RainCheck: Overcoming Capacitive Interference Caused by Rainwater on Smartphones

Ying-Chao Tung
Paul G. Allen School | DUB Group
University of Washington
Seattle, WA, USA
yct56@cs.washington.edu

Mayank Goel
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
mayank@cs.cmu.edu

Isaac Zinda
Harvey Mudd College
Claremont, CA, USA
me@isaaczinda.com

Jacob O. Wobbrock
The Information School | DUB Group
University of Washington
Seattle, WA, USA
wobbrock@uw.edu

ABSTRACT

Modern smartphones are built with capacitive-sensing touchscreens, which can detect anything that is conductive or has a dielectric differential with air. The human finger is an example of such a dielectric, and works wonderfully with such touchscreens. However, touch interactions are disrupted by raindrops, water smear, and wet fingers because capacitive touchscreens cannot distinguish finger touches from other conductive materials. When users' screens get wet, the screen's usability is significantly reduced. RainCheck addresses this hazard by filtering out potential touch points caused by water to differentiate fingertips from raindrops and water smear, adapting in real-time to restore successful interaction to the user. Specifically, RainCheck uses the low-level raw sensor data from touchscreen drivers and employs precise selection techniques to resolve water-fingertip ambiguity. Our study shows that RainCheck improves gesture accuracy by 75.7%, touch accuracy by 47.9%, and target selection time by 80.0%, making it a successful remedy to interference caused by rain and other water.

CCS CONCEPTS

• **Human-centered computing** → **Interaction techniques**;

KEYWORDS

Rain water, capacitive touch sensing, smartphones, touch, gesture, swipes, situational impairments, improved sensing.

ACM Reference Format:
Ying-Chao Tung, Mayank Goel, Isaac Zinda, and Jacob O. Wobbrock. 2018. RainCheck: Overcoming Capacitive Interference Caused by Rainwater on Smartphones. In *2018 International Conference on Multimodal Interaction (ICMI '18), October 16–20, 2018, Boulder, CO, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3242969.3243028

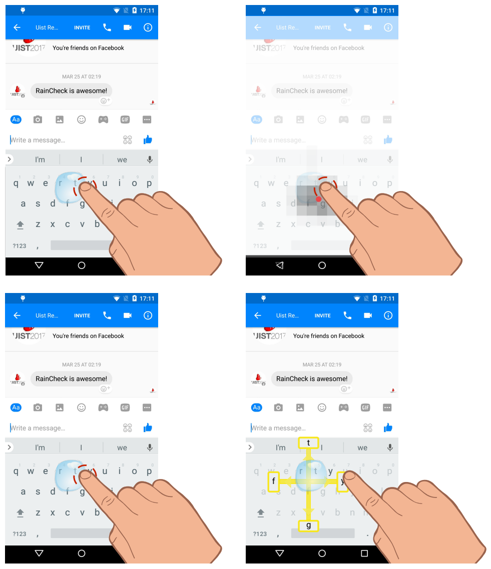

Figure 1: The upper figures show how rainwater interferes with sensing finger location. The water (top-left) shifts the sensed finger location to the left and down (top-right). RainCheck detects unusual capacitive signatures (bottom-left) and, among other features, offers a target selection technique to resolve ambiguity (bottom-right). In this case, RainCheck suggests a gesture (swipe) to select UI elements that were affected by the water-covered area.

1 INTRODUCTION

People often use their smartphones while walking, commuting, or in other outdoor situations. Currently, touch is the most dominant way users interact with modern mobile devices. However, commodity capacitive-sensing touchscreens built on smartphones are not customized to sense only human finger touches. They are engineered to detect any conductive or dielectric material, and the human skin is just one of many possibilities.

Therefore, raindrops, water smears, or even sweat can make it hard for the touchscreen to detect real human finger touch. For example, in the United States, approximately 2.8% of the population has hyperhidrosis or excessive sweating of the palms [8]. Smartphone touchscreens generally cannot differentiate between water and fingers. When touchscreens are wet and fingers come into contact with them, the water disrupts the sensed location of the finger (see Figure 1, top right). The result is what we might call a "situational fat finger problem." When a person is using his or her smartphone in the rain, usability decreases, and the person tends to focus much more on getting touches to work, ruining situation-awareness and posing a potential safety hazard.

Much research has explored how to improve the users' text entry and target selection accuracy while walking [6, 14, 17, 22]. Kane et al. [12] proposed the term Walking User Interfaces (WUIs) and evaluated screens whose elements grew in size when the user was walking. Moreover, Yamabe [30] adapted the size of fonts and images automatically while walking by using accelerometer information. That said, how weather might affect the interaction between users and smartphones while walking remains unexplored.

To address the problem of wet fingers or touchscreens, we present RainCheck (Figure 1), a system that differentiates between rainwater and human touches. By extracting features from the low-level raw capacitive data stream from commodity touchscreen drivers, RainCheck adapts to spurious touch events and uses a target disambiguation technique that works well even when the touchscreen is wet.

