Beyond the Touchscreen: An Exploration of Extending Interactions on Commodity Smartphones

CHENG ZHANG, Georgia Institute of Technology
ANHONG GUO, Carnegie Mellon University
DINGTIAN ZHANG, Georgia Institute of Technology
YANG LI, Google
CALEB SOUTHERN, ROSA I. ARRIAGA, and GREGORY D. ABOWD,
Georgia Institute of Technology

Most smartphones today have a rich set of sensors that could be used to infer input (e.g., accelerometer, gyroscope, microphone); however, the primary mode of interaction is still limited to the front-facing touch-screen and several physical buttons on the case. To investigate the potential opportunities for interactions supported by built-in sensors, we present the implementation and evaluation of BeyondTouch, a family of interactions to extend and enrich the input experience of a smartphone. Using only existing sensing capabilities on a commodity smartphone, we offer the user a wide variety of additional inputs on the case and the surface adjacent to the smartphone. Although most of these interactions are implemented with machine learning methods, compact and robust rule-based detection methods can also be applied for recognizing some interactions by analyzing physical characteristics of tapping events on the phone. This article is an extended version of Zhang et al. [2015], which solely covered gestures implemented by machine learning methods. We extended our previous work by adding gestures implemented with rule-based methods, which works well with different users across devices without collecting any training data. We outline the implementation of both machine learning and rule-based methods for these interaction techniques and demonstrate empirical evidence of their effectiveness and usability. We also discuss the practicality of BeyondTouch for a variety of application scenarios and compare the two different implementation methods.

CCS Concepts: • **Human-centered computing → Gestural input**; **Mobile devices;**

Additional Key Words and Phrases: Mobile interactions, smartphones, inertial sensors, microphone, rule-based method, machine learning

ACM Reference Format:
Cheng Zhang, Anhong Guo, Dingtian Zhang, Yang Li, Caleb Southern, Rosa I. Arriaga, and Gregory D. Abowd. 2016. Beyond the touchscreen: An exploration of extending interactions on commodity smartphones. ACM Trans. Interact. Intell. Syst. 6, 2, Article 16 (August 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2954003

The review of this article was managed by Giuseppe Carenini and Shimei Pan, associate editors of the special issue on Highlights of IUI 2015.
This material is based in part on work supported by the National Science Foundation Graduate Research Fellowship under grant DGE-1148903.
Authors' addresses: C. Zhang, D. Zhang, C. Southern, R. I. Arriaga, and G. D. Abowd, School of Interactive Computing, Georgia Institute of Technology; emails: {chengzhang, dingtianzhang, caleb.southern}@gatech.edu, arriaga@cc.gatech.edu, abowd@gatech.edu; A. Guo, Human-Computer Interaction Institute, Carnegie Mellon University; email: anhongg@cs.cmu.edu; Y. Li, Google; email: yangli@acm.org.

1. INTRODUCTION

Currently, the primary mode of interaction with a smartphone is limited to the front-facing touchscreen and a small number of physical buttons along the sides. However, there are various scenarios when it is not convenient to touch the screen or the buttons, such as when the phone is inside a pocket, when it is on a table and the user's hands are dirty or wet, when touching the screen occludes the display, or when the user is holding the phone in one hand while the other hand is occupied with another task. To address these limitations, we implemented and evaluated a family of interaction techniques, called *BeyondTouch*, that allow a user to perform additional inputs in-pocket, one-handed, two-handed and on-table, extending the input language of a smartphone in several interesting ways (Figure 1).

1.1. Contributions and Overview

Although it is possible to accommodate many of these interaction scenarios by adding capabilities to the smartphone through external sensors (e.g., a capacitive touchscreen on the back of the phone [Hiraoka et al. 2003; Saponas et al. 2011]), it is also possible to augment the input language of a commodity smartphone through proper leverage of its existing sensing platform. Current smartphones are equipped with an increasing number of sensors, specifically accelerometers, gyroscopes, and microphones, all of which can be exploited to support a wider variety of software-detected input events without the need for additional hardware.

Previous work has suggested the possibility of sensing a small set of input events. Researchers have already demonstrated the importance of in-pocket interaction with customized hardware [Hudson et al. 2010; Saponas et al. 2011; Zhang et al. 2013], the possibility of detecting tap events using inertial sensors [Saponas et al. 2011; Zhang 2013], and using advanced machine learning techniques to infer user input [McGrath and Li 2014; Xu et al. 2012; Zhang et al. 2015]. BeyondTouch is distinguished by the breadth of input events enabled, the discussions on practical issues when applying machine learning on mobile interactions (e.g., personalization on models), the comparison between rule-based and machine learning methods, and other challenges deploying them in real applications.

This article is an extended version of Zhang et al. [2015], in which we discussed the opportunities and challenges of applying the machine learning technique on recognizing gestures on a smartphone with only built-in sensors. Beyond our previous work, this article also describes the design, evaluation, and gestures implemented with impact and robust rule-based methods.

BeyondTouch (see Figure 1) extends the input language of a smartphone by offering the following:

- On-case interactions (one-handed, two-handed) to avoid occluding the limited screen area with the fingers
- Indirect interactions (on-table, in-pocket) by directing interactions to where the phone is placed (e.g., a table, a pocket).

We provide a review of related work next. We then describe the technical approaches behind the implementation of BeyondTouch, the machine learning and rule-based approaches to recognize the various input events. We provide an empirical evaluation of the performance of BeyondTouch and discuss the implications of these results for practical use. Our results suggest that rule-based methods can provide improved accuracies while detecting corner taps over machine learning methods. We also show that with a little bit of personalized training data, the performance for some machine learning interactions can be improved.

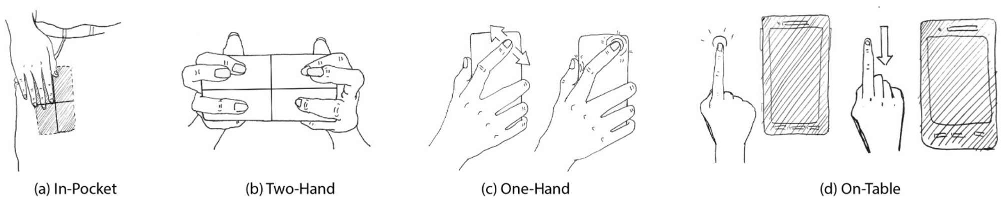

Fig. 1. BeyondTouch interaction techniques.

