMU channel and the video channel, to ensure the recall of user input steps, we set a relatively wide time window of 1000ms [t_0 - 300, t_0 + 700) for each timestamp t_0, and filter out the keyframes that don't fall into the time window. In this process, the redundant frames containing the animation from non-user input (e.g., system notification, popup windows) are filtered out, further improving the precision of user input steps.

Gesture Relabeling. We used the soft keyboard recognition model to recognize Typing as Typing is highly relevant with soft keyboard, and relabeled the multiple Taps as one Typing. For Tap and Long

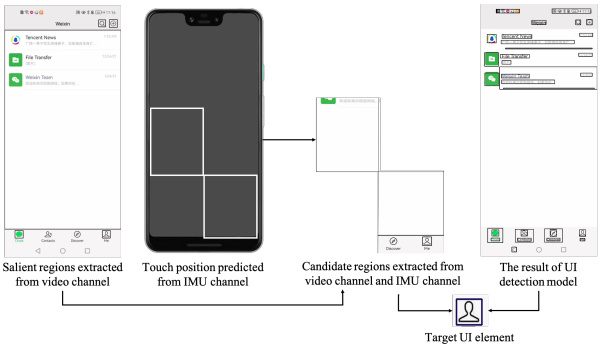


Figure 4: Pipeline to predict the target UI element. The system firstly extract the candidate regions by combining video channel with IMU channel. Then, it selects the position that has the highest IoU with the UI elements that the UI Element Detection Model outputs.

tap, we directly use the result of the IMU Data Processing Module because it is difficult to distinguish these two gestures from the video channel. For the scroll-based gestures (Scroll up, Scroll down, Scroll left, Scroll right), only when the inter-frame optical flow shows the feature points' horizontal or vertical movement and the IMU data shows the corresponding scroll-based pattern, we label the gesture as scroll-based gestures. The direction depends on the direction of feature points movement that optical flow displayed.

With such a multi-modal fusion strategy, we can correct some errors caused by the individual channel. For example, tapping a specific UI element may sometimes cause screen content's rightward/leftward movement animation. In this case, using the optical flow pattern only may result in the failure of recognizing the correct gesture, which requires the IMU channel for gesture recognition. Sometimes the IMU Data Processing Module may fail to recognize the direction of finger movement in such a short time window. In this case, the inter-frame optical flow can help to calculate the correct direction of a scroll-based gesture. However, recognizing Long Tap is difficult for both channels, and we discuss this limitation in the discussion section.

Target UI Element Prediction. With keyframes and gestures for each step, we tend to predict the target UI element on which the gesture is performed. Our target is to narrow the range of visual searching when the tutorial creator is trying to find and highlight the position of the target UI element in the tutorial creation process. With such information, we can highlight the position of the target UI element in the semi-finished tutorial so that the tutorial creator can check the target UI element of each step quickly without searching on the interface for several seconds. We define a correct prediction result as a bounding box that contains the target UI elements. We limit the size of this bounding box to no larger than 1/3 size of the whole screen.

We leverage the position prediction result of the IMU channel, salient regions calculated from the Video Processing Module, and the result of the UI Element Detection Model to predict the position of the target UI element. As Fig 4 shows, each channel calculates several candidate bounding boxes. We select the bounding boxes from the salient regions inside the top 2 results of the IMU channel. Then we further calculate the IoU between the selected bounding boxes and the output of the UI Element Detection Model, and select the bounding box with the highest IoU. Sometimes, no bounding box of inter-frame difference is inside the top 2 results of the IMU channel (e.g., some UI elements have no visual feedback when users touch on it). In this case, we select the top 2 results of the IMU channel in the case they are adjacent and select the top 1 result with its horizontal adjacent in case the top 2 results are bidiagonal.

Generation of Instructional Text. We generate initial instructional text to present what to do with the target UI element in the interface. The template of the initial instructional text is defined as "gesture + text label" of the target object. With the position of the target UI element where tutorial creators contact, we use the OCR technique to get the text label of the target object. Then we simply combine the gesture and text label to generate the initial instructional text (e.g., we combine the gesture "Tap" and text label "wallet" together to generate a simple instructional text "Tap the wallet"). If there is no text label, the initial instructional text will only contain the gesture type. The initial instructional text provides a template and entrance, which allows the user to refine the instructional text. We believe it is more convenient for tutorial creators to directly refine the instructional text than to create an instructional text all by themselves.

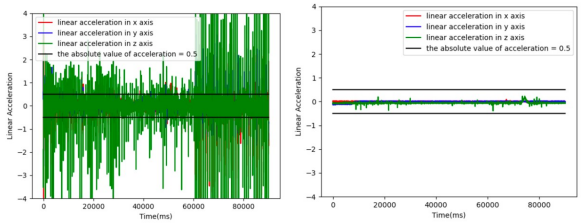
3.3.3 Output of Input Step Analyzer. As shown in Fig 1, the output of the Input Step Analyzer is the metadata of tutorial steps. For each step, the metadata includes the keyframe, the gesture of the user input, the position of the target UI element, and the initial instructional text. With such metadata, the Tutorial Generator of SmartRecorder generates a semi-finished tutorial that contains a visual guide of the target object and initial instructional text. The semi-finished tutorial with metadata will be sent to the Tutorial Refinement Editor. Fig 9 shows the semi-finished tutorial of a certain step and its display in the Tutorial Refinement Editor.

3.4 Technical Evaluation

In this section, we present the evaluation of the IMU-based Input Step Analyzer. We first report the performance of the IMU Data Processing Module. Then we present the system performance with and without the IMU channel. The result shows although there are some challenges in the IMU Data Processing Module, the IMU channel significantly improves the system performance. Video channel and IMU channel have the potential to make up for each other's defects.

3.4.1 Evaluation of the IMU Data Processing Module. We split the whole IMU dataset that we collected into train set, validation set, and test set (8:1:1). The validation set helps to deal with the overfitting problem, and we stop training when the results on the validation set converge. We report the performance of each model on the test set as follows.