We first report some observations of people using their smartphones in the rain, describing behaviors such as wiping phone screens on various parts of the body to remove rainwater. We then report results from a controlled study showing that RainCheck improves gesture accuracy by 75.7%, touch accuracy by 47.9%, and target selection time by 80.0%, reducing target selection time in the presence of water from 2.5 to 0.5 seconds. To achieve the latter two results, RainCheck utilizes a well-known precise selection technique, Escape [31], to solve the "situational fat finger problem" that occurs when the human finger comes into contact with water and the screen at the same time.

The contributions of this paper are as follows: (1) the development of RainCheck, which uses low-level capacitive sensor information and employs a well-known precise selection technique to improve interaction on wet smartphone touchscreens; (2) empirical results from a controlled study showing that RainCheck significantly improves touch accuracy and reduces interaction time compared to the same touchscreen without RainCheck.

2 RELATED WORK

In this section, we briefly review work related to capacitive sensing technologies, situational impairments and precise selection techniques for touchscreens, the three areas of work most related to this project.

2.1 Capacitive Sensing Technologies

Generally, capacitive-sensing technology can detect anything that is conductive or has a dielectric differential with air, which can cause confusion when multiple such objects are in contact with the screen. To address this issue, new phones often use a combination of self- and mutual-capacitance. Combining self- and mutual-capacitance can help because water and human-touch have different capacitive signatures in most cases. When the screen has either water or a finger, this approach works most of the time. However, it fails often enough to degrade the usability of the device, and can still lead to users becoming frustrated.

Recently, hardware manufacturers have developed different approaches to improve the noise rejection capabilities of touchscreens [5, 16, 26]. However, to the best of our knowledge, no prior work has proposed any software-only solution to disambiguating touches from water. Unlike hardware-based solutions, software-based solutions can be applied to smartphones on the market using only firmware or software upgrades. The low incremental cost of software-based solutions might also make it an attractive alternative to hardware-based solutions, which can be more expensive.

2.2 Situational Impairments

Prior work has conceived of how "situational impairments" [24] may worsen a user's interactions with mobile devices [33]. Situational impairments might be caused by numerous factors [29], including ambient temperature, light levels, noise, and bodily movements, to name a few. Goel et al. [7] argued that current mobile devices still lack appropriate awareness of context, situation, and environment, limiting successful interaction between devices and users. Moreover, Pascoe et al. [20] indicated that using mobile devices while walking is the basic requirement of fieldwork users with mobile systems. Other research on text entry [17, 32]and target selection [14, 22] while walking has tried to make mobile devices more useful while users are on-the-go.

Context-awareness has also been highlighted as one of the key user abilities that can be compromised when mobile devices are used in the wild [15, 19, 20]. For example, Sarsenbayeva et al. [21] explored how cold environments affect users' ability to interact with mobile devices.

2.3 Precise Selection Techniques for Touchscreens

The challenge of enabling precise touch interactions on touchscreens has been investigated by numerous researchers (e.g., [3]). The "fat finger problem" has also been recognized and discussed for years (e.g., [23]). Precision-Handle [1], Shift [27], Escape [31], and LucidTouch [28] are four of the many solutions to address this problem. Projects have also addressed text entry on ultra-small touchscreens [4, 18, 25].

The interference of users' touch by water can be viewed as a type of fat finger problem because the touch area is enlarged (and distorted) by the size of the water's contact area. RainCheck leveraged concepts from prior work in the design of its selection technique to overcome this interference.

3 OBSERVATIONS OF SMARTPHONE INTERACTIONS IN THE RAIN

To better understand the problem of rainwater interference on capacitive touch screens, we conducted informal field observations of eight people using their smartphones on a rainy day in <location

anonymized>. The observations were conducted in late winter 2017 over three consecutive rainy days. On the days in question, weather service data show 1.3 - 2.0 cm rainfall occurred. Participants were observed in a public setting while walking with their smartphones in the rain (Figure 2B), attempting to perform tasks of their own choosing. These tasks included reading and replying to email, reading news, texting friends, checking bus arrival times, and scrolling a Facebook or Twitter feed. Our observations of people performing these tasks lasted about 10 minutes per participant, for about 90 minutes of total observation time.

The rainwater interference was so bad that participants often gave up on whatever particular task they were doing, switching to other tasks only to encounter similar problems. Starting with a dry smartphone screen, it took only about 20 seconds before rainwater became a significant hindrance to successful interaction (Figure 2A). Many taps on the screen went unrecognized, or were recognized as occurring in locations far from the actual tap position. Single-finger swipes were often mistaken for two-finger pinch actions that inadvertently zoomed in on photos or maps.