2. RELATED WORK

Now we review past work on using sensors to enable novel interaction techniques and position our contribution of providing a broad range of interactions for a smartphone without having to add any new sensing capabilities to the device.

Some of the earliest demonstrations of broadening interactions with mobile devices were presented by Hinckley et al. [2000], who instrumented a PDA with a variety of extra sensing capabilities to demonstrate a wide variety of interactions that could augment the user experience. This work inspired much of the related research that we have reviewed. Although most of the related work has taken a similar approach of adding sensors to the device to demonstrate interactions that can be supported, contemporary smartphone platforms include increasingly sophisticated sensors, and there is an opportunity to produce many discrete interactions without adding sensors.

2.1. On-Case Interactions

One of the main problems with touchscreens is that users' fingers occlude the screen. Several researchers have experimented with shifting the interaction away from the touchscreen to the side and back of devices to eliminate occlusion or increase efficiency and accessibility of the device. This can be accomplished by adding additional hardware, such as a keypad [Hiraoka et al. 2003; Li et al. 2008] or a touchpad [Baudisch and Chu 2009; Schwesig et al. 2004; Scott et al. 2010; Sugimoto and Hiroki 2006; Wobbrock et al. 2003], or through computer vision [Schmieder et al. 2013; Wigdor et al. 2007].

Other research has explored potential gestures supported by built-in sensing capabilities on the phones. Hinckley and Song explored techniques using only inertial sensors and touch input for handheld devices that leverage the multimodal combination of touch and motion, providing touch-enhanced motion and motion-enhanced touch gestures as well as the detection of tap events [Hinckley and Song 2011]. Tap events have been further investigated by recognizing side taps [McGrath and Li 2014], detecting taps on the back case of a phone using the microphone as a sensor relying on off-device computation to process the input [Robinson et al. 2011], or distinguishing between a single gentle or firm tap on the case [Heo and Lee 2011]. Besides using tap gestures while holding the phone, there are often situations in which a simple tap directed to some location on the device is an easy and often preferred eyes-free way to interact [Ronkainen et al. 2007], such as interacting with the phone while it is inside a pocket. Taking the phone out for interactions demands a high level of attention, both cognitively and visually, and is often socially disruptive. Whack Gestures explored quick access to a device in a pocket through striking its case forcefully with the palm or heel of the hand [Hudson et al. 2010]. Goel et al. demonstrated the possibility of inferring hand postures while holding the phone using only built-in sensors but requiring a somewhat undesirable constant vibration source [Goel et al. 2012].

2.2. Off-the-Phone Interaction

In some contexts, it is desirable to interact with the phone without directly touching it. The space around the phone can be used to interact with the phone. Airlink exploits the

Doppler effect to detect the gestures above the phone [Chen et al. 2014]. Others have explored off-the-phone interactions while a phone is placed on a flat surface (e.g., a table), requiring customized pickups attached on the surface to capture the acoustic signals on the surface, such as a modified stethoscope [Harrison and Hudson 2008] or contact piezoelectric sensors [Paradiso et al. 2002]. Goel et al. used both the customized sensors on the surface as well as the phone sensors (e.g., inertial sensors) to support a wider set of around-phone gestures [Goel et al. 2014]. What is missing from the literature is an exploration of the kind of off-the-phone gestures that can be detected without using any additional hardware other than the built-in sensors. Although previous work has explored recognizing gestures by differentiating sounds generated by gestures [Harrison and Hudson 2008], the on-table interaction in BeyondTouch can detect both sliding and tapping gestures with only built-in sensors. Specifically, we introduce the use of dual microphones available in most modern smartphones for recognizing gestures.

2.3. Snooping Users' Input on Smartphones

Interestingly, several researchers outside the HCI community have explored nefarious uses of sensing mobile device interaction, uncovering information security risks afforded by easy access to smartphone sensors such as the accelerometer to detect users' input on the touchscreen [Cai and Chen 2011; Miluzzo et al. 2012; Owusu et al. 2012; Xu et al. 2012]. However, most of these techniques rely on offline inference. Thus, how they will work for real-time interaction tasks is unclear.

3. MOTIVATION AND APPLICATIONS

BeyondTouch was motivated by many common smartphone usage scenarios, where the touchscreen or on-case buttons limit convenient interactions. To address these limitations, we implemented four classes of interaction scenarios: in-pocket, one-handed, two-handed, and on-table.

3.1. In-Pocket

Many people place their phones in a pocket, bag, or armband, thus requiring extra effort to retrieve the phone to perform otherwise simple interactions. We implemented in-pocket interaction, which allows the user to tap on any of the four corners of the phone's case while the device is inside a pocket or on an armband (Figure 1(a)). The four distinct corners are easily reached without removing the phone or looking at the screen, and tapping is convenient to perform while the user is in motion. This in-pocket interaction is inherently eyes-free, minimizing the visual distraction to the user and increasing accessibility for users with visual impairments.

With the in-pocket interaction capability, we simplify responding to an incoming phone call either by dismissing it or sending one of several prepared text responses (e.g., "On my way," "Busy, call later") when the phone is in a pocket. Another example is interacting with a music app during exercise, with the phone attached to an armband. In a music player application, the back corners of the case could be mapped to volume up, volume down, track forward, and previous track.

3.2. One-Handed

When the user is interacting with the phone with a single hand, it is not easy to move the thumb across the entire touchscreen area, especially as screen sizes increase. In addition, the other hand may be occupied doing something else. BeyondTouch allows the user to tap on the side of the phone with the thumb, and to tap or slide on the back of the phone with the index finger (Figure 1(c)).

The one-handed tapping interactions can be used for applications such as browsing through images and Web pages. They can also be used for camera functions, such as taking selfies, zooming in and out, and so forth. Other applications include accessing functions with coarse input gestures. For example, the user can simply turn the flashlight on and off by tapping on the back of the phone. The one-handed sliding interactions are naturally matched with applications that require scrolling operations, such as proceeding to the next/previous voicemail or a song in a playlist.

3.3. Two-Handed

Two-handed interactions address usage scenarios where the touchscreen is available but the users' hands may occlude the small screen area of the phone while holding it in two hands. We implemented two-handed interactions to address this problem, employing a similar user experience to a traditional game controller on a laptop or a game console. The user can hold the phone in landscape orientation with two hands and tap on virtual buttons on the back of the case (Figure 1(b)).

In the current implementation, we offer up to six soft buttons on the back and side of the case. Applications include games and navigation, with the advantage of exposing the entire screen area because the fingers are touching the back of the case rather than covering the touchscreen.