Contact Detection Model. We first took an overview of the sensor data in the IMU dataset that we collected and filtered some abnormal data. In data collection, we collected the sensor data from



(a) Normal linear acceleration signal offered by a participant in data collection process

(b) Abnormal linear acceleration signal offered by a participant in data collection process

Figure 5: Comparison between normal (a) and abnormal (b) linear acceleration signal collected in IMU dataset

Table 1: Technical evaluation of contact detection model

Model	Accuracy	Precision	Recall	F1
Contact detection	0.982	0.977	0.975	0.976

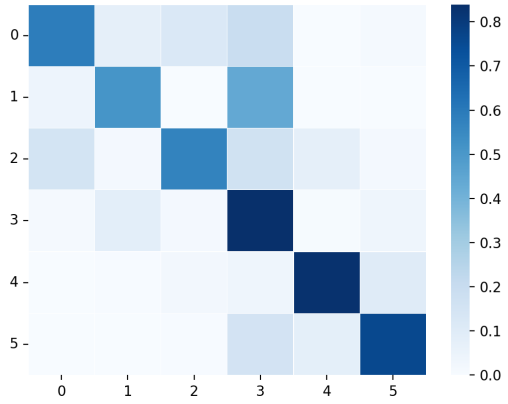


Figure 6: Normalized confusion matrix of position prediction model

participants' own devices, while some devices failed to offer normal sensing signals. For example, Fig 5a shows the normal linear acceleration signals in a data collection process. Since we asked the participants to tap the screen with normal pressure, gentle pressure, and heavy pressure each for about 30s when collecting tap signals, the figure shows normal variation feature and signal range (even the signals of gentle taps are higher than 0.5). However, Fig 5b shows the abnormal linear acceleration signal that all signal points of taps are lower than 0.5. As a result, we filtered out such samples before evaluation, which account for 24.4% of the original positive dataset. Then we evaluated our contact detection model on the test set. As Table 1 shows, the 2-class contact detection model achieves an accuracy of 98.2% with high precision (97.7%) and recall (97.5%), which can be further improved by combining the video channel with the IMU channel.

Position Prediction Model. The accuracy of this 6-class position prediction model seems to be limited (72.8%), so we perform further analysis to check whether the model could be applied in practice. As the normalized confusion matrix in Fig 6 shows, most mispredictions occur in two adjacent grids. For example, some gestures in grid 1 are predicted to grid 0, which is on the left side of the target, or to grid 3, which is below the target. Considering that the contact on the border of two adjacent grids could be difficult to distinguish, we also calculate the top 2 accuracy, which appears much better (90.3%). This indicates that this model could still be useful for predicting the touch position, especially when combined with the video channel.

Gesture Recognition Model. The gesture recognition model distinguishes six different gestures (Tap, Long tap, Scroll up, Scroll down, Scroll left, Scroll right). The model accuracy is 84.9%. As is shown in Table 2, the biggest challenge for the model is the recognition of Long Tap, which can be easily confused with Tap. The recall of all other gestures is above 82.1%, and Tap achieves the highest (94.8%). The misrecognized scroll gestures are most likely confused with Tap since some scroll gestures may be too fast or too gentle for the IMU to capture the scroll features. To tackle such a challenge, we leverage optical flow features from the video channel in our multi-modal fusion stage.

3.4.2 Performance of SmartRecorder with and without IMU Data Processing Module for Calculating the Tutorial metadata Automatically . To evaluate the performance of our approach for calculating the tutorial metadata from the screencast and IMU data, we made 20 recordings of smartphone interaction tasks with two different devices (Honor 20 and Huawei Mate 30 pro) and labeled 104 user input steps as our test set. We first removed the IMU Data Processing Module out and let SmartRecorder generate semi-finished tutorials with the video channel only. Without IMU Channel, it is hard to recognize the gesture of each step from the video channel solely, so it only segments the user input steps and predicts the target UI element of each step. Then we evaluated the whole system (with IMU channel) on the same dataset.

As Table 3 shows, without the IMU channel, the average recall of user input steps achieved 99.38%, and the average precision is 93.90%. The mean accuracy of target UI element prediction is 69.77%. With the IMU Data Processing Module, the average recall of user input steps remains 99.38%, and the average precision achieved 97.50%. The mean accuracy of target UI element prediction is 86.68% and the mean accuracy of gesture prediction for each step is 96.04%. Wilcoxon signed-rand test revealed that the IMU channel significantly improves the precision of the SmartRecorder for automatically segmenting user input steps ($p < 0.05$) and the accuracy for predicting target UI elements ($p < 0.01$).

4 DESIGN OF SMARTRECORDER

Based on the IMU-based Input Step Analyzer, we propose SmartRecorder to generate video tutorials of smartphone interaction tasks with machine automation and minimal human involvement. It consists of two front-end modules: Screencast Module (collects the screencast

Table 2: Technical evaluation of gesture recognition model

Gesture	Precision	Recall	Confusion Matrix					
			Tap	Long tap	Scroll up	Scroll down	Scroll left	Scroll right
Tap	0.856	0.948	**0.948**	0.013	0.01	0.008	0.013	0.008
Long tap	0.649	0.387	0.371	**0.387**	0.097	0.016	0.081	0.048
Scroll up	0.834	0.846	0.067	0.007	**0.846**	0.02	0.04	0.02
Scroll down	0.877	0.821	0.064	0.019	0.026	**0.821**	0.026	0.045
Scroll left	0.85	0.828	0.073	0.013	0.033	0.033	**0.828**	0.02
Scroll right	0.869	0.84	0.053	0.013	0.04	0.04	0.013	**0.84**

