BrushLens: Hardware Interaction Proxies for Accessible Touchscreen Interface Actuation

Chen Liang
University of Michigan
Ann Arbor, MI, USA
clumich@umich.edu

Yasha Iravantchi
University of Michigan
Ann Arbor, MI, USA
yiravan@umich.edu

Thomas Krolikowski
University of Michigan
Ann Arbor, MI, USA
tkroliko@umich.edu

Ruijie Geng
University of Michigan
Ann Arbor, MI, USA
gengr@umich.edu

Alanson Sample
University of Michigan
Ann Arbor, MI, USA
apsample@umich.edu

Anhong Guo
University of Michigan
Ann Arbor, MI, USA
anhong@umich.edu

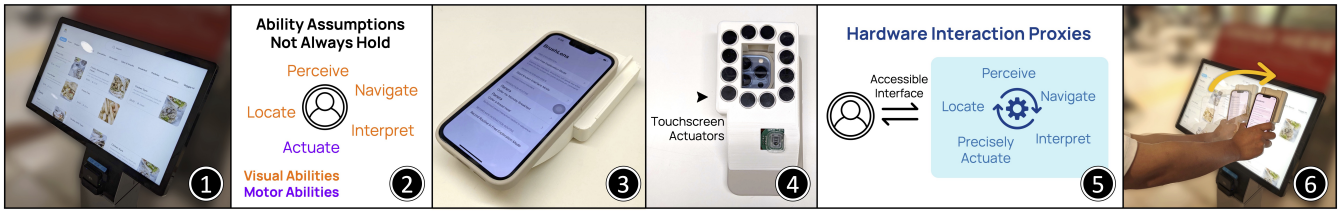


Figure 1: Touchscreen devices are widely adopted but not always accessible, such as restaurant kiosks (1). Assumptions about users' visual and motor abilities (2) to perceive visual information, locate, navigate, interpret interface layout, and precisely perform pre-defined gestures do not always hold, making devices inaccessible for some users. We propose BrushLens, a hardware phone case (3) that is equipped with an array of touchscreen actuators (4). It acts as a hardware interaction proxy that perceives, locates, and actuates touchscreen on behalf of users (5), and allows users to interpret and give interaction intentions through the accessible interface running on their personal devices. This allows users to "Brush" on the interface (6) while the actuators constantly monitor their positions through camera and sensor input, and perform a touch gesture directly if any of them is on top of the target button, making inaccessible devices accessible for people with diverse abilities.

ABSTRACT

Touchscreen devices, designed with an assumed range of user abilities and interaction patterns, often present challenges for individuals with diverse abilities to operate independently. Prior efforts to improve accessibility through tools or algorithms necessitated alterations to touchscreen hardware or software, making them inapplicable for the large number of existing legacy devices. In this paper, we introduce BrushLens, a hardware interaction proxy that performs physical interactions on behalf of users while allowing them to continue utilizing accessible interfaces, such as screenreaders and assistive touch on smartphones, for interface exploration and command input. BrushLens maintains an interface model for accurate target localization and utilizes exchangeable actuators for physical actuation across a variety of device types, effectively reducing user workload and minimizing the risk of mistouch. Our evaluations reveal that BrushLens lowers the mistouch rate and empowers visually and motor impaired users to interact with otherwise inaccessible physical touchscreens more effectively.

KEYWORDS

Touchscreen appliances, accessibility, interaction proxy, computer vision, touch actuation

ACM Reference Format:
Chen Liang, Yasha Iravantchi, Thomas Krolikowski, Ruijie Geng, Alanson Sample, and Anhong Guo. 2023. BrushLens: Hardware Interaction Proxies for Accessible Touchscreen Interface Actuation. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3586183.3606730

1 INTRODUCTION

Touchscreen devices have become ubiquitous in everyday life, playing a critical role in performing various everyday tasks. From flight check-in kiosks to food ordering systems, interacting with these devices independently has become essential in various usage scenarios. However, despite their widespread adoption, such devices are often designed with assumptions about users' abilities and interaction patterns, making them less accessible to completely inaccessible for users with diverse abilities. Individuals with visual impairments may have difficulty perceiving the necessary information to operate these touchscreens, and the risk of triggering unintended actions

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606730

due to mistouches during interaction may bring additional concerns while using them [28]. Furthermore, the precision and force required to register touch events on many touchscreen devices (especially resistive touchscreens) may pose additional challenges for individuals with motor impairments, such as tremor and spasm, thus limiting their ability to use these devices independently.

Prior research has explored various solutions to enhance the accessibility of touchscreen devices. For example, several approaches have been proposed to improve touchscreen interaction experience for motor impaired users through the concept of ability-based design [66]. These include modifying touch detection algorithms to better recognize users' input gestures [47] or adapting the interface to support additional input gestures [64]. However, these methods often require hardware or software modifications of existing touchscreen devices, making them inapplicable to a large number of inaccessible devices that are already in use. Other systems aim to make graphical interface content more accessible [14, 15, 67] and provide camera-based guidance for visually impaired individuals to navigate and actuate touchscreen appliances [26, 28]. However, these solutions still necessitate precise interactions from users, where the risk of mistouches remains a concern.

Inspired by prior work on software interaction proxies [71] and accessibility tools for touchscreen interactions [42], our work focuses on making the interaction more accessible and risk-free through the concept of *hardware interaction proxies*, which introduces an intermediary layer between the user and the inaccessible touchscreen devices. Specifically, the proxy should ***perceive and interpret*** the user interface (UI) and convert to an accessible format for users to explore, ***locate*** target UI elements and give meaningful feedback for users to navigate on the interface, and ***actuate*** the interface on behalf of the user. This allows users to interact with the device using a more accessible interface such as screenreaders and touch assistance on smartphones, delegating the tasks of interpreting, locating, and actuating inaccessible touchscreen interfaces to the proxy, ultimately reducing user workload and minimizing the risk of mistouch. Furthermore, the use of such a proxy ensures compatibility with a wide range of legacy touchscreen devices that are already in use, especially ones that are challenging to retrofit with accessibility functions or those that do not support other direct control or communication protocols, such as Bluetooth.

As an instantiation of this idea, we introduce BrushLens, a hardware interaction proxy to enhance access to various inaccessible touchscreen devices. BrushLens uses a model of the interface layout with real-time camera and sensor readings from the user's smartphone to locate itself relative to the screen, providing audio directional guidance for users to navigate through the interface. Through the BrushLens app running on users' personal smartphones, BrushLens provides an accessible interface that works seamlessly with users' existing assistive technologies running on their smartphones (e.g., screenreader and touch assistance). We demonstrate two possible actuators — solenoid and capacitive screen autoclicker — for BrushLens to support various types of touchscreens in daily life.

Through a technical evaluation and demonstration, we showed that BrushLens can run in real time, giving users timely guidance to navigate touchscreens and locate target UI elements. With a stationary testing setup, we demonstrated that BrushLens actuators can reach 100% success rate on activating touchscreens, and consistently activate screens within 2 pixels from the desired actuation position. Using two types of actuators, BrushLens can actuate capacitive touchscreens of different sizes, resistive touchscreens, and physical touchpads and buttons, showing its compatibility with various touchscreen devices in the wild.

Our user studies showed that BrushLens enables visually impaired users to interact with touchscreens that were previously inaccessible. Specifically, the autoclicker actuator made only a single mistouch over a total of 250 clicks, showing the competence of enabling risk-free touchscreen interaction for visually impaired users. Depending on users' different abilities and challenges, BrushLens additionally showed an effective reduction in the number of mistouches and inactive touches for motor impaired users with tremor or spasm of up to 73.9% percent (from 46 to 12 errors). Finally, users with low finger-sensitivity appreciated the additional haptic feedback provided by BrushLens when the screen was actuated.

Our results demonstrate that BrushLens effectively supports users with diverse abilities to interact with touchscreens that were inaccessible or challenging to operate before. By employing the concept of hardware interaction proxies, BrushLens provides a platform that bridges the gap between inaccessible touchscreens and accessible interfaces that users are already familiar with, allowing users to delegate challenging tasks to the proxy. We envision that, through additional technical effort, BrushLens could enable more accurate touchscreen actuation, which further brings BrushLens towards the goal of risk-free interaction, and can be portable enough for daily uses. The extendable design of the system creates possibilities of supporting additional user groups with diverse abilities, and the design of interchangeable actuators also opens up future directions of supporting more complex touch gestures, empowering BrushLens to be used in various daily scenarios.

2 RELATED WORK

Our work is built on prior research in touchscreen appliances accessibility issues, methods and systems for making touchscreen appliances accessible, diverse user needs of operating touchscreen devices and assistive technologies, and the idea of using interaction proxies to increase interface accessibility.

2.1 Touchscreen Accessibility Issues for Visually and Motor Impaired Users

Many touchscreen interfaces in the wild have accessibility issues that pose challenges for people with various abilities to use them independently [25, 34, 40, 61]. From perception to actuation, these accessibility issues arise from multiple aspects of the interaction procedure, thus requiring collective effort from all these aspects to make the interaction fully accessible [3, 63].

For visually impaired users, inaccessible touchscreen poses challenges in interface *perception*, *navigation*, and *actuation* [34, 37, 40, 52, 62]. The lack of multimodal feedback makes it challenging or impossible for blind people to explore and find UI elements [34]. Although both research solutions (e.g., Slide rule [34]) and commercial screenreaders (e.g., Apple VoiceOver [6] and Android TalkBack [22]) have already existed for a while, existing assistive technologies do not scale up very well for large touchscreens like kiosks [24, 37, 46]. The diverse touchscreen hardware and software also

makes it infeasible to modify and deploy new assistive technology on all devices. The input gestures presumed by touchscreen devices also bring potential accessibility concerns, as the interaction patterns differ from those of sighted users [34, 37, 62], and usually require additional space and time to perform, which could be incorrectly interpreted by touchscreens as other gestures.

Motor impaired users usually face additional challenges with interface *actuation* [4, 17, 25, 32, 45] due to the challenges of performing gestures presumed by touchscreens (e.g., extend the finger, sufficient fine motor control) [32, 47, 61]. Touchscreens may not correctly interpret various tapping gestures, such as palm or fist press, that motor impaired users may use to actuate the screen [47]. Conditions such as slow movements, tremor, or spasm also increase interaction errors [17, 18, 32, 47]. These factors significantly increase the risk (incorrect and invalid touches) and time to complete tasks compared to able-bodied users. These findings motivate our research to discover possible solutions to make existing touchscreen devices accessible for people with diverse abilities.

2.2 Methods and Systems for Making Touchscreen Appliances Accessible

Various works have been proposed to make touchscreen appliances accessible. Prior work on text recognition with computer vision [20, 60], UI elements detection on the display [14, 67], interface metadata generation [70], and expected interactions prediction [54] also created building blocks for touchscreen assistive technologies.

Some solutions proposed direct modifications of the touchscreen system. Slide rule [34] creates gesture mappings for visually impaired users to operate touchscreens. Smart Touch [47] collects motor impaired users' touch patterns and uses template matching to better detect diverse gestures. Other work models users' behavior in advance to adjust the system's responses to users' input [11, 48]. Accessible interface designs [64, 72] or ability-based personalized interfaces [21] also support varying abilities. However, the assumption that the target touchscreen software or hardware can be directly modified may not always hold, making them less applicable to be used on various touchscreen devices in the wild.

Other work explored external solutions. VizLens [26] detects users' finger and give audio guidance for navigation. Unfold and Touch [43] uses a foldable stand to activate touches. StateLens [28] reverse-engineers the underlying interface state diagram and guides users to operate the screen. However, the actuation problem remains unsolved, as users still have to carefully move fingers around the touchscreen to avoid mistouch and precisely activate the exact button. This becomes critical as visually impaired users may not notice a mistouch was triggered, and the precise actuation itself is challenging for people with tremor or spasm.