3.4. On-Table

On-table interaction explores possible interactions around the phone by using only built-in sensors on commodity smartphones. When the user is cooking and a phone call comes in, his phone may be on a table and his hands may be dirty or wet. How would he answer the phone call without polluting the screen by touching it? BeyondTouch allows the user to tap and slide on the table around the phone to interact with the device indirectly (Figure 1(d)). Furthermore, on-table interactions can be extended to many applications that demand simple and straightforward interactions, such as responding to a phone call or controlling the music player.

4. IMPLEMENTATION OF BEYONDTOUCH

Table I describes the complete set of input events that BeyondTouch supports and related sensors applied. We implemented the BeyondTouch interactions on commodity smartphones by interpreting data from a combination of three common built-in sensors: the accelerometer, gyroscope, and microphones. We applied machine learning or rule-based techniques to identify the characteristic signature of each input event. The machine learning methods were implemented using Weka [n.d.], a publicly available tool that provides implementations of various machine learning algorithms that can be run in real time on a smartphone.

4.1. Choice of Detection Technique

In the early exploration of this project, we applied a rule-based method to recognize the four corner taps for on-case interactions. Based on our observation, the rotation of the phone case presents distinct quantitative signatures when tapping on each of the four corners of the phone case. We implemented a straightforward rule-based solution to detect the four corners, which worked well. The advantages of the rule-based approach are (1) lightweight software implementation (given the limited processing power of a smartphone), (2) ease of implementation, and (3) no requirement for training and personalization. However, machine learning, by contrast, is appropriate for interactions that are more complex, where the difference between input events are not obvious from observation of the sensor input streams. We will describe both implementation methods in the following sections.

Table I. Interaction Techniques

Scenario	Interaction	Input Events	Sensors	Techniques
One-handed	Tap	Back-single-tap, back-double-tap, side-tap	Gyroscope, accelerometer, the microphone at bottom	Machine learning
	Tap-slide	Back-single-tap, back-double-tap, back-slide-up, back-slide-down	Gyroscope, accelerometer, the microphone at bottom	
In-pocket	Tap	Tap on the four corners while the phone is inside the pocket	Gyroscope, accelerometer, two microphones	Rule based
Two-handed	4-point- tap	Top-left, top-right, bottom-left, bottom-right	Gyroscope, accelerometer, the microphone at bottom	Rule based
	6-point- tap	Up-left, up-right, top-left, top-right, bottom-left, bottom-right	Gyroscope, accelerometer, the microphone at bottom	Machine learning
On-table	Slide, tap	Slide, tap on the surface around the phone	Gyroscope, accelerometer, two microphones	Machine learning

4.2. Choice of Built-In Sensors

In our initial investigation, we examined data from the accelerometer to determine if a tap occurred anywhere on the case of a phone, inspired by work such as Whack Gestures [Hudson et al. 2010]. To detect where the user tapped on the case, we then added the gyroscope, which better reflects the difference in the tapping events on four corners.

In pilot testing, we observed that when the user held the phone with a tight grip, these two motion sensors often failed to detect the tap event, even with a strong tap on the back case. To address this problem, we added the microphones as a third input stream. We only consider loudness from the microphone buffer, measured in decibels (dB). To calculate the decibel level, we first normalize each audio sample and calculate the average root mean square (P_{rms}) of the samples in the buffer. We then calculate the decibel value (Equation (1)), where P_{ref} is equal to the standard reference sound pressure level of 20 micropascals.

$$L_P = 20 \times log_{10}\left(\frac{P_{rms}}{P_{ref}}\right) dB \tag{1}$$

In addition, a sensor fusion of microphone and inertial sensors are complementary to filter out most of the noise, which improves the robustness of the techniques.

4.3. Rule-Based Implementation

We implemented in-pocket and two-handed (4-point-tap) with rule-based methods, which only had tapping events on the four corners of a phone case.

4.3.1. Rationale. To detect corner taps on the device, we consider how a phone reacts when a tapping event occurs. We first need to detect the occurrence of a corner tap (tap detection) and then determine which of the four corners was tapped (corner classification). We apply rule-based techniques on a sliding window of the input streams from three sensors (accelerometer, gyroscope, and microphone) to detect and then classify input events.

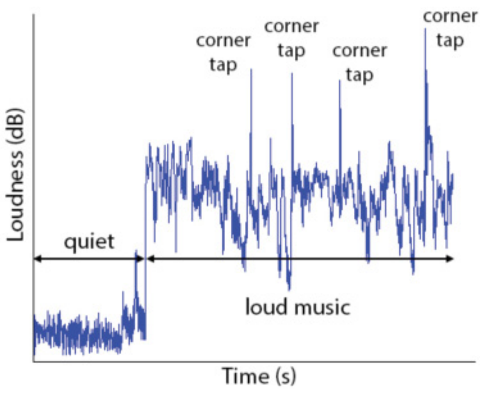

Fig. 2. Microphone data when the back of the smartphone is tapped. In this plot, the ambient environment noise is first relatively low and then increases as music is played. The four spikes on the right are the tapping events, which are easily distinguishable from the ambient noise.

4.3.2. Tap Detection. We examined the device's accelerometer and gyroscope data when the case is tapped. One concern with microphone data is that ambient noise from the environment might cause false triggers. However, the sound generated by the on-case tap event is significantly louder, as measured in decibels, than the audio environment noise. This is because the solid case conducts sound better than air and the tapping event occurs in the proximity to the microphone sensors. Figure 2 plots the decibel value changes when tapping happens with high-volume pop music played from a nearby speaker, placed 0.2m away from the phone. Combining all three sensor streams therefore allows for a robust tap detection algorithm.

In our implementation, we employed slightly different tap detection thresholds for in-pocket and two-handed scenarios based on the level of movement and the loudness observed for each of these scenarios. Basically, those criteria examine the decibel values from the sound, and values and derivatives of the z-axis of accelerometer as well as the x-axis and y-axis of the gyroscope, since the peak of a tap was obvious in those axes.

For in-pocket interaction, we set two criteria for the detection of the tap event. The first criterion is that both the maximum decibel value and the z-axis value of the accelerometer must pass a certain threshold. Once the first criterion is met, we examine whether the second criterion is satisfied in one of two ways. The first way is if the x-axis value of the gyroscope and its derivative both satisfy certain thresholds. The second way is if the y-axis value of the gyroscope and its derivative both satisfy certain thresholds. Because the tap event happens in a very short time window, the use of the derivative of the sensor value filters out the undesired lower frequency movement resulting from a slower shake of the phone as opposed to a legitimate tap. We approximate the derivative of sensor data as the change in value divided by the change in time.

