Ultra-Low-Power Mode for Screenless Mobile Interaction

Jian Xu, Suwen Zhu, Aruna Balasubramanian*, Xiaojun Bi*, Roy Shilkrot*

Department of Computer Science
Stony Brook University
Stony Brook, New York, USA
{jianxu1, suwzhu, arunab, xiaojun, roys}@cs.stonybrook.edu

ABSTRACT

Smartphones are now a central technology in the daily lives of billions, but it relies on its battery to perform. Battery optimization is thereby a crucial design constraint in any mobile OS and device. However, even with new low-power methods, the ever-growing touchscreen remains the most power-hungry component. We propose an Ultra-Low-Power Mode (ULPM) for mobile devices that allows for touch interaction without visual feedback and exhibits significant power savings of up to 60% while allowing to complete interactive tasks. We demonstrate the effectiveness of the screenless ULPM in text-entry tasks, camera usage, and *listening* to videos, showing only a small decrease in usability for typical users.

CCS Concepts

•Human-centered computing → Touch screens; Smartphones; Mobile phones; •Software and its engineering → Power management;

Author Keywords

Power Saving; Mobile System; Text Entry; Touchscreen;

INTRODUCTION

Smartphones play a crucial role in people's daily lives. It is now hard to imagine living *without* a smartphone ready at hand, yet this happens to nearly everyone on a daily basis - when the device is out of battery. Power consumption has always been a major concern for mobile phone users, manufacturers and application providers, as batteries drain more quickly with the ever-increasing usage. This forces some users to charge their phones more than once daily when under intense usage (such as browsing on a high-speed cellular connection), or limit their use when outside. It is not uncommon to see people carrying bulky power banks with them, or have a mobile charger ready on their person for "casual charging" in public spaces.

When the battery charge goes critically low (less than 5%), completing a specific task before the phone shuts down becomes a race against time. Certain operations, such as texting a long email, watching a video stream or capturing videos are prohibitively draining, as the phone may expire mid-task and work may go unsaved. This critical situation is so common that there's a very popular app called Die With Me [7] allowing users to share their desperation on low battery levels.

Figure 1: Picture of use cases of ULPM. The left shows the normal cases of using smartphones to take videos, write text messages and watch online video; The right side shows users do not need to turn on the screen for interaction by applying ULPM, which saves power consumption especially at critically low battery level.

To alleviate this problem, we propose our Ultra-Low-Power Mode (ULPM) system, which is designed to extend mobile battery lifetime, by turning off the screen while keeping the foreground app touch-interactive. Our system allows users to complete important or power consuming tasks at critical battery levels with non-visual interaction, relying on residual or "muscle" memory.

Non-visual interaction with touchscreen and mobile devices is a daily activity for many visually impaired (VI) persons. At high levels of expertise, VI persons can achieve incredible speeds and a wide range of usage of standard phones in ac-

*Alphabetic order

UIST '18, October 14–17, 2018, Berlin, Germany

ISBN 978-1-4503-5948-1/18/10…$15.00

DOI: https://doi.org/10.1145/3242587.3242614

cessibility modes using touch gestures [28, 23]. Non-VI users can also, to a large extent, interact with mobile devices without visual feedback. Research on imaginary user interfaces [18, 19] and invisible keyboards [38] has shown that users develop strong memories on the interface locations and are able to interact with devices with little visual cues. Inspired by this, we hypothesize non-VI persons can make efficient use of non-visual interaction utilizing their *visual memory*, and reap the benefit of a long battery life without sacrificing much in capability.

However, there are several system challenges involved in making this screenless touch interaction feasible. When a user presses the power button to turn off the display, applications become non-responsive since the OS disables the input events to the application. All foreground apps become inactive and the UI and the graphics stack are destroyed. An alternate to switching the display completely off is to set the screen to *zero* brightness in software. The zero brightness mode reduces the backlight but does not completely switch off the display. Our measurements show that *zero* brightness draws 50% more power compared to switching the display off.

Our work, ULPM (Ultra-Low-Power Mode) enables the mobile device to be touch-responsive even when the display is turned off. The key problem is that when the smartphone screen is turned off, all applications (except background services) are stopped and the entire UI stack from the kernel down to the screen hardware is destroyed. ULPM utilizes specialized kernel-level code that alters the behavior of the OS touch event submodule as well as the touchscreen hardware. Specifically, ULPM keeps the application active and keeps alive the connection between the keyboard event and the application. ULPM then caches a copy of the application UI tree (which is an intermediate representation of the UI), but destroys the graphics stack. This has two advantages: the application UI can be updated even if the UI is not rendered on the screen, and when the display is switched back on the entire UI does not have to be recreated, saving power.

Another challenge is designing a non-visual interaction scheme for inherently visual, non-trivial and highly-practiced tasks, such as text-entry. On one hand, the finger-eye coordination heavily used in these tasks is broken, and therefore performance level can drop. On the other hand, the visual information can be redundant, as non-visual mobile interaction has shown to be possible at acceptable words-per-minute (WPM) rates [38].

To this end, in ULPM we explored a new screenless keyboard interaction and an exploratory research of other screenless interactions, such as “listening" to a YouTube video, and using the camera to capture a video when the screen is switched off. Compared to the two use cases—listening to videos and using the camera with the screen switched off, typing on a virtual keyboard requires the most visual feedback and user attention.

We designed a new keyboard layout that achieves better screenless typing with acceptable typing speed and accuracy. Specifically, we removed non-alphabetic keys including the *space bar* and *delete button*, and used swipe gestures to perform specific actions such as committing the word, using the space bar, and deleting words. We also placed more blank space in the keyboard to avoid mis-tapping and added audio feedback (which consumes negligible power). We conducted user studies with 16 participants. The users achieved an average typing speed of 32 WPM and 8.66% Word-Error-Rate (WER) compared to 42 WPM and 1.86% WER on a visible keyboard. For the use case of capturing a video with the display switched off, we conducted a user study with 12 participants. We found that users were able to track the scene even without constant visual feedback, relying on their motor memory and scene position approximation.

Importantly, we show that ULPM decreases power consumption significantly compared to reducing the brightness levels to 50% or even 0%. Recall that reducing the brightness to 0% is a software approach; instead, ULPM switches the display off completely at the hardware level. For text entry, ULPM reduces the power consumption by 66% compared to when the display brightness is set to a constant 50%. ULPM reduces power consumption by 22% compared to when the phone brightness is 0% *and* the power-save mode is turned on. Results are quantitatively similar for the other two test cases—listening to YouTube videos and capturing videos with the display switched off.

RELATED WORK

Our work builds on prior related works on invisible user interfaces and power saving techniques.