To cope with these problems, participants engaged in a variety of rainwater-thwarting behaviors. For example, participants often wiped off their phone screens using their shirt or sweater, or the side or back pocket of their pants. This wiping behavior resulted in "water smear", a thin veneer of water spread evenly across the smartphone screen. A few participants wiped not only their devices but the tips of their thumbs, trying to remove water from their hands as well as their hardware. It was clear from these observations that rainwater indeed presents a significant situational impairment and barrier to mobile interaction.

4 THE DESIGN OF RAINCHECK

In this section, we describe the design of RainCheck, detailing the key choices that we made to overcome rainwater interference on smartphone screens.

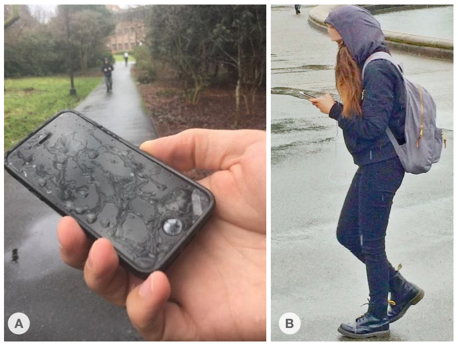

Figure 2: (A) Rainwater on a smartphone screen after only 20 seconds of exposure during a rain shower. The drops on the screen cause significant interference when attempting to use the smartphone, rendering it almost unusable. (B) A woman walks in the rain during one of our observations. Although this image might suggest otherwise, it was actually raining quite hard at the time this photo was taken. Note the wet pavement and puddles visible.

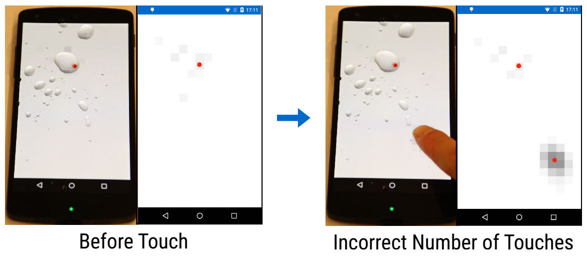

Figure 3: The user touches the screen with a single finger, but the touchscreen is confused due to the presence of water and detects two touch points (two red dots). However, the shape and distribution of sensor values in the finger-touch area were very different than those of water drops, giving us an opportunity for improvement.

4.1 Gaining Access to Low-level Raw Capacitance Data

In order to process the capacitive signature of rainwater, we had to gain access to the low-level output from a smartphone touchscreen driver. After surveying various smartphones that would make this technically feasible, we chose to implement RainCheck on an LG Nexus 5 running Android 5.0.2 with a Linux kernel [1] specifically modified to expose the debug interface of a Synaptics ClearPad 3350, the touchscreen driver. We obtained capacitive sensor values from this programmatic interface: an 8-bit 15×27 pixel capacitive image at 20 frames per second, in which each image pixel corresponds to a 4.1×4.1 mm square on the screen [11]. The interface also provided the inferred touch locations that were used by Android's application layer. With the debug interface, we visualized the raw data and touch points provided by the firmware on the smartphone to understand how the capacitive touchscreen reacts to water drops or water smear. The next section describes our observations of the interference caused.

4.2 Water Interference Pattern

To understand the way water interferes with our testbed smartphone, we systematically applied various water patterns mimicking rainwater using an eye-dropper. Our resulting observations revealed two types of sensing errors: sensing a phantom touch and sensing incorrect touch positions. In sensing a phantom touch, the smartphone treated some water drops as finger-touch points (Figure 3), which explains why sometimes a user's single-touch was interpreted as a two-finger pinch in our field observations. Moreover, in many cases, both fingers and touchscreens were wet, and most modern touchscreens interpret the water to be the part of the finger, so the inferred touch center shifts (Figure 4). This problem is the second problem mentioned above, that of sensing incorrect touch positions.

4.3 The RainCheck Prototype

RainCheck's solution to the water interference problem has two parts: (1) differentiating between water and human touch based upon the characteristics of each, and (2) utilizing a disambiguating

[1]The modified kernel and dataset will be made open-source.

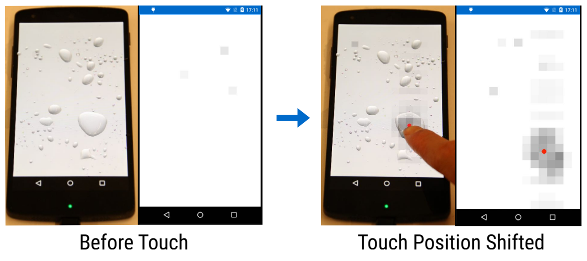

Figure 4: Apart from being detected as touch points, water drops also affect the center of the touch area. When the user touches a water drop and the touchscreen simultaneously (right), the touch-position extracted from the blob's center shifts because blob size also includes the size of the water.

interaction technique in cases where differentiating between water and a human finger fails.