Table 3: Performance of SmartRecorder with and without IMU channel

Devices	Tasks	Steps	precision of steps		recall of steps		target UI prediction accuracy		gesture accuracy
			without IMU	with IMU	without IMU	with IMU	without IMU	with IMU	with IMU
device 1	T1	5	100%(5/5)	100%(5/5)	100%(5/5)	100%(5/5)	80%(4/5)	100%(5/5)	100%(5/5)
	T2	3	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	66.67%(2/3)	100%(3/3)	100%(3/3)
	T3	8	100%(7/7)	100%(7/7)	87.50%(7/8)	87.50%(7/8)	42.86%(3/7)	100%(7/7)	100%(7/7)
	T4	3	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	66.67%(2/3)	66.67%(2/3)	100%(3/3)
	T5	8	100%(8/8)	100%(8/8)	100%(8/8)	100%(8/8)	75%(6/8)	75%(6/8)	87.50%(7/8)
	T6	7	87.50%(7/8)	100%(7/7)	100%(7/7)	100%(7/7)	57.14%(4/7)	71.43%(5/7)	100%(7/7)
	T7	5	83.33%(5/6)	100%(5/5)	100%(5/5)	100%(5/5)	80%(4/5)	60%(3/5)	100%(5/5)
	T8	4	100%(4/4)	100%(4/4)	100%(4/4)	100%(4/4)	75%(3/4)	100%(4/4)	75%(3/4)
	T9	3	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)
	T10	5	100%(5/5)	100%(5/5)	100%(5/5)	100%(5/5)	60%(3/5)	80%(4/5)	100%(5/5)
device 2	T1	5	100%(5/5)	100%(5/5)	100%(5/5)	100%(5/5)	80%(4/5)	100%(5/5)	100%(5/5)
	T2	3	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	66.67%(2/3)	100%(3/3)	100%(3/3)
	T3	8	100%(8/8)	100%(8/8)	100%(8/8)	100%(8/8)	50%(4/8)	75%(6/8)	100%(8/8)
	T4	3	100%(3/3)	100%(3/3)	100%(3/3)	100%(3/3)	66.67%(2/3)	66.67%(2/3)	100%(3/3)
	T5	8	88.89%(8/9)	100%(8/8)	100%(8/8)	100%(8/8)	75%(6/8)	87.50%(7/8)	100%(8/8)
	T6	7	70%(7/10)	87.50%(7/8)	100%(7/7)	100%(7/7)	57.14%(4/7)	71.43%(5/7)	100%(7/7)
	T7	5	62.50%(5/8)	62.50%(5/8)	100%(5/5)	100%(5/5)	80%(4/5)	80%(4/5)	100%(5/5)
	T8	4	100%(4/4)	100%(4/4)	100%(4/4)	100%(4/4)	75%(3/4)	100%(4/4)	75%(3/4)
	T9	4	100%(4/4)	100%(4/4)	100%(4/4)	100%(4/4)	75%(3/4)	100%(4/4)	100%(4/4)
	T10	6	85.71%(6/7)	100%(6/6)	100%(6/6)	100%(6/6)	66.67%(4/6)	100%(6/6)	83.33%(5/6)
Mean value		5.20	93.90%	97.50%	99.38%	99.38%	69.77%	86.68%	96.04%

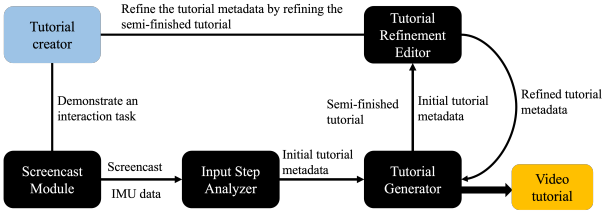


Figure 7: Technique pipeline of SmartRecorder

and IMU data during the process of an interaction task demonstration) and Tutorial Refinement Editor (enables tutorial creators to refine the tutorial metadata) and two back-end modules: Input Step Analyzer (calculates the initial metadata of the tutorial from the screencast and IMU data of demonstration) and Tutorial Generator (generates semi-finished tutorial for refinement and final video tutorials). We present the technique pipeline of SmartRecorder (see Fig 7) in this section.

4.1 Front end: Screencast Module

Users can use Screencast Module to record the process of completing a smartphone interaction task. It not only records the screen animation but also records the smartphone IMU data. When users stop recording, the screencast, as well as the corresponding IMU data, will be sent to Input Step Analyzer for user input analysis. We designed the interface of this module based on an open-source project[2].

[2]https://play.google.com/store/apps/details?id=com.app.kk.screenrecorder

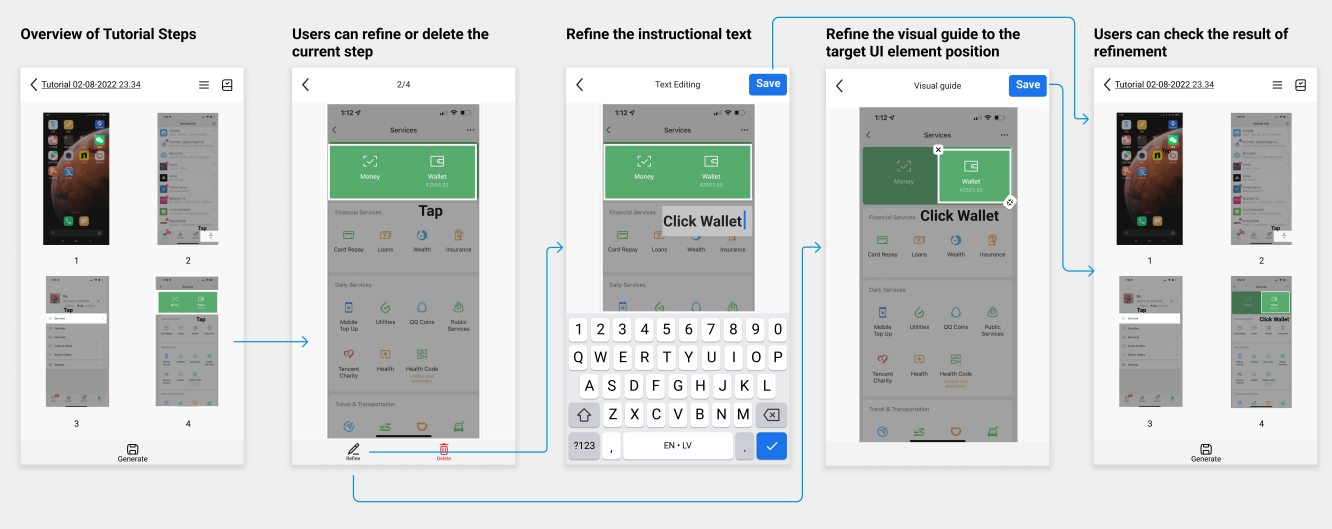


Figure 8: Interaction flow to check and refine the metadata of semi-finished tutorial

4.2 Back end: Input Step Analyzer

We use the technique we introduce in Section 3 to process the screencast and IMU data that Screencast Module collects. Input Step Analyzer calculates the initial tutorial metadata for generating a semi-finished tutorial.

4.3 Front end: Tutorial Refinement Editor

Considering that the tutorial metadata that Input Step Analyzer automatically extracts from a screencast cannot achieve 100% correctness, we designed an easy-to-use editor to help non-experts check and refine the metadata of the semi-finished tutorial quickly. This editor was designed based on the performance of the Input Step Analyzer. We summarize its characteristics as follows. (1) There is no need for users to add the keyframe of each step. Users just need to check the semi-finished tutorial and delete the redundant steps. Based on our analysis of professional software [37], the most difficult to learn and time-consuming features are those tools that users should leverage to add content actively. However, checking and deleting content is relatively easy. We ensure the high recall (99.38%) of key steps so that users do not need to insert keyframes actively. They just need to check the steps and delete redundant steps. (2) For each step, users do not need to draw instructional content; they only need to move the bounding box to highlight the target UI element more accurately and refine the instructional text. Users can do this in a templated interface and refine the instructional text directly instead of using some tools (like the rectangle tool and text box tool in Premiere) to draw the instructional content. 3) There is no need for users to consider what the instructional content looks like; they only need to ensure its correctness.