New Ways of Interaction

Previous research has explored the possibilities of supporting interactions without visual feedback. Findlater et al. [8] showed that expert typists exhibit spatially consistent key press distributions even when the keyboard is not visible. Mottelson et al. [27] developed a swipe-based invisible text entry method on a smartwatch. Users could enter text at 10.6 WPM after 30 minutes of training. Lu et al.'s work [24] showed that users could perform eyes-free typing at 17-23 WPM on a touchpad with the keyboard displayed on the TV screen. Commercial keyboards, such as the Fleksy Keyboard [9] also supports an invisible keyboard mode. Gustafson et al. [18] showed how participants can draw basic characters on an imaginary sketchpad. Zhu et al. [38] presented an invisible keyboard with an adapted spatial model.

Other research into *alternative* modes of interaction with the device screen suggested using multitouch gestures detected through the pocket fabric, from simple touch strokes to full alphanumeric text entry [30], as well as utilizing mobile motion sensor data for interaction with minimal attention from the user [21].

There has also been considerable work in *accessible* user interfaces for users with visual impairments. For example, Kane et al. [23] designed Slide Rule to address interfaces inaccessible to blind users. White et al. [34], Jayant et al. [22] and Vázquez et al. [33] helped users with visual impairments capture photos by providing continuous audio feedback to properly aim the camera toward an object.

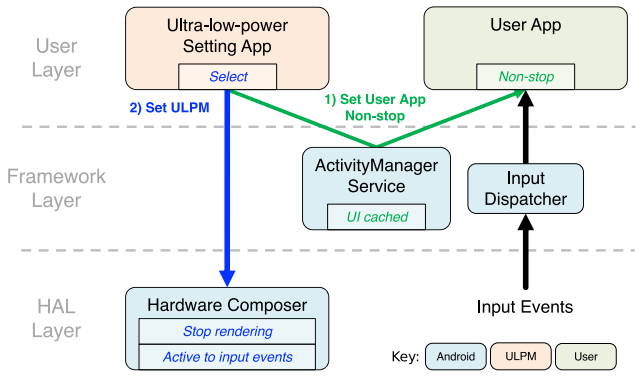


Figure 2: High-level system design of our Ultra-Low-Power Mode platform. A user-level interaction (e.g. power button press) can signal the Hardware Composer (HWC) to go into ULPM.

The focus of these new interaction paradigms is not about power savings. As we discuss in the next section, switching the backlight completely off for power savings creates new system challenges because the application becomes non-responsive.

Power Saving System Techniques

He et al. [20] dynamically scale the resolution of the smartphone display depending on the distance between the user and the screen for saving power while not affecting user experience.

Dalton et al. [6] proposes a system that turns the display on/off based on the presence of a user. UIWear [36] automatically generates wearable apps by extracting UI from corresponding smartphone apps. UIWear still performs UI updates even when the application moves to the background and the application UI is destroyed. However, UIWear is not designed for screenless interactions so that it does not make the phone app responsive when the screen turned off.

Samsung offers an Ultra Power Saving Mode [29] that saves power consumption by changing the color of the screen to grayscale and restricting other functionalities such as WiFi and Bluetooth. Our system goes one more step further by disabling screen displaying rather than changing screen color.

SYSTEM DESIGN AND IMPLEMENTATION

The goal of ULPM is to allow user interactions when the smartphone display is turned off for ultra-low-power mobile usage.

Motivation and Challenges in designing ULPM

Our initial power measurements show why designing ULPM while still conserving power is challenging.

Figure 3 shows comparative energy consumption when the display is completely switched off versus when the display is switched to different brightness levels (0%, 50%, etc) as allowed by the operating system. We show the energy consumption when the display brightness is 100% and under low power saver mode for comparison. The low power saver

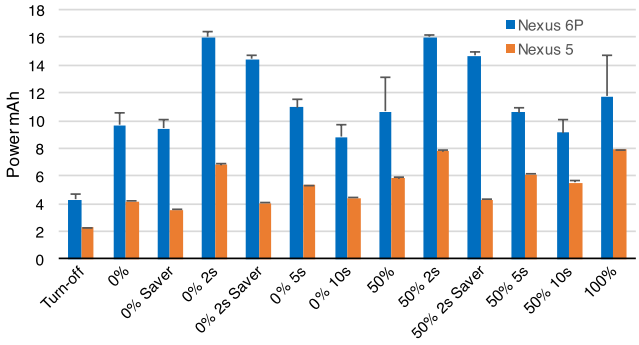


Figure 3: Power Consumption of varying brightness, using power save mode, and when the display is switched on and off periodically at a certain interval. Varying the screen brightness only has a minor effect on power savings compared to completely switching the screen off. Switching the screen on and off at periodic intervals results in large power drain. For example, switching the screen off and on at two-second interval consumes more power (even when using the power-save model) compared to keeping the screen on at a constant 50% brightness.

mode [17] is an android mechanism that saves power by disabling some energy consuming functionalities, such as Google Maps navigation. These experiments were performed on a Nexus 5 and Nexus 6P over a one-minute interval. The measurements were conducted 5 times, and the figure shows the average and error bars show the 95th percentile confidence interval. The figure also shows the power consumption when the display is switched on and off at a certain interval but we discuss those results later.

The percentage difference in energy consumption between when the display is completely switched off and when the display is at 0% brightness is 55% for Nexus 6P and 46% for Nexus 5. However, the difference between 0% brightness and full brightness is only 18% for Nexus 6P and 46% for Nexus 5. This suggests that when the display is at 0% brightness (very dim screen, practically unusable), there is still considerable power consumption compared to when the display is completely switched off. The reason is that rendering is not stopped when the screen is at 0% brightness.

We design ULPM to operate when the display is completely switched off with no rendering, for power savings. A series of activities are triggered when the display is switched off. The phone turns off the display matrix, the capacitive touch grid sensor, and the rendering pipeline. In addition, the operating system stops all applications, tears down the UI tree [12], and releases memory buffers. The UI tree is a representation of the UI that is used for rendering. Finally, the phone enters an "interaction sleep cycle" that keeps various background services running while shutting down all user-facing operations including keyboard interactions. This sleep mode is highly energy efficient by not only switching off the power-hungry display [5], but also stopping computationally-heaving rendering activities.

Two key challenges emerge when operating with the display switched off. The obvious challenge is that users can no longer interact with the application because the application's user-facing operations are shut down and the UI tree is destroyed. A second challenge is that there is a large overhead when switching the display back on because the UI stack needs to be reconstructed and rendered.

