consume power and therefore preserves phone battery life. Our validation shows that its functionality is compatible with most of the top 50 Android apps. In our usability study, Interactiles showed promise in improving task completion time and increasing user satisfaction for certain interactions. Our results support the value of a hybrid software-hardware approach for providing tangibility while maximizing compatibility across common apps.

Use Case Scenario

The Interactiles system is designed to work with mobile screen readers while providing tactile feedback. Its hardware base is a 3D-printed plastic shell that snaps on a phone, and the shell contains three hardware components: a number pad, a scrollbar, and a control button at the bottom of the scrollbar. Users may flip in and out hardware components for different tasks. When all hardware components are flipped out and the software is off, users have full access to the touchscreen as normal. These hardware components work with our software to achieve five main functionalities: *element navigation*, *bookmarking*, *page scrolling*, *app switching*, and *number entry*. These features are described in the following user scenario and also demonstrated in our supplementary video.

Consider a user named Jane. To use Interactiles, she turns on the software and flips in the scrollbar. When the software is active, a floating window appears on the screen to receive touch events from the scrollbar. She browses the Amazon mobile app to buy a backpack. Starting with the *element navigation* mode, she moves the scroll thumb to hear information for each product on a page. When she navigates through all products on the current page and the scroll thumb arrives at the bottom of the scrollbar, she long presses the scroll thumb to move to the next page.

While Jane is browsing, she finds a backpack with good reviews and a reasonable price. She would like to browse through all the products before purchasing, but she also wants to be able to go back to this backpack later. Thus, she long presses the control button to *bookmark* this backpack.

After going through a long list of backpacks, Jane realizes she likes the bookmarked one better. She presses the control button once to activate the *page scrolling* mode. She slides up the scroll thumb to scrolls up pages. Her phone vibrates and announces the bookmark content when it scrolls to the page containing the bookmarked backpack.

While Jane is shopping, a new message notification pops up. She presses the control button to activate *app switching* mode and hears “Mode: App” as confirmation. She moves the scroll thumb to hear the recently opened apps. When she hears “Messages”, she double-taps the scroll thumb and the Messages app opens.

Jane presses the control button again to switch back to *element navigation* mode. Moving the scroll thumb, she hears the subject line for each message (*e.g.*, “What’s the address for coffee on Monday?”) and then navigates to the reply text field. She double-taps the scroll thumb; the text field is active for input, the phone soft keyboard pops up, and a floating window for the number pad appears on the screen. Jane flips in the physical number pad for *number entry*. She enters numbers using the number pad and enters letters using the phone soft keyboard as normal.

Overview

In the following sections, we first explore challenges faced by people with visual impairments in touchscreen interactions as well as prior software and hardware solutions to these challenges. We highlight an important gap in adding tangible interaction to mobile phones. Next, we introduce the Interactiles system to address this gap, and describe details of its hardware/software implementation. Then, through user studies, we evaluate the improvement of Interactiles on task performance and user experience. Finally, we discuss the system limitations and potential future work.

RELATED WORK

This section explores the specific challenges individuals with visual impairments encounter when using touchscreens and prior research efforts to address these problems. We define our main design goals from the unaddressed aspects of these challenges.

Challenges in Graphical Interface Accessibility

Built-in screen readers (*e.g.*, TalkBack [9] and VoiceOver [2]) on mobile touchscreen devices have become widely adopted by people with visual impairments. However, visual information is lost during the text-to-speech conversion (*e.g.*, spatial layout or view hierarchies). Qualitative studies [4,20,22] have investigated difficulties faced by people with visual impairments using screen readers: Baldwin et al. [4] conducted a long-term field study of novices learning general computer skills on desktops, McGookin et al. [22] investigated touchscreen usage across many types of devices such as phones and media players, and Kane et al. [17] specifically focused on mobile touchscreens. These studies consistently found problems with locating items within graphical interfaces that were designed to be seen rather than heard. Furthermore, once objects are located, it is still a challenge to understand their location in the context of the app, remember where they are, and relocate them. This location and relocation process is time-consuming, with users needing to make multiple passes through content and listen to increasingly longer amounts of audio. Even then, they may not realize that their desired target is not reachable in the current context [4]. As Kane *et al.* and McGookin *et al.* demonstrated, this basic location problem exists not only on desktops but also on mobile touchscreens. We would expect this problem to be compounded on phones because their targets are smaller and denser.

Because of these issues, users often carry multiple devices with overlapping functionality that each offer better accessibility for specific tasks [17,22]. A few examples include the media players with tactile buttons [22], or the popular Victor Reader Stream [16], a handheld media

device specifically for people with visual impairments. Although iPods and smartphones support similar features to these devices, their flat screens make them more complex to use. However, carrying multiple devices can be difficult to manage [17] and costly [22] (*e.g.*, the Victor Reader Stream costs $369). For a population that is more likely to live in poverty than those with sight [1], this represents a substantial barrier to equal information access.

These basic problems of locating, understanding, interacting with, and relocating objects via a touchscreen lead to the secondary problems of needing to manage multiple devices and the high cost of additional devices. Prior work to address these problems has taken different approaches, ranging from software-focused (*e.g.,* [17,19]), hardware-focused (*e.g.*, [8]), to hybrid (*e.g.*, [4,20]) in improving accessibility.

Software-based Touchscreen Assistive Technologies

Software-based approaches to improving accessibility introduced new interaction techniques designed to ease cognitive load and shorten task completion time. For example, Access Overlays [19] improves task completion times and spatial understanding through software overlays designed to help with target acquisition. However, although Access Overlays showed accessibility improvements on large touchscreens, such an approach may not work on mobile devices due to the significantly smaller screen size.