Fig 8 shows the interaction flow to check and refine the metadata of the semi-finished tutorial. The refinement editor presents an overview of tutorial steps, and the creator can check whether the tutorial content is correct. If there is a redundant step, the creator can delete it. Users can also get into the refinement interface to refine the position of the visual guide and the instructional text (e.g., in Fig 8, the second image shows a highlight box containing two UI elements, and the instructional text only indicates the gesture. The creator can click the "refine" button, narrow the range of the visual guide, and write the instructional text in detail). We require users to use the words which describe the gesture, such as tap" and "typing" when editing the instructional text so that SmartRecorder can correct the corresponding gesture in the metadata accordingly. During this refinement process, the SmartRecorder automatically corrects the metadata from Input Step Analyzer. For example, besides removing the redundant step, SmartRecorder extracts the keywords of the refined instructional text to get the gesture information and update the gesture label in tutorial metadata. The record of the target UI element position will be updated, and the instructional text stored in metadata will also be changed to what the creator finally writes. When the user clicks the button "Generate," the refined metadata will be sent to the Tutorial Generator, which generates the final video tutorial.

4.4 Back end: Tutorial Generator

4.4.1 Generating Semi-finished Tutorial. With the keyframe image of each step, we first highlight the position of the target UI element recorded in the metadata file and show the instructional text near the highlight region. These two elements are the basic content of a tutorial, with which the recipient can get what to do in each step. The generated tutorial is in a static format to display the tutorial metadata (see Fig 9), and it can be sent to the Tutorial Refinement Editor for further refinement.

4.4.2 Video Tutorial Generator. As video tutorials have the advantage of showing input gestures [38], we pre-designed short cartoon animations of every gesture and stored them in the back end for inserting them to the final video tutorial. According to the requirements for the video length [30], each gesture animation lasts about 3 seconds to show the corresponding gesture, not too long or too short. After adding instructional text and highlight region to the

Table 4: Interaction tasks and operation steps

No.	Name	Operation steps
T1	Check the bill in "WeChat"	Tap the "WeChat" icon -> Tap the "Me" tab -> Tap the "Pay" button -> Tap the "wallet" button -> Tap the "bill" button
T2	Start the simple mode of smartphone	Scroll the screen and find the setting -> Tap the "Setting" icon -> scroll the menu list and find the item "system and update" -> Tap the item "system and update" -> scroll the menu list and find the item "simple mode" -> Tap the item "simple mode"
T3	Search the "Baidu" app in minus 1 screen	Keep swiping the screen rightward into the minus 1 screen -> Tap the searching box -> Input the word "Baidu" -> Tap the appeared "Baidu" icon
T4	Mobile service payment with Alipay	Tap the "Alipay" icon -> Tap the "mobile service payment center" button -> Tap the phone number input box -> Input the phone number -> Tap an amount button -> Tap the "pay right now" button
T5	Delete a message	Tap the "Message" icon -> Long press on the message that need to be deleted -> Tap the "delete" button -> Tap the "confirm" button
T6	Create a memo	Tap the "Memo" icon -> Tap the "create" icon -> Input the memo content -> Tap the "save" icon

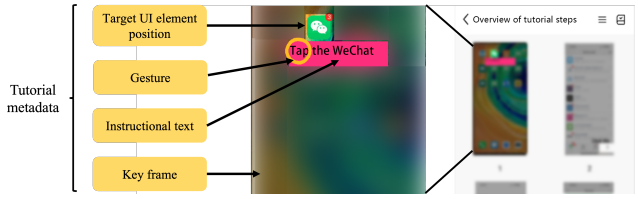


Figure 9: A step of a semi-finished tutorial, which contains the tutorial metadata, and its display in the Tutorial Refinement Editor

keyframe of each step, we merge every keyframe with every frame of its corresponding gesture animation according to the position of the target UI element and generate a short video piece of every step. Then we insert the generated instructional video piece into the original screencast according to the timestamp of each keyframe so that the video tutorial displays the instructional content step-by-step. We also refer to some prior works to improve accessibility in terms of interface font size, object size, and element position [35].

5 USER STUDY: EVALUATING VIDEO TUTORIAL CREATION EFFICIENCY AND GUIDING EFFECT OF SMARTRECORDER

We conducted a two phase user study to evaluate the performance of SmartRecorder from the end-user perspective. At the beginning of this two-phase user study, we designed 6 interaction tasks (see Table 4). Then in phase 1, we evaluate creators' efficiency of using SmartRecorder to create video tutorials. We recruited expert participants (EP) who have expertise in video editing and non-expert participants (NP) who have no expertise in video editing to create video tutorials of the interaction tasks shown in Table 4 in this phase. In phase 2, we evaluate the guiding effectiveness of the video tutorials generated by SmartRecorder in terms of guiding tutorial recipients to complete interaction tasks. We recruited another 18 participants to evaluate the guiding effectiveness of the video tutorials that are created during phase 1 in this phase.

5.1 Phase 1: Video Tutorial Creation Efficiency Evaluation

5.1.1 Participants. Through online contact, we recruited 12 participants who are college students or researchers to play the role of tutorial creator in this phase. 6 (3 female and 3 male; aged from 22 to 30, M = 26.33, SD = 2.74) of them were expert users of video editing with more than 1 year of video editing experience and all majored in design (abbreviated as EP). The other 6 participants (3 female and 3 male; aged from 20 to 24, M = 21.83, SD = 1.57) had no prior video editing experience (abbreviated as NP). Demographic information of the participants is listed in Table 5, Appendix. We provided the each EP with 100 yuan and each NP with 200 yuan as the compensation.

5.1.2 Apparatus. We conducted this experiment in a meeting room and provided each participant with a smartphone (Huawei Mate30 Pro), which had the SmartRecorder and video editing tools that expert participants usually use installed. For P4, who usually uses Adobe Premier, we allowed him to use his own desktop with Adobe Premier to participate in our experiment. We also prepared a video tutorial that was created artificially to show participants what elements should be included in the video tutorial.

5.1.3 Procedure. Although most of the EPs major in information design, we still showed all of them the video tutorial example and introduced the elements that they should display in their created video tutorials before the creation process. For the NPs, we did not do this because SmartRecorder can generate the tutorial automatically.

For each EP, we conducted the experiment through the following steps.

Step I: We taught each participant how to use SmartRecorder first and let them get familiar with it for about 3 minutes.

Step II: We showed the path to complete an interaction task before the participant started to create the tutorial of it.