Similarly, we also set two criteria for the two-handed interaction. The first criterion is thresholds for the maximum decibel value and the difference between the maximum and minimum decibel values in the buffer. The second criterion can be satisfied in one of two ways: (1) thresholds for the y-axis of the gyroscope and its derivative or (2) thresholds for the z-axis of the accelerometer and its derivative.

The code used on each device, along with specific thresholds, is available at http://www.czhang.org/code_beyondtouch/.

4.3.3. Corner Classification. Once the tap events have been segmented out, classifying them into the four corners is straightforward based on the physical characteristics of tapping at these locations (Figure 3). Each tap event contains two peaks for the

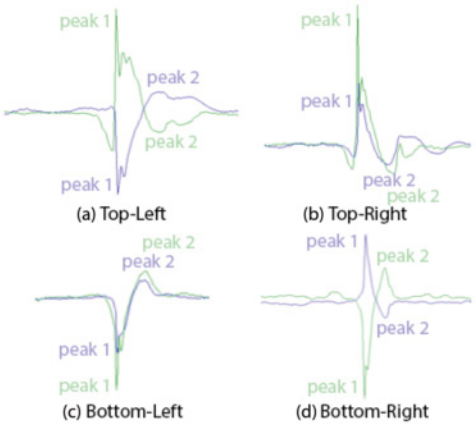

Fig. 3. The clearly distinguished gyroscope signatures of corner taps. Gyro-y is green and gyro-x is purple.

gyroscope rotation along the x-axis and y-axis, with the second peak being opposite of the first peak and lower in amplitude. Therefore, there are two alternative solutions to detect the tap corners. The first is to compare the signs of the peaks on the x-axis and y-axis of the gyroscope (e.g., for top-left from a user's perspective, the peak on the x-axis is negative while the y-axis is positive). The second solution is to compare the amplitude of the first peak and the second peak on the x-axis and y-axis (e.g., for taps on the top two corners, the amplitude of the positive/first peak is always larger than the negative/second peak). We chose the second solution. However, the only difficulty comes from a change in orientation of the phone itself. As the phone is rotated, the mapping of top/bottom and left/right changes. Two kinds of orientation need to be detected. One is whether the phone is facing the human body, which we do not detect. This is because it is hard to recognize this orientation with the IMU and microphone. We made sure that the phone was facing the body in our study. However, we believe that using a light sensor or camera could possibly solve this problem. The other kind of orientation is whether the phone is horizontally or vertically placed in the pocket, which results in four possibilities. We examine the accelerometer data to classify this orientation and then remap the detected corners according to the orientation of the phone from the user's perspective such that a user does not need to worry about the orientation of the phone while putting it into a pocket as long as the phone faces the body.

4.3.4. Calibration on Different Phones. One advantage of using rule-based techniques over a statistical machine learning approach is the ease of calibration for different phones without collecting training data for each phone. We successfully tested our interactions on a Samsung Galaxy S3, a Samsung Galaxy S4, and an HTC One with minimal calibration effort. To calibrate our corner classification method for each of these devices on two-handed interaction, the only parameter we need to adjust is the aspect ratio, which is the length divided by the width of the phone. We use this parameter to normalize the sensor data from the x-axis and y-axis of the gyroscope. To calibrate for in-pocket interaction, we also adjusted the threshold for the z-axis of the accelerometer level for tap detection, allowing all phones to have the same sensitivity to tap strength despite differences in weights and dimensions. Although the adjustment is intuitive to humans, it would be expensive for a statistical machine learning method to learn, especially because it requires sufficient training data from each device to capture device variation. We acquired similar performance results for all three phones in the user study that will be presented later. If a developer wants to apply this algorithm on a new phone model, we would suggest to first change the dimension ratio of the phone case and then adjust the other parameters. This can be done by observing the values of those parameters based on a few gesture samples.

4.4. Machine Learning Implementation

4.4.1. On-Case Implementation. We implemented one-handed (tap, tap-slide) and two-handed (6-point-tap) interactions using a combination of rule-based (segmentation) and machine learning approaches (classification).

We use a traditional sliding window method with a window size of 0.64 seconds. Since a tap or slide event usually last 0.5 seconds, we round it up to 0.64 seconds for the convenience of computation in the Fourier transform. Since the sample rate of gyroscope and accelerometer in the phones we used are 100Hz, each frame contains 64 samples with 50% overlap with adjacent frames. For each frame, the system will extract features to determine whether an event happens (segmentation) and which event it is (classification).

To detect the occurrence of an input event, we use a rule-based approach to detect whether an input event (either tap or slide) has occurred by examining the combination of thresholds for sensors, including the difference between the maximum and minimum value from the x-axis/z-axis of the accelerometer and the y-axis/z-axis of thegyroscope. We chose these criteria based on our observation on collected sensor data, but a more advanced machine learning event detection method can be applied to further reduce false errors in the future.

For classification, we extract three sets of features for each 0.64 second length buffer from data of three axes of gyroscope and accelerometer. The features that we selected were based on our observation and understanding of the data and gestures, which are also common statistical features used in related works. We plotted out the data in both time and frequency domains, decided which features to use, estimated the time each gesture took, and chose the appropriate window size. The first set is the time-domain features. For each axis of the data from the gyroscope and accelerometer, we calculate the seven features over each frame, including the root mean square, derivative of the root mean square, maximum and minimum value, mean, variance, and median. We also calculate the ratio between maximum and minimum for the three axes of the gyroscope and the total energy for both the gyroscope and accelerometer, which indicates the magnitude of the whole event. In addition, the maximum decibel value and the derivative of neighboring decibel values are also computed as features. Therefore, we have $7 \times 6 + 3 + 2 + 2 = 49$ features related to amplitude.

The second set of features includes frequency and energy, which are calculated for each axis of both the accelerometer and gyroscope. We compute the fast Fourier transform (FFT) using JTransforms [Wendykier n.d.] to calculate the frequency energy on each frequency band. We build 31 bins over a frequency of 100Hz for each axis as well as the standard deviation of frequency energy over 31 bins, since we observe that our input events are equally distributed in terms of energy over frequencies. In total, we have $31 \times 6 + 6 = 192$ features.