Figure 3 shows the energy consumption when the mobile device is periodically switched on and off at 2-second and 5-second intervals. More energy is spent when the display is switched on and off at 2-second intervals compared to steadily holding the brightness at 50%.

ULPM design

We designed ULPM to overcome the two challenges described above. ULPM (i) switches the screen off while allowing users to interact with the application, and (ii) considerably reduces the overhead of periodically switching the display back on. This second part is crucial because we envision that the users periodically switch the display on for sanity checks.

ULPM turns off the display at the hardware layer to go beyond the zero brightness level supported by the operating system. However, the key is to *not* destroy the UI tree and instead cache it at Activity Manager, which manages the life-cycle of UIs. Our observation is that, while not rendering the UI saves power, storing the UI tree does not increase power consumption (as we show in the Evaluating Section §5). Importantly, by caching the UI tree, it does not have to be reconstructed when the display is turned on, reducing the switching overhead.

To implement the ULPM system, we modified existing Android OS by inserting a new mode called "Ultra-Low-Power". Figure 2 shows the overall system design. The Ultra-Low-Power setting enables users to configure certain apps to be responsive when the display is switched off.

Recall that when the display is off, applications receive a stop signal which makes the application inactive. Instead, ULPM sends a binder message to *ActivityManagerService*, which is the module for managing the life-cycle of UI in Android OS. Based on users' preference, ULPM changes the application to *Non-stop* status allowing apps to update their UI and respond to user interactions. Importantly, the *ActivityManagerService* caches the UI tree. This allows the application to continually update the UI tree, but the UI is not rendered on the screen for saving power.

We stop UI rendering by modifying the hardware composer [14] (that renders the final image on the display) but allow user interactions. User input events are generated when users tap on the screen through the *InputDispatcher* [13] as shown in Figure 2. Our modifications allow user inputs to be delivered to the application.

We implemented these changes in Android 7 (Nougat) and evaluated over Nexus 5 and Nexus 6P phones. The phones need to be rooted and loaded with the customized ROM that imports our ULPM feature. However, this design can be ported to other Android versions and other OSes, as the design of Hardware Composer has been widely used across different OSes, such as Tizen. We also envision that these changes be incorporated by the phone vendors to save power.

USE CASES

We envision the use of ULPM for three use cases—text entry, "listening" to video content, and capturing videos. These form a representative set of tasks that present-day mobile users perform on their phones. Text entry, for example, accounts for 40% of the mobile device users time [2]. Users are increasingly watching videos on mobile devices. For example, YouTube reports that video was 80% of total web traffic in 2017, and mobile video consumption increases by 100% year-over-year [10]. Similarly, with advanced lenses and sensors built into mobile phones, the sales for digital cameras has dropped by nearly 80% since 2010 [32] as more users capture photos and videos with their smartphones [4].

Technically, ULPM does not require any modification to the off-the-shelf apps. However, we need to design new interaction techniques so users can interact with the app even without visual feedback. The text entry use case is the most complicated and so we largely focus on it; the other two cases are more straightforward and do not need any customization.

Preliminary Survey of Desired Use Cases

We conducted a brief survey of 35 users (students and staffs in a university) where we asked them *"what tasks would you want to perform when your battery capacity is low?"*. The results showed that text entry was one of the most predominant tasks: over 82% users chose text entry related apps (e.g., Email, Messenger) as the most wanted task.

Although only a small number of users chose using the phone camera (4 users) or watching videos (2 users) as a low-battery task, power saving was a desired feature for these apps and users tend to use these apps more often with the power saving feature enabled. This reported low usage is partially due to the lack of power saving mode for these apps and their power draining nature.

We asked the same 35 participants *"When the phone battery goes low, how likely would you use text entry, capturing video, and video watching apps, separately? If we've developed a system X that drastically saves power when phone battery goes low, what's your answer?"* The median ratings (1 - least likely, 5-most likely) increased from 4 to 5 for text entry, from 2 to 3 for capturing video, and from 2 to 3 for listening to videos. 14 users said they would likely or very likely use video capturing in low power condition with the power saving feature. 12 users said they would likely or very likely listen to videos with the power saving feature.

Use case #1: Text Entry

Text entry is one of the most commonly performed tasks with mobile devices. A study on iPhone usage [2] showed that more than 40% of users' mobile activities involved intensive text entry, including sending emails, typing messages, searching information online, making phone calls, updating the calendar, and many others. As a result, it is critical that ULPM is able to support user interaction with mobile devices, even when the display is switched off.

We take inspiration from recent work [38] on typing on an invisible keyboard that leverages user's motor memory. Their motivation is to extend the display area of useful content which is often obstructed by the keyboard. In ULPM, we go further by supporting text entry when the screen is off.

In the ULPM typing mode, users do not get visual feedback and can only rely on their motor memory. Since users would not be able to utilize the suggestion bar to correct their typing mistakes, we enabled auto-correction whenever the literal string was an Out-of-Vocabulary word to reduce typing errors. Below we describe some of the design decisions we made in ULPM.

Re-designing the keyboard layout: The original version of the keyboard as shown in Figure 8a does not work with ULPM because it greatly increases the probability of mistyping. For example, if users accidentally press the *delete* button, *space* bar, *language switch* or *suggestion* bar instead of the alphabetic keys, users will lose their context of where they are typing because those buttons could trigger other UI layouts.

Instead, we re-designed the layout and removed keys that are not alphabets and are not required during blind typing. To avoid mis-tapping, we placed more blanks in the keyboard layout. We also ensure that the relative height of keyboard remains unchanged because users are already used to the keyboard size and position. Figure 8b shows the redesigned keyboard.

The key idea of the keyboard re-design is to leverage users' motor memory of the Qwerty layout in text entry tasks even when the screen is turned off. To this end, the layout of the keyboard is mostly the same as an Android keyboard, for both the screen-on (Figure 8a) and the screen-off (Figure 8b) versions. Specifically, the positions and sizes of alphabetic keys remain unchanged, while we replaced functional keys such as *Backspace* and *Space* with blank space.

However, functionalities such as *delete* and *space* are essential for typing. We replace these keys with two gestures, *swipe-left* and *swipe-right* respectively. When users want to delete content, they can swipe left to delete an entire word instead of a single character. Deleting words is known to be more efficient on invisible keyboards [38]. After finishing typing each word, users swipe right to commit the word and append a space to the text.

Adding audio feedback: Since there is no visual feedback, we use the Android TextToSpeech [11] API to provide audio feedback during text entry. After a word is committed (right swipe), the entered word will be read to the users. We found that the increase in energy consumption when providing audio feedback was less than 1%.