Step III: The participant leveraged their daily used video editing tools (baseline) to make the video tutorial of the given interaction task. We ignored the time they spent on screen recording and recorded the time they spent on video editing.

Step IV: The participant leveraged SmartRecorder to make the video tutorial of the given interaction task and we recorded the time they spent on refining the semi-finished tutorial in this process.

Step V: Repeated Step II, III, and IV on the next interaction task and exchanged the sequence of Step III and Step IV for counterbalance.

Step VI: We asked the participants to fill out a questionnaire about the user preference (1 = the lowest user preference, 5 = the highest user preference), fatigue (1 = the most fatiguing, 5 = the least fatiguing), ease-of-use (1 = the most difficult to use, 5 = the easiest to use) and learnability (1 = the most difficult to learn, 5 = the easiest to learn) regarding baseline and SmartRecorder. Participants were required to rate the above four indicators on 5-point Likert scales for each tutorial creation process. The scale direction is the higher the better.

For each NP, we conducted the above steps without III, and they only rated the score for SmartRecorder in Step VI. The study lasted about 1 hour with each NP and about 2 hours with each EP.

5.1.4 Results and Discussion. **Data Collection**. Through the experiment, we got 36 completion time records (6 participants x 6 interaction tasks) from EPs with baseline, 36 completion time records from EPs with SmartRecorder, and 36 completion time records from NPs with SmartRecorder. We also collected 36 (6 participants x 6 interaction tasks) scores on each indicator from EPs for baseline, 36 scores from EPs for SmartRecorder, and 36 scores from NPs for SmartRecorder.

Time for Creating the Tutorials. The average time that EPs spent on video editing with the baseline tool is 349.67s (SD = 253.24) and is 132.64s (SD = 69.38) with SmartRecorder. The average time that NPs consumed is 85.25s (SD = 43.68), with SmartRecorder. The completion time distribution suggested non-normality, confirmed by a Shapiro-Wilk test. Wilcoxon signed-rank test revealed that compared with baseline, EPs spent significantly less time ($p<0.001$) when using SmartRecorder to create video tutorials. The time cost is reduced by 62.07%. Mann-Whitney's U test revealed that NPs also spent significantly less time ($p<0.001$) when using SmartRecorder than EPs spent when they used baseline, and time cost is reduced by 75.62%.

Subjective Experiences for Creating the Tutorials. Fig 10, shows the box plot of subjective scores that the baseline and SmartRecorder got in terms of ease-of-use, fatigue, learnability, and user preference. The distribution of the ratings is non-normal, confirmed by a Shapiro-Wilk test. Among EPs, Wilcoxon signed-rank test revealed that SmartRecorder got significantly higher ratings on each indicator (ease-of-use: Median = 4, IQR = 1; fatigue: Median = 4, IQR = 0.25; learnability: Median = 4, IQR = 1; user preference: Median = 4, IQR = 0.25) than baseline (ease-of-use: Median = 2.5, IQR = 1; fatigue: Median = 3, IQR = 1.25; learnability: Median = 3, IQR = 1.25; user preference: Median = 3, IQR = 3) with following p values (ease-of-use: $p<0.001$; fatigue: $p<0.001$; learnability: $p=0.004<0.01$; user preference: $p<0.001$). Mann-Whitney's U test showed that NPs also rated significantly higher scores ($p<0.001$) on each indicator (ease-of-use: Median = 4, IQR = 1; fatigue: Median = 4, IQR = 2; learnability: Median = 5, IQR = 1; user preference: Median = 4, IQR = 1) for SmartRecorder when compared with the scores that EPs assigned to baseline. There is no significant difference between EPs' scores and NPs' scores for SmartRecorder in terms of ease-of-use, fatigue, and user preference. For the indicator "learnability", NPs assigned significantly higher scores than EPs ($p = 0.019 <0.05$). It possibly resulted from that SmartRecorder facilitated NPs to conveniently achieve what they were unable to do in the past while it didn't have such an impact on EPs.

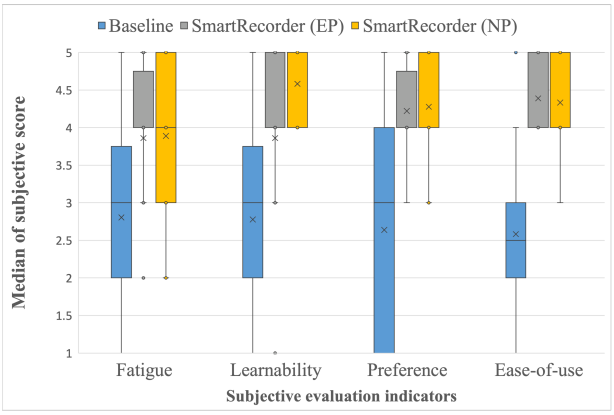


Figure 10: Results of subjective evaluation on baseline and SmartRecorder

The result of this study shows that with SmartRecorder, people without video editing skills can create video tutorials with even less time than those with video editing skills and use familiar video editing tools. From the subjective feedback, SmartRecorder can significantly reduce the fatigue and the learning cost of creating video tutorials and is much easier to use than EPs daily used tools.

5.2 Phase 2: Guiding Effectiveness Evaluation

5.2.1 Participants. In this phase, we recruited 18 participants (10 females, 8 males; aged from 26 to 73, M = 53.39, SD = 17.52) through online contact based on the criteria that participants should be unfamiliar with the interaction tasks that we listed in Table 4. All of the participants have middle school degrees or above, and all of them have 3 years or above smartphone-using experience. Demographic information of our participants in this phase is listed in Table 6, Appendix. We provided each of them with 100 yuan for their participation.

5.2.2 Apparatus. This experiment was conducted in a meeting room, and two smartphones were used during the experiment. One smartphone was used (Honor V20) to play the video tutorials made in phase 1, and the other one (Huawei Mate30 Pro) was used to complete the interaction tasks listed in Table 4. The tutorials created in the prior phase were re-organized into 6 groups of baseline tutorials (created with baseline tools by EPs) and 6 groups of SmartRecorder tutorials (created with SmartRecorder by NPs). For each group, each tutorial was from different authors and aimed at different interaction tasks with latin square to counterbalance the author's impact.

We assigned each group to three participants, and the assignment was counterbalanced on gender. Each participant got two groups of tutorials. One group is the baseline tutorial, and the other group is the SmartRecorder tutorial.