Standalone assistive devices, such as robots, are also explored. Take My Hand [53] uses the robot to take the users' finger to the destination. TouchA11y [42] uses a bot with extendable reel to reach and touch the target on behalf of the user. However, the required setup and the use of robots could potentially limit their practicality in everyday situations, and may introduce certain social and privacy concerns such as attracting unnecessary attention [2, 55].

These solutions motivate our work to focus on (1) minimum required modification of the touchscreen system and setup on users' side, (2) privacy and social considerations when used in public space, (3) diverse support of target touchscreen device types, and (4) interactions that require less precise motor control and localization.

2.3 Using Proxies to Improve Accessibility

The idea of proxies has been proposed in various domains. For example, a transformation proxy was used to make webpage more personalized without additional modification on either the original site or the client side [7]. For accessibility applications, content transformation proxy was used to transcode existing web pages and images without alt text [10, 58]. Other approaches reinterpret interaction behaviors before sending to the system, such as modifying pointing and cursor behavior based on users' ability [29, 30] and remap unimanual input into bimanual interactions in VR [69] for motor impaired users. Similar ideas have been used on physical interfaces as well, such as DIY braille overlay on inaccessible physical interfaces [27], automatically generated tactile maps as interactive overlays [23, 59] and data visualization [12]. Interaction proxy was also proposed to enhance mobile application accessibility [71] and touchscreen interfaces [36, 42], where the user can interact with a proxy interface layer that converts the user input to one or more inputs that can be interpreted by the inaccessible user interface. This leaves both the original application and the manifest interface unchanged, making it compatible to various use cases.

Inspired by these works, BrushLens applies the idea of interaction proxies [42, 71], but focuses on hardware interaction proxy that performs physical interactions. This additional middle layer between the user and the inaccessible touchscreen allows the proxy to still interact with the touchscreen in a way presumed by most touchscreens, but also support accessible interface for users to command and delegate the actuation to the hardware device. The goal is to bridge rather than make modification on either the device's or the user's end, bringing compatibility to a wider range of use scenarios. This also enables users to focus on gross movement for easier interaction, leaving the proxy to perform precise actuation.

3 DESIGN GOALS FOR HARDWARE INTERACTION PROXIES

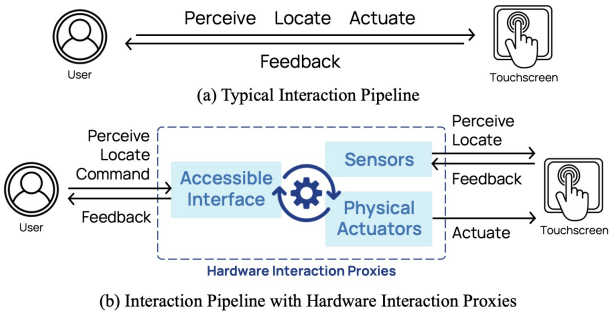


Figure 2: Typical touchscreen interaction pipeline (a) with ability assumptions between the users and the device, and the alternative pipeline with hardware interaction proxies (b), where the proxies bridge the assumption gap, and make the interaction procedure more accessible to the user.

The idea of hardware interaction proxies (which we referred to as 'proxies' below) aims to bridge users (and their preferred assistive technologies) and the target interaction device. For inaccessible touchscreens, a typical interaction pipeline (figure 2) requires users to perceive and interpret screen content, locate target interface elements, and actuate the screen accordingly. Screens will in return provide certain feedback for users' interaction. However, this pipeline is based on various ability assumptions, including assumptions made by touchscreen devices about users' ability to perceive, locate, and actuate interface elements, and assumptions made by users about devices' ability to interpret their interaction intentions, and give feedback that is accessible to them.

To bridge this gap about ability assumptions, hardware interaction proxies should act as a middle layer to transform information and intentions from each side, into a more accessible and interpretable format to the other side. Specifically, it should allow both sides to offload tasks that require ability assumptions to the proxies, which includes the following three major parts:

Perception and Localization: Proxies should effectively retrieve the information on the inaccessible touchscreen and represent them to the user in an accessible manner. It should also be able to locate interface elements, as well as itself, relative to the interface to provide useful and accessible feedback for users.

Actuation: Proxies should perform actuation on behalf of the users. Users should be able to initiate the actuation through accessible interfaces, and proxies should perform the actuation in a way that is interpretable by touchscreens. Collectively with the localization module, actuation should be performed precisely on the desired target to avoid interaction risks, and should response fast enough to offset the needs of fine motor control from the user.

Accessible Interface: Proxies should support accessible interfaces for people with diverse abilities and needs. Through the accessible interface, users should be able to retrieve target touchscreen interface information, get directional guidance and feedback, and interact with proxies. Information provided should be accessible based on users' needs and preserve privacy and social concerns.

Based on the findings from prior research, we summarized specific design considerations that hardware interaction proxy should aim to achieve throughout the interaction pipeline, including:

(1) **Risk-free Interaction:** Proxies should accurately actuate touchscreens, and reduce visually or motor impaired users' concerns about potential risks of mistouches [28, 47, 50, 61].
(2) **Support Various Interaction Needs:** Proxies should support various interaction needs, including different touchscreen technologies (e.g., capacitive/resistive touchscreens, physical touchpads) and gestures (e.g., tap, swipe) that can cover a wide range of accessibility needs in daily life [4].
(3) **Minimize Required Changes on Device and User Side:** Proxies should bridge inaccessible devices and users (and their accessible devices), requiring minimum changes to either of them. This allows higher compatible with various legacy touchscreen devices, and enables users to use assistive technology they already mastered for interaction, which reduces concerns of unable to find needed support for a specific hardware/software [46].
(4) **Privacy and Social Concerns:** Proxies should have similar or higher level of privacy as direct interaction to reduce privacy concerns of using assistive technology in public spaces by visually and motor impaired users[2, 35, 49]. It should also follow the social concerns and etiquette as required by the user (e.g., no loud audio feedback or noise) [2, 55].
(5) **Portable Form Factor:** Proxies should be easy to carry around, attach and detach to existing accessible devices, and requires minimum setup to reduce the concern of limited portability of the assistive technology [49].

In the following sections, we will refer back to these design considerations in our design process.

4 BRUSHLENS SYSTEM

We implemented the BrushLens system as an instantiation of the hardware interaction proxy mentioned in Section 3. As shown in figure 3, the BrushLens system contains three major components: processing unit, hardware phone case, and accessible user interface. The processing unit performs device *localization* and processes all sensor and interface data, the hardware phone case performs *actuation* on behalf of users, and the user's smartphone runs BrushLens app and provides an *accessible interface* for user to explore inaccessible interface and operate BrushLens. Implementation details and technical performance evaluations are described below.

4.1 Actuation

BrushLens aims to support risk-free interaction and various interaction needs. It uses an array of actuators (figure 4) for precise actuation. The hardware platform supports a wide range of power configurations for different actuators that meet various interaction needs. We demonstrate two possible actuator configurations below, and discuss additional interaction techniques in Section 6.4.

4.1.1 Hardware Structure. BrushLens focuses on responsive, risk-free interactions compatible with various touchscreen devices, such as capacitive touchscreen, resistive touchscreen, and physical touchpad. We integrate the hardware platform as a phone case to support actuation control, illustrated in figure 4. The case contains an Arduino Nano 33 IoT to control the actuators and receives commands through Bluetooth Low Energy (BLE). These commands control when and which actuator to activate. An I/O expander and a variable dual-side DC Boost converter (±3 to ±30V, 20W) are used to support different numbers and types of actuators for diverse actuation needs, including physical actuation (e.g., press-down microwaves touchpad) and digital actuation (e.g., capacitive touchscreens). A 7.4V 200mAh battery is used to power all components.

The hardware platform is engineered to support different *quantities* of actuators, which expands the area of actuation. As a result, BrushLens can activate the target button if any actuator is on top of it, which increases the likelihood of activation while the user moves the case, thereby reducing the need of precise motor control from the user as compared to having just a single actuator.

The adjustable power supply further broadens BrushLens compatibility with different *types* of actuators. With different supported operating voltage, BrushLens enables a wider selection of actuators that meet diverse interaction requirements. We will showcase two example configurations of actuators next.

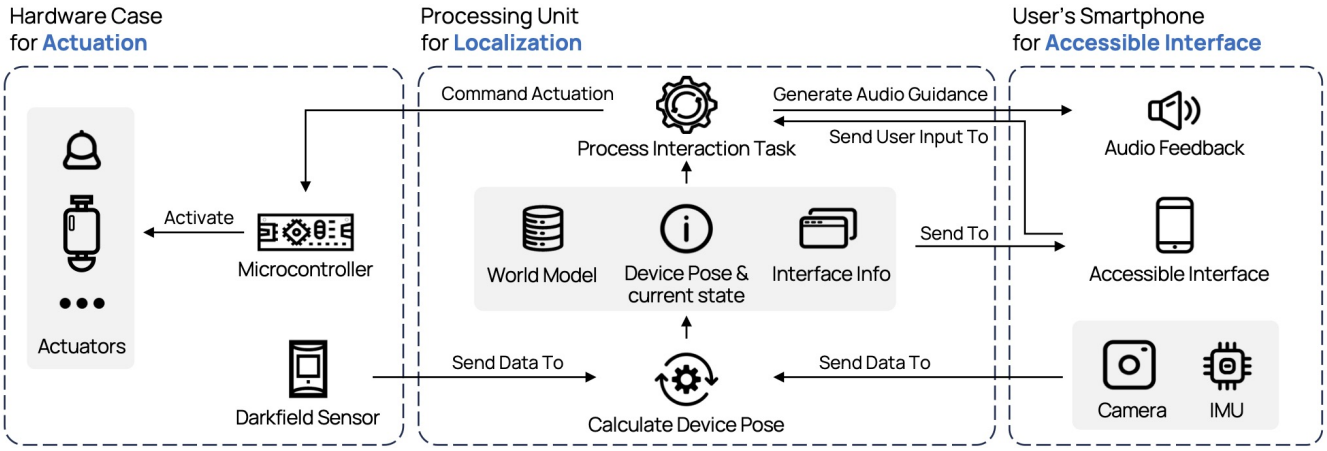


Figure 3: BrushLens overall structure. It shows an accessible interface on the user's smartphone for the user to explore the interface and give interaction intentions. It continuously sends data to the processing unit, which uses phone sensor data to locate the device relative to the screen and give directional information towards the target button. Once it decides a click can be safely performed, it will activate the corresponding actuator that is on top of the button to perform the touch.

4.1.2 Actuators. We show two possible actuator configurations — solenoid and capacitive screen autoclickers — for different interaction scenarios. Below we list the features of each actuator, and demonstrate how, by switching to different actuators, BrushLens can utilize these features to further optimize towards specific interaction requirements and users' needs.

Solenoids: Solenoid actuators can perform mechanical movements (i.e., push and pull) by supplying current to generate an electromagnetic force, which can perform a physical 'press' on various interactive appliances. While they can activate force sensitive interface (e.g., physical buttons and resistive touchscreen), a small modification is required to better actuating capacitive touchscreens. To achieve this, we add an additional conductive touch head using heat-shrink tube, as shown in figure 4. The head is made of conductive rubber and fiber, and its shape mimics a fingertip. All touch heads are connected to the BrushLens case and are grounded whenever the user is holding the case. As an example, we use small 4.5V solenoids as actuators (see figure 4), each sized $20 \times 12 \times 11$ mm. We fit eight solenoids around the BrushLens case to increase the actuation area. A custom PCB hosts the supporting hardware for the solenoids and makes the case more compact.