In the case of mobile phones, Slide Rule re-maps gestures to support screen navigation separately from activating targets [17], similar to the mechanism now supported by TalkBack and VoiceOver. NavTap [12] presented a navigational text entry method that divided the alphabet into rows to reduce cognitive load and improved typing speed over a 4-month study. BrailleTouch [26] allowed users to enter Braille characters using three fingers on each hand and offered a significant speed advantage over standard touchscreen braille keyboards. For the task of number entry, DigiTaps [3] encoded different gestures to the ten numeric digits. These gestures required a varying number of fingers to tap, swipe, or both. Although the DigiTaps system showed improvement over VoiceOver in number entry speed, it required users to remember the 10 gesture patterns and may also benefit from a tangible component.

Although text entry is an important activity, mobile phones are also used for a much wider variety of activities such as perusing bus schedules, weather information, and personal calendars. These all require some form of text entry, but also actions specific to each app [18]. Indeed, the plethora of apps on mobile phones [27] makes the creation of new, general interaction patterns challenging.

Tangible Accessibility

Although software solutions are already helpful for accessibility improvement, physical modifications are a promising complement to software. There have also been efforts to introduce or modify hardware to improve touchscreen accessibility.

One area of research has been in creating physical interaction aids for focused applications such as maps and graphics. Tactile maps [29] overlaid on a touchscreen were fabricated using conductive filament that could transfer touch. In TacTILE [15], a toolchain was presented for creating arbitrary tactile graphics with both raised areas and cutout regions to present information. Both of these approaches resulted in positive feedback by making it easier to explore spatial information, but the tangible hardware was limited to a single screen on a single application. Additionally, the hardware was not attached to the phone and thus required the user to carry it separately.

For larger touchscreens, such as those found on tabletops or kiosks, Touchplates [20] introduced a set of guides to provide tactile feedback. They are versatile and inexpensive because of their use of standard touchscreen software and can be implemented by cutting holes in inexpensive materials such as cardboard and acrylic sheets. The visual tags allowed the software event listeners to configure to an arbitrary touchplate location. However, the interaction innovations of Touchplates may not all work on smaller mobile screens. The need to remove, reattach, and store guides such as the ones presented by Touchplates while on-the-go could make it hard to manage them for a mobile setup. The cutout mechanism would also be harder to interact with on a smaller screen. Tangible interfaces have also been used to enhance screen reader access in desktop computers without a touchscreen. The Tangible Desktop [4] showed significant improvements in task completion time on a personal computer by replicating traditional desktop metaphors in the physical world. However, its setup of a potentiometer-powered scrollbar and physical icons would be difficult to carry around. Hybrid-Brailler [31] combined physical and gestural interaction to provide fast and accurate Braille input. Its physical interface is on the back of the phone where it does not block gestural interaction on the screen. Therefore, the physical interface communicates with phones via Bluetooth, which requires power and increases the cost and complexity of the system.

A variety of commercial designs and products also provided hardware solutions to text entry. The Touchfire keyboard cover [30] added a rubber overlay for iPad keyboards. Although useful, many reviewers were not impressed by the $50 price [21], and the small size of a phone keyboard does not suit the same approach. There were also customizable stickers and screen protectors with raised dots [14], which were only useful on a single screen, and customizable phones with Braille buttons [25] which only offered basic functionality such as calling and failed to meet everyday use requirements.

In summary, there appears to be an untapped opportunity to use tangible means to improve accessibility in mobile touchscreen computing. Prior work has demonstrated the potential of new software-enabled interaction techniques which make it easier to locate objects, but these do not fully

use tangibility as a memory aid and tool to ease cognitive load. It has also been shown that inexpensive, easily fabricated tactile pieces have great promise but do not extend well to a mobile setup because they have many pieces which are difficult to manage on-the-go or lack broad deployability on a mobile phone due to their narrow applications.

Design Goals

Considering the challenges highlighted in the related work, we identify the following design goals.

Tangible with mobile focus: Prior research [4,20] shows the positive impact of tangible interaction, but most such work has been designed for large touchscreens. The prevalence of mobile devices and apps urges us to design for mobile touchscreens, which present a different set of challenges than larger interfaces.

Portable, contained, and non-blocking: Because we design for mobile devices, our solution should be portable enough to carry on a daily basis. All hardware components should be contained in one piece so that users do not have to carry additional pieces and assemble them. In addition, the system should work with built-in screen readers, and should not block normal touchscreen interaction when not in use.

Compatible with various mobile apps: In previous research [15,22], a tactile overlay is limited to one screen in a single app. Carrying a piece of hardware for each screen or app is not practical on a daily basis. We would like to design some common functionalities that can be used in different mobile apps.

Low-cost: There are commercial products that provide tactile feedback (*e.g.,* Braille phone [25]). However, they may not be affordable to people with disabilities, a population more likely to face socio-economic barriers [1]. Therefore, we would like to design an inexpensive solution for better deployability.

Unpowered: Many rich tactile interactions have been enabled through electrical [5] and electrical-mechanical experiences [4]. However, these systems increase the cost of assistive technologies and decrease portability, which is important for mobile platforms. Therefore, we would like to design an unpowered solution.

INTERACTILES DESIGN AND IMPLEMENTATION

Interactiles is an inexpensive, unpowered, general-purpose system that increases tactile interaction on touchscreen phones. Its hardware allows users to interact and receive tactile feedback. Its software receives touch input on hardware and invokes the corresponding action on the current running app. The system is designed to work with built-in screen readers and mobile apps without modification to them. Currently, our software supports devices running Android 5.0 and above (82.3% of Android devices [10]).

Before we started to design Interactiles, we spoke with two screen reader users to learn what improvements they desired in mobile touchscreen interaction. Their feedback motivated us to design the selected functions. They expressed disappointment at the disappearance of physical phone buttons. They also cited interactions and situations where they have particular difficulty, such as losing one's place while using explore-by-touch on a moving bus. This motivated us to design the *element navigation* feature, as the scroll thumb keeps one's place on the screen. Both of them did not like entering numbers, especially in a time-sensitive situation. They expressed enthusiasm for physical buttons that would help them with such problems.