As a result, we compute 241 features for each frame to detect the one-handed and two-handed interaction gestures. We pass these features to a support vector machine (SVM), provided by the Weka machine learning library [Weka n.d.] and then use this model to classify gestures.

4.4.2. On-Table Implementation. The on-table interaction is also implemented using a combination of rule-based and machine learning approaches.

In the segmentation stage of our implementation, we use a rule-based approach to detect whether an input event (either tap or slide) has occurred by examining the combination of thresholds for the three sensors, including the loudness level of the bottom microphone and the maximum and minimum value from the *z*-axis of the accelerometer or *y*-axis of the gyroscope. This is similar to the methods we use to detect on-case gesture events and can be potentially improved with advanced methods.

To classify the input event, we extract features from the data streams of the microphones, gyroscope, and accelerometer. To capture the full duration of a slide event, we used a sliding window of 1 second. Then we divide the 1-second window into 10 frames. For each frame, we extract the following basic time-domain features for all sensors: (1) level, (2) derivative of level, (3) maximum and minimum level, and (4) energy. A feature vector of each window is the combination of the feature vector of the 10 frames. We put the feature vector of a window into the pretrained model for classification.

We compared the performance of different machine learning algorithms on our collected training data using 10-fold cross validation provided by Weka. Both SVM and k-nearest neighbor (KNN) provided greater than 97.5% for precision and recall. We elected to use KNN in our implementation, as KNN would react better to a new user's input data.

4.4.3. Building a Training Model. A trained machine-learning model usually can be categorized based on whether it is session dependent or user dependent. A session-dependent model requires recollecting training data before using the model each time. A user-dependent model demands collecting training data from each user. Both of them would increase the barrier of applying them in real-world usage. Therefore, we adopted a user-independent model in our implementation that applies a precollected training dataset and does not require any data collection from each user. Based on our pilot study results, we found that the performance of one-handed and two-handed interactions were dependent on how the user holds the phone and performs the gestures. However, the phone's position was not changed when performing gestures for on-table interaction. Therefore, we collected a larger set of data for building the model for one-handed and two-handed interactions.

For on-table interaction, we first conducted a small pilot study with the authors. A classifier was trained with the training samples only from one author. We tested that model with other authors, and the accuracy was very high. We thought that was because the gestures for on-table interaction did not require any direct contact with the phone, which greatly reduced the variance between gestures. As a result, we collected training data from only one of the researchers, who performed approximately 600 gestures for each slide event and the tap event on a table. We varied the tapping position and strength to approximate the behavior of different users.

To build the training set for two-handed (6-point-tap) and one-handed interactions, we collected training data from 10 users, including two authors. Four of them provided examples for both two interactions, whereas others provided examples for only one of the interactions. As a result, each interaction has 10 trainers and approximately 1,000 instances for each input event. During the training process, we asked each user to repeat each input event 100 times on a Galaxy S3, with approximately a 1-second gap between each gesture. The data collection program did not provide any feedback to users. Therefore, the users performed the input gestures in the manner most comfortable and natural to them. For one-handed interaction, all data was collected with the right hand only.

After collecting all of the data, we manually labeled each input event in the training set by visualizing and applying thresholds to the data. If the researcher was not sure about the label of an instance, we discarded it to avoid polluting the training set. Around 85% of the gestures ended up in the training set.

5. EVALUATION METHOD

We evaluated the accuracy and usability of BeyondTouch for the interactions listed in Table I. We presented users with a series of stimuli for the interactions and recorded the responses interpreted by the rule-based methods or pretrained machine learning models. All interactions were performed in a lab environment.

5.1. Participants and Apparatus

5.1.1. Rule-Based Interactions. For rule-based interactions, a study with 11 participants (1 female) was conducted in a lab setting environment to evaluate the system at the early stage of the project. The average accuracy was 92.27% for in-pocket interaction and 93.59% for two-handed interaction. However, the evaluation was conducted on one device (a Samsung Galaxy S3) and without any ambient audio noise, which was different from real-world scenarios. To further evaluate the accuracy of our classification methods as well as user experience, we recruited another 12 participants (4 female). Each participant was assigned to use one of three phones (Samsung Galaxy S3, Samsung Galaxy S4, HTC One).

5.1.2. Machine Learning Interactions. For machine-learning based interactions, we recruited 12 participants (1 female) for two-handed (6-point-tap) and one-handed interaction using a Samsung Galaxy S3 phone and another 11 participants (6 females) for on-table interaction. Seven of them used a Samsung Galaxy S3, and the rest used a Samsung Nexus phone.

5.2. Study Procedure

In the study, each participant went through an introduction phase—a practice phase before starting the real tasks. We first explained the interactions to the participants and allowed them to practice the gestures, receive feedback after performing each gesture, and ask questions. Then each participant performed practice sessions, responding to stimuli for each gesture in the assigned interaction technique.

5.2.1. Rule-Based Interactions. For rule-based interactions, each participant performed two practice sessions, responding to 10 cues requesting them to tap a particular corner. For in-pocket interaction, the participant was given audio cues and audio feedback. For two-handed (4-point-tap) interaction, the cue was displayed on the touchscreen, and the user also received visual feedback of green (correct) or red (incorrect) after responding to the cue. Then each participant performed two evaluation sessions, responding to 20 randomly generated cues for each of the four corners in each interaction technique (in-pocket and two-handed).

To evaluate the robustness of the interactions, we added background noise with three sources simultaneously (75dB for highway noise, 60dB for people talking noise, and 75dB for random music) to the evaluation. We asked the users to report any self-made errors in this phase. For in-pocket interaction, we asked the participants to walk around the table in our lab, on which three speakers were placed, to simulate the situation that the user is in motion. The participants then placed the phone inside a pants pocket in any orientation with the screen facing their body. For two-handed interaction, the participants sat still next to the table.