While a solenoid can simulate the single tapping gesture, it is comparatively larger and heavier to bring with the user. Since most touchscreen devices are capacitive, we demonstrate another actuator that specifically actuates capacitive touchscreens and is lighter and thinner, which we describe below.

Capacitive Touchscreen Autoclickers: Autoclickers alter the capacitance of the screen in the location they are in contact with the screen, thereby simulating a fingertip touch [38, 44]. Due to these properties, they can only be used on capacitive screens and must stay in contact with the screen for valid actuation. However, with no moving parts, the clicker can activate almost instantly after turned on, resulting in improved responsiveness compared to the solenoid. The clicker can perform up to 35 clicks/sec, enabling BrushLens to actuate screens more responsively and accurately even under fast movement or to click small buttons. Since the clicker is a solid-state component, it opens up the possibility of being integrated into smaller form factors (e.g., custom PCB) that makes the BrushLens smaller and lighter to be carried around for daily use cases.

We build a BrushLens case with 12 clicker actuators (figure 4 bottom). The boost converter provides ±26V DC to power the clickers, and the I/O expander connects all clickers to the Arduino that activate the corresponding clicker to perform touches.

4.1.3 Performance Evaluation. Below we evaluate BrushLens actuation performance, including actuation responsiveness, accuracy, consistency, and device compatibility. These results are from the technical evaluation of the system, and additional user evaluations can be found in Section 5.4.

Actuation Delay: We measure the actuation delay to demonstrate the current system responsiveness. The delay starts from when the an actuation command was sent to the Arduino through the BLE protocol and ends when the touchscreen recognizes the actuation. This demonstrates how fast BrushLens can react after the target object has been recognized and the actuation is triggered. The shorter the delay, the more accurately it actuates buttons, and the faster the user can brush on the touchscreen without accidentally overshooting or missing the target button.

We use BrushLens to perform 300 actuations on a capacitive touchscreen to measure the actuation delay, with a random sleep time between 0.5s - 1.5s between actuations to simulate the user input. As shown in figure 5, the solenoid delay ranges from 37ms to 148ms ($\mu = 75.27, \sigma = 19.56$), and 27ms to 107ms for the clicker ($\mu = 61.54, \sigma = 18.12$). Due to the mechanical movement, solenoid inevitably has higher actuation delay, which is a tradeoff to consider between responsiveness and device compatibility (which we will describe next). The distribution is skewed right, with most delays between 30ms - 100ms, showing that the actuation module can process and perform actuations in a timely manner, allowing BrushLens to quickly reacts to detected buttons and user movements.

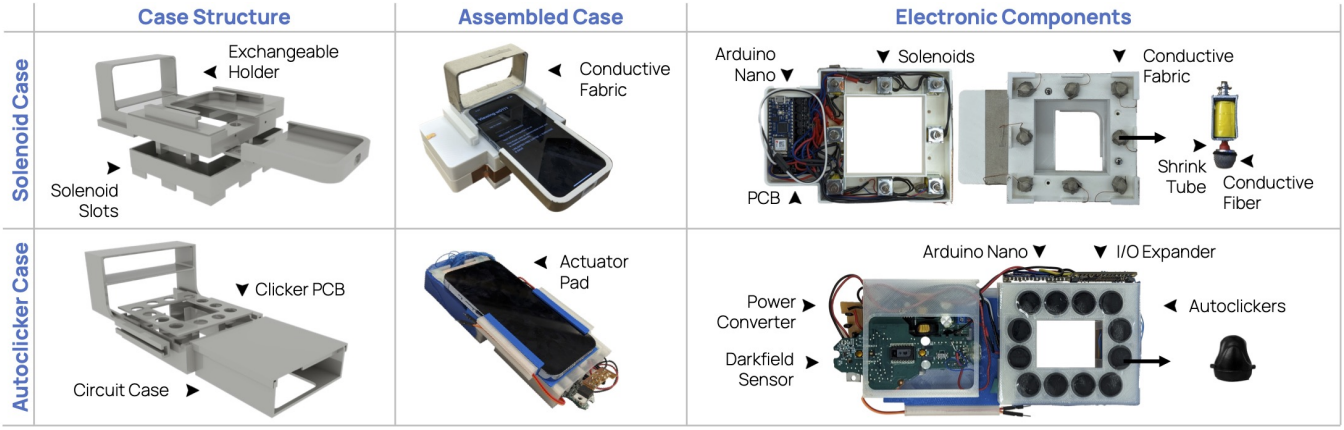


Figure 4: Solenoid and autoclicker cases with images of case rendering, the assembled case, and electronic components inside.

Device Compatibility: We tested solenoid and autoclicker on different touchscreens with various touch technology used in daily life. The compatibility result is summarized in figure 7.

Autoclicker: Although autoclickers can only activate capacitive touchscreens, we show that it works on various types and sizes of capacitive touchscreen devices, including large monitors (Tabletop Phillips 242B9T Touchscreen Monitor, 24"), tablets (SAMSUNG S7 FE, 12.4", Amazon Fire HD 10, 10.1", Apple iPad Pro Gen 4), portable devices (Google Pixel 5, 6"). We expect these devices to be representative for a broad range of touchscreens found in the wild.

Solenoid: The physical push enables solenoids to actuate various types of touchscreen appliances that support press/tap gestures. We tested BrushLens with solenoid actuators on three different touchscreen technology that covers the majority of appliances in daily life, including (1) capacitive touchscreen, (2) resistive touchscreen, and (3) physical touchpad and button. It shows that touches were successfully registered by these devices, representing a high compatibility of using solenoids to actuate touchscreens.

Touch Registration Rate: We analyzed the registration rate of both actuators on the capacitive touchscreen to evaluate the false positive and false negative rates for actuations. We positioned the actuator at the same place on the touchscreen, performed 100 consecutive actuations, and counted the number of touches that were registered. Both the solenoid and autoclicker were able to reach 100% precision and recall on actuating the screen. These results show that the actuators are able to effectively register touches without misses and do not trigger unintended activation.

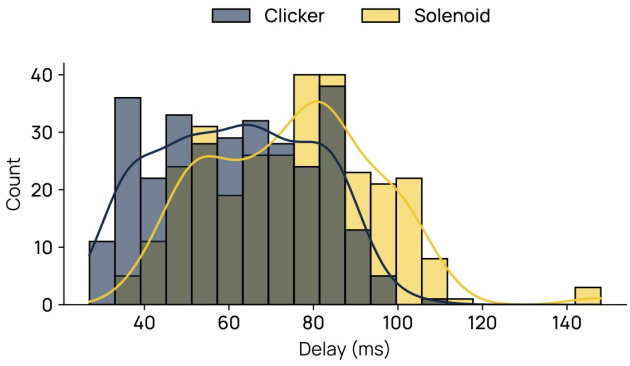


Figure 5: Histogram and kernel density estimate plot of the system actuation delay (in milliseconds).

Actuation Location Consistency: We evaluated the actuation location consistency of both actuators by checking the variance of registered touch locations given a fixed physical actuation position. This is to verify whether the actuator can consistently actuate the exact same place on the touchscreen as intended, which is critical for precise actuation. We used both actuators to press the same location 300 times, measured the registered touch locations, and analyzed the distribution of the location offset between the desired and the registered locations on the screen. Figure 6 shows that the position x offset is within ±1 pixel, and y offset is within -1 to 2 pixel range, where 88.5% of touches are within 1 pixel (L1 distance) than the desired location, showing a high accuracy and consistency using solenoids and clickers to register touches on the touchscreen.

4.2 Localization

BrushLens uses a processing unit for device localization, which processes all sensor data, including camera, IMU, and darkfield sensor, to collectively calculates the device pose relative to the interface screen. This is critical for BrushLens to determine where and when to activate actuators, and generate informative guidance for users to move the device towards the target.

4.2.1 Software Structure. To determine the pose and position, BrushLens retrieves the camera and IMU data from the user's smartphone, and sends it to the processing unit via wireless communication for processing. The processing unit maintains a model of the target interface, including images of each user interface, type and location of UI elements, and transitions between interface states.

We would like to emphasize that, to verify the idea of hardware interaction proxies, which is the main contribution of this project, we assume that the interface model is known to the processing unit, and choose not to reproduce various prior work on UI perception that proves the possibility of public interface reverse engineering,

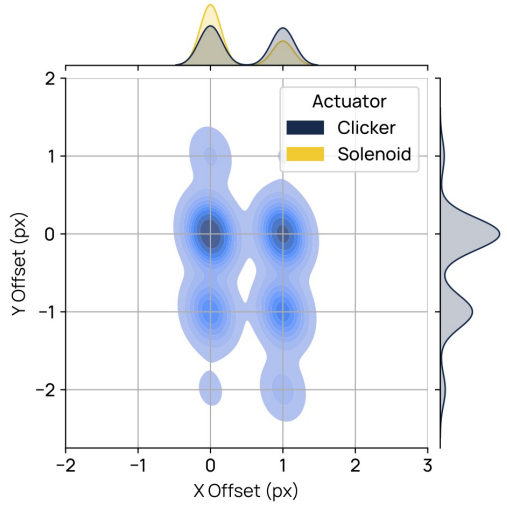


Figure 6: Kernel density estimation of registered touch position offset in pixels on a 24" touchscreen with 93 PPI.

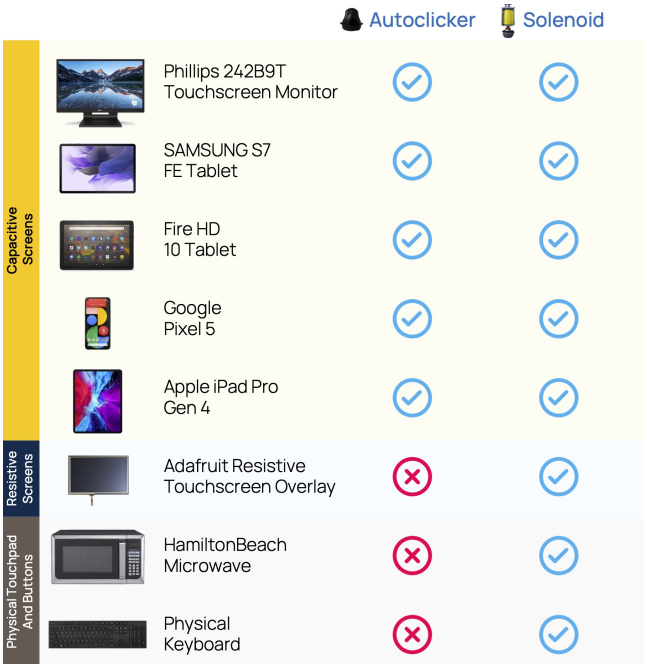


Figure 7: Actuator compatibility with common touchscreen appliances with different touchscreen technologies.

automatic (graphical) UI element recognition, and expected touchscreen interaction prediction [13, 16, 28, 54, 57, 65, 68]. Similar assumption was also made by prior work on making inaccessible touchscreens accessible, such as TouchA11y [42]. When needed, these can be incorporated as part of the interaction pipeline, which we will further elaborate in Section 6.3.

As shown in figure 8, using the current camera view, the processing unit attempts to match the camera view to the interface image to determine the BrushLens case' position, and uses the smartphone's IMU to update the device pose. In our implementation, we separated the computing program from the mobile client to a separate desktop device for idea verification and visualization purposes. Whenever needed, it can be incorporated back as a stand-alone iOS app running on users' smartphone without external device support.

Our system includes the following key components for precise localization:

Feature Matching: We use Speeded Up Robust Features (SURF) for feature detection utilizing its advantages of speed, scale-invariance, and robustness against image transformations [8]. This allows the program to find the potential match of the camera view within the interface image in real time, and be able to handle interfaces with different sizes and view angles.