Hardware Components

The unpowered and portable Interactiles hardware provides tactile feedback to users. Inspired by prior research on capacitive touch hardware components [7], our hardware leverages conductive material to connect a user's finger to an on-screen contact point, thus registering a touch event.

Our prototype costs less than $10 in materials. Most parts of the hardware are 3D-printed with inexpensive PLA plastic. Some parts are handmade from silicone, conductive fabric, conductive thread, and standard fasteners (*e.g.*, nuts, bolts, and washers). The print requires about 10 hours and assembly requires about 2 hours. As it is a one-time print, the time is unlikely prohibitive. In addition, users may receive help from volunteer communities. For example, e-NABLE is successful in adapting and assembling AT (*e.g.*, prosthetic limbs) for people with disabilities. The hardware base is a 3D-printed plastic shell that snaps on a phone, and three hardware components are attached to the shell:

The **Scrollbar** is attached to the right side of the phone shell. It consists of a 3D-printed PLA frame and a scroll thumb. The scroll thumb is a piece of metal bent at a right angle, encased in PLA, and covered with conductive fabric. We chose fabric because it transmits capacitive touch reliably, feels comfortable, and does not scratch the touchscreen. The scrollbar is attached to the shell using a hinge; it can be flipped out from the screen when not in use.

The **Control Button** is attached to the bottom of the scrollbar. It is a silicone button sewn through with conductive thread to transmit touch. The button was cast using a 3D-printed mold to control button size and shape. Silicone material was chosen to make sure the button is comfortable to press. Because there lacks a castable, affordable, conductive material, we sewed conductive thread to make the silicone conductive.

The **Number Pad** is attached to the left side of the phone shell. It is a 3D-printed PLA plate that contains a 4x3 grid of silicone buttons. Similar to the scrollbar, the number pad is hinged and can be flipped out when not in use.

Software Proxies

The Interactiles software is an accessibility service which runs in an app-independent fashion. To increase deployability and generalizability, our software is implemented with standard Android accessibility APIs. It does not require rooting a phone or access to app source

code and is compatible with Android TalkBack. Our implementation approach was inspired by *Interaction Proxies* [32], a strategy to modify user input and output on the phone that can be used for accessibility repairs such as adding alternative text or modifying navigation order.

Interaction Proxies [32] consist of one or more *Floating Windows*, which are visible on top of any currently running app. As these proxies sit above the app in z-order, they can modify both the output and input of the underlying app. Output is modified by drawing inside the floating windows. Input is modified using *Event Listeners* which intercept all touch events on the floating window using the Android accessibility API and consume them before they reach the underlying app. Interaction Proxies can further leverage accessibility APIs for *Content Introspection* of the underlying app as well as *Automation* of actions.

Using these abstractions, Interaction Proxies implemented proof-of-concept accessibility repairs [28] that are compatible with various mobile apps. In this paper, we describe our implementation which, although it leveraged concepts from *Interaction Proxies*, was entirely novel.

The **Interactiles Service** captures touch events generated when users touch the conductive portion of a hardware component, interprets them, and takes appropriate action. It captures events using floating windows with attached event listeners and delivers events using a combination of content introspection and automation.

The event listeners monitor *AccessibilityEvents* [11] generated in the currently running app and floating windows to perform actions. User interaction with TalkBack generates events different from standard touch events so that we cannot use the standard gesture listener. We implemented a custom listener to recognize the following events that occur within the floating windows: 1) tap, 2) double tap, 3) slide (press and move scroll thumb), and 4) long press with different durations.

The floating windows in Figure 2 sit underneath the relevant piece of hardware, and thus do not interfere with user interactions on the uncovered screen area. The presence/absence of each floating window and its content are determined by the current Interactiles mode, along with the state of the current app or apps that are running. In particular, our software assumes the scrollbar is always active until the phone keyboard pops up for text entry. Whenever the keyboard is active, the number pad floating window appears, and the scrollbar floating window is temporarily removed to avoid blocking the keyboard. The user flips in the physical number pad and flips out the scrollbar accordingly. When the keyboard is deactivated, the number pad floating window disappears, and the scrollbar reappears. When all hardware components are flipped out and the software is off, the user has full access to the touchscreen as normal.

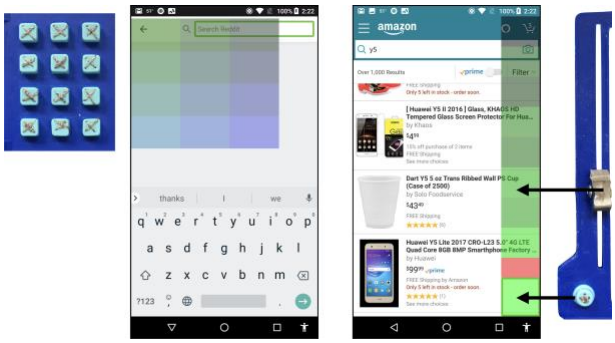

Figure 2. Floating windows created for number pad (left), scrollbar (right), and control button (right bottom). The windows can be transparent; we use colors for demonstration.

The scrollbar floating window is implemented using a dynamic list of buttons representing the items to be scrolled through. When the scroll thumb slides to a button, TalkBack announces the alternative text of the button. The scrollbar has three modes: *App Switching*, *Element Navigation*, and *Page Scrolling*. A short press on the control button at the bottom of the scrollbar will switch modes, announce the new mode, and update the buttons in the floating window under the scrollbar. The number and size of buttons are dynamically determined by the number of elements on the current screen for *Element Navigation* mode and are constant in the other two modes. The scrollbar floating window also has an active button reserved for the control button. For *Number Entry*, the number pad floating window is implemented using a static 4x3 grid of buttons that becomes active when a text box is in focus.