5.2.3 Procedure. We allowed each participant about 1 minute to get familiar with the smartphone that we provided before they started to complete the interaction tasks. During the experiment, participants were asked to complete the interaction tasks listed in Table 4 with the guidance of two groups of video tutorials. For each interaction task, we played the corresponding video tutorials on one smartphone, and the participant completed the interaction task on the other smartphone. We looped the video until the participant finished the task. The participant could revisit the video at any time in this process. We recorded the time from the start of watching the video tutorial to the completion of the interaction task. If the participant could not understand the tutorial content or was unable to complete the task, we allowed them to give up that task. After they completed all of the interaction tasks once, we asked the participants to rate their satisfaction (1 = the lowest satisfaction, 5 = the highest satisfaction) to each video tutorial that they just learned from on 5-point Likert scales and allowed them to have a rest for about 5 minutes. Then, we changed to the other group of video tutorials. The participant needed to complete the interaction tasks again with the guidance of a new group of tutorials. The sequence to play two groups of tutorials was counterbalanced. We took brief interviews with the participants about their comments on the two types of tutorials after they finished all interaction tasks. Each participant took about 1 hour to finish the whole process.

5.2.4 Results and Discussion. There are 108 (6 interaction tasks x 18 participants) interaction tasks required to be completed in total with the guidance of baseline tutorials and SmartRecorder tutorials. The result shows that the video tutorials created with SmartRecorder could facilitate participants to complete the interaction tasks with higher completion rate, less completion time and higher satisfaction.

Completion Rate. The completion rate of the baseline group is 96.27%(104/108). P2 failed in T3 for being unable to understand the goal of this task and T5 for their inability to understand the meaning of "long press" in instructional text. P10 failed in T1 for being unable to catch up with the instructional content and finally gave up. P12 also failed in T5 for unbale to understand the meaning of "long press". The completion rate of the SmartRecorder group is 99.07%(107/108). P2 failed in T3 for still being unable to understand the task goal.

Completion Time. For the tasks that were completed successfully, we got 104 completion time records of the baseline group and 107 completion time records of the SmartRecorder group. The average completion time of the baseline group is 68.88s (SD = 48.46), and the average completion time of the SmartRecorder group is 53.47s (SD = 39.16). Shapiro-Wilk test confirmed the non-normality of completion time. Mann-Whitney's U test showed that with the tutorial generated from SmartRecorder, the average completion time of an interaction task was significantly less ($p = 0.005 < 0.01$) than that with baseline tutorials.

Subjective Satisfaction. We collected 108 (6 interaction tasks x 18 participants) satisfaction scores on baseline tutorials and tutorials generated from SmartRecorder. Shapiro-Wilk test suggested the non-normality of ratings and Wilcoxon signed-rank test revealed that tutorials generated from SmartRecorder got significantly higher user satisfaction (Median = 5, IQR = 1) than baseline tutorials (Median = 4, IQR = 2) with $p<0.001$.

Cause of Higher Satisfaction. Compared with the baseline group, the main cause of higher user satisfaction and less completion time for the SmartRecorder group was the smooth pace of the video tutorial. In the baseline group, some tutorials only lasted for less than 1 second, which led to participants' failure to keep up with. Take P4 as an example *"I like the tutorial that the machine generated, I feel it has a buffer time for me to watch. The other version plays too fast and ends before I can't make sense of it."* P7 also indicated that *"The prior version (baseline group) is sometimes too fast, and it's no doubt that the second type (SmartRecorder group) is better. But it will be much better if it can be more slowly and pause longer time in the instructional piece to allow us more time to make sense of it."*. We also observed that the participants needed to revisit and replay the tutorial videos in the baseline group much more times which led to a longer completion time. Since we add the gesture animation to each instructional video piece, the length of the instructional piece is extended, which results in a more friendly instruction pace. Additionally, compared with manually created video tutorial, auto-generated video tutorials make it easier to keep the pace of the video steady by uniformly setting the length of the instructional video clips. However, some participants like P7 proposed that the pace of the generated tutorial video remained to be optimized. We believe this is an interesting point that deserves further exploration.

6 DISCUSSION

We firstly discuss the novelty of our work. Then we indicate the limitation and future work of this study.

6.1 The Novelty of Our Work

We discuss the novelty of our work from two aspects. From the perspective of technique, we leverage smartphone IMU to analyze user input characteristics, which presents a new method to process the media information on smartphones and generate tutorials about smartphone usage. From the perspective of the human-computer collaboration, we improve the recall of tutorial steps to let creators delete the redundant steps and refine the tutorial content directly with an easy-to-use editor, which lowers the time cost and skill requirement. We also propose that facilitating end-users to be a checker and refiner of machine automation can be a promising way of human-computer collaboration.

6.1.1 Leveraging smartphone IMU to Analyze User Input Characteristic. TapNet [18] explored the potential of smartphone IMU to recognize multiple tap properties, including tap direction and location, in the context of off-screen interaction. We further explore the feasibility of recognizing the screen-based gestures (Tap, Long tap, Scroll up, Scroll down, Scroll left, Scroll right) and apply this technique to tutorial generation. To tackle the challenge of extracting user interaction trace from screencast, smartphone IMU is a channel that has great potential to use with low privacy risk and

without complicated user settings (compared with using Accessibility API [28, 39, 44]). Considering the popularity of IMU sensors on smartphones regardless of the operating system, using IMU to assist screencast processing will be more pervasive and accessible than some existing methods. The technique evaluation also shows that combining the video channel with the IMU channel will compensate for each other's defects and improve the system performance when extracting tutorial metadata from the demonstration.

6.1.2 Human-Computer Collaborative Tutorial Creation. Since it is hard to ensure the correctness of the tutorial content with fully automated approaches, we involve the creators as a role of a checker and refiner of the machine automation. In SmartRecorder, the editing complexity is reduced from video editing to image refinement, which reduces the skill requirement, decreases the time cost, and facilitates the creation of non-experts. The back-end Input Step Analyzer supports the design of Tutorial Refinement Editor. It automatically extracts the key information of each tutorial step, which reduces the human work on information retrieval. With a high recall of key steps, the tutorial creator only needs to delete the redundant steps and refine the tutorial metadata in several images, which liberates them from timeline operation and information reorganization. We believe that facilitating end-users to be a checker and refiner of machine automation can be a promising way to improve human-computer collaborative work. It not only reduces the human workload and skill requirements but also keeps the correctness of the result.

6.2 Limitation and Future Work

We discuss this study's limitations and future work from the perspective of IMU data processing, the target group of tutorial recipients.