Touchscreen Size Estimation: To precisely determine the physical location on the screen, BrushLens estimates the physical size of the current screen, and converts pixel (px) results (interface reference frame) into millimeters (mm) (physical screen reference frame). When the device is on the touchscreen, given the camera Field of View (FoV), the case height, and the transformation matrix of the camera view to the interface image, the program calculates the ratio of converting pixel units to mm. This is then used to determine the physical position of the BrushLens and its actuators on the screen, which is essential for correct actuation on screens of different sizes.

Onscreen Detection: To avoid falsely triggering actuators in midair, the program continuously monitors the 3D pose of the device, making sure it is in touch with the screen. This is determined by (1) the transformation matrix of the feature matching results, checking whether there exists a significant perspective transformation, and (2) smartphone IMU readings, checking whether the device is suddenly tilted along X or Y axis. Whenever the device is tilt or lifted, the program pauses, notifies, and waits the user to place the device back onto the touchscreen to avoid invalid touches.

World Model: Combining device location, pose, and interface information, the program constantly updates its world model to make decisions upon users' requests. This includes deciding when and which solenoid will be triggered, generating directional guidance, and decomposing high level commands into a sequence of actuations based on interface transitions.

Actuation Control: BrushLens tracks the location of all the actuators through its world model, and if any actuator is within the target actuating area, it will send the command to the Arduino through BLE protocol. To further avoid overshoot or early click, a dynamic unsafe boundary (10% of width and height) is set within the actual target area, and the actuator is triggered only if it is inside the safe area (i.e., not near the edge of the target area).

4.2.2 Sensor Choices for Accurate Localization. To better optimize the localization accuracy, we use the macro lens camera on iPhone 13 Pro with smaller minimum focal distance for clear close-up view, and darkfield sensor for precise movement tracking on reflective and transparent materials, which are described below:

Camera with Smaller Minimum Focal Distance: We use an iPhone 13 Pro as it has an ultra-wide camera with a smaller minimum focal distance of 20 mm [5]. This improves camera focus on the touchscreen display, having a clear view of the interface

Figure 8: BrushLens maintains a world model of the interface and device location in both the touchscreen interface coordinate (in px) and touchscreen device physical coordinate (in mm). The interface on the left reveals the current state BrushLens calculates, and the image in the middle is the camera view, which has a narrow FoV due to close distance to the touchscreen interface, as shown in images on the right (38mm for the solenoid case and 18mm for the clicker case).

even though the BrushLens pad thickness is only 18 mm, which further benefits the quality of feature detection.

Darkfield Sensor for Precise Movement Tracking and Better Touchscreen Compatibility: Since a matching of SURF results is not guaranteed in cases with insufficient visual features (e.g., solid background, small FoV), we use a darkfield sensor [1], which has been used in commercial laser mouses, to track device movement to supplement device localization in featureless situations. When a movement is detected, the sensor sends 2D movement deltas to the processing unit via Bluetooth to update the BrushLens position. It also works on different types of surfaces, including glasses or other reflective surfaces that are commonly used with touchscreen kiosks but are challenging for regular optical sensors to track movement. This makes BrushLens robust to a wider range of interfaces and touchscreen devices in daily scenarios.

4.2.3 Performance Evaluation. Since prior work has evaluated SURF for localization [28, 42], we did not reproduce rigorous evaluation of localization accuracy. To simulate a typical use case, we moved the BrushLens to a sequence of 20 random points on a 24" food ordering interface, it shows an average of 47 px (σ = 34.27) (12.8mm (σ = 9.36)) error between the actual and the estimated position. Notice that the device is moved to the 20 points in sequence, thus the error may accumulate while moving, until a feature matching is found to reset the error. This shows the feasibility of using the current system setup for localization, while demonstrating possible technical improvements for better accuracy.

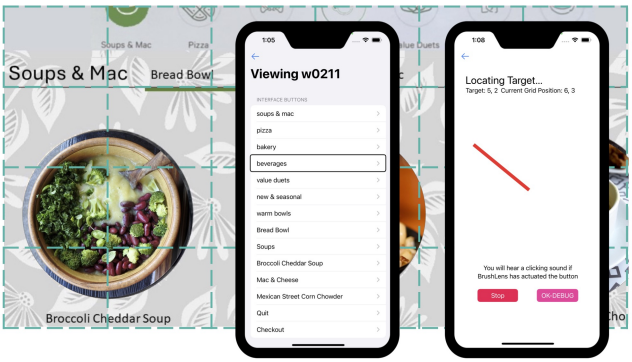
4.3 Accessible Interface

With the design goal of supporting various interaction needs, BrushLens supports accessible interfaces for users with diverse abilities to explore and interact with inaccessible touchscreens. We demonstrate three example interfaces for visually and motor impaired users to show the feasibility of supporting additional assistive interface through the BrushLens interface.

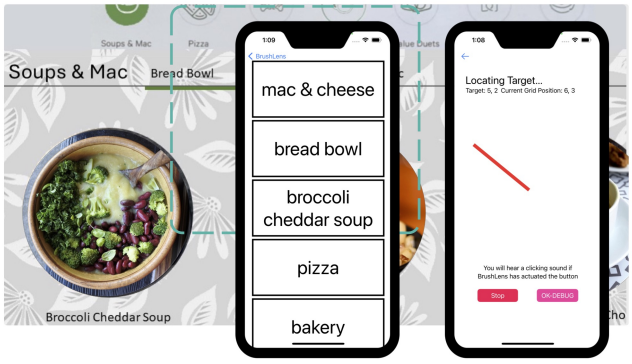
4.3.1 Visual Ability Support. BrushLens supports an interface exploration mode (figure 9a) that lists UI elements of the target touchscreen on the users' smartphone, allowing users to use assistive technologies, such as VoiceOver, to explore inaccessible interfaces. With the interface model, the processing unit sends the JSON file of the UI layout, including clickable buttons and labels, to the mobile BrushLens app. User can explore the options and select the button they want to press, and the BrushLens app will give audio guidance for the user to move towards the target button on the touchscreen.

For audio guidance, we use a grid system as an example to demonstrate how audio feedback could give users sufficient movement guidance. Other guidance systems used in prior works [26, 28, 51, 56] can also be integrated into the BrushLens app. To create the grid guidance system, the processing unit divides the touchscreen interface into adjustable grids, which is preset to 1"×1" in our implementation. The number of rows and columns are determined by the actual screen size measured by the BrushLens' localization system. This ensures that user can maintain a constant mental model of the cell size and map the grid to touchscreens of different sizes. The processing unit constantly sends the grid coordinate of the target button and the current device location to the BrushLens app, which speaks out these coordinates for the user to move the device accordingly. Upon actuation, BrushLens app initiates a vibration and also an audio description saying which button was pressed, to keep the user notified about BrushLens' actuation.

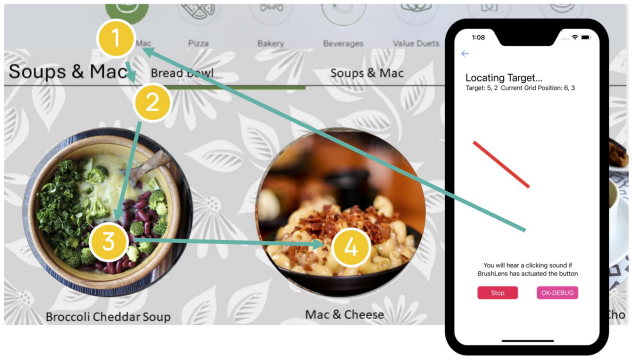
4.3.2 Motor Ability Support. BrushLens supports a button magnifier interface for motor ability support, and provides an extendable slot for additional physical accessories to be attached for motor impaired users to more easily move BrushLens device on touchscreens. Given the wide range of needs and assistive technologies for motor impaired users (e.g., physical switches, sip-and-puff devices), below we present some examples that would be helpful for people with spasm, tremor, or problem with grasping, and show how BrushLens could connect to other assistive solutions.



(a) Exploration Mode (Left) and Grid-Based Audio Guidance (Right) with VoiceOver



(b) Button Magnifier Mode Showing Nearby Buttons (Left) and Directional Guidance (Right)



(c) Interaction Routine Mode for Sequential Button Input

Figure 9: Screenshots of BrushLens app for visual (a), motor (b), and frequent input support modes (c).

Button Magnifier Mode: When the target button is small and in a densely packed area, users may need multiple touches for the touchscreen to register the touch on the button, or may mistakenly trigger undesired nearby buttons. As shown in figure 9b, to reduce these limitations, BrushLens acts as a button magnifier, detecting buttons near the BrushLens device, and projecting them at a larger scale on users' smartphones, allowing users to use additional assistive technology (e.g., physical switch access, assistive touch) and more easily select the button they want to press. As shown in figure 9b right, once the user selects the target, BrushLens shows an directional arrow pointing to the target button, where the length is proportional to how far the button is. The user can roughly move towards that direction, and BrushLens continuously monitors the precise position of all actuators and the actuation target, which allows the user to delegate the job of doing fine motor control and performing touchscreen-assumed gestures to the BrushLens proxy.

Exchangeable Handles: Users can attach additional handles or grips through the sliding rail on the side of BrushLens to support various personal needs. As an example, we 3D-printed the flexible handle using soft PLA for users to grab or slide-in their hands to move the device, giving users more ways of using BrushLens.

4.3.3 Frequent Usage Support. Touchscreen interactions have been shown to take much more time for visually and motor impaired users [26, 28, 42]. This additional inefficiency comes from not only the interaction itself, but also the time spent on repetitive steps, such as listening to the menu description every time. This becomes unnecessary if the user is already familiar with the interface layout, and want to perform frequent interaction routines. Interactions that require sequential input can easily magnifies this efficiency difference, causing additional frustrations to users. This includes interactions that require multiple touches, or typing on the virtual keyboard (e.g., credit card number). In these cases, touching individual buttons not only takes more time and effort, but also introduces potential privacy concerns for sensitive information.

To further simplify the interaction experience, BrushLens supports interaction routine mode, as shown in figure 9c. Similar to interaction routine in other domains, this allows users to record a sequence of input (e.g., phone number, or a list of interactions to order a bread meal) to be reused later. With a saved interaction routine, BrushLens can directly navigate users through a sequence of buttons to press, saving the time needed to explore the menu and select the button for each button press.

5 USER EVALUATION

We evaluate how BrushLens supports users with various abilities in accessing inaccessible touchscreen devices with the following research questions:

- RQ1: Can BrushLens support users interact with touchscreen devices with common interaction tasks?
- RQ2: Can BrushLens make it possible for visually impaired users to use touchscreen devices?
- RQ3: Can BrushLens improve the interaction experiences for motor impaired users to use touchscreen devices?
- RQ4: How effective are solenoids and clickers in facilitating access to touchscreen devices?
- RQ5: What are the user experiences with BrushLens, and what possible improvements could be made to the system?

5.1 Participants

We recruited 10 participants (6 female and 4 male) with visual (B1 to B6) or motor impairments (M1 to M4). Self-reported visual impairments include blindness, light perception, and low vision, and self-reported motor impairments include low finger sensitivity or

Table 1: Participant demographic information. In total, there are 10 unique participants and 14 trials. Four participants used both solenoid and autoclicker actuators for additional comparison and feedback on their user experience. The difficulty using touchscreens is self-reported on a 7-point Likert scale (1 - Strongly disagree and 7 - Strongly agree).