App Switching: Screen reader users typically switch between apps using a physical button or a software button at the bottom of the screen, and then navigate through a list of opened apps. However, locating this button and switching through apps can be difficult. For example, TalkBack requires three swipes to navigate to the next app (announce app name, open app info, close app). In our *App Switching* mode, the user simply slides the scroll thumb to hear the name of each opened app (represented as a button in the floating window). Double-tapping the scroll thumb opens the app. To minimize required precision, Interactiles defaults to switching between the four most recently opened apps, a number that can be customized if desired.

Element Navigation: TalkBack and VoiceOver users can swipe left/right to navigate between all UI elements on the screen in a linear order. Users can swipe quickly without listening to the full text of each item (*e.g.*, in a long list). Alternatively, users can use explore-by-touch, where they move one finger around the screen to hear about on-screen elements. It is a faster way to skip content in a long list, but risks missing elements with small or hard-to-reach targets. Interactiles uses the physical scrollbar to navigate through app UI elements; each element creates a button with its alternative text in the floating window. Users slide the scroll thumb to move the focus between elements and hear

the content of the focused element. A quick slide can skip elements. Double-tapping the scroll thumb clicks the focused element. When the scroll thumb arrives at the top/bottom of the scrollbar, users can long press the scroll thumb to move to the previous/next page.

Page Scrolling: Screen reader users may scroll pages instead of elements, with a two-finger or three-finger swipe on the screen. Advanced TalkBack users can also assign a two-stroke gesture to scroll pages with one finger. TalkBack announces the page location after scrolling (*e.g.*, "showing items 11 to 20 of 54"). Interactiles supports a similar function with tactile feedback on the physical scrollbar. If the current app screen is scrollable, users can slide up/down the scroll thumb to scroll to the previous/next page and hear the page number. When the scroll thumb arrives at the top/bottom of the scrollbar, users can quickly slide down/up the scroll thumb without scrolling screen. Thus, users can keep scrolling in an infinite list.

Bookmarking: It can be challenging for TalkBack users to relocate a UI element in an app. Interactiles uses the physical control button to bookmark a UI element in an app. In *Element Navigation* mode, when users navigate to an element of interest, they can long press the control button and hear "Bookmarked" with the alternative text of that element. Later, when users move to a screen that contains the bookmark (a new screen is open or the screen scrolls), the phone vibrates and announces "Bookmark found" with the alternative text of the bookmarked element. Currently, our system allows one bookmark per app.

Number Entry: TalkBack users enter numbers by switching to the symbol mode of soft keyboard. However, locating first the symbol keyboard and then a specifc number on the keyboard is challenging. In addition, typing a combination of letters and numbers requires frequent keyboard mode switching. Interactiles uses the physical number pad to enter numbers; as seen in Figure 1 (Left) and Figure 2 (Left), users can type letters on the soft keyboard at the same time. The number pad floating window contains a 4x3 grid of buttons, matching the position of the physical buttons. The number pad uses a layout similar to a phone keypad: the first three rows have number 1 to 9, and the bottom row has "read the entered text", number 0, and "backspace". When a button is pressed, our software updates the content of the active text field by appending a number or removing the last character.

System Improvement from Pilot Study

After we developed a fully working prototype of the software and hardware, we conducted a pilot study with a blind participant (a 38-year-old male who has used TalkBack for 7 years) and iterated based on his feedback. Originally, to go to the next page of elements when in element navigation mode, the user had to switch to page mode, scroll down a page, and switch back to element mode. Based on our pilot study, we changed the system to allow a "next page" action while in element mode. We also added more verbal feedback to help the user. The final system announces the page number while *page scrolling*, the bookmark content when found, and the entered characters in a text box. In the final system, pages were not directly mapped to scrollbar locations; the user goes to the next page when at the bottom of the scrollbar by moving the scroll thumb quickly up (faster than 10 cm/s) to get more room, and then down slowly again.

System Validation Across Android Apps

To test robustness, we validated Interactiles compatibility in 50 Android apps from Google Play Store. Our sample apps were 5 top free apps in each of 10 categories (*i.e.*, Book, Communication, Entertainment, Fitness, Navigation, Medical, News, Productivity, Shopping, Social). For each app, we tested *page scrolling*, *element navigation*, and *bookmark* on a screen that allows scrolling. We tested *number entry* on a screen that allows text entry. When possible, we tested the main screens of apps (*e.g.*, News Feed screen in the Facebook app). The results are summarized in Table 1.

Page Scrolling	Element Navigation	Bookmark	Number Entry	App Switching
49/50	42/50	49/50	49/49	50/50

Table 1. Number of apps in which Interactiles features worked as expected, out of 50 sample apps. One app did not include any text field to test number entry.

Page scrolling worked as expected in all but one app. In the main screen of Google Play Book, our system did not scroll the page vertically, but instead scrolled the large horizontal banner at the top. As future work, we may allow developers to annotate which element should be scrolled in specific apps or may allow scrolling in multiple scrollable elements.

Element navigation did not work as expected in 8 apps. In 3 apps, there are more than 30 UI elements on a screen so that each corresponding button on the floating window was too small for the scroll thumb to individually select. In 5 apps, our software ignored important UI elements (*e.g.*, bottom tab buttons in Reddit). These apps did not provide alternative text on those UI elements, and therefore our software ignored them. As future work, we may also enable developers to annotate what elements should be ignored and what should be included. We can also design new hardware to support infinite scrolling (*e.g.*, a scroll wheel) so that each element will have a larger, constant size selection area.

Bookmarks could be created in all apps, and found in all but one app. To display a list of products, the Amazon app used a *WebView*, which exposed out-of-screen content to accessibility APIs. Thus, our system incorrectly found the bookmark even if it is not visible on the current screen. We solved this by checking the visibility of elements. As future work, we may examine other UI components that expose out-of-screen content.

Number entry worked as expected in all apps (except Rosa, which did not contain any text field to test). The number

pad floating window appeared when the keyboard popped up and the pressed number was correctly updated in the active text field. The backspace and announcement functions worked as well.