Use case #2: Listening to Videos

ULPM allows users to listen to online videos when the screen is switched off to save power. Users already use paid services such as Youtube Red [37, 3] to *listen* to video recordings. ULPM will allow users to switch off the screen and listen to videos for any video application, not requiring a paid service. In the section of EVALUATING ULPM POWER SAVINGS (§5), we show that this mode of interaction can significantly reduce power consumption.

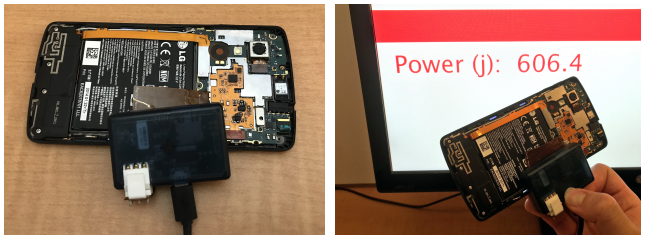


Figure 4: The BattOr hardware we used for power draw measurement.

Use case #3: Capturing photos and videos

Users keep their phone screens on while capturing videos, although theoretically they can capture the videos without the display. Switching the display off while capturing videos using ULPM can save power. We envision that users periodically calibrate the position and angle of the smartphone by occasionally turning the screen on. Our user study shows that users do not require the display to be switched on to capture videos. Capturing videos without seeing the screen would just induce acceptable deviation. The power measurement also shows using ULPM in this context saves power consumption.

EVALUATING ULPM POWER SAVINGS

The goal of the system evaluation is to measure power savings when using ULPM. To figure out the maximum power saving value we can get, we turned on power-save mode when measured the case of ULPM mode.

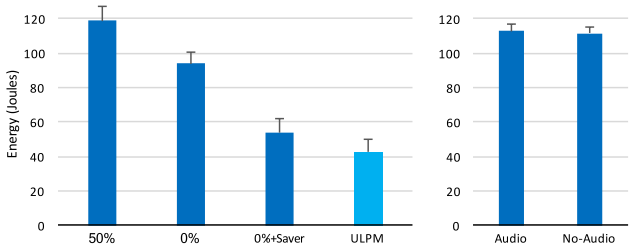
All our experiments were conducted on Nexus 6P and Nexus 5. We use the BattOr [31] hardware tool to measure the power consumption on Nexus 5. BattOR, is a small, noninvasive hardware device that interposes between a device and its battery (see Figure 4). BattOR collects power measurements on the order of tens of microseconds and is known for high accuracy. BattOR does not work on Nexus 6P. Instead, we use dumpsys [16] software to measure the power consumption on Nexus 6P. The energy saving is estimated as *(E1 - E2) / E1*, where E1 is the old power energy consumption and E2 is the new energy consumption.

Text Entry

To emulate the typing gesture, we utilize the *adb* [15] tool to issue a typing event every second.

Figure 5a shows the power draw when typing under different conditions. ULPM (with power-save mode turned on) saves 66% power compared to when the screen brightness is at a constant 50% without power-save mode. Brightness set at 0% with the Android power saver mode enabled [17] saves more power because it disables a number of services, such as background data syncing. However, ULPM even saves 22% power compared to the case of 0% brightness with Android power saver mode enabled.

Typing with audio feedback. We measured the power consumption for typing with and without audio feedback. During the



(a) Power draw while typing (b) Feedback

Figure 5: Power Consumption: (a) typing on various brightness levels, Android power-save mode or ULPM enabled. (b) using audio feedback has an insignificant increase in power draw.

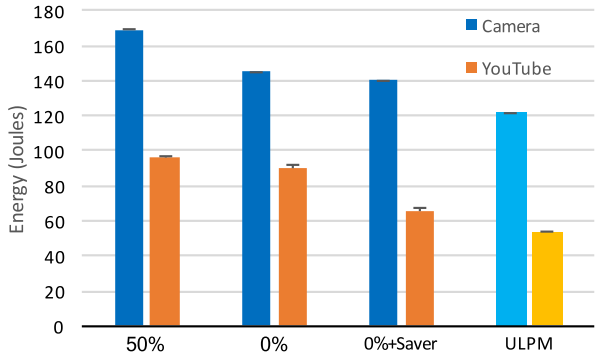


Figure 6: Power consumption while playing YouTube videos and using the camera for capturing videos.

measurement, we tuned the audio volume to the magnitude of 30%, which was used in our user studies. The volume is at a comfortable magnitude for users. The measurement was repeated 5 times each for 60 seconds. Figure 5b shows that the audio feedback consumes negligible power.

YouTube streaming

In this experiment, we measured the power consumption when playing a YouTube video for 60 seconds, and average over 5 runs. As before, we also experimented with different screen brightness versus ULPM. Figure 6 shows that ULPM (with power-save mode turned on) saves 30% power compared to when the screen is at 50% brightness (without power save mode). When the Android power save mode is turned on, even when the screen brightness is set to 0%, ULPM saves 18% power.

Capturing videos

We measured the power improvements when using ULPM (with power-save mode turned on) to capture videos. We took a 60-second video and repeated the experiment 5 times under different brightness levels. Figure 6 shows, by turning off the screen, ULPM saves 47% power over the case of half brightness. When compared to 0% brightness with power-save mode, ULPM saves 13% power. While this may not seem dramatic, in 0% brightness mode the visual feedback is barely usable, unless in complete darkness, and yet excess power is drained.

USER STUDY: TEXT ENTRY ON ULPM

We conducted a set of user studies to answer the following research questions:

- *Can users type when the screen is turned off? If so, how is the comparative typing speed and error rate?*
- *Does visual feedback help by switching on the screen occasionally? Is more frequent switching-on better?*

Experiment Setup

The study was a within-subject design. The independent variable was the four conditions for typing.

- ***Regular keyboard with screen on***: Participants used the regular keyboard as shown in Figure 8a. During the study, the screen was always switched on. This was the baseline of the user study. After finishing each trial, users should press the green triangle button in Figure 8 to proceed to the next trial.
- ***ULPM keyboard with screen off***: Participants used the ULPM keyboard as shown in Figure 8b. ULPM was enabled. When the study started, participants were asked to manually turn off the screen by pressing the power button. During the study, participants were *not allowed* to switch on the screen unless they finished the current sentence. Upon finishing each trial, participants pressed the power button to turn on the screen to proceed to the next trial.
- ***ULPM keyboard with screen on for each word (i.e., word-wise)***: Participants used the ULPM keyboard and ULPM was enabled. The screen was on when the study initially started. The screen turned off automatically once the participants started typing each word. The screen turned on automatically after the participants swiped right to commit a word. We call this set of user study as the *wordwise* condition.
- ***ULPM keyboard with turning on screen by users' preference (i.e.,user-controlled)***: Participants used the ULPM keyboard and ULPM was enabled. When the study started, participants were asked to manually turn off the screen by pressing the power button. During the study, participants were *allowed* to switch on the screen occasionally by their own preference. We call this set of user study as the *user-controlled* condition. Upon finishing each trial, participants pressed the power button to turn on the screen to proceed to the next trial.