5.2.2. Machine Learning Interactions. For machine learning interactions, each participant performed practice sessions, responding to stimuli for each gesture in the assigned interaction technique 10 times in a row. Similar to the two-handed (4-point-tap) rule-based interaction, we presented the participant with blue-colored visual stimuli on the smartphone screen, indicating which gesture to make and changing the color green and red as the feedback. Each participant responded to 40 stimuli (2 sessions $\times$ 20 stimuli) for each gesture with random order in one-handed and two-handed interactions. All

Table II. Accuracy of Each Interaction Technique

Interaction	In-Pocket	Two-Handed (4-point-tap)	Two-Handed (6-point-tap)	One-Handed-Tap	One-Handed-Tap-Slide	On-Table
Accuracy	93.75%	92.60%	71.28%	88.47%	72.92%	93.74%
	(SD = 3.35%)	(SD = 5.23%)	(SD = 12.89%)	(SD = 3.55%)	(SD = 6.53%)	(SD = 4.64%)

Table III. One-Handed-Tap Confusion Matrix

	Classification		
User Input	*BackSingleTap*	*BackDoubleTap*	*SideTap*
BackSingleTap	92.92%	4.79%	2.29%
BackDoubleTap	24.17%	73.96%	1.88%
SideTap	0.83%	0.63%	98.54%

participants were asked to use their right hand to perform tasks in one-handed interaction. For the on-table interaction, each participant responded to 90 stimuli (3 sessions $\times$ 30 stimuli), randomly chosen between tap and slide gestures. For each interaction, we also asked the users to report any self-made errors in this phase. The process took less than 1 hour for each participant in the user study, and there was no compensation.

6. RESULTS

We report accuracy for each interaction as the percentage of responses where the gesture classification matched the stimulus and the user impressions of each interaction based on a questionnaire administered at the end of the evaluation phase.

For in-pocket, one- and two-handed (6-point-tap, 4 point-tap) interactions, there were 480 examples ($20 \times 2 \times 12$) for each gesture, and 990 inputs ($30 \times 3 \times 11$) for on-table interaction. Table II presents the overall accuracy and standard deviation for each interaction.

6.1. In-Pocket Interaction

For in-pocket interaction, the overall accuracy is 93.75% (SD = 3.35%), with different phones' accuracy of 90.78% for Galaxy S3, 94.84% for Galaxy S4, and 95.63% for HTC One. The 12 participants self-reported a total of 15 user-generated input errors.

6.2. Two-Handed-4-Point Interaction

For two-handed-4-point interaction, the overall accuracy is 92.60% (SD = 5.23%), with different phones' accuracy of 91.25% for Galaxy S3, 95.78% for Galaxy S4, and 90.78% for HTC One. The participants did not report any user-generated input errors.

6.3. Two-Handed-6-Point Interaction

For two-handed-6-point interaction, the overall accuracy is 71.28%. The confusion matrix is reported later in Table VII. The up-right corner (tapping on the right half of upper edge on the case) was the most accurate (83.33%), and the bottom-right corner was the least accurate (50.83%).

Some users reported that up and top points are much easier to perform than bottom points. Some users also reported that the test sessions were so long (240 taps in two sessions) that it made them feel tired. This also led to 12 user self-made errors.

6.4. One-Handed-Tap Interaction

For one-handed-tap interaction, the overall accuracy is 88.47%. The confusion matrix is reported in Table III. Side-tap was the most accurate (98.54%), and double-tap was the

Table IV. One-Handed-Tap (Back-Tap Combined + Side) Confusion Matrix

	Classification	
User Input	*BackTap(Combined)*	*SideTap*
BackTap(Combined)	97.92%	2.08%
SideTap	1.46%	98.54%

Table V. One-Handed-Tap-Slide Confusion Matrix

	Classification			
User Input	*BackSingleTap*	*BackDoubleTap*	*BackSlideDown*	*BackSlideUp*
BackSingleTap	84.38%	4.38%	2.71%	8.54%
BackDoubleTap	20.00%	73.96%	0.63%	5.42%
BackSlideDown	4.58%	0.21%	41.25%	53.96%
BackSlideUp	2.50%	0%	5.42%	92.08%

Table VI. One-Handed-Tap-Slide (Back-Tap Combined) and Slide (Combined) Confusion Matrix

	Classification	
User Input	*BackTap(Combined)*	*Slide(Combined)*
BackTap(Combined)	91.35%	8.65%
Slide(Combined)	3.65%	96.35%

least accurate (73.96%). There is a lot of confusion between double-tap and back-tap due to the variation of strength and interval of taps between different users.

If we collapse back-tap and double-tap into one gesture, higher accuracy can be achieved (97.92%) (Table IV).

For one-handed-tap-slide interaction, the overall accuracy is 72.92%. The confusion matrix is reported in Table V. Back-slide-up was the most accurate (92.08%), and back-slide-down was the least accurate (41.25%). There is a lot of confusion from back-slide-down to back-slide-up. There is also a lot of confusion from double-tap to back-tap.

Collapsing back-tap and double-tap into one gesture, and back-slide-down and back-slide-up into another gesture, improves accuracy to 91.35% for taps and 96.35% for slides (Table VI).

6.5. On-Table Interaction

For the on-table interaction, 11 participants averaged 93.74% accuracy (SD = 4.64%) over a total of 990 input events. Tap gestures were recognized with 95.68% accuracy, and slide gestures were recognized with 91.11% accuracy.

7. DISCUSSION

Like any recognition technique, BeyondTouch is not perfect. Here we discuss the results to uncover a better understanding of how and when BeyondTouch is suitable.

7.1. Accuracy and User Impressions

Some users reported that tapping continuously for so long (tapping and sliding more than 500 times) made them feel less natural and comfortable. Although the experimental procedure allowed us to gather much data under different situations, we recognize that it put unrealistic demands on users that would not be the case in everyday life.

Similar to real applications, we provided real-time feedback while participants were testing our technique. But we observed that some users adjusted their gestures accord-

Table VII. Two-Handed-6-Point Confusion Matrix

	Classification					
User Input	*UpLeft*	*UpRight*	*TopLeft*	*TopRight*	*BottomLeft*	*BottomRight*
UpLeft	77.92%	14.58%	3.96%	2.92%	0%	0.63%
UpRight	5.63%	83.33%	3.13%	6.46%	0%	1.46%
TopLeft	2.50%	8.75%	70.83%	15.42%	1.04%	1.46%
TopRight	0%	10.21%	8.75%	79.17%	0%	1.88%
BottomLeft	0%	7.29%	21.46%	2.92%	65.63%	2.71%
BottomRight	0.21%	17.29%	9.38%	19.38%	2.92%	50.83%

ing to the feedback given by the system, especially if they saw misclassified instances. We take this as a self-learning process and may potentially have positive influence in the user study result.

Next we discuss other observations for each scenario that impacted results.