App switching worked in all apps, as this feature did not rely on specific app screen content.

USABILITY STUDY

To complement the feedback that informed our design of Interactiles, we conducted a study comparing Interactiles with TalkBack (the built-in Android screen reader). This study collected qualitative reactions to Interactiles and compared task completion times and user preferences.

Participants

We recruited participants (N = 5) through word of mouth, university services, and mailing lists of organizations of blind people. 3 participants were blind while 2 had some level of impaired vision. In the comparative study, all participants use mobile screen readers, primarily iOS VoiceOver. Table 2 shows their background.

ID	Age	Gender	Vision	Screen Reader Proficiency
P1	24	Male	Blind	Intermediate
P2	58	Female	Blind (L) Low Vision (R)	Basic
P3	29	Male	Blind	Advanced
P4	31	Female	Low Vision	Advanced
P5	43	Female	Blind	Intermediate

Table 2. Information on our study participants, who were all VoiceOver users. The proficiency was self-rated as basic, intermediate, or advanced.

Comparative Study Method

We conducted a comparative study in a university usability lab and in public libraries. We employed a within-subjects design to examine the completion time, accuracy, and user preference between two interaction methods: A Nexus 6P phone with TalkBack (the control condition), and the same phone with TalkBack and Interactiles (the experimental condition). Participants had the option to set up the phone with their preferred TalkBack volume and speed settings.

Participants were asked to complete four tasks that isolated specific functionality, followed by a more open-ended task to explore the system as a whole. At the beginning of each task, we explained the corresponding Interactiles feature(s) and gave the participant a training task to gain familiarity with Interactiles. The participant was also given time to do the training task with TalkBack.

For each participant and each task, we randomized the ordering of the conditions to counterbalance order effects. Participant feedback was audio recorded by the researchers. After each task, the participant was asked to give qualitative feedback and answer Likert scale questions about the speed, ease, intuitiveness, and confidence while using Interactiles and TalkBack. Upon completion of the tasks, the participant was asked to provide general feedback about Interactiles and TalkBack and their preference.

Tasks

The specific tasks in the usability study were designed to test each Interactiles feature. Tasks were chosen by considering the difficulties faced in using common apps and how Interactiles might be used in such situations. These tasks covered target acquisition (*locate)*, browsing speed (*locate, relocate, app switch)*, data entry (*mixed text/number entry)*, and spatial memory (*relocate*).

App switch: With four apps open, the participant was asked to switch from one of the four apps to another. Then the participant was asked to switch to another of the four apps. This was repeated four times in total.

Locate: The participant was asked to find a specific song in a Spotify playlist. In the Interactiles condition, the participant was encouraged to use *element navigation* to search through the current page of songs. They can slide to navigate elements and long press at the bottom of the scrollbar to move to the next page. When the participant found the song in the Interactiles condition, we encouraged the participant to *bookmark* the song for the *relocate* task described next.

Relocate: After the participant located the song in the *locate* task, the playlist was scrolled back to the top and the participant was asked to find the song again. In the Interactiles condition, we encouraged the participant to use *page scrolling* to find the bookmark from the *locate* task. Because participants were novices with respect to Interactiles, we explicitly encouraged the participant to use Interactiles functionality.

Mixed text/number entry: The participant was asked to compose a message in the default Messages app, which required entering both numbers and text. The message was dictated to the participant as follows: "My number is [phone number]. Meet me at [address]".

Holistic: This task consisted of three steps. The participant was first asked to find a specific product by name on pre-curated Amazon shopping lists without using the search bar (a water bottle using Talkback and a backpack using Interactiles). When the participant found the product, they were asked to switch to Messages to enter a contact phone number and a shipping address. In the end, the participant was asked to switch back to Amazon and add the product to their shopping cart. This task was designed to encourage participant to use all Interactiles features.

Data Collection and Analysis

Our data included times for each task, accuracy for each task, task-specific Likert scales, general Likert scales, and qualitative feedback.

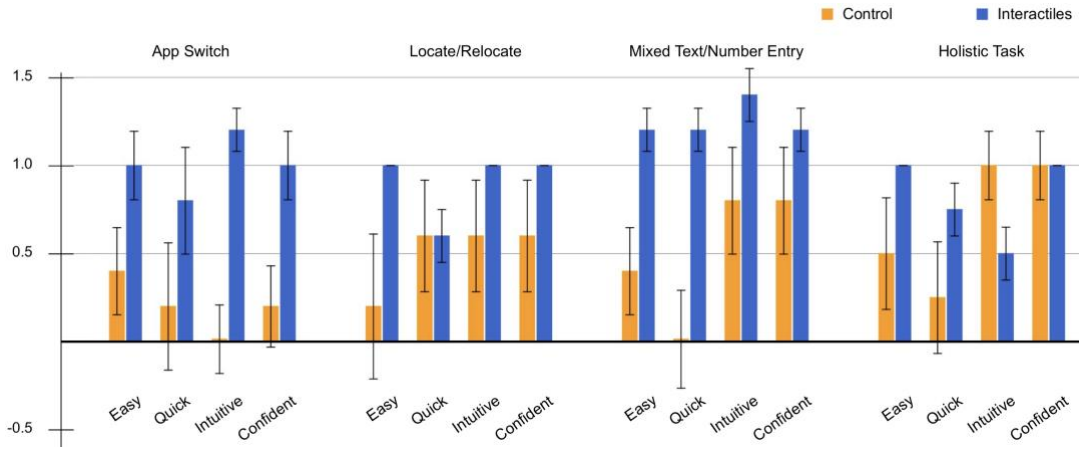

Figure 3. The average Likert scale rating (strongly disagree = -2, strongly agree = 2) with standard deviation from participants for tasks. Participants were asked how easy, quick, intuitive, and how confident they felt completing each task with the control condition (only TalkBack) and Interactiles. Locate/relocate was rated as one task. P5 did not complete or rate the holistic task.