6.2.1 IMU Data Processing. For the contact detection model, we filter out some gentle touches that cause the fluctuation of linear acceleration below 0.5. If the user touches the screen too gently, it will be hard for the contact detection model to detect the contact. However, as the mean value of a positive sample is 2.1, the accuracy will be high when the user touches the screen with normal pressure. For the gesture recognition model, since the similar tap properties between Long Tap and Tap, we find that distinguishing these two gestures is difficult, which affects our gesture classification accuracy. As for the position prediction model, since the contact on the border of two adjacent grids is difficult to distinguish, our 6-class position prediction is not good yet. As a result, we leverage the top 2 results of position in the following UI element position prediction. In the future, we will collect more data for model training and optimize the model for IMU data processing. We believe the improvement of the IMU data processing model can reduce the complexity of the following technique pipeline and further improve the performance of the Input Step Analyzer.

6.2.2 Specific Study on Recipient Groups. As this work mainly focuses on the creation process of the video tutorial, we focus less on the different requirements of different recipient groups. As general video tutorials can not always satisfy all persons, there remains much research space for exploring the specific needs of varying recipient groups. For example, when creating a tutorial for children or people with visual impairments, there should be some special requirements for the tutorial content design [23, 34]. Also, the user studies lack qualitative research, which leads to our weak contribution to the recipients' personalized feedback and design opportunities for different recipient groups. In the future, we will conduct more user studies to divide recipients into different groups and explore the specific needs of different recipient groups so that we can optimize the tutorial generated by SmartRecorder for different people.

7 CONCLUSION

This paper presents SmartRecorder, a novel tool that generates video tutorials of smartphone tasks with machine intelligence and minimal human effort. With the screencast and IMU data (vibration characteristics generated by user touch) collected in the task completion process, SmartRecorder leverages computer vision and sensing algorithms to extract key information (keyframes, target UI element position, gesture) of each input step in the screencast and generate the tutorial video. Through a two-phase user study, we evaluated the efficiency and effectiveness of SmartRecorder. The results show that SmartRecorder facilitates non-experts to create video tutorials with significantly less time (reduced by 75.62%) and higher satisfaction from tutorial recipients. Using IMU data to improve the performance of automatic interaction trace extraction is novel among existing methods. Additionally, SmartRecorder provides tutorial creators with an easy-to-use tutorial refinement editor designed based on the high recall for key steps to revise the tutorial content, which minimizes human effort and ensures the correctness of tutorial content. We also discuss that involving human creators as a checker and refiner for the result that machine intelligence creates can be a promising way for human-computer collaboration.

ACKNOWLEDGMENTS

This work is supported by the Natural Science Foundation of China under Grant no. 62132010, and by Beijing Key Lab of Networked Multimedia, the Institute for Guo Qiang, Tsinghua University, Institute for Artificial Intelligence, Tsinghua University (THUAI), and by 2025 Key Technological Innovation Program of Ningbo City under Grant No.2022Z080.

REFERENCES

[1] Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2012. Waken: Reverse Engineering Usage Information and Interface Structure from Software Videos. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 83–92. https://doi.org/10.1145/2380116.2380129

[2] Lingfeng Bao, Zhenchang Xing, Xin Xia, and David Lo. 2018. Vt-revolution: Interactive programming video tutorial authoring and watching system. *IEEE Transactions on Software Engineering* 45, 8 (2018), 823–838.

[3] Lawrence Bergman, Vittorio Castelli, Tessa Lau, and Daniel Oblinger. 2005. DocWizards: A System for Authoring Follow-Me Documentation Wizards. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Seattle, WA, USA) *(UIST '05)*. Association for Computing Machinery, New York, NY, USA, 191–200. https://doi.org/10.1145/1095034.1095067

[4] Carlos Bernal-Cárdenas, Nathan Cooper, Madeleine Havranek, Kevin Moran, Oscar Chaparro, Denys Poshyvanyk, and Andrian Marcus. 2022. Translating Video Recordings of Complex Mobile App UI Gestures Into Replayable Scenarios. *IEEE Transactions on Software Engineering* (2022).

[5] Carlos Bernal-Cárdenas, Nathan Cooper, Kevin Moran, Oscar Chaparro, Andrian Marcus, and Denys Poshyvanyk. 2020. Translating Video Recordings of Mobile App Usages into Replayable Scenarios. In *Proceedings of the ACM/IEEE 42nd*

International Conference on Software Engineering (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 309–321. https://doi.org/10.1145/3377811.3380328

[6] Diogo Cabral and Nuno Correia. 2017. Video editing with pen-based technology. *Multimedia Tools & Applications* 76, 5 (2017), 6889–6914.

[7] Juan Casares, A Chris Long, Brad A Myers, Rishi Bhatnagar, Scott M Stevens, Laura Dabbish, Dan Yocum, and Albert Corbett. 2002. Simplifying video editing using metadata. In *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*. 157–166.

[8] Peggy Chi, Nathan Frey, Katrina Panovich, and Irfan Essa. 2021. Automatic Instructional Video Creation from a Markdown-Formatted Tutorial. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 677–690.

[9] Peggy Chi, Zheng Sun, Katrina Panovich, and Irfan Essa. 2020. Automatic Video Creation From a Web Page. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 279–292.

[10] 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* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 93–102. https://doi.org/10.1145/2380116.2380130

[11] Deephow. 2022. *Deephow*. https://www.deephow.cn/#solution

[12] Jonathan D. Denning, William B. Kerr, and Fabio Pellacini. 2011. MeshFlow: Interactive Visualization of Mesh Construction Sequences. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 66, 8 pages. https://doi.org/10.1145/1964921.1964961

[13] 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* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 373–382. https://doi.org/10.1145/2047196.2047245

[14] Google. 2022. *Documentation for app developers*. https://developer.android.com/docs

[15] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating Photo Manipulation Tutorials by Demonstration. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 66, 9 pages. https://doi.org/10.1145/1576246.1531372

[16] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. *Chronicle: Capture, Exploration, and Playback of Document Workflow Histories*. Association for Computing Machinery, New York, NY, USA, 143–152. https://doi.org/10.1145/1866029.1866054

[17] Jeff Huang and Michael B. Twidale. 2007. Graphstract: Minimal Graphical Help for Computers. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA) *(UIST '07)*. Association for Computing Machinery, New York, NY, USA, 203–212. https://doi.org/10.1145/1294211.1294248

[18] Michael Xuelin Huang, Yang Li, Nazneen Nazneen, Alexander Chao, and Shumin Zhai. 2021. TapNet: The Design, Training, Implementation, and Applications of a Multi-Task Learning CNN for Off-Screen Mobile Input. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 282, 11 pages. https://doi.org/10.1145/3411764.3445626

[19] Amir Jahanlou and Parmit K Chilana. 2022. Katika: An End-to-End System for Authoring Amateur Explainer Motion Graphics Videos. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[20] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion amplifiers: sketching dynamic illustrations using the principles of 2D animation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4599–4609.