7.1.1. In-Pocket Interaction. We observed that when users tried to reach the farthest corner and tap bottom-left, sometimes they touched other corners first. This tended to increase errors for the bottom-left corner. In addition, there are some false negatives happening for the bottom-left corner. We did not rigorously collect ground truth for that but informally observed approximately 1 out of 10 taps not registered. In addition, we noticed that the depth of the pocket matters. For deep pockets, reaching the bottom corners is harder. Moreover, for shallow pockets, walking creates more noise for the top-right corner. In addition, walking made it relatively harder for users to locate the positions of the corners and increased more user errors in the process.

7.1.2. One-Handed Interaction. When users tried to perform back-slide-down, they would first wave their index fingers up. Although there is no contact with the phone, the shake of the phone is very similar to back-slide-up, resulting in many false classifications.

7.1.3. Two-Handed Interaction. For machine learning 6-point-tap interaction, we observed that the variations in the way users hold the phone, the positions they tap, and the strengths of the taps affected individual results quite a bit. For rule-based 4-point-tap interaction, we noticed that for users with very long fingers, it is harder for them to tap on the corners. Since they tap more toward the center, more confusion exists, because the closer the tap position to the center, the less obvious the characteristic of the rotation. In addition, when users adjust hands positions while holding the phone, some false positives are introduced.

7.1.4. On-Table Interaction. Both slide and tap gestures in on-table interaction required the user to touch the surface around the phone. Since we used microphones to recognize the gestures, if users did not generate a loud enough sound, it would increase the difficulty of detecting the occurrence of an event. From our observation, during the practice, most users could learn very fast to use their fingertips and nails together to perform the gestures, which were well recognized by the system. However, user fatigue resulted in false negatives in the latter stages of the experiment.

7.2. Comparison Between Rule-Based and Machine Learning Methods

As the user study results show, rule-based interaction provided greater than 90% accuracy without requiring any training data from the users. As a comparison, machine learning interactions requested precollected training data and a prebuilt training model and provided a relative lower accuracy. However, the results cannot be directly compared, as most of the machine learning interactions were more complicated than rule-based interactions.

To compare the effectiveness of the rule-based method against a machine learning method, we reused the training data collected for two-handed-6-point interaction. We only used the instances for the top-left, top-right, bottom-left, and bottom-right corners, which overlaps between 4-point-tap (rule-based) and 6-point-tap (machine learning) interactions.

First, we built user-independent models with a machine learning implementation and evaluated using leave-one-participant-out cross validation, where the mean accuracy was 86.53% with a standard deviation of 7.38%. We then passed the same dataset through our rule-based classification and compared the classification result with the ground truth. As a result, we received a mean accuracy of 91.27% with a standard deviation of 6.84%. This additional experiment indicates that the simple rule-based method is able to work better across different users without any prerequisite data collection on detecting the four corner tap event.

7.3. Personalization for Machine Learning Interactions

In the user study, the one-handed and two-handed(6-point-tap) machine learning interactions showed an obvious drop in terms of accuracy compared to the result that we computed on the training dataset using 10-fold cross validation and the leave-one-participant-out method. This gap is mostly caused by individual differences, such as a user's finger length, tap position, and strength of tap. Recall that there were 16 total trainers and that we collected training data for each scenario from a different subset of 10 of the overall 16. We further tried to encourage each trainer to perform the input events in whatever way he or she felt most comfortable, thus introducing lots of opportunities for these individual differences to be revealed in the models generated. A personalization procedure should improve the results of the machine learning model for any given user.

Our approach to personalization is a revision on the leave-one-participant-out method. Instead of leaving one trainer's complete training set out, we methodically incorporated a stratified and randomly selected subset of that person's training data into the model that was then used to test the remainder of that trainer's input data. We refer to this as "leave-one-participant-partially-out" to reflect that some of an individual's data is incorporated into the learned model that is used for classification.

Specifically for one-handed and two-handed(6-point-tap) interactions, we collected approximately 100 training instances for each event from 10 different trainers. Instead of using 9 trainers' data to build the model to evaluate the 10th trainer's data, we gradually added a portion of instances from the 10th trainer's data into the training set to build a new model. Then we used this new model to evaluate the rest of the 10th trainer's dataset. To avoid bias, we ran the sample selection process 10 times, then built and evaluated the model for each of them. Next, we averaged results from the 10 sample runs and report those in the following. Note that although the ratio of test data to training data changes based on the amount of training events extracted from the 10th trainer, the difference in those ratios is negligible for purposes of comparison.

To evaluate this method, we compare the accuracy results for different sizes of the trainers' input set (randomly selected subsets of size 10, 20, 30, 40, and 50 of the approximately 100 overall events) against the baseline of using none of that person's training set to build the model (i.e., using zero of the overall events).

In Figure 4, we show the personalization results for the three usage scenarios. For each graph, the red/bolder line represents the averaged result for the leave-one-participant-partially-out method across all trainers in our training set. The *x*-axis indicates how many of the trainer's input events were added to build the personalized machine learning model. For example, the one-handed-tap interaction scenario provides three input events: back-tap, back-double-tap, and side-tap. Each user provided