4.3.1 Water vs. Human-Touch. From the low-level raw capacitive data received from RainCheck's modified kernel, we generated one grayscale image per frame. Next, we extracted the potential touch area by blurring and detecting blobs in the image. Then, we analyzed characteristics of each identified blob to possibly detect a difference between a blob representative of a water drop and blobs typically formed by human touch. We extracted the blob's shape, size, and the distribution of the raw capacitive values covered by the blob. Compared to real finger-touch points, a "touch point" made by water had a lower peak sensor value, and the shape was smaller and irregular (Figure 3, right). We devised a heuristic-based model that used the peak sensor value, and the ratio of width and height of each blob, to differentiate between water and finger-touch. Similar solutions were used and explored for the issue of unintended touches caused by wrist, palm, forearm, or stylus on touchscreen devices [2, 9, 10]. The "touch point" made by those body parts or conductive devices, compared to the human finger touch, were in more irregular or larger shapes. The detailed pseudo code of RainCheck was shown in **??**.

4.3.2 Adapting to Spurious Touch Events. A tap interaction must produce a precise touch position; otherwise users cannot select targets accurately. By contrast, swipes and other gestures require relatively less precise touch positions because gestures unfold over time, producing more data on which to base a smartphone's response. Therefore, to make tap interactions successful in the presence of rainwater, we applied a tap-then-swipe interaction, which was similar to the approach presented in Escape [31], to improve target selection while selectable objects are dense, to help users select targets even with water interfering with the touch area.

When features are extracted from touch blobs, RainCheck tests for abnormal touch events (such as unusual touch size, irregular shape, or lower peak sensor values in the touch area); it then checks if the touch area is covered by any selectable UI elements. Next, RainCheck utilizes its tap-and-swipe selection technique to allow users to select their intended target. For example, in Figure 1, the water drop is covering keys "R", "T", "F", and "G". Once RainCheck detects a touch within that water drop, it displays a pop-up that lets the user select these keys by quickly swiping in different directions. Thus, the selection is made by swiping, and the actual position of the swipe is of relatively little importance, much like a marking menu [13]. Figure 5 shows the control flow for RainCheck and the control condition, discussed further in the next section [2].

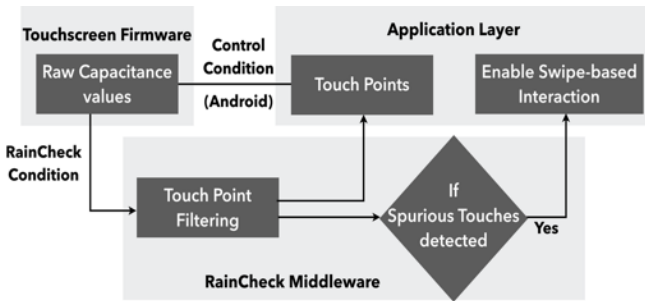

Figure 5: The control flow of RainCheck and its various components.

5 RAINCHECK EVALUATION

To evaluate the effectiveness of RainCheck, we conducted two studies to check whether RainCheck would improve (1) touch recognition, and (2) target selection performance in the presence of water. For a laboratory evaluation, the consistency of the environment is important, so we decided not to rely on actual outdoor rain and instead apply water to a touchscreen surface in a controlled fashion using a spray bottle.

5.1 Study 1: Gesture Performance

Our field observations indicated that most gesture recognition errors with water drops or water smear on a touchscreen were due to the smartphone recognizing the incorrect number of touch points. In this study, we wanted to assess the accuracy of recognizing the correct number of human finger-touch points while performing gestures.

5.1.1 Participants. Twelve participants (8 males, 4 females) ranging in age from 20 to 29 years (M = 22.9, SD = 2.6) were recruited. All participants had more than two years of experience with touch screen smartphones. Eleven participants reported that they "sometimes" attempted to use their smartphones in the rain. Only one participant had "little experience" with using smartphones in the rain because, in her telling of it, she was not comfortable using a wet touch screen.

5.1.2 Apparatus. Participants used our custom experiment software on an LG Nexus 5 that has a 4.95-inch capacitive touch screen with 1080 × 1920 pixels and our customized Linux kernel allowing access the touchscreen debug interface. The application recorded capacitive sensor values and touch points' positions from the debug interface at 20 frames per second.

[2]The RainCheck algorithm and touch data in our study will be made open-source.

5.1.3 Procedure. Participants were presented with six different touch-gesture tasks in random order under two different environment settings, a dry touchscreen and a wet touchscreen. Each task was presented 30 times, so each participant was required to perform 360 touch gestures in all (6 tasks × 30 times × 2 environments). To simulate how people used their smartphones in the rain, we sprayed water on the touchscreen with a spray bottle and then wiped the screen with an Adidas water-resistant sports jacket (Figure 6). The result was a realistic combination of water drops and water smear, much like what appears in Figure 2.

With 12 participants, 6 touch gestures (each repeated 30 times), and 2 environments, a total of 12×6×30×2 = 4320 touch gestures were made in this portion of the study.