ID	Age	Gender	Group	Difficulty Using Touchscreens	Self-reported Impairments	Actuator Used
B1	59	Male	BVI	7	Blind	Solenoid (S4), Clicker (S11)
B2	70	Female	BVI	7	Light perception	Solenoid (S5), Clicker (S13)
B3	38	Male	BVI	7	Blind	Solenoid (S6), Clicker (S8)
B4	61	Male	BVI	7	Blind	Solenoid (S7)
B5	71	Female	BVI	7	Blind	Clicker (S12)
B6	59	Female	BVI	7	Low vision	Clicker (S14)
M1	60	Female	Motor	2	No finger sensitivity, low finger dexterity	Solenoid (S1)
M2	33	Female	Motor	5	Cerebral palsy, spasm	Solenoid (S2), Clicker (S10)
M3	53	Female	Motor	2	Post-polio paraplegic, upper body limitations (reach)	Solenoid (S3)
M4	71	Male	Motor	5	Tremor, spasm, slow movement	Clicker (S9)

dexterity, cerebral palsy, tremor, spasm, and limited reach. Six participants used BrushLens with either the solenoid or the autoclicker, and four participants used both actuators across two study sessions for additional comparison. Participants' demographic information are shown in table 1. We conducted 7 solenoid sessions and 7 autoclicker sessions, which are numbered as S1 to S14 and listed in the last column in the table for corresponding participants.

5.2 Apparatus

We used a 24" Phillips 242B9T capacitive touchscreen monitor, positioned flat on the adjustable table, to simulate a tabletop public kiosk. Upon participants' request, the monitor could be adjusted within ±30 degree pitch angle. Participants were provided with an iPhone 13 Pro as the accessible personal device, which was attached to the BrushLens hardware case. It ran iOS 15.6.1 with VoiceOver and Assistive Touch enabled, and could be customized upon participants' request. The phone is connected through a local wireless access point to the BrushLens processing unit, which was running on a laptop with Intel Core i7 9750H CPU.

5.3 Procedure

The studies were conducted in person at the authors' institution, and took approximately 2 hours per session. Participants were each compensated $50 and reimbursed for their transportation costs. The study was approved by the institution's IRB.

The study started with a brief pre-survey about demographic questions, prior touchscreen experiences, and self-rated difficulty interacting with touchscreen devices in the wild, which are reported in table 1. Next, participants familiarized themselves with the touchscreen kiosk, the provided iPhone and its accessibility settings, and the BrushLens hardware and user interface.

To simulate real usage scenarios, participants were asked to perform four types of tasks. Motor impaired participants were first requested to perform tasks independently without BrushLens, and then use BrushLens' Button Magnifier Mode (figure 9b) to perform tasks again for comparison. Visually impaired participants used the interface exploration model (figure 9a) with VoiceOver to complete the tasks. As directly accessing the touchscreen is impossible and inappropriate, we avoided the independent interaction tasks for visually impaired users. The task details are listed below:

(1) **Different-sized Button Clicking Task:** Participants were asked to use BrushLens to actuate 50 buttons of 5 different widths (10mm, 20mm, 30mm, 40mm, 50mm) and 5 movement amplitude (10cm, 20cm, 30cm, 40cm, 50cm) to evaluate the actuation accuracy and success rate for various button sizes at random locations on the screen. Note that we specifically included the small 10mm buttons, which is uncommon on large kiosks, to evaluate whether users could use BrushLens to actuate buttons with minimum suggested width [41], but has shown to cause much more inaccurate touches for users than larger buttons, especially for motor impaired users [18].
(2) **Food Ordering on a Reproduced Bakery Restaurant Kiosk:** We reproduced the kiosk interface of a bakery restaurant, and asked participants to use BrushLens to explore the interface, make a selection, and use BrushLens to actuate the button following its guidance. Participants were given 5 menu items to order, which required a total of 18 button clicks. This is to evaluate whether BrushLens can support users to independently complete tasks similar to those in daily use cases, and to get user feedback on system usability.
(3) **Typing Task on Virtual QWERTY Keyboard:** Participants were asked to type two given strings on the touchscreen using the virtual on-screen QWERTY keyboard, which contain both letters and numbers. This is to evaluate the effectiveness of BrushLens for typing tasks, which has been shown to be challenging and time-consuming for visually and motor impaired users [9, 39].
(4) **Interaction Routine for Sequential Input:** Participants were asked to use BrushLens interaction routine mode to order two lists of items (13 and 10 button clicks respectively). The goal is to get user feedback on the effectiveness and usability of the automatic sequential input feature.

After each task, participants rated the learnability, comfort, usefulness, satisfaction, and independent use along a 7-point Likert scale (1 for strongly negative and 7 for strongly positive), and provided feedback on their experiences. At the end, we conducted a semi-structured interview to get additional feedback regarding the

overall user experience and potential improvements for BrushLens. The studies were video and audio recorded for further analysis.

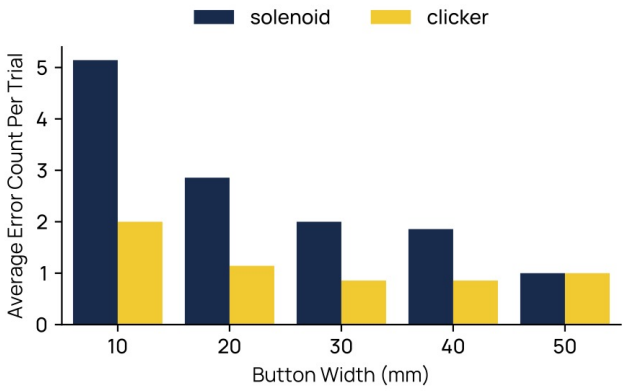


Figure 10: Distribution of click errors (misclicks and inactive clicks) for different button sizes (in mm).

5.4 Results

We summarized the results based on the research questions mentioned previously, and we additionally use the following evaluation criteria to analyze BrushLens' actuation performance, including:

(1) **Misclick:** Touch gesture was performed, but landed outside of the target area, or recognized as valid gesture but interpreted as other gestures by the screen (e.g., press and hold). Misclick rate is the number of misclicks over all the touch gestures performed.
(2) **Inactive:** Touch gesture was performed, but failed to be recognized as a valid gesture by the screen (i.e., no response at all from the screen). Inactive rate is the number of inactive touches over all the touch gestures performed.
(3) **Accuracy:** Number of correct actuation over all actuations performed.

5.4.1 RQ1: Can BrushLens support users interact with touchscreen devices with common interaction tasks? **Participants were able to use BrushLens to perform tasks similar to those in daily use scenarios.** All participants were able to complete the food ordering task (task 2) and interaction routine task (task 4) successfully, with a 100% completion rate. This indicates that BrushLens can successfully assist users with visual and motor impairments to perform common touchscreen interaction tasks similar to daily use scenarios. For other tasks, BrushLens presented varied benefits and performance for different users and situations, which we will evaluate in detail in the following sections.

5.4.2 RQ2: Can BrushLens make it possible for visually impaired users to use touchscreen devices? **BrushLens enabled visually impaired users to navigate and actuate inaccessible touchscreens.** All visually impaired participants reported in the pre-survey (table 1) that touchscreens in the wild are completely inaccessible for them to use independently unless with additional accessibility features. Participants reported that they could only ask sighted people for help, either in person or through online services such as Be My Eyes [19], to use inaccessible touchscreens, which may take a long wait time to get such help (B2), and the volunteers may not be skilled enough to provide helpful or timely guidance (B2, B5). Other approaches such as putting Braille overlays (B1), using prior experiences about UI layout (e.g., back button is usually at the upper left corner) (B1, B3), or using other accessible devices (e.g., Braille keyboard) (B2), are either not practical to be used in public space, or too risky to trigger unintended interactions. Using BrushLens, all visually impaired participants were able to perform the 4 tasks that were impossible to do before, and were able to explore UI elements, navigate to the target locations, and actuate the touchscreen through the hardware interaction proxy.

The analysis of touch accuracy in task 1 shows that using BrushLens, visually impaired users can reach an average of 84.2% actuation accuracy, which is 76.5% for the solenoid group and 90.4% for the clicker group. Specifically, visually impaired users who used the clicker actuator made only 1 misclick in a total of 250 touches over all 5 sessions, showing the effectiveness of BrushLens in reducing accidental touches, which is a major concern from both the study participants (B1, B3) and from prior studies [28, 61].

We further analyzed the user experiences of visually impaired participants through responses from Likert scale questions, where detailed responses can be found in figure 11. With an average score of 6.4 (6.2 for task 2 ($\sigma = 0.83$), 6.7 for task 3 ($\sigma = 0.5$), and 6.3 for task 4 ($\sigma = 1.65$)), **participants found BrushLens very useful for performing touchscreen interaction tasks, specifically for keyboard typing tasks.**

> *"I really feel confident for this task. [Using BrushLens for this task] is fantastic, brilliant. Having the idea of the keyboard layout in my head is helpful, [...] and it is pretty easy to do [the typing task]." (B3)*

5.4.3 RQ3: Can BrushLens improve the interaction experiences for motor impaired users to use touchscreen devices? **BrushLens reduced the number of misclicks and inactive touches for motor impaired users, depending on users' abilities and needs.** While BrushLens enabled visually impaired participants to navigate and actuate various buttons that were impossible before, the benefit and performance varied for motor impaired people, which we further analyzed below.

All participants with motor impairments reported facing challenges using touchscreen devices in the wild; however, the specific challenges and corresponding needs varied. M2 was not able to perform the gestures pre-assumed by most touchscreens, and needed multiple touches for the screen to recognize the gesture. The spasm condition also reduced the touch accuracy and caused additional mistouches. For M2, BrushLens was able to reduce error touches by a large percentage of 73.9% from 46 (10 mistouches, 36 inactive touches) to 12 (7 mistouches, 5 inactive touches), which greatly decreased the total number of additional touches needed for a successful actuation. As M2 mentioned:

> *"I felt without the case, I kept tapping. With the case, it was easier because it tapped for me."* (M2)

Table 2: Performance data. M: Misclick (touch outside of button, or interpreted as other gestures, e.g., hover, long press). I: Inactive (touched but not recognized by screen). Acc: BrushLens Accuracy. SM/SI: Misclicks/Inactives when interacting by participants themselves. BM/BI: Misclicks/Inactives with actuators. SID: Session ID. Rate is calculated over 50 clicks per trial.

Actuator	Group	SID	SM	SI	BM	BI	MRate	IRate	Acc	Group MRate	Group IRate	Group Acc	Total MRate	Total IRate	Total Acc
Solenoid	Motor	S1	0	0	9	7	0.18	0.14	0.68	0.193	0.093	0.713	0.151	0.105	0.742
		S2	10	36	7	5	0.14	0.10	0.76						
		S3	3	5	13	2	0.26	0.04	0.70						
	BVI	S4	/	/	4	5	0.08	0.10	0.82	0.120	0.115	0.765			
		S5	/	/	8	4	0.16	0.08	0.76						
		S6	/	/	6	6	0.12	0.12	0.76						
		S7	/	/	6	8	0.12	0.16	0.72						
Clicker	Motor	S9	0	4	4	3	0.08	0.06	0.86	0.060	0.110	0.830	0.020	0.097	0.883
		S10	7	26	2	8	0.04	0.16	0.80						
	BVI	S8	/	/	1	6	0.02	0.12	0.86	0.004	0.092	0.904			
		S11	/	/	0	5	0	0.10	0.90						
		S12	/	/	0	6	0	0.12	0.88						
		S13	/	/	0	4	0	0.08	0.92						
		S14	/	/	0	2	0	0.04	0.96						

We additionally evaluated the input strings from S9 and S10 by participants M4 and M2 in task 3, as both participants faced challenges typing on virtual keyboard (task 3) due to tremor or spasm, especially when buttons are densely arranged. We use Levenshtein distance to compare participant-typed and BrushLens-typed string with the ground truth to evaluate string differences. The average distance was 5.5 ($\sigma = 2.69$) for user-typed strings, and 4.0 ($\sigma = 2$) for BrushLens-typed strings. Due to the limited sample size, a statistical significance of the string distance difference cannot be concluded. However, it shows the potential of improving input string accuracy through the hardware interaction proxy by reducing mistouches.