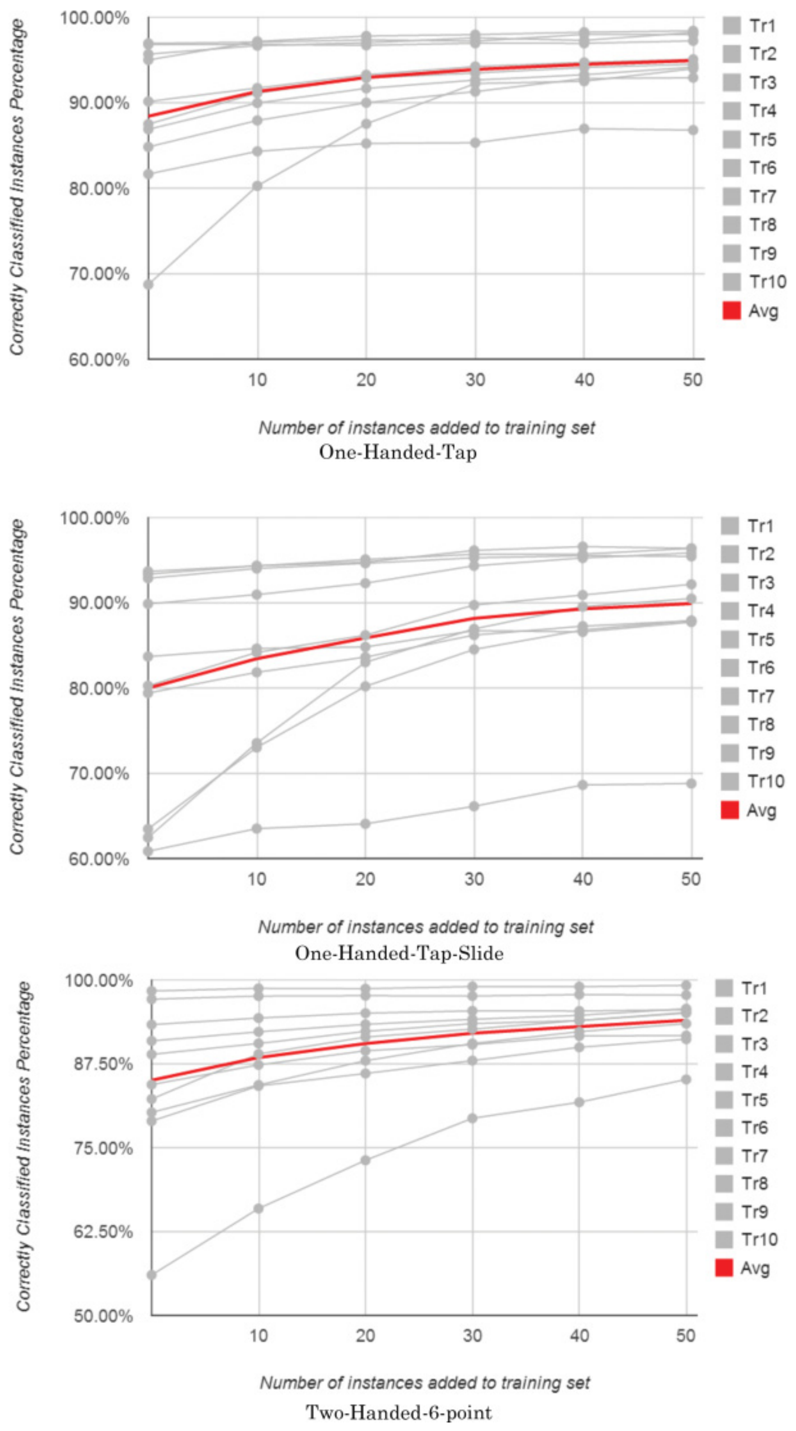

Fig. 4. Personalization results.

around 100 instances for each input event. An x-axis value of 10 represents random selection of 10 instances for each input event from the 10th user's dataset. Since the one-handed-tap-slide scenario has three distinct input events, a total of $10 \times 3 = 30$ instances were incorporated into the personalized training set.

Figure 4 reveals an interesting insight. There is a pretty significant spread of accuracy results across the trainers, with each scenario having at least one trainer's classification accuracy being much poorer than the rest of the trainers with no personalization (x-axis value of zero). That classification gap decreases when personalization is introduced. Without personalization, the average accuracy across all 10 trainers is 80.00% (one-handed-tap-slide), 88.42% (one-handed-tap), and 85.06% (two-handed) with standard deviations of 0.13, 0.08, and 0.12, respectively. Using 30 personal training events (per input type), the average accuracies increase to 88.17%, 93.89%, and 92.06%, whereas the standard deviations decrease to 0.09, 0.04, and 0.06. Figure 4 demonstrates these positive trends for all three machine learning scenarios.

Furthermore, beyond using 30 examples of each event from the trainer, there is no appreciable improvement of the personalization. This is also good, as it means that we need collect only a small set of an individual BeyondTouch user training data to sufficiently personalize that user's model.

A closer examination of the results in Figure 2 shows that what personalization is doing is more about improving the classification for the poorest performers in the leave-one-participant-out results. The better-performing trainers do not improve that much. This suggests that it might be worthwhile in practice determining on the fly how well classification accuracy is for a given user. If that accuracy is below some acceptable threshold, BeyondTouch can recommend additional training from that user to personalize and improve his results.

For example, it may be possible to infer incorrect real-time classifications by BeyondTouch by sensing the user repeating an interaction or developing some other situation-specific heuristic. A running count of these errors can help to determine if a threshold of error tolerance is met. If so, training can be suggested to the user. With this approach, BeyondTouch could be used "out of the box," and for a smaller percentage of users, only after having some experience with BeyondTouch, a user could decide whether it would be worthwhile to improve accuracy by taking some time out to give personalized training data.

7.4. Challenges to Practical Deployment

Several issues beyond recognition rates impact the practical utility of BeyondTouch.

7.4.1. Calibration on Different Phones. Smartphones vary in terms of form factor and sensor capabilities, which calls into question how our results work across devices. In our user study, we have already demonstrated that our rule-based interactions can be calibrated on different phones and provide similar recognition accuracies. For machine learning interactions, we did not test our technique on a large variety of phones. However, since the sensors we used are the most common ones and we applied a machine learning technique in the implementation, calibration across phones is a matter of collecting enough training data prior to installation on a given phone and then allowing for personalization as described earlier. Although this data collection may be time consuming, it does not directly impact the user of the device until and unless personalization is warranted.

7.4.2. Usage of Multiple Microphones. Most smartphones have more than one microphone. Usually, the manufacturer positions one of the microphones at the bottom to receive the user's voice input and the other at the top to detect the ambient sound (to improve the call quality by reducing environmental noise). These two microphones can also be leveraged for interaction purposes, as demonstrated in our on-table interaction. According to basic physical principles, if the distances between a sound source and two audio receivers are different, the loudness level of the audio received by the two receivers should be different. We found that to be true when the physical touch event was close to or on the phone case. We imagine that richer potential interaction events can be detected by combing the data from multiple microphones and inertial sensors.

7.4.3. Interactions by Contexts. In practice, many of the interactions that we developed are designed to work in specific contexts. For example, the one-handed interaction is designed for the scenario when the user prefers to hold and interact with the phone in one hand, whereas two-handed interaction is tuned for a user holding the phone in two hands. If the phone runs two-handed interaction while holding it in one hand, the false positives and false negatives would be increased. Hence, an automatic detection of context about the position and orientation of the phone [Wiese et al. 2013] is a necessary step for applying BeyondTouch in real-world scenarios.

7.4.4. Energy Consumption. BeyondTouch interactions are largely dependent on sensors. However, recently released smartphones (e.g., Moto X, iPhone 5S/6) are beginning to feature a low-power, always-on coprocessor for managing the sensors. With this kind of new hardware, we expect that the energy consumption of the sensor usage will be greatly reduced going forward and would not be a concern while using these sensor-based techniques.

8. CRITIQUE OF