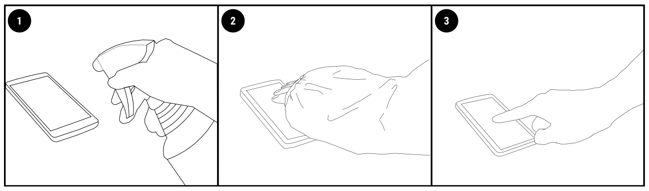

Figure 6: The procedure for each trial is shown above. The first step was to spray water on the touchscreen. The second step was to wipe the screen with the provided jacket. The third step was for participants to perform the task shown on the screen.

5.1.4 Tasks. According to our field observations, people often used messaging, navigation, and social media applications in the rain, among others. Common touch gestures shown in these applications could be categorized into six types: Swipe Up, Swipe Down, Swipe Right, Swipe Left, Pinch Open, and Pinch Closed. We evaluated our system with these touch gestures (see Figure 7).

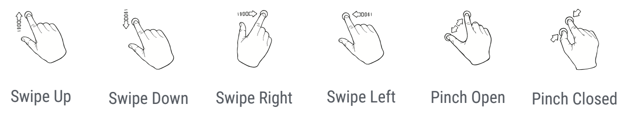

Figure 7: The six touch gestures participants performed in our first study to evaluate the performance of RainCheck.

5.1.5 Results. The results of the touch-gesture tasks are shown in Figure 8. In total, we recorded 4320 gesture trials for two environments, a dry touchscreen and a wet touch screen, i.e., 2160 trials for each. To get our data for these trials, we extracted only screen frames that had human finger touches in them, and then sent the corresponding touch points provided by the current system to an Android simulator to measure their accuracy. Afterwards, RainCheck processed the extracted frames and then sent the touch points it detected to the Android simulator, too. In the wet condition, error rates of the unmodified system and RainCheck were 21.34% and 5.2%, respectively. Overall, RainCheck resulted in a 75.7% reduction in errors in the presence of water on the touchscreen (461 error gestures vs. 112 error gestures). In the wet condition, normality was violated (w = 0.822, p < .001). A nonparametric analysis was therefore used. There was a significant effect of Method (RainCheck - Regular) on error rate with Wilcoxon's signed-rank test (p < .001). RainCheck significantly outperformed than the regular system on LG Nexus 5.

In the dry condition, the accuracy of RainCheck was 99.25% over 2160 gestures, which is awfully close to the perfection of the regular system (accuracy: 100%). In total, 16 cases were recognized incorrectly by RainCheck (swipe right: 2, swipe left: 1, pinch closed: 5, pinch open: 5, swipe down: 3, and swipe up: 0). The reason for these errors was that RainCheck filtered out some smaller and ambiguous touch points in some frames while performing these gestures, so the simulator could not recognize the correct gesture. We believe that the situation can be avoided if we collect touch point data from more users or apply a machine learning technique in the future. Even still, we are quite close to perfect.

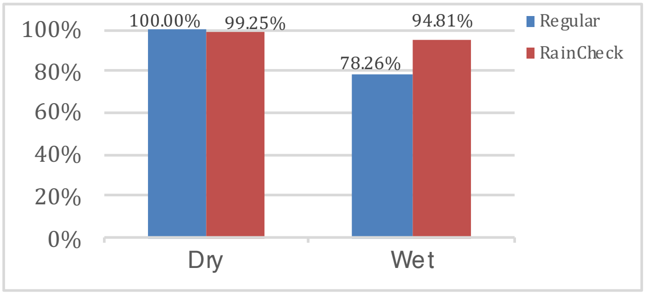

Figure 8: The chart shows the accuracy of touch gestures with the unmodified system and with RainCheck. With a wet touchscreen, RainCheck decreased the error rate from 21.3% to 5.2%.

5.2 Study 2: Target Selection

In our second study, we wanted to evaluate RainCheck's disambiguation technique for improving the accuracy of target selection with water drops on the touchscreen.

5.2.1 Participants & Apparatus. We recruited five participants (all males and right-handed, aged 22 to 27). Our apparatus was much as before in Study 1. Specifically, in each target-selection trial, our custom software recorded the number of touch attempts and the trial completion time, so we could analyze and compare the performance of RainCheck to the unmodified system. We measured the task time from the first finger touch to when the finger was lifted from the correct target.

5.2.2 Procedure & Tasks. Participants were presented with 9 buttons in a 3 × 3 grid layout (9). Each button had a size of 45 × 45 dp, which is the same as the size of a key on a typical Android soft keyboard. The target button was displayed in a different color from the other eight buttons, and appeared randomly at different places within the grid layout. Participants were required to complete 30 tasks with and without RainCheck active, and in two different environments (a wet vs. dry screen), for 120 trials in all.

One of our particular interest was the performance of RainCheck's target disambiguation technique, so participants could use tap-and-swipe when RainCheck detected an unusual touch event in the wet touchscreen condition.