M1 struggled to determine if a button has been pressed due to a lack of fingertip sensitivity, particularly when using devices that require force for actuation, such as resistive touchscreens or physical touchpads. M3 experienced difficulty reaching buttons on large devices or those positioned far away due to upper body limitations. For both M1 and M3, the challenges predominantly revolved around the larger gross motor movements needed prior to actuation or the feedback after actuation, rather than the accuracy of performing the touch gesture within a specific range. Since the current BrushLens prototype is primarily optimized to reduce challenges with fine motor control, it did not offer significant additional benefits on interaction accuracy, as demonstrated in table 2.

However, BrushLens still offered certain advantages beyond enhancing touch accuracy. As M1 noted, the solenoid's physical actuation, combined with the clicking sound and vibration of the BrushLens phone case, gave additional "tactile feedback of having the buttons pushed" (M1). This was particularly appreciated by M1 and M2. M3, who used a reacher daily, found the reacher incompatible with capacitive screens and difficult to use on hard physical buttons. She envisioned the possibility of "attaching the case to the end of the reacher, so it can press buttons for me" (M3).

5.4.4 RQ4: How effective are solenoids and clickers in facilitating access to touchscreen devices? **Both the solenoid and the clicker were capable of actuating touchscreens, with a respective average clicking accuracy of 74.2% and 88.3%.** We examined the performance using results from the button-clicking task (task 1), as it encompasses a broad spectrum of button interaction situations.

Table 2 summarized the actuation performance from task 1. While both actuators can reach 100% stationary actuation accuracy in the technical evaluation, the user evaluation results show that solenoid actuators reached a clicking accuracy of 74.2%, and clicker actuators reached 88.3% during movement. This shows that both actuators, in their current states, still require further improvements to be sufficiently robust to users' diverse usage and movement patterns. Specifically, as we have observed during user studies, the actuators were more likely to cause errors when the case was being moved too fast. This is because the overall delay, including system delay and mechanical actuation delay, was not short enough for the actuator to touch the button within its range. As previously shown in figure 5 in the technical evaluation, solenoids had higher actuation delay in general. This usually caused overshoot and led to mistouches while users were moving the case. As shown in figure 10, this was true especially for smaller buttons, as it requires higher precision and provides less touch error tolerance. Slightly lifting or tilting the case while the actuators were activated could cause inactive touches due to a lack of contact with the screen. All these reveal that, while the actuators were capable of performing touching gestures most of the time, additional technical effort is needed to make it more responsive and robust to various use cases.

While both actuators had similar inactive rates, clicker had significantly (t-test, $p < 0.01$) lower misclick rate than solenoid. Also, as shown in figure 10, clicker caused fewer errors on average, and had a more uniform error distribution among different button widths. In comparison, solenoid had more errors overall, especially on smaller size buttons. This also echoes our findings in the technical evaluation, showing that the mechanical movement brings additional delay after a command is initiated, which makes solenoids less robust to users' fast movement that may cause mistouches.

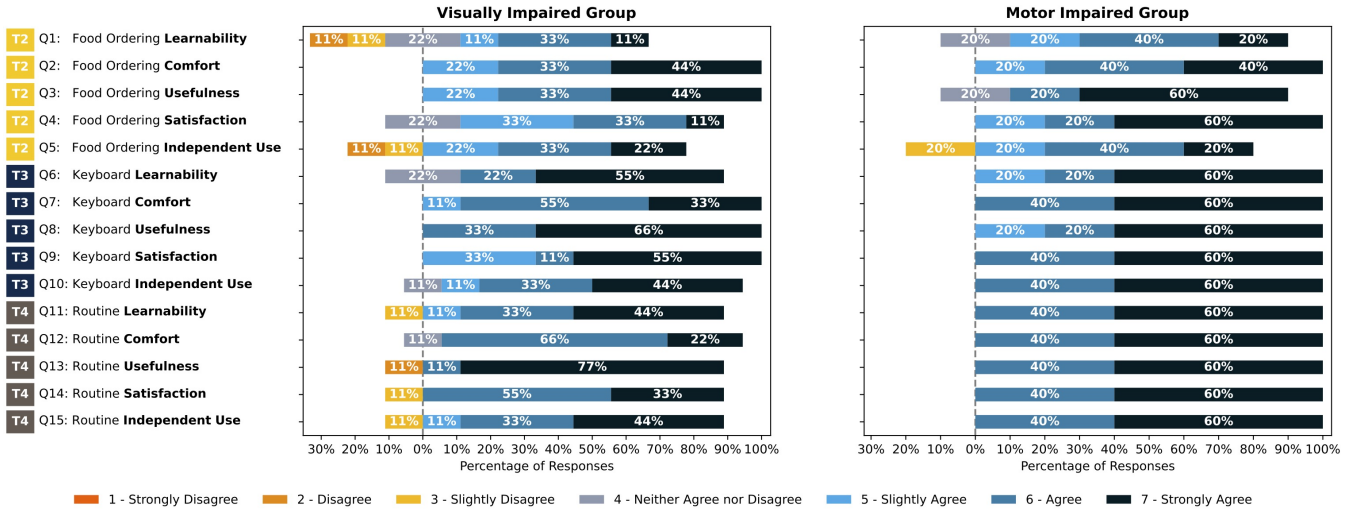


Figure 11: Summary of Likert scale question responses. We asked learnability, comfort, usefulness, satisfaction, and independent use for task 2 (Food Ordering), 3 (Keyboard Input), and 4 (Sequential Input).

5.4.5 RQ5: What are the user experiences with BrushLens, and what possible improvements could be made to the system? **Participants felt confident using BrushLens to perform tasks independently.** The average score for Likert scale questions on independent use was 5.29 ($\sigma = 1.59$), 6.29 ($\sigma = 0.91$), and 6.21 ($\sigma = 1.12$) for task 2, 3, and 4 respectively. This shows the potential of BrushLens to support independent use for both user groups in various usage scenarios. Our participants also envisioned using BrushLens to input frequent flyer numbers at the airport kiosks (M1), using cashier screens at stores (B1), actuating smart home appliances without putting extra tape or custom Braille overlay (B6), and more.

It also shows that, **BrushLens gave users the sense of control, which brought users agency and confidence about their actuation.** By operating BrushLens, participants could better understand the actions performed by the system, and could combine additional information (e.g., a rough location estimation) to double check if the case was doing the right thing. B3 mentioned the importance of having control over the interaction process, and B1 mentioned:

> *I like using BrushLens to push buttons myself. At least I know what's going on, I can know where I pushed something, and I can be more confident. And this is important, because I need something that is completely under my hand. (B1)*

This shows that by supporting but not fully undertaking all aspects of the interaction, BrushLens actively engages the user in the interaction process, which brings users more agency and confidence.

While BrushLens was integrated as a wireless phone case to make it compact to use, participants reported that **the current form factor still needs to be more portable.** It would be ideal if it can be "put into my pocket", where "with the current case I need to find something extra to place it" (M2). B1, B2, B3, M4 also mentioned the need of using small touchscreens, such as printer control panels, where the current form factor seems to be too big for them. This shows that additional engineering effort is needed to make BrushLens compact enough for more use cases. However, as B6 mentioned, she "definitely see a possibility of this", and were excited to see it used on various appliances.

Participants also mentioned that **the guidance system was somewhat confusing at first and needed extra time to get familiar with.** This led to low scores on Q1: learnability and Q5: independent use. The inconsistent behavior caused by the lost of tracking and the conservative logic on activating actuators led to lower score for Q5: independent use. These show the importance of having consistent, clear, and intuitive feedback. As B3 mentioned, the lack of these factors will hinder the system utility in other tasks as well, such as interaction routine mode. These show that additional work is needed to find an optimal guidance system to more informatively guide the user in different interaction conditions, and to generate intuitive guidance to shorten the learning curve.

6 DISCUSSION AND FUTURE WORK

In this section, we discuss privacy and social considerations of using assistive technologies in public spaces, additional supports for diverse user groups and interaction needs, and more broadly a vision for risk-free actuation of touchscreen devices. Additionally, we detail the limitations of the current BrushLens implementation and provide avenues to address these in future work.

6.1 Privacy and Social Considerations of Assistive Technology Use in Public Spaces

Prior research has discussed the potential privacy [2, 35, 49] and social concerns [2, 55] visually and motor impaired users may have when using assistive technology the public spaces. These include concerns surrounding providing sensitive information publicly as well as potential embarrassment caused by the loud audio feedback generated by the assistive device, which is inappropriate in some contexts. Participants in our study also echoed these findings, as they do not wish others to know what they are typing, and want

the interactions they perform, as well as the device they use, to not draw significant attention from others. This includes the shape and size of the device, how they use them, and the noise it generates.

BrushLens was developed with these privacy and social considerations in mind, which are reflected in our design choices. The BrushLens interface, which runs on users' smartphones, allows local storage of sensitive information. The BrushLens phone case also reduces the likelihood of shoulder surfing, by providing a cover to hide all actuators along with the resulting actuation and interaction. Users can interact with their devices and receive audio feedback privately via earphones, preventing sensitive information from being overheard. Since BrushLens is designed as a phone case, users can naturally use the phone for interactions. The device can be moved across the screen in a manner similar to the "tap to pay" interaction frequently employed at ordering kiosks, and the autoclicker actuator is capable of performing touch gestures without any additional noise, minimizing actions that might draw unwanted attention.

Nonetheless, the current BrushLens form factor requires further refinement for optimal use in public spaces. 6 out of 10 participants in our study indicated that the device is somewhat bulky and could be more lightweight. This feedback highlights the need for ongoing engineering efforts to enhance portability and ease of use in everyday situations. We note, however, that the BrushLens case does not require any permanent modifications to the phone and future iterations should also maintain this feature to allow users to decide when and how to use hardware interaction proxies.

6.2 Towards Risk-free Actuation for Hardware Interaction Proxies

Actuation accuracy is critical for mitigating potential interaction risks, which typically arise from inadvertently triggering unintended targets or gestures, or from devices registering invalid inputs. Our technical and user evaluations demonstrate that risk-free actuation is attainable for stationary actuations. Yet, users have diverse movement and interaction patterns that pose additional challenges. Furthermore, existing actuation delays increase the likelihood of overshooting while pressing small targets, and the assumption that the actuators are aligned well with the screen is often dependent on how the user holds the device. Ultimately, it cannot be assumed that each actuation will be performed under best-case scenarios given the variety of ways we observed users interacted with BrushLens. Despite all these challenges, we reiterate that BrushLens provided significant improvements when interacting with touchscreens, reducing mistouches by over 70%.

Although the current actuation accuracy is yet ideal, we foresee that further technical enhancements could improve and bring BrushLens closer to providing risk-free actuation. A better system integration of all processing elements into a single BrushLens app would eliminate unnecessary delays, enabling direct command transmission to microcontrollers for higher responsiveness and accuracy. Additional feedback and error handling mechanism could also be incorporated, such as audio feedback for overly fast movement, and richer information on target distance and area to help users adaptively decide their movements. Also, the system could incorporate predictive methods, such as Kalman filter, to adaptively control the timing of initiating the actuation command.

Moreover, enabling risk-free actuation requires comprehensive performance testing across all conditions, which is a limitation of our work. Throughout the study, we positioned the target device flat on the screen to simulate tabletop usage scenarios and have not verified performance consistency across devices at other angles, such as large vertical kiosks on walls. While there are no technical limitations that would prevent BrushLens from working when vertical (i.e., solenoids and autoclickers are not sensitive to orientation), additional rigorous user studies will offer further supporting evidence for a broader range of use cases in everyday life.