The task was a phrase transcription task. The phrases were randomly selected from the Mackenzie and Soukoreff phrase set [26]. The same set of phrases was used for each condition, but randomized in order. The first four phrases were used as a practice session. Each task consisted of four blocks, and each block consisted of eight phrases. A short break of one minute was enforced between each block to help participants get rid of fatigue. The order of these four conditions was counterbalanced using a Latin square [35].

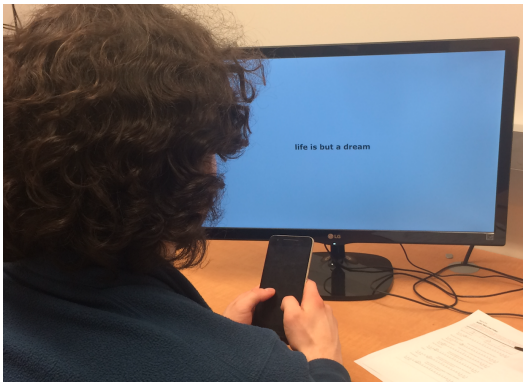
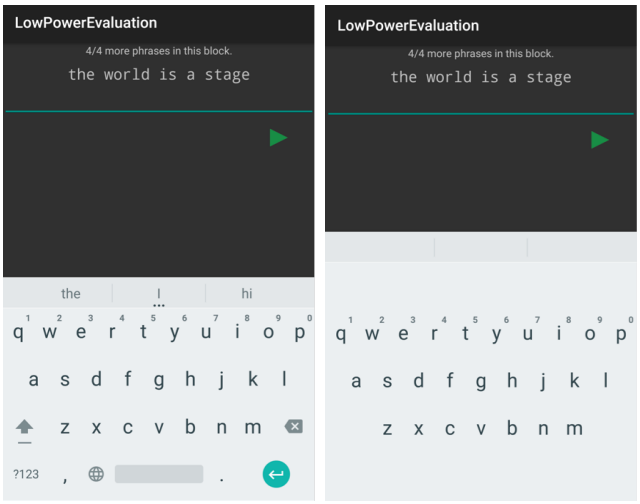


Figure 7: User typing with the screen turned off during the user study. The monitor screen was used to show the current phrase content user was transcribing.



(a) Regular keyboard. (b) ULPM keyboard.

Figure 8: Screenshots of the keyboards used in the user studies. The left one (a) is a regular keyboard used in Android system by default; the right one (b) is the keyboard specially designed for ULPM for better performance.

Figure 7 shows the setup of the user study. Participants were seated in an office setting and instructed to type on the device with two thumbs. The monitor was used to show the phrase content while participants were transcribing in the screen off condition, as they might forget what they should transcribe when the screen was turned off. Participants were instructed to type as fast and natural as possible. To be consistent, audio feedback was turned on for all conditions.

Figure 8 shows the application we developed for this user study. It collects comprehensive data regarding user typing, such as timestamps for each typing event and the events of switching the screen on or off, the words typed, etc. Throughout the study, we used a Nexus 6P phone running with Android 7 OS equipped with ULPM.

Participants

We conducted a user study with 16 users (6 female) aged between 20 and 34 ($M = 25.3$). 5 were undergraduate students, and 11 were graduate students. The average of their self-reported familiarity with the Qwerty layout and touchscreen typing was 4.7 out of 5 (1: not familiar, 5: very familiar).

Results

Text entry speed. We calculated the text entry speed following Mackenzie [25]:

$$WPM = \frac{|T-1|}{S} \times 60 \times \frac{1}{5}, \quad (1)$$

where T is the target string and S is the elapsed time in seconds from the first to the last touch in the sentence.

The average speed for each condition is shown in Figure 9. The average speed for the screen on condition (42.18 WPM) was faster than the other three conditions. ANOVA showed the input condition had a main effect on the average text entry speed ($F_{3,45} = 17.72, p < 0.001$). Pairwise comparison with Holm adjustment showed the difference was significant between the screen on and the other conditions ($p < 0.001$ for wordwise and user-controlled, $p = 0.00176$ for screen off).

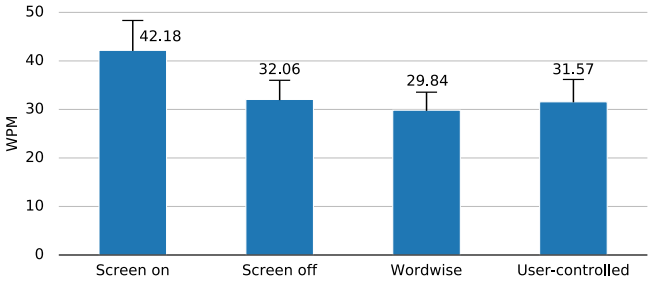


Figure 9: Means (95% confidence interval) of text entry speed of each condition.

Learnability. A more fine-grained analysis showed that users can learn typing with screen off quickly. Figure 10 shows the average input speed of each block for the four conditions. Users practiced on the ULPM keyboard for a total of 4 blocks, ~3 minutes per block (~12 minutes in total). The average typing speed in the first block was 28.64 WPM, and improved to 36.28 WPM in the last block.

ANOVA showed that the block number had a main effect on the average text entry speed ($F_{3,45} = 13.09, p < 0.001$). Pairwise comparisons with Holm adjustment showed the differences were significant between the first block and the third/fourth block (both $p < 0.001$), as well as between the second block and the third/fourth block ($p = 0.01113$ and $p = 0.00261$ respectively).

Error rate. We measured error rate using word error rate [1], which is the minimum word distance between the target string and the final transcribed string, divided by the length of the target string. The error rate was much higher for the screen off condition. The average error rate (SD) was 1.86% (1.66%) for

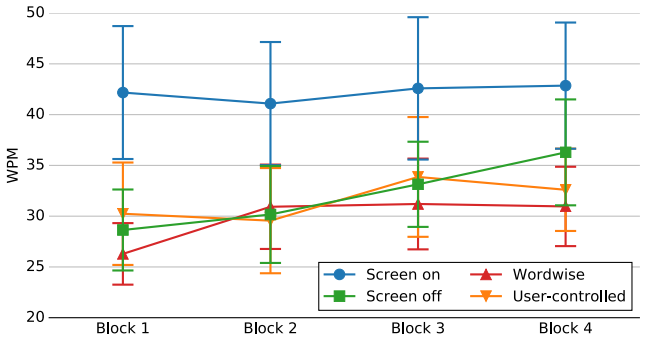


Figure 10: Means (95% confidence interval) of text entry speed by block.