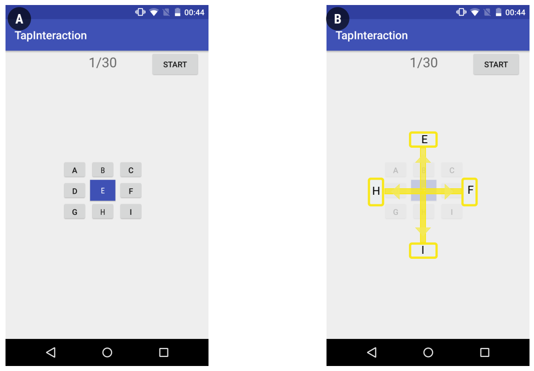

Figure 9: At left is the screenshot of the initial state of the task. At right is when RainCheck detected an unusual touch area and offered four potential selectable UI targets that were covered by the touch.

5.2.3 Results. For the second study, RainCheck performed as well as the unmodified system on the dry touchscreen (RainCheck: M = 1.07 attempts, SD = 0.44; Regular: M = 1.03 attempts, SD = 0.33), and also outperformed the unmodified system on the wet touchscreen (M = 1.11 attempts, SD = 0.68). For the unmodified system without RainCheck, the average number of attempts in the wet condition was 2.13 (SD = 5.00). Thus, in the wet condition, RainCheck was 47.9% more accurate than the unmodified system.

Our results also showed that RainCheck's average selection time demonstrated the benefits of its target disambiguation technique. Average selection times of the unmodified system and RainCheck were 2.543 seconds (SD = 7.314) and 0.553 seconds (SD = 1.681), respectively. RainCheck reduced the selection time by 80.0%, which saved almost two seconds per task (Figure 10). For the second study, although only 5 participants were recruited, we still reported the statistical results. The paired samples t-test was used to analyze the performance on completion time and the number of attempts for the unmodified system with and without RainCheck. No significant difference was found on either the completion time ($t(4) = 3.42$, $p = 0.026$) or the number of attempts ($t(4) = 2.42$, $p = 0.072$). The detectable effect might need more samples, but our small sample demonstrated the promising result.

6 DISCUSSION

It is clear that RainCheck considerably improves gesture and touch accuracy, and target selection time, in the presence of a wet touchscreen. In other words, RainCheck performs as it was meant to. Happily, even for dry touch screens, RainCheck does not seem to "get in the way," performing just as well as an unmodified touchscreen. Going further, an interesting finding was that two of our five participants in the second study tried to avoid contact with water while performing the target selection task. They reported that they knew the water drops near the target buttons would probably

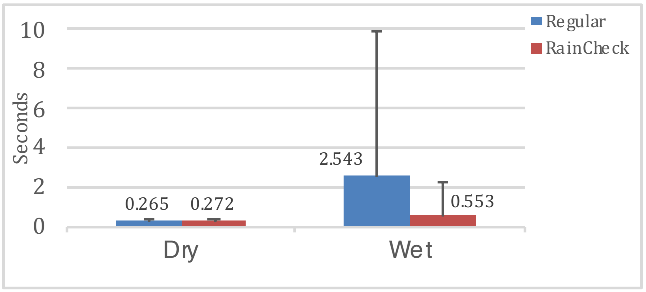

Figure 10: Time taken (in seconds) for successful selection in our second study. On average, RainCheck reduced the selection time in the wet condition from about 2.5 seconds to 0.5 seconds, or by 80.0%. Error bars represent +1 standard deviation.

affect their touch accuracy, so they would keep their fingers away from the water drops if the drops were close to the target. Although we did not evaluate RainCheck for text entry, when asked, over half of our participants agreed that our target selection technique would improve touchscreen typing accuracy in the rain. In short, participants' experiences in our study generated optimism that RainCheck could help them in the rain.

7 LIMITATION & FUTURE WORK

As with all studies, ours had limitations. Our RainCheck prototype also had limitations. One limitation is that we did not evaluate the performance of the proposed approach on different smartphones. The performance might be affected by different hardware implementations of the capacitive sensing. Another limitation is that we did not collect any multi-touch data to see whether multi-touch would affect the performance of RainCheck. More participants in a larger study would add validating support to RainCheck's performance. And implementing RainCheck on other devices beyond the LG Nexus 5 are all ways in which this work could be extended beyond these limitations.

Future studies could include investigating the effectiveness of RainCheck in outdoor rainy conditions, even though these would be rather uncontrollable. We could conduct a series of studies with common mobile applications people use while walking, such as messaging and navigation applications, to evaluate RainCheck's performance in real-world conditions.

Currently, we only used low-level raw sensor data from the capacitive touchscreen to infer finger-touches and improve touch accuracy. We could perhaps discover how to leverage a pressure-sensing touchscreen to distinguish water, which would exert light pressure, from finger touches exerting greater pressure on the screen.