6.3 Assumptions on Known Interface Information

With its core focus on the concept of a hardware interaction proxy, BrushLens is built upon the assumption that the interface information is known to the system. Nevertheless, BrushLens could also utilize the user's smartphone camera to incorporate interface recognition as part of the interaction procedure, thereby providing an integrated solution to the touchscreen accessibility problem.

Since prior solutions on interface recognition usually require interface image as the input, the system could either stitch the interface image from partial views while the user 'brushes' the device across the screen to scan the whole interface, or prompt the user to capture the full view of the interface if possible. The system could then apply previous research to recognize UI elements and their bounding boxes [65, 68], determine tappable regions [54, 57], and reconstruct the underlying interface state diagram [28].

Although technically feasible, integrating these components requires further work to fully evaluate the user experience for both visually and motor impaired users with this updated interaction pipeline. For instance, prior work has shown that visually impaired people may face challenges capturing images as desired [33], which could bring additional frustration if users are asked to capture images multiple times. The fatigue and discomfort of multiple movements may bring additional challenges for motor impaired users [32]. These considerations highlight the need for future work of a thorough and careful design and evaluation before incorporating the interface recognition component into the pipeline.

6.4 Support Diverse Groups of Users and Interaction Needs

People with diverse abilities may have different interaction needs. While the current BrushLens primarily focuses on visually impaired users and motor impaired users with limited fine motor control, we envision it could be further extended to match additional interaction needs, such as users having difficulty reaching the target.

Users' needs also go beyond one's ability and depend on how users use, and want to use, the device in their daily life. For example, participant B3 mentioned the specific need of using small touchscreens like card readers, and preferred the form factor to be more compact. Also, while the current prototype is tested with high-end devices with ultrawide cameras, as long as the camera can focus on the screen, BrushLens could also work with standard smartphone camera systems with adjustments on case height and size to match camera parameters. However, as more and more smartphones are

equipped with ultrawide cameras (e.g., Google Pixel 7, SAMSUNG Galaxy S22), we foresee easier integration with these devices.

In addition, touchscreen interactions typically involve various gestures, such as swiping, pinching, and drag-and-drop. We envision expanding BrushLens to accommodate additional gestures beyond tapping through modifying the actuator or mechanical designs. For instance, previous research has demonstrated the feasibility of using densely aligned electrodes [31], activated sequentially, to simulate a swipe gesture, which could be incorporated into BrushLens' actuator layout design. Additional mechanical design, such as incorporating additional solenoids to push actuators horizontally could achieve a swipe gesture, though this may entail more intricate mechanical designs that could compromise portability.

7 CONCLUSION

We presented BrushLens, a hardware interaction proxy for accessible touchscreen interface actuation. Our technical evaluation and user study demonstrated that BrushLens empowers visually impaired users to interact with touchscreen devices while providing motor impaired users with reduced mistouches, inactive touches, and enhanced haptic feedback. By locating itself relative to elements on the screen accurately, BrushLens can give helpful feedback to assist users in navigating touchscreen interfaces. The user studies also identified potential enhancements, including increasing system speed to compensate for users' diverse movement and interaction patterns and improving the case geometry for improved portability. We showcased the viability of employing a hardware interaction proxy to increase touchscreen accessibility and outlined various opportunities to expand BrushLens' functionality. These enhancements include supporting a broader range of interaction gestures and enabling individuals with diverse abilities to benefit from BrushLens as an interaction proxy for touchscreen interface actuation across various daily use scenarios.

ACKNOWLEDGMENTS

We sincerely thank our participants who contributed to our studies for their time, and the reviewers for their valuable feedback and suggestions. This research was supported in part by a Google Research Scholar Award. We thank organizations and individuals who helped with the study recruitment, Ruei-Che Chang for his help with interaction event logging, Ruiyi Wang for her help with early stage prototyping, and Andi Xu for her help with the user study.

REFERENCES

[1] 2009. Logitech Darkfield Laser Tracking: the World is Your Mouse Pad: an Innovation Brief. https://www.logitech.com/images/pdf/briefs/Logitech_Darkfield_Innovation_Brief_2009.pdf

[2] Ali Abdolrahmani, Ravi Kuber, and Stacy M. Branham. 2018. "Siri Talks at You": An Empirical Investigation of Voice-Activated Personal Assistant (VAPA) Usage by Individuals Who Are Blind. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (Galway, Ireland) *(ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 249–258. https://doi.org/10.1145/3234695.3236344

[3] Nancy Alajarmeh. 2021. Non-visual access to mobile devices: A survey of touchscreen accessibility for users who are visually impaired. *Displays* 70 (2021), 102081. https://doi.org/10.1016/j.displa.2021.102081

[4] Lisa Anthony, YooJin Kim, and Leah Findlater. 2013. Analyzing User-Generated Youtube Videos to Understand Touchscreen Use by People with Motor Impairments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1223–1232. https://doi.org/10.1145/2470654.2466158

[5] Apple. 2022. Share your best iPhone macro photos for Apple's Shot on iPhone Challenge. https://www.apple.com/newsroom/2022/01/share-your-best-iphone-macro-photos-for-apples-shot-on-iphone-challenge/. Online; accessed 15 January 2023.

[6] Apple. 2023. Apple VoiceOver. https://www.apple.com/accessibility/vision/. Online; accessed 15 January 2023.

[7] Rob Barrett, Paul P. Maglio, and Daniel C. Kellem. 1997. How to Personalize the Web. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '97)*. Association for Computing Machinery, New York, NY, USA, 75–82. https://doi.org/10.1145/258549.258595

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. SURF: Speeded Up Robust Features. In *Computer Vision – ECCV 2006*, Aleš Leonardis, Horst Bischof, and Axel Pinz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 404–417.

[9] Mohammed Belatar and Franck Poirier. 2008. Text Entry for Mobile Devices and Users with Severe Motor Impairments: Handiglyph, a Primitive Shapes Based Onscreen Keyboard. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility* (Halifax, Nova Scotia, Canada) *(Assets '08)*. Association for Computing Machinery, New York, NY, USA, 209–216. https://doi.org/10.1145/1414471.1414510

[10] Jeffrey P. Bigham, Ryan S. Kaminsky, Richard E. Ladner, Oscar M. Danielsson, and Gordon L. Hempton. 2006. WebInSight: Making Web Images Accessible. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility* (Portland, Oregon, USA) *(Assets '06)*. Association for Computing Machinery, New York, NY, USA, 181–188. https://doi.org/10.1145/1168987.1169018

[11] Pradipta Biswas and Patrick Langdon. 2012. Developing Multimodal Adaptation Algorithm for Mobility Impaired Users by Evaluating Their Hand Strength. *International Journal of Human–Computer Interaction* 28, 9 (2012), 576–596. https://doi.org/10.1080/10447318.2011.636294 arXiv:https://doi.org/10.1080/10447318.2011.636294

[12] Craig Brown and Amy Hurst. 2012. VizTouch: Automatically Generated Tactile Visualizations of Coordinate Spaces. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (Kingston, Ontario, Canada) *(TEI '12)*. Association for Computing Machinery, New York, NY, USA, 131–138. https://doi.org/10.1145/2148131.2148160

[13] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design Search and Knowledge Discovery through Auto-Created GUI Component Gallery. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 180 (nov 2019), 22 pages. https://doi.org/10.1145/3359282

[14] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang. 2020. Unblind Your Apps: Predicting Natural-Language Labels for Mobile GUI Components by Deep Learning. 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, 322–334. https://doi.org/10.1145/3377811.3380327

[15] Jieshan Chen, Amanda Swearngin, Jason Wu, Titus Barik, Jeffrey Nichols, and Xiaoyi Zhang. 2022. Towards Complete Icon Labeling in Mobile Applications. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 387, 14 pages. https://doi.org/10.1145/3491102.3502073

[16] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, and Guoqiang Li. 2020. Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1202–1214. https://doi.org/10.1145/3368089.3409691

[17] K. B. Chen, A. B. Savage, A. O. Chourasia, D. A. Wiegmann, and M. E. Sesto. 2013. Touch screen performance by individuals with and without motor control disabilities. *Appl Ergon* 44, 2 (2013), 297–302. https://doi.org/10.1016/j.apergo.2012.08.004

[18] Sacha N. Duff, Curt B. Irwin, Jennifer L. Skye, Mary E. Sesto, and Douglas A. Wiegmann. 2010. The Effect of Disability and Approach on Touch Screen Performance during a Number Entry Task. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54, 6 (2010), 566–570. https://doi.org/10.1177/154193121005400605 arXiv:https://doi.org/10.1177/154193121005400605

[19] Be My Eyes. 2023. Be My Eyes - See the world together. https://www.bemyeyes.com/. Online; accessed 15 January 2023.

[20] Giovanni Fusco, Ender Tekin, Richard E. Ladner, and James M. Coughlan. 2014. Using Computer Vision to Access Appliance Displays. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility* (Rochester, New York, USA) *(ASSETS '14)*. Association for Computing Machinery, New York, NY, USA, 281–282. https://doi.org/10.1145/2661334.2661404

[21] Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2007. Automatically Generating User Interfaces Adapted to Users' Motor and Vision Capabilities. 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, 231–240. https://doi.org/10.1145/1294211.1294253

[22] Google. 2023. Android TalkBack. https://support.google.com/accessibility/android/answer/6006564. Online; accessed 15 January 2023.

[23] Timo Götzelmann and Aleksander Pavkovic. 2014. Towards Automatically Generated Tactile Detail Maps by 3D Printers for Blind Persons. In *Computers Helping People with Special Needs*, Klaus Miesenberger, Deborah Fels, Dominique Archambault, Petr Peňáz, and Wolfgang Zagler (Eds.). Springer International Publishing, Cham, 1–7.

[24] William Grussenmeyer and Eelke Folmer. 2017. Accessible Touchscreen Technology for People with Visual Impairments: A Survey. *ACM Trans. Access. Comput.* 9, 2, Article 6 (jan 2017), 31 pages. https://doi.org/10.1145/3022701

[25] Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2010. Towards Accessible Touch Interfaces. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility* (Orlando, Florida, USA) *(ASSETS '10)*. Association for Computing Machinery, New York, NY, USA, 19–26. https://doi.org/10.1145/1878803.1878809

[26] Anhong Guo, Xiang 'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P. Bigham. 2016. VizLens: A Robust and Interactive Screen Reader for Interfaces in the Real World. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 651–664. https://doi.org/10.1145/2984511.2984518

[27] Anhong Guo, Jeeeun Kim, Xiang 'Anthony' Chen, Tom Yeh, Scott E. Hudson, Jennifer Mankoff, and Jeffrey P. Bigham. 2017. Facade: Auto-Generating Tactile Interfaces to Appliances. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5826–5838. https://doi.org/10.1145/3025453.3025845

[28] 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* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 371–385. https://doi.org/10.1145/3332165.3347873

[29] Amy Hurst, Jennifer Mankoff, Anind K. Dey, and Scott E. Hudson. 2007. Dirty Desktops: Using a Patina of Magnetic Mouse Dust to Make Common Interactor Targets Easier to Select. 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, 183–186. https://doi.org/10.1145/1294211.1294242

[30] Amy Hurst, Jennifer Mankoff, Anind K. Dey, and Scott E. Hudson. 2007. Dirty Desktops: Using a Patina of Magnetic Mouse Dust to Make Common Interactor Targets Easier to Select. 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, 183–186. https://doi.org/10.1145/1294211.1294242

[31] Kaori Ikematsu, Kunihiro Kato, and Yoshihiro Kawahara. 2021. LightTouch Gadgets: Extending Interactions on Capacitive Touchscreens by Converting Light Emission to Touch Inputs. 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 509, 11 pages. https://doi.org/10.1145/3411764.3445581