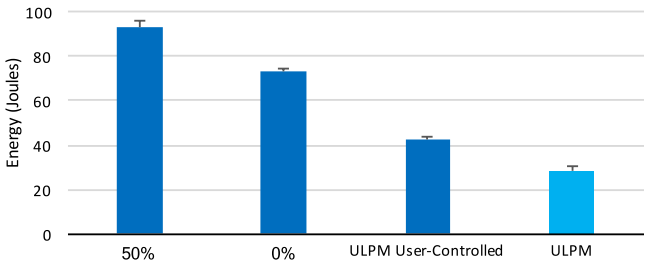


Figure 11: Power consumption by emulating the user study of User-Controlled case. We can see that it still saves 42% power with ULPM than the 0% brightness case.

the screen on condition, 8.66% (8.15%) for the screen off condition, 4.49% (6.27%) for wordwise, and 4.27% (4.16%) for the user-controlled condition. An error rate of 4% is low compared with other text entry methods, such as QWERTY [27, 38]. Moreover, the error rate could be reduced by occasionally turning on the screen, as evidenced from the user-controlled case, in which the error rate reduced by half from 8.66% to 4.27%. ANOVA showed a main effect of the input condition on the error rate ($F_{3,45} = 6.587, p < 0.001$). Pairwise comparisons with Holm adjustment showed the difference was significant for screen on vs. screen off ($p = 0.0198$) and screen off vs. wordwise ($p = 0.0051$).

Subjective measures. Users were asked to rate the four conditions on a $1-5$ scale (1: very dislike, 5: very like) at the end of the study. The average rating was 4.66 for the screen on condition, 3.34 for screen off, 2.84 for wordwise, and 3.84 for user-controlled.

Participants also commented about their experience. Six users commented that they dislike the wordwise condition, which turns the screen on and off too frequently that disturbs their typing performance. Eight users felt more comfortable when typing with the screen off compared with turning the screen on frequently.

Power measurement for User-Controlled case. The app we developed for the user study collected timestamps of each typing events during the study. The data collected can be used to understand how often participants would turn on the screen during typing, and how long it lasts during each occasional turning-on session.

It turned out that on average, participants would turn on the screen to take a peek at the screen every 9.3 seconds. Once the screen was turned on by participants, it lasted 0.93 second on average until it was turned off again.

We measured the power consumption by emulating the use case of participants switching the screen on and off arbitrarily in the user-controlled case, since it is the most popular case among the three screen-off typing conditions, according to the participants' subjective feedback. As discussed in the previous section, high frequency of switching the screen on and off consumes even more power than keeping the screen turned-on all the time. We measured the power consumption to confirm that in the most popular case of screen-off typing, and with the screen on/off pattern we collected from the user study, ULPM is still able to save energy.

We emulated this case by repeatedly switching the screen on every 9 seconds and then turning off the screen after 1 second. We ran the test for 60 seconds, 5 times. We compared the results with other cases as well, including keeping screen on at 0%, 50% brightness, and ULPM without turning on and off back-and-forth.

As Figure 11 shown, the power consumption of emulating the user-controlled user study saves over 50% power than half brightness case. From the result, we can see that the user-controlled case combined with ULPM still consumes 42% less power than keeping the screen at zero brightness.

The reason for not consuming so much power as Figure 3 shown is that the UI in ULPM isn't destroyed and reconstructed repeatedly even when switching the screen on and off back-and-forth. Therefore, in the the case of switching the screen on arbitrarily, i.e. User-Controlled case, ULPM still saves a reasonable magnitude of power.

USER STUDY: ULPM VIDEO CAPTURING

We conducted another user study to answer the following question:

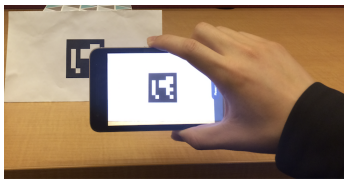
- *Can users capture an acceptable video when the screen is turned off ?*

Experiment Setup

We invited participants to capture a 20-second video by holding a Nexus 6P phone on hand. The video was about a static object on the table, at ∼1m away. Participants were divided into two groups. The first group was asked to first capture videos with the phone screen always on and capture videos with the screen off afterwards. Study conditions (screen-on and ULPM) were counterbalanced. Participants were asked to keep the object in the center of the video as much as possible.

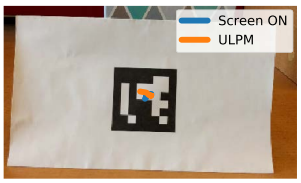
Participants.

We conducted a user study with 12 participants (4 female) aged between 23 and 28 ($M = 25.1$). These participants were all graduate students. Their average rating of the familiarity

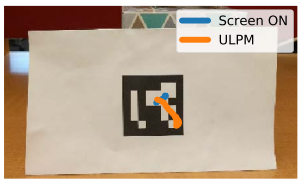


(a) Camera with screen on.

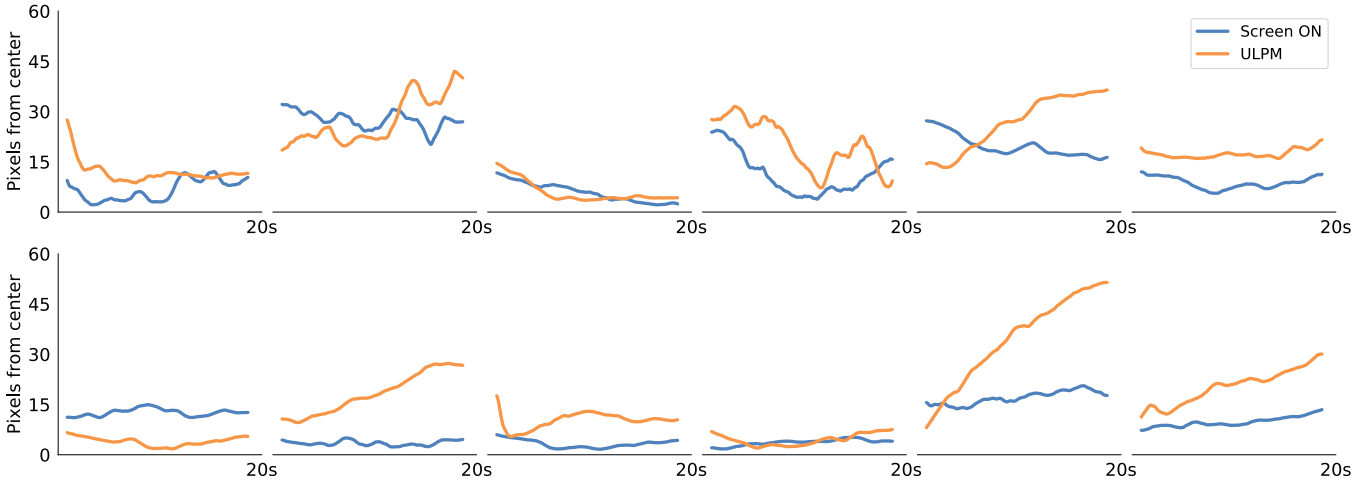
(b) Camera with screen off.