8 CONCLUSION

We have presented RainCheck, a new touch-based interactive system for commodity smartphones that reduces capacitive interference caused by rainwater. RainCheck works by obtaining low-level raw capacitance readings; detecting and filtering out unusual capacitance blobs using heuristics involving blob shape, size, and pixel-value distribution; and triggering a precision target-selection technique when detecting potential water interference with a user's tap interactions. The result is that for wet touchscreens, RainCheck improves gesture accuracy by 75.7%, touch accuracy by 47.9%, and target selection time by 80.0%. RainCheck demonstrates one way in which the situational impairments caused by rainwater can be successfully overcome.

9 ACKNOWLEDGEMENT

We thank all of our participants for their time and feedback. We would also like to thank Vivek Shankar for his assistance and contribution.

REFERENCES

[1] Pär-Anders Albinsson and Shumin Zhai. 2003. High Precision Touch Screen Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 105–112. https://doi.org/10.1145/642611.642631

[2] Michelle Annett, Anoop Gupta, and Walter F. Bischof. 2014. Exploring and Understanding Unintended Touch During Direct Pen Interaction. *ACM Trans. Comput.-Hum. Interact.* 21, 5, Article 28 (Nov. 2014), 39 pages. https://doi.org/10.1145/2674915

[3] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. 2006. Precise Selection Techniques for Multi-touch Screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 1263–1272. https://doi.org/10.1145/1124772.1124963

[4] Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 2014. Swipeboard: A Text Entry Technique for Ultra-small Interfaces That Supports Novice to Expert Transitions. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 615–620. https://doi.org/10.1145/2642918.2647354

[5] Cypress.com. 2018. TrueTouchÂő Touchscreen Controllers | Cypress Semiconductor. Retrieved January 21, 2018 from http://www.cypress.com/products/truetouch-touchscreen-controllers

[6] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. WalkType: Using Accelerometer Data to Accomodate Situational Impairments in Mobile Touch Screen Text Entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2687–2696. https://doi.org/10.1145/2207676.2208662

[7] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. GripSense: Using Built-in Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 545–554. https://doi.org/10.1145/2380116.2380184

[8] Aamir Haider and Nowell Solish. 2005. Focal hyperhidrosis: diagnosis and management. *CMAJ* 172, 1 (2005), 69–75. https://doi.org/10.1503/cmaj.1040708 arXiv:http://www.cmaj.ca/content/172/1/69.full.pdf

[9] Ken Hinckley, Michel Pahud, Hrvoje Benko, Pourang Irani, François Guimbretière, Marcel Gavriliu, Xiang 'Anthony' Chen, Fabrice Matulic, William Buxton, and Andrew Wilson. 2014. Sensing Techniques for Tablet+Stylus Interaction. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 605–614. https://doi.org/10.1145/2642918.2647379

[10] Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. 2010. Pen + Touch = New Tools. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 27–36. https://doi.org/10.1145/1866029.1866036

[11] Christian Holz, Senaka Buthpitiya, and Marius Knaust. 2015. Bodyprint: Biometric User Identification on Mobile Devices Using the Capacitive Touchscreen to Scan Body Parts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3011–3014. https://doi.org/10.1145/2702123.2702518

[12] Shaun K. Kane, Jacob O. Wobbrock, and Ian E. Smith. 2008. Getting off the Treadmill: Evaluating Walking User Interfaces for Mobile Devices in Public Spaces. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '08)*. ACM, New York, NY, USA, 109–118. https://doi.org/10.1145/1409240.1409253

[13] Gordon Kurtenbach and William Buxton. 1994. User Learning and Performance with Marking Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. ACM, New York, NY, USA, 258–264. https://doi.org/10.1145/191666.191759

[14] Min Lin, Rich Goldman, Kathleen J. Price, Andrew Sears, and Julie Jacko. 2007. How do people tap when walking? An empirical investigation of nomadic data entry. *International Journal of Human-Computer Studies* 65, 9 (2007), 759 – 769. https://doi.org/10.1016/j.ijhcs.2007.04.001

[15] Alexander Mariakakis, Mayank Goel, Md Tanvir Islam Aumi, Shwetak N. Patel, and Jacob O. Wobbrock. 2015. SwitchBack: Using Focus and Saccade Tracking to Guide Users' Attention for Mobile Task Resumption. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2953–2962. https://doi.org/10.1145/2702123.2702539

[16] Microchip.com. 2018. 2D maXTouch. Retrieved January 21, 2018 from https://www.microchip.com/design-centers/capacitive-touch-sensing/2d-touch/maxtouch

[17] Sachi Mizobuchi, Mark Chignell, and David Newton. 2005. Mobile Text Entry: Relationship Between Walking Speed and Text Input Task Difficulty. In *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices &Amp; Services (MobileHCI '05)*. ACM, New York, NY, USA, 122–128. https://doi.org/10.1145/1085777.1085798

[18] Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-small Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2799–2802. https://doi.org/10.1145/2470654.2481387