[32] Curt B. Irwin and Mary E. Sesto. 2012. Performance and touch characteristics of disabled and non-disabled participants during a reciprocal tapping task using touch screen technology. *Applied Ergonomics* 43, 6 (2012), 1038–1043. https://doi.org/10.1016/j.apergo.2012.03.003

[33] Chandrika Jayant, Hanjie Ji, Samuel White, and Jeffrey P. Bigham. 2011. Supporting Blind Photography. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility* (Dundee, Scotland, UK) *(ASSETS '11)*. Association for Computing Machinery, New York, NY, USA, 203–210. https://doi.org/10.1145/2049536.2049573

[34] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. 2008. Slide Rule: Making Mobile Touch Screens Accessible to Blind People Using Multi-Touch Interaction Techniques. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility* (Halifax, Nova Scotia, Canada) *(Assets '08)*. Association for Computing Machinery, New York, NY, USA, 73–80. https://doi.org/10.1145/1414471.1414487

[35] Shaun K. Kane, Chandrika Jayant, Jacob O. Wobbrock, and Richard E. Ladner. 2009. Freedom to Roam: A Study of Mobile Device Adoption and Accessibility for People with Visual and Motor Disabilities. In *Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility* (Pittsburgh, Pennsylvania, USA) *(Assets '09)*. Association for Computing Machinery, New York, NY, USA, 115–122. https://doi.org/10.1145/1639642.1639663

[36] Shaun K. Kane, Meredith Ringel Morris, Annuska Z. Perkins, Daniel Wigdor, Richard E. Ladner, and Jacob O. Wobbrock. 2011. Access Overlays: Improving Non-Visual Access to Large Touch Screens for Blind Users. 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, 273–282. https://doi.org/10.1145/2047196.2047232

[37] Shaun K. Kane, Jacob O. Wobbrock, and Richard E. Ladner. 2011. Usable Gestures for Blind People: Understanding Preference and Performance *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 413–422. https://doi.org/10.1145/1978942.1979001

[38] Mike Keats. 2018. The Updated Pegg Turbo 7 – Phone Auto Clicker. https://peggelectronics.com/pegg-turbo-7-auto-clicker-with-attitude/. Online; accessed 15 January 2023.

[39] Lars Emil Knudsen and Harald Holone. 2012. A Multimodal Approach to Accessible Web Content on Smartphones. In *Computers Helping People with Special Needs*, Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–8.

[40] Ravi Kuber, Amanda Hastings, and Matthew Tretter. 2020. Determining the accessibility of mobile screen readers for blind users. *UMBC Faculty Collection* (2020).

[41] Seungyon Lee and Shumin Zhai. 2009. The Performance of Touch Screen Soft Buttons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 309–318. https://doi.org/10.1145/1518701.1518750

[42] Jiasheng Li, Zeyu Yan, Arush Shah, Jonathan Lazar, and Huaishu Peng. 2023. Toucha11y: Making Inaccessible Public Touchscreens Accessible. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 748, 13 pages. https://doi.org/10.1145/3544548.3581254

[43] Weiyue Lin, Ting Li, Liu Liu, and Qian Zhu. 2023. "Unfold and Go Touch": A Portable Method for Making Existing Touchscreens Accessible to Blind and Low Vision People in Self-Service Terminals. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, Article 302, 7 pages. https://doi.org/10.1145/3544549.3585819

[44] mikeselectricstuff. 2022. Phone touchscreen auto-clicker. https://www.youtube.com/watch?v=ugPr3HV4yBc. Online; accessed 15 January 2023.

[45] Kyle Montague, Hugo Nicolau, and Vicki L. Hanson. 2014. Motor-Impaired Touchscreen Interactions in the Wild. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility* (Rochester, New York, USA) *(ASSETS '14)*. Association for Computing Machinery, New York, NY, USA, 123–130. https://doi.org/10.1145/2661334.2661362

[46] J. Morris and J. Mueller. 2014. Blind and Deaf Consumer Preferences for Android and iOS Smartphones. In *Inclusive Designing*, P. M. Langdon, J. Lazar, A. Heylighen, and H. Dong (Eds.). Springer International Publishing, Cham, 69–79.

[47] Martez E. Mott, Radu-Daniel Vatavu, Shaun K. Kane, and Jacob O. Wobbrock. 2016. Smart Touch: Improving Touch Accuracy for People with Motor Impairments with Template Matching *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1934–1946. https://doi.org/10.1145/2858036.2858390

[48] Martez E. Mott and Jacob O. Wobbrock. 2019. Cluster Touch: Improving Touch Accuracy on Smartphones for People with Motor and Situational Impairments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300257

[49] Maia Naftali and Leah Findlater. 2014. Accessibility in Context: Understanding the Truly Mobile Experience of Smartphone Users with Motor Impairments. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers &; Accessibility* (Rochester, New York, USA) *(ASSETS '14)*. Association for Computing Machinery, New York, NY, USA, 209–216. https://doi.org/10.1145/2661334.2661372

[50] Hugo Nicolau, Kyle Montague, Tiago Guerreiro, André Rodrigues, and Vicki L. Hanson. 2015. Typing Performance of Blind Users: An Analysis of Touch Behaviors, Learning Effect, and In-Situ Usage. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility* (Lisbon, Portugal) *(ASSETS '15)*. Association for Computing Machinery, New York, NY, USA, 273–280. https://doi.org/10.1145/2700648.2809861

[51] Uran Oh, Shaun K. Kane, and Leah Findlater. 2013. Follow That Sound: Using Sonification and Corrective Verbal Feedback to Teach Touchscreen Gestures. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility* (Bellevue, Washington) *(ASSETS '13)*. Association for Computing Machinery, New York, NY, USA, Article 13, 8 pages. https://doi.org/10.1145/2513383.2513455

[52] Beryl Plimmer, Andrew Crossan, Stephen A. Brewster, and Rachel Blagojevic. 2008. Multimodal Collaborative Handwriting Training for Visually-Impaired People. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 393–402. https://doi.org/10.1145/1357054.1357119

[53] Adil Rahman, Md Aashikur Rahman Azim, and Seongkook Heo. 2023. Take My Hand: Automated Hand-Based Spatial Guidance for the Visually Impaired. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 544, 16 pages. https://doi.org/10.1145/3544548.3581415

[54] Eldon Schoop, Xin Zhou, Gang Li, Zhourong Chen, Bjoern Hartmann, and Yang Li. 2022. Predicting and Explaining Mobile UI Tappability with Vision Modeling

and Saliency Analysis. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 36, 21 pages. https://doi.org/10.1145/3491102.3517497

[55] Kristen Shinohara and Jacob O. Wobbrock. 2011. In the Shadow of Misperception: Assistive Technology Use and Social Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 705–714. https://doi.org/10.1145/1978942.1979044

[56] Jing Su, Alyssa Rosenzweig, Ashvin Goel, Eyal de Lara, and Khai N. Truong. 2010. Timbremap: Enabling the Visually-Impaired to Use Maps on Touch-Enabled Devices. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services* (Lisbon, Portugal) *(MobileHCI '10)*. Association for Computing Machinery, New York, NY, USA, 17–26. https://doi.org/10.1145/1851600.1851606

[57] Amanda Swearngin and Yang Li. 2019. Modeling Mobile Interface Tappability Using Crowdsourcing and Deep Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3290605.3300305

[58] Hironobu Takagi and Chieko Asakawa. 2000. Transcoding Proxy for Nonvisual Web Access. In *Proceedings of the Fourth International ACM Conference on Assistive Technologies* (Arlington, Virginia, USA) *(Assets '00)*. Association for Computing Machinery, New York, NY, USA, 164–171. https://doi.org/10.1145/354324.354371

[59] Brandon Taylor, Anind Dey, Dan Siewiorek, and Asim Smailagic. 2016. Customizable 3D Printed Tactile Maps as Interactive Overlays. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility* (Reno, Nevada, USA) *(ASSETS '16)*. Association for Computing Machinery, New York, NY, USA, 71–79. https://doi.org/10.1145/2982142.2982167

[60] Ender Tekin, James M. Coughlan, and Huiying Shen. 2011. Real-time detection and reading of LED/LCD displays for visually impaired persons. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*. 491–496. https://doi.org/10.1109/WACV.2011.5711544

[61] Shari Trewin, Cal Swart, and Donna Pettick. 2013. Physical Accessibility of Touchscreen Smartphones. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility* (Bellevue, Washington) *(ASSETS '13)*. Association for Computing Machinery, New York, NY, USA, Article 19, 8 pages. https://doi.org/10.1145/2513383.2513446

[62] Gregg Vanderheiden, Jonathan Lazar, Amanda Lazar, Hernisa Kacorri, and J. Bern Jordan. 2023. *Kiosks and Information-Transaction Machine Access (1999–)*. Springer International Publishing, Cham, 89–97. https://doi.org/10.1007/978-3-031-09214-5_8

[63] Gregg C Vanderheiden and J Bern Jordan. 2012. Design for people with functional limitations. *Handbook of human factors and ergonomics* (2012), 1407–1441.

[64] Chat Wacharamanotham, Jan Hurtmanns, Alexander Mertens, Martin Kronenbuerger, Christopher Schlick, and Jan Borchers. 2011. Evaluating Swabbing: A Touchscreen Input Method for Elderly Users with Tremor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 623–626. https://doi.org/10.1145/1978942.1979031

[65] Thomas D. White, Gordon Fraser, and Guy J. Brown. 2019. Improving Random GUI Testing with Image-Based Widget Detection *(ISSTA 2019)*. Association for Computing Machinery, New York, NY, USA, 307–317. https://doi.org/10.1145/3293882.3330551

[66] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-Based Design: Concept, Principles and Examples. *ACM Trans. Access. Comput.* 3, 3, Article 9 (apr 2011), 27 pages. https://doi.org/10.1145/1952383.1952384

[67] Mulong Xie, Sidong Feng, Zhenchang Xing, Jieshan Chen, and Chunyang Chen. 2020. UIED: A Hybrid Tool for GUI Element Detection. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1655–1659. https://doi.org/10.1145/3368089.3417940

[68] Mulong Xie, Sidong Feng, Zhenchang Xing, Jieshan Chen, and Chunyang Chen. 2020. UIED: A Hybrid Tool for GUI Element Detection. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1655–1659. https://doi.org/10.1145/3368089.3417940

[69] Momona Yamagami, Sasa Junuzovic, Mar Gonzalez-Franco, Eyal Ofek, Edward Cutrell, John R. Porter, Andrew D. Wilson, and Martez E. Mott. 2022. Two-In-One: A Design Space for Mapping Unimanual Input into Bimanual Interactions in VR for Users with Limited Movement. *ACM Trans. Access. Comput.* 15, 3, Article 23 (jul 2022), 25 pages. https://doi.org/10.1145/3510463

[70] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. 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 275, 15 pages. https://doi.org/10.1145/3411764.3445186

[71] Xiaoyi Zhang, Anne Spencer Ross, Anat Caspi, James Fogarty, and Jacob O. Wobbrock. 2017. Interaction Proxies for Runtime Repair and Enhancement of Mobile Application Accessibility. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6024–6037. https://doi.org/10.1145/3025453.3025846

[72] Yu Zhong, Astrid Weber, Casey Burkhardt, Phil Weaver, and Jeffrey P. Bigham. 2015. Enhancing Android Accessibility for Users with Hand Tremor by Reducing Fine Pointing and Steady Tapping. In *Proceedings of the 12th International Web for All Conference* (Florence, Italy) *(W4A '15)*. Association for Computing Machinery, New York, NY, USA, Article 29, 10 pages. https://doi.org/10.1145/2745555.2747277