Time data was analyzed within participants in comparing their performance using Interactiles against the control condition but could not be directly compared across participants because each participant used TalkBack at a different speed. We use bar charts to show trends in the Likert scale data. However, our sample was too small to calculate statistics for either timing or Likert scale data. The qualitative data was organized into themes by one researcher and discussed as a group until agreement was reached on what was learned.

Results

In terms of speed, Interactiles improved performance times for the *app switching* and *number entry* tasks, as can be seen in Figure 4. Participants were uniformly positive about the number pad but were mixed on the usefulness of the scrollbar and control button even though the scrollbar did result in better task completion time for the previously mentioned task. Average Likert scale ratings for each condition and task can be seen in Figure 3.

App switch: As seen in Figure 4, participants performed app switching with Interactiles almost twice as quickly as they did with just TalkBack. Participants also had a positive response to this task. *"Compared to hunting down the overview button every time, it was relatively quick and painless, especially when I figured out how to operate the slider with the thumb."*

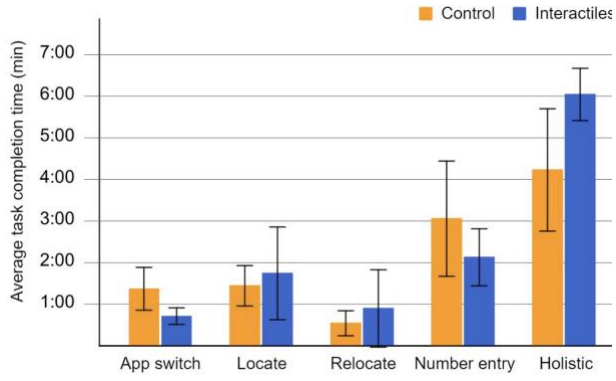

Figure 4. Average task completion time of each task in the study. P4 did not complete app switching on the control condition, and P5 did not complete the holistic task.

Locate/Relocate: The scrollbar had slower task completion times for 3 out of 5 participants than Talkback for *locate*, in which participants used *element navigation.* For *relocate*, 3 out of 5 participants had faster task completion time using Interactiles. The individual times can be seen in Figure 5. Qualitative feedback from participants confirmed that the scrollbar does not provide a performance advantage over the default Talkback element navigation methods (*i.e.*, swipe navigation or explore-by-touch). Though participants did use physical location to find elements (P1 said, *"This song should be in the middle of the scrollbar…"*), they were unsure the exact scrollbar location of the element and slid the scroll thumb slowly to avoid skipping elements. We observed that participants varied in how they touched the scroll thumb (index finger, thumb, or a two-finger pinch) and how hard they pressed the scroll thumb against the screen. P1 said, *"I feel like the scrollbar with the way it scrolls between pages isn't quite as intuitive as I would think,"* while P2 had trouble knowing when to use element navigation versus page scrolling. The confusion of where and how to use the scroll thumb was also noted by P3, who said, *"I would have kind of hoped that I could use my thumb on the scrollbar without having to lift and put it back down [to move to the next page]."* Additionally, some behaviors registered unintended touch events on the screen (*e.g.*, a long press was interpreted as two short presses because the touch contact was lost in the middle). However, participants did see potential value in the scrollbar. Though P5 felt that using the scrollbar required her to uncomfortably hold the phone, she said, *"It's just nice being able to touch something,"* and later followed up with *"I think for beginners on the phone, that scrollbar would be better."*

Although *relocate* was faster using Interactiles than TalkBack for 3 out of 5 participants with the help of the

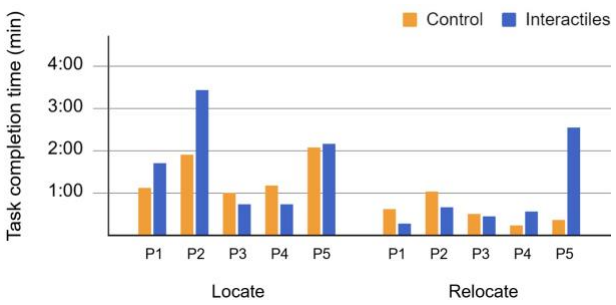

Figure 5. Individual task completion time of locate and relocate tasks.

bookmark feature, participants were skeptical of its value due to the current implementation. Participants expressed desire for being able to automatically go to a bookmark. P3 said, *"I question the usefulness of bookmarking mostly because digitally when I think of bookmark I can immediately tap a button and go to this control versus having to scroll around a find it."*

Mixed text/number entry: Using the number pad was much faster than the keyboard alone for all participants. Participants also showed strong preference for the number pad, as can be seen in Figure 3. For example, P1 said *"Where I would use this piece of hardware the most is the number pad… [to] enter the extension 156 during a phone call… I can't do that fast enough [with the default keyboard]."* Similarly, P3 liked the number pad *"for entering long strings of numbers... It saves time switching to the number/symbol keyboard and is more intuitive because the layout of the numbers on the symbol keyboard is horizontal and that takes a bit of getting used to."* P4 was quite enthusiastic, saying *"This I love. This is genius,"* but also suggested *"rather than the key that reads aloud what's on the screen, I might change that to something that's more commonly used, maybe a pound sign or perhaps a period or a hyphen."*

Holistic value: Functions overloaded to a single component created confusion for participants. For example, there were two ways of going to the "next page" (long press at the bottom of the scrollbar in *element navigation*, and a slide in *page scrolling*). These two ways confused our participants due to overloaded functionality on a single component and the mode switch required to go between them. In the words of P4, *"It felt a little clumsy figuring out up/down, when to push, how long to hold it."* On the other hand, when the mode switches and associated functionality were clear, Interactiles was valued. To use the number pad, participants had to make a conscious effort to flip the number pad onto the screen and the scrollbar off the screen. The number pad feature, although requiring a mode switch of sorts, resulted in high participant satisfaction.