(c) User 3 tracking.



(d) User 11 tracking.



(e) Tracking results from users. A 20s measurement had ∼600 frames at 30 FPS. Box-filter smoothing ($N = 31$) was applied to reduce noise.

Figure 12: Photo of capturing a video of the marker in different conditions. Participants were asked to keep aligning the marker in the center of the video. (a) is capturing a video with the screen turned on; (b) is capturing a video with the screen turned off and ULPM enabled. (c) and (d) are the screenshots of the videos taken by user 3 and user 11 respectively. (e) shows the tracking results after analyzing the videos.

with using the phone camera was 4.6 out of 5 (1: not familiar, 5: very familiar).

Results

We analyzed the video recordings from users to measure how successful they were in filming the static object. Offset values are the pixels offset Euclidean distance from the center of the object to the center of the video image. The size of the analyzed image is 576×324 pixels. For image analysis we used OpenCV[1] and Aruco[2] to find the marker in the image and calculate the offset distance.

Figure 12e shows the results of tracking after analyzing the videos from both conditions. An independent-samples t-test was conducted to compare the offset in the screen-on and ULPM conditions. There was a significant difference in the scores for screen-on (10.914 ± 7.742) and ULPM (16.853 ± 11.249) conditions; $t(12322) = -36.25$, $p = 0.000$. These results suggest that using the ULPM has an effect on keeping the object centered, as hypothesized. However, the average difference in offset is roughly 6 pixels ($\sim 0.9\%$ of the diagonal image size), so we conclude using the ULPM may present but a minor effect on centering the object.

[1] https://opencv.org/

[2] https://www.uco.es/investiga/grupos/ava/node/26

DISCUSSION

We have shown that setting the mobile system to an interactive screenless mode makes for compelling power savings over traditional power-save modes, and we hope this will encourage OS and device manufacturers to consider adding this capability in their core offering. Screenless interaction was shown to take a mild hit on effectiveness, and not cripple the overall action. Users were able to use their memorization of common UI elements and visualize imaginatively their on-screen actions. Text entry suffered a drop of just 8 WPM with an increase of ∼3% in word error rate after little practice, and using the screenless camera had a very mild effect on object-centering success.

Our method is easily portable to most Android devices, using a relatively simple modification of the OS kernel. While using our mode in other OSs, such as iOS, may pose a bigger engineering challenge, there are other considerations for the success of the ULPM. Low-power screens, such as OLED-based technologies, are taking a bigger portion of the market. ULPM may have a lesser effect in such devices, thus we still need to investigate the power gains in a wider variety of devices.

The limitation of our investigation lies in the choice of users, which is derived from the study control variables. In so far as we test for QWERTY input, we only select users with a

solid grasp of the QWERTY layout so their residual visual memory can aid in finger placement. Interaction with a phone, though, usually spans much more than just QWERTY input. We selected to examine video consumption (YouTube) and production (shooting a video with the camera), which are very common usages of present-day mobile phones. Many more major applications have yet to be tested, such as social media and news feed consumption.

Additionally, we discovered certain limitations of the implementation. Some users noted that the ULPM feels like typing in a masked password text box and that sometimes leads them to opt for deleting their entire input and starting from the top, rather than attempting to correct without proper visual feedback. We are therefore looking into adding better correction mechanisms, as well as support for special characters and numeric input.

CONCLUSION

Energy conservation is a tough problem in smartphones that attracted a number of systems researchers, such as [20, 5]. The first contribution of this paper a system called *Ultra-Low-Power-Mode*, i.e. ULPM, which enables an operation mode with minimal power consumption by switching off the screen. ULPM allows certain apps to remain active in the background while the screen is off and respond to user interactions. We solve the challenge of enabling apps to be background-active by setting the UI status to *Non-stop*. To keep the screen responsive even when it is turned off, we import a new power mode called *Ultra-Low-Power-Mode* in the Android Hardware Abstract Layer (HAL) that enables the touchscreen to remain active to touch events.

We validated the feasibility of screenless smartphone interaction through several user studies. Studying screenless keyboard typing, results show an acceptable rate of 36 WPM when the screen is turned off, which is on-par with typing with the screen on at 42 WPM. We've also conducted a study around capturing videos with the screen off, which showed only a mild drop in object-centering performance (a difference of $\sim$0.9% in off-center disparity).

We conclude that the ULPM is a viable option for a very low-power mobile phone operation relying on visual and habitual practice of long-time users, inspired by non-visual phone usage of VI persons. When the screen is off, visual and muscle memory engage and the user's blind finger placements turn out to be more accurate than random. The ULPM is based on a readily deployable kernel modification, which can be implemented in the majority of Android phones worldwide.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. We thank our user study participants. This work was supported in part by NSF CSR-1717973.