[19] Antti Oulasvirta, Sakari Tamminen, Virpi Roto, and Jaana Kuorelahti. 2005. Interaction in 4-second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 919–928. https://doi.org/10.1145/1054972.1055101

[20] Jason Pascoe, Nick Ryan, and David Morse. 2000. Using While Moving: HCI Issues in Fieldwork Environments. *ACM Trans. Comput.-Hum. Interact.* 7, 3 (Sept. 2000), 417–437. https://doi.org/10.1145/355324.355329

[21] Zhanna Sarsenbayeva, Jorge Goncalves, Juan García, Simon Klakegg, Sirkka Rissanen, Hannu Rintamäki, Jari Hannu, and Vassilis Kostakos. 2016. Situational Impairments to Mobile Interaction in Cold Environments. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, New York, NY, USA, 85–96. https://doi.org/10.1145/2971648.2971734

[22] Martin Schedlbauer and Jesse Heines. [n. d.]. Schedlbauer, M. and Heines, J. Selecting While Walking Keywords Selecting While Walking: An Investigation of Aiming Performance in a Mobile Work Context.

[23] Andrew Sears and Ben Shneiderman. 1991. High precision touchscreens: design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies* 34, 4 (1991), 593 – 613. https://doi.org/10.1016/0020-7373(91)90037-8

[24] Andrew Sears and Mark Young. 2003. The Human-computer Interaction Handbook. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, Chapter Physical Disabilities and Computing Technologies: An Analysis of Impairments, 482–503. http://dl.acm.org/citation.cfm?id=772072.772105

[25] Tomoki Shibata, Daniel Afergan, Danielle Kong, Beste F. Yuksel, I. Scott MacKenzie, and Robert J.K. Jacob. 2016. DriftBoard: A Panning-Based Text Entry Technique for Ultra-Small Touchscreens. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 575–582. https://doi.org/10.1145/2984511.2984591

[26] STMicroelectronics. 2018. Touchscreen Controllers. Retrieved January 21, 2018 from http://www.st.com/en/mems-and-sensors/touchscreen-controllers.html?querycriteria=productId=SC1717

[27] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 657–666. https://doi.org/10.1145/1240624.1240727

[28] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid Touch: A See-through Mobile Device. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 269–278. https://doi.org/10.1145/1294211.1294259

[29] Jacob O. Wobbrock. 2006. The Future of Mobile Device Research in HCI. Retrieved May 2, 2018 from http://faculty.washington.edu/wobbrock/pubs/chi-06.05.pdf

[30] T. Yamabe and K. Takahashi. 2007. Experiments in Mobile User Interface Adaptation for Walking Users. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*. 280–284. https://doi.org/10.1109/IPC.2007.94

[31] Koji Yatani, Kurt Partridge, Marshall Bern, and Mark W. Newman. 2008. Escape: A Target Selection Technique Using Visually-cued Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 285–294. https://doi.org/10.1145/1357054.1357104

[32] Koji Yatani and Khai N. Truong. 2009. An evaluation of stylus-based text entry methods on handheld devices studied in different user mobility states. *Pervasive and Mobile Computing* 5, 5 (2009), 496 – 508. https://doi.org/10.1016/j.pmcj.2009.04.002

[33] Yeliz Yesilada, Simon Harper, Tianyi Chen, and Shari Trewin. 2010. Small-device users situationally impaired by input. *Computers in Human Behavior* 26, 3 (2010), 427 – 435. https://doi.org/10.1016/j.chb.2009.12.001

A RAINCHECK ALGORITHM

```
Data: Raw Capacitive sensor values & Touch points reported
      by the touch controller
Result: Determine the touch point made by rainwater or
        fingers
/* Exact values of these initialized variables
   were determined by touch data from the pilot
   study, which can be varied with different
   hardware.                                        */
$R_f$ <- The range of the aspect ratio of the finger touch data;
$P_f$ <- The average peak capacitance sensor value of the finger
  touch data;
$STD_f$ <- The standard deviation of the sensor value of the
  finger touch data;
RainCheck(touch_point, sensor_vals){
    /* get statistics data from sensors around the
       touch point.                                 */
    touch_area <- connected_component_algo(sensor_vals,
      touch_point->pos);
    peak_val <- getPeakVal(touch_area);
    val_std <- getStdOfSensorVal(touch_area);
    aspect_ratio <- getAspectRatio(touch_area);
    /* Check the stats of the capacitance value.  */
    if peak_val is less than ($P_f$ - $STD_f$) then
        /* The touch point is not a finger touch. */
    end
    /* Check the aspect ratio of the touch area.  */
    if aspect_ratio is not in $R_f$ then
        if peak_val is larger than ($P_f$ - $STD_f$) then
            /* The finger contacts with the water, so
               the peak value is in the range of
               human touch, but the aspect ratio is
               affected by the rainwater.          */
            enableSwipeInteraction(touch_point);
        else
            /* The touch point is not a finger touch.
               */
        end
        /* The touch point is a finger touch.     */
    end
```