Customizability

One sentiment expressed by P1, P2, and P3 was the desire to personalize Interactiles to their own needs. For example, P1 expressed enthusiasm for the number pad but did not feel he needed the scrollbar. P2 felt the hardware was uncomfortable to hold and wished she could move components around. P3 was used to holding his phone by his ear; it was difficult to use the scrollbar in that position. All participants interacted with the scroll thumb a little differently (index finger, thumb, or a two-finger pinch). The user feedback suggests that a tactile approach to mobile touchscreen accessibility needs to be physically customizable in both the types and locations of components.

DISCUSSION

Our results demonstrate that Interactiles is particularly useful for app switching and number entry, which are tasks that currently require a mode switch, but may not be as useful for tasks that are already quick even without tangibility such as locate and relocate. Our analysis also explores interactions that may be more helpful to map to a scrollbar and provides design recommendations for future work in tangible mobile touchscreen accessibility.

As shown in Figure 4, Interactiles did not provide a major speed advantage for all tasks. However, it did improve task completion time for app switching and number entry. Because participants were beginners with both Interactiles and TalkBack, this suggests that Interactiles may be of value for novice users. Given more time to learn, participants might also be more comfortable with Interactiles and find greater value in its functionality.

Interactiles was least helpful for locate and relocate. It failed to serve as a memory aid and was not reliable enough to be trusted by participants. A secure clip for holding the scrollbar to the screen to maintain screen contact would help to reduce uncertainty that resulted from inconsistent touch events. The scrollbar may be more useful for known mappings (*e.g.*, a menu) than unknowns (*e.g.*, infinite scroll). In the case of relocate, although Interactiles improved task performance for 3 out of 5 participants, participants wanted an additional feature to automatically arrive at bookmarks. Given the speed benefit of bookmarking, this could be of great value. A future implementation might include a short strip of buttons that could be used as bookmarks, similar to saved radio station buttons on cars.

Interactiles was slower for element navigation than Talkback (*i.e.*, swipe navigation or explore-by-touch). Because of the space limitations in the mobile platform, many apps use linear layouts to deliver content. Even though swipe navigation and explore-by-touch do not have tactility, they work fast enough to help the user form a coherent understanding of the app, especially when content is linear. One reason may be the common use of one dimensional layout in many small screen mobile apps. Even though swiping and explore-by-touch do not have tactility or much of a physical mapping, they work fast enough to help the user form a coherent understanding of the app, especially if content is linear. We believe this is the reason the scrollbar did not rate highly with participants, even though it did result in faster completion times for all participants in the app switching task and was faster for 3

out of 5 participants in the relocate task. Participants still provided positive feedback on having tangible feedback on the physical scrollbar.

One of the most difficult challenges for tangible or screen reader interaction on mobile platforms is infinite scroll. Ideally, there should be no distinction between elements and pages. The Interactiles approach of chunking elements into discrete pages that requires users to stop processing elements to go to the next page hurts a user's memory and understanding of the content. However, software implementations of infinite scroll not only load the next page only as needed, but also may even change the order of elements each time the user begins scrolling. Overlapping pages may help with this, but it may simply be that swiping is a better model for interacting with infinite scroll content.

Interactiles was most valuable both in task completion times and participant ratings for app switching and number entry. This suggests the interactions to target on mobile might be those that currently already require a mode switch, particularly a difficult one such as opening the symbol keyboard to enter symbols and numbers.

CONCLUSION AND FUTURE WORK

We have presented Interactiles, a system for enhancing screen reader access to mobile phones with tangible components. The success of Interactiles for app switching and number entry provides support for a hybrid hardware-software approach. Supporting materials (*e.g.*, 3D-printing files) are available at: https://github.com/tracyttran/Interactiles.

Interactiles enhances accessibility by enabling mobile touchscreen tactility across multiple existing apps. It is the first mobile system compatible across apps, as opposed to a static overlay that only works with one screen configuration. As shown in our technical validation, Interactiles functions effectively in most of the top 50 Android Apps. Another major advantage is its low cost and assembly from readily available materials. Both advantages indicate that Interactiles is highly deployable.

We would also like to explore the value of Interactiles for additional tasks such as menu access, copy/paste, and text editing/cursor movement. There is also an opportunity to enable T9 text entry [23] using the number pad and explore its design choices (*e.g.*, how to effectively present word prediction candidates).

For future work, our study suggests the importance of creating a toolchain for hardware customization. Advances needed here would include an ability to create models for "plug and play" components that could be snapped in and out of the phone shell and a facility for configuring mappings between components and tasks such as bookmarking, app switching, and text entry.

Support for adding new types of components and invoking their associated modes would also add flexibility to Interactiles. A simple solution might be to place a physical component on the screen and move a finger around its edges, using a gesture recognizer to identify the component according to its shape. If components are not all of a unique shape, an alternative would be for components to include small conductive strips that create a unique pattern when a finger is swiped over them, which can then similarly be recognized to identify the component.

ACKNOWLEDGEMENTS

This work was funded in part by the National Science Foundation under award IIS-1702751, the National Institute on Disability, Independent Living and Rehabilitation Research under award 90DPGE0003-01, and a Google Faculty Award.