REFERENCES

1. Xiaojun Bi, Shiri Azenkot, Kurt Partridge, and Shumin Zhai. 2013. Octopus: Evaluating Touchscreen Keyboard Correction and Recognition Algorithms via. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 543–552. DOI: http://dx.doi.org/10.1145/2470654.2470732
2. Barry Brown, Moira McGregor, and Donald McMillan. 2014. 100 Days of iPhone Use: Understanding the Details of Mobile Device Use. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI '14)*. ACM, New York, NY, USA, 223–232. DOI:http://dx.doi.org/10.1145/2628363.2628377
3. Business Insider. 2017a. News about Youtbe Red. (2017). http://www.businessinsider.com/heres-where-youtube-red-stands-after-one-year-2017-1
4. Business Insider. 2017b. People will take 1.2 trillion digital photos this year – thanks to smartphones. (2017). http://www.businessinsider.com/12-trillion-photos-to-be-taken-in-2017-thanks-to-smartphones-chart-2017-8
5. Xiaomeng Chen, Abhilash Jindal, Ning Ding, Yu Charlie Hu, Maruti Gupta, and Rath Vannithamby. 2015. Smartphone Background Activities in the Wild: Origin, Energy Drain, and Optimization. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. ACM, New York, NY, USA, 40–52. DOI: http://dx.doi.org/10.1145/2789168.2790107
6. Angela B. Dalton and Carla S. Ellis. 2003. Sensing User Intention and Context for Energy Management. In *Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9 (HOTOS'03)*. USENIX Association, Berkeley, CA, USA, 26–26. http://dl.acm.org/citation.cfm?id=1251054.1251080
7. Dries Depoorter and David Surprenant. 2018. Die With Me. (2018). http://diewithme.online
8. Leah Findlater, Jacob O. Wobbrock, and Daniel Wigdor. 2011. Typing on Flat Glass: Examining Ten-finger Expert Typing Patterns on Touch Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2453–2462. DOI: http://dx.doi.org/10.1145/1978942.1979301
9. Fleksy. 2016. Fleksy Keyboard - GIFs, Custom Extensions, and Themes. (2016). http://fleksy.com/
10. Forbes. 2017. 17 Stats And Facts Every Marketer Should Know About Video Marketing. (2017). https://www.forbes.com/sites/miketempleman/2017/09/06/17-stats-about-video-marketing/
11. Google. 2016a. Android API documentation: TextToSpeech. (2016). https://developer.android.com/reference/android/speech/tts/TextToSpeech.html
12. Google. 2016b. Android API documentation: Use node tree debugging. (2016). https://developer.android.com/guide/topics/ui/accessibility/node-tree-debugging.html

13. Google. 2016c. Android Open Source Project documentation: Input. (2016). https://source.android.com/devices/input/

14. Google. 2016d. Android Open Source Project: Hardware Composer. (2016). https://source.android.com/devices/graphics/arch-sf-hwc

15. Google. 2016e. Android Studio documentation: Android Debug Bridge (adb). (2016). https://developer.android.com/studio/command-line/adb.html

16. Google. 2016f. Android Studio documentation: dumpsys. (2016). https://developer.android.com/studio/command-line/dumpsys.html

17. Google. 2017. Google support: Manage your battery. (2017). https://support.google.com/nexus/answer/6187458?hl=en

18. Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. 2010. Imaginary Interfaces: Spatial Interaction with Empty Hands and Without Visual Feedback. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 3–12. DOI: http://dx.doi.org/10.1145/1866029.1866033

19. Sean Gustafson, Christian Holz, and Patrick Baudisch. 2011. Imaginary Phone: Learning Imaginary Interfaces by Transferring Spatial Memory from a Familiar Device. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 283–292. DOI: http://dx.doi.org/10.1145/2047196.2047233

20. Songtao He, Yunxin Liu, and Hucheng Zhou. 2015. Optimizing Smartphone Power Consumption Through Dynamic Resolution Scaling. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. ACM, New York, NY, USA, 27–39. DOI: http://dx.doi.org/10.1145/2789168.2790117

21. Scott E. Hudson, Chris Harrison, Beverly L. Harrison, and Anthony LaMarca. 2010. Whack Gestures: Inexact and Inattentive Interaction with Mobile Devices. In *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '10)*. ACM, New York, NY, USA, 109–112. DOI: http://dx.doi.org/10.1145/1709886.1709906

22. 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 (ASSETS '11)*. ACM, New York, NY, USA, 203–210. DOI: http://dx.doi.org/10.1145/2049536.2049573

23. 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 (Assets '08)*. ACM, New York, NY, USA, 73–80. DOI: http://dx.doi.org/10.1145/1414471.1414487

24. Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. 2017. BlindType: Eyes-Free Text Entry on Handheld Touchpad by Leveraging Thumb's Muscle Memory. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 18 (June 2017), 24 pages. DOI: http://dx.doi.org/10.1145/3090083

25. I. Scott MacKenzie. 2015. A Note on Calculating Text Entry Speed. (2015). http://www.yorku.ca/mack/RN-TextEntrySpeed.html

26. I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems (CHI EA '03)*. ACM, New York, NY, USA, 754–755. DOI:http://dx.doi.org/10.1145/765891.765971

27. Aske Mottelson, Christoffer Larsen, Mikkel Lyderik, Paul Strohmeier, and Jarrod Knibbe. 2016. Invisiboard: Maximizing Display and Input Space with a Full Screen Text Entry Method for Smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16)*. ACM, New York, NY, USA, 53–59. DOI:http://dx.doi.org/10.1145/2935334.2935360

28. 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 (ASSETS '15)*. ACM, New York, NY, USA, 273–280. DOI: http://dx.doi.org/10.1145/2700648.2809861

29. Samsung. 2018. Samsung Ultra Power Saving Mode. (2018). https://www.samsung.com/ca/support/mobile-devices/galaxy-s7-how-do-i-use-ultra-power-saving-mode-on-my-samsung-galaxy-s7/

30. T. Scott Saponas, Chris Harrison, and Hrvoje Benko. 2011. PocketTouch: Through-fabric Capacitive Touch Input. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 303–308. DOI: http://dx.doi.org/10.1145/2047196.2047235

31. Aaron Schulman. 2016. BattOr: Portable Power Monitor for Mobile Phones. (2016). http://cseweb.ucsd.edu/~schulman/battor.html

32. Statista.com. 2017. Have Digital Camera Sales Bottomed Out? (2017). https://www.statista.com/chart/5782/digital-camera-shipments/

33. Marynel Vázquez and Aaron Steinfeld. 2014. An Assisted Photography Framework to Help Visually Impaired Users Properly Aim a Camera. *ACM Trans. Comput.-Hum. Interact.* 21, 5, Article 25 (Nov. 2014), 29 pages. DOI:http://dx.doi.org/10.1145/2651380

34. Samuel White, Hanjie Ji, and Jeffrey P. Bigham. 2010. EasySnap: Real-time Audio Feedback for Blind Photography. In *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 409–410. DOI: http://dx.doi.org/10.1145/1866218.1866244

35. Wikipedia. 2017. Latin square. (2017). https://en.wikipedia.org/wiki/Latin_square

36. Jian Xu, Qingqing Cao, Aditya Prakash, Aruna Balasubramanian, and Donald E. Porter. 2017. UIWear: Easily Adapting User Interfaces for Wearable Devices. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom '17)*. ACM, New York, NY, USA, 369–382. DOI: http://dx.doi.org/10.1145/3117811.3117819

37. YouTube. 2018. Youtbe Red. (2018). https://www.youtube.com/red

38. Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/3173574.3174013