[21] Kimia Kiani, Parmit K Chilana, Andrea Bunt, Tovi Grossman, and George Fitzmaurice. 2020. "I Would Just Ask Someone": Learning Feature-Rich Design Software in the Modern Workplace. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–10.

[22] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. Crowdsourcing Step-by-Step Information Extraction to Enhance Existing How-to Videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 4017–4026. https://doi.org/10.1145/2556288.2556986

[23] Kenneth A Kobak, Wendy L Stone, Elizabeth Wallace, Zachary Warren, Amy Swanson, and Kraig Robson. 2011. A web-based tutorial for parents of young children with autism: results from a pilot study. *Telemedicine and e-Health* 17, 10 (2011), 804–808.

[24] 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* (Virtual Event, USA) *(SUI '21)*. Association for Computing Machinery, New York, NY, USA, Article 16, 11 pages. https://doi.org/10.1145/3485279.3485289

[25] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012. *Delta: A Tool for Representing and Comparing Workflows*. Association for Computing Machinery, New York, NY, USA, 1027–1036. https://doi.org/10.1145/2207676.2208549

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (may 2017), 84–90. https://doi.org/10.1145/3065386

[27] Ben Lafreniere, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2014. Investigating the Feasibility of Extracting Tool Demonstrations from In-Situ Video Content. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 4007–4016. https://doi.org/10.1145/2556288.2557142

[28] Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017. SUGILITE: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 6038–6049.

[29] Bruce D Lucas, Takeo Kanade, et al. 1981. *An iterative image registration technique with an application to stereo vision*. Vol. 81. Vancouver.

[30] Nichole A Martin and Ross Martin. 2015. Would you watch it? Creating effective and engaging video tutorials. *Journal of Library & Information Services in Distance Learning* 9, 1-2 (2015), 40–56.

[31] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2011. *Ambient Help*. Association for Computing Machinery, New York, NY, USA, 2751–2760. https://doi.org/10.1145/1978942.1979349

[32] Britta Meixner, Katarzyna Matusik, Christoph Grill, and Harald Kosch. 2014. Towards an easy to use authoring tool for interactive non-linear video. *Multimedia Tools and Applications* 70, 2 (2014), 1251–1276.

[33] Cuong Nguyen and Feng Liu. 2015. *Making Software Tutorial Video Responsive*. Association for Computing Machinery, New York, NY, USA, 1565–1568. https://doi.org/10.1145/2702123.2702209

[34] Amy Pavel, Gabriel Reyes, and Jeffrey P Bigham. 2020. Rescribe: Authoring and Automatically Editing Audio Descriptions. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 747–759.

[35] Andraž Petrovčič, Sakari Taipale, Ajda Rogelj, and Vesna Dolničar. 2018. Design of mobile phones for older adults: An empirical analysis of design guidelines and checklists for feature phones and smartphones. *International Journal of Human–Computer Interaction* 34, 3 (2018), 251–264.

[36] 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* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 135–144. https://doi.org/10.1145/2047196.2047213

[37] Adobe Premiere. 2022. *Premiere Pro*. https://www.adobe.com/products/premiere.html

[38] Jorge Ribeiro and Ana Correia de Barros. 2014. Efficiency of a video and a tutorial in teaching older adults to interact with smartphones. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 34–45.

[39] André Rodrigues, Leonardo Camacho, Hugo Nicolau, Kyle Montague, and Tiago Guerreiro. 2018. Aidme: Interactive Non-Visual Smartphone Tutorials. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct* (Barcelona, Spain) *(MobileHCI '18)*. Association for Computing Machinery, New York, NY, USA, 205–212. https://doi.org/10.1145/3236112.3236141

[40] Alborz Rezazadeh Sereshkeh, Gary Leung, Krish Perumal, Caleb Phillips, Minfan Zhang, Afsaneh Fazly, and Iqbal Mohomed. 2020. VASTA: a vision and language-assisted smartphone task automation system. In *Proceedings of the 25th international conference on intelligent user interfaces*. 22–32.

[41] Cheng-Yao Wang, Wei-Chen Chu, Hou-Ren Chen, Chun-Yen Hsu, and Mike Y. Chen. 2014. EverTutor: Automatically Creating Interactive Guided Tutorials on Smartphones by User Demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 4027–4036. https://doi.org/10.1145/2556288.2557407

[42] Sarah Weir, Juho Kim, Krzysztof Z. Gajos, and Robert C. Miller. 2015. Learnersourcing Subgoal Labels for How-to Videos. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 405–416. https://doi.org/10.1145/2675133.2675219

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

[44] Mingyuan Zhong, Gang Li, Peggy Chi, and Yang Li. 2021. *HelpViz: Automatic Generation of Contextual Visual Mobile Tutorials from Text-Based Instructions*. Association for Computing Machinery, New York, NY, USA, 1144–1153. https://doi.org/10.1145/3472749.3474812

A PARTICIPANT DEMOGRAPHICS IN THE USER STUDY

Table 5: Expert and Non-expert participants in the first phase of user study

Label	No	Age	Gender	Profession	Video editing tool
EP	P1	29	F	Researcher working in human factor	Viamaker
EP	P2	22	F	Student majoring in information design	Viamaker
EP	P3	26	F	Student majoring in information design	VUE
EP	P4	30	M	Student majoring in architecture design	Adobe Premier
EP	P5	24	M	Student majoring in information design	Viamaker
EP	P6	27	M	Student majoring in information design	Viamaker
NP	P7	24	M	Student majoring in computer science	None
NP	P8	23	M	Student majoring in computer science	None
NP	P9	20	M	Student majoring in computer science	None
NP	P10	20	F	Student majoring in social science	None
NP	P11	23	F	Student majoring in social science	None
NP	P12	21	F	Student majoring in management	None

Table 6: Demographic information of participants in the second phase of user study

No	Age	Gender	Education level	Years of smartphone usage
P1	70	F	Junior middle school	5
P2	67	M	Senior middle school	4
P3	63	F	Senior middle school	8
P4	73	F	college	6
P5	68	F	college	10
P6	61	F	Senior middle school	7
P7	64	M	college	6
P8	61	F	Senior middle school	6
P9	65	M	Senior middle school	7
p10	65	F	Senior middle school	3
P11	70	M	Junior college	9
P12	61	M	Senior middle school	4
P13	29	M	Post graduate.	above 10
P14	28	M	Post graduate	above 10
P15	33	F	Post graduate	above 10
P16	26	F	Post graduate	above 10
P17	26	F	Post graduate	above 10
P18	33	M	Post graduate	above 10