REFERENCES

1. Afb.org. Statistical Snapshots from the American Foundation for the Blind. Retrieved from http://www.afb.org/info/blindness-statistics/2
2. Apple. Accessibility. Retrieved from http://www.apple.com/accessibility/osx/voiceover/
3. Shiri Azenkot, Cynthia L. Bennett, and Richard E. Ladner. 2013. DigiTaps: Eyes-Free Number Entry on Touchscreens with Minimal Audio Feedback. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2013)*, 85–90. https://doi.org/10.1145/2501988.2502056
4. Mark S. Baldwin, Gillian R. Hayes, Oliver L. Haimson, Jennifer Mankoff, and Scott E. Hudson. 2017. The Tangible Desktop. *ACM Transactions on Accessible Computing (TACCESS)* 10, 3: 1–28. https://doi.org/10.1145/3075222
5. Olivier Bau, Ivan Poupyrev, Ali Israr, and Chris Harrison. 2010. TeslaTouch: Electrovibration for Touch Surfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2010)*, 283–292. https://doi.org/10.1145/1866029.1866074
6. Blitab. Blitab | Feelings Get Visible. Retrieved from http://blitab.com/
7. Tzu-wen Chang, Neng-Hao Yu, Sung-Sheng Tsai, Mike Y. Chen, and Yi-Ping Hung. 2012. Clip-on Gadgets: Expandable Tactile Controls for Multi-touch Devices. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI 2012)*, 163–166. https://doi.org/10.1145/2371664.2371699
8. Yasmine N. El-Glaly, Francis Quek, Tonya Smith-Jackson, and Gurjot Dhillon. 2013. Touch-screens Are Not Tangible. In *Proceedings of the International Conference on Tangible, Embedded and Embodied Interaction (TEI 2013)*, 245–253. https://doi.org/10.1145/2460625.2460665
9. Google. Get started on Android with TalkBack. Retrieved from

https://support.google.com/accessibility/android/answer/6283677?hl=en

10. Google. Dashboards | Android Developers. Retrieved from https://developer.android.com/about/dashboards/index.html

11. Google. AccessibilityEvent. Retrieved from https://developer.android.com/reference/android/view/accessibility/AccessibilityEvent.html

12. Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2009. NavTap: a Long Term Study with Excluded Blind Users. In *Proceedings of the ACM Conference on Computers and Accessibility (ASSETS 2009)*, 99–106. https://doi.org/10.1145/1639642.1639661

13. 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 ACM Conference on Human Factors in Computing Systems (CHI 2017)*, 5826–5838. https://doi.org/10.1145/3025453.3025845

14. A T Guys. Tactile Screen Protector for iPhone 5C Phone Otterbox Case. Retrieved from http://www.atguys.com/store/index.php?main_page=product_info&products_id=172

15. Liang He, Zijian Wan, Stacy Biloa, Leah Findlater, and Jon E. Froehlich. 2017. TacTILE: a Preliminary Toolchain for Creating Accessible Graphics with 3D-printed Overlays and Auditory Annotations. In *Proceedings of the ACM Conference on Computers and Accessibility (ASSETS 2017)*, 397–398. https://doi.org/10.1145/3132525.3134818

16. Humanware. Victor Reader Stream. Retrieved from https://store.humanware.com/hus/victor-reader-stream-new-generation.html

17. 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 ACM Conference on Computers and Accessibility (ASSETS 2008)*, 73–80. https://doi.org/10.1145/1414471.1414487

18. 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 ACM Conference on Computers and Accessibility (ASSETS 2009)*, 115–122. https://doi.org/10.1145/1639642.1639663

19. Shaun K. Kane, Meredith Ringel Morris, Annuska Z. Perkins, Daniel Wigdor, Richard E. Ladner, Jacob O. Wobbrock, and Meredith Ringel Morris. 2011. Access Overlays: Improving Non-Visual Access to Large Touch Screens for Blind Users. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2011)*, 273–282. https://doi.org/10.1145/2047196.2047232

20. Shaun K Kane, Meredith Ringel Morris, and Jacob O. Wobbrock. 2013. Touchplates: Low-cost Tactile Overlays for Visually Impaired Touch Screen Users. In *Proceedings of the ACM Conference on Computers and Accessibility (ASSETS 2013)*. https://doi.org/10.1145/2513383.2513442

21. Lifewire. Review: Touchfire Keyboard for the iPad. Retrieved from https://www.lifewire.com/review-touchfire-keyboard-ipad-1994430

22. David McGookin, Stephen Brewster, and WeiWei Jiang. 2008. Investigating Touchscreen Accessibility for People with Visual Impairments. In *Proceedings of the Nordic Conference on Human-Computer Interaction: Building Bridges (NordiCHI 2008)*, 298–307. https://doi.org/10.1145/1463160.1463193

23. Nuance. T9 Text Input. Retrieved from https://www.nuance.com/mobile/mobile-solutions/text-input-solutions/t9.html

24. World Health Organization. 2011. World Report on Disability. Retrieved from http://www.who.int/disabilities/world_report/2011/en/

25. OwnFone. Make Your OwnFone. Retrieved from https://www.myownfone.com/make-your-ownfone

26. Mario Romero, Brian Frey, Caleb Southern, and Gregory D. Abowd. 2011. BrailleTouch: Designing a Mobile Eyes-free Soft Keyboard. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI 2011)*, 707–709. https://doi.org/10.1145/2037373.2037491

27. Statista. 2016. Number of Available Applications in the Google Play Store. Retrieved from http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

28. Stevezhanggeek. CHI2017Demo. Retrieved from https://github.com/appaccess/CHI2017Demo

29. Brandon Taylor, Anind Dey, Dan Siewiorek, and Asim Smailagic. 2016. Customizable 3D Printed Tactile Maps as Interactive Overlays. In *Proceedings of the ACM Conference on Computers and Accessibility (ASSETS 2016)*, 71–79. https://doi.org/10.1145/2982142.2982167

30. Touchfire. Touchfire Ultra-Protective Magnetic Case and 3D Keyboard. Retrieved from https://touchfire.com/

31. Daniel Trindade, André Rodrigues, Tiago Guerreiro, and Hugo Nicolau. 2018. Hybrid-Brailler: Combining Physical and Gestural Interaction for Mobile Braille Input and Editing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2018)*. https://doi.org/10.1145/3173574.3173601

32. Xiaoyi Zhang, Anne 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 ACM Conference on Human Factors in Computing Systems (CHI 2017)*, 6024–6037. https://doi.org/10.1145/3025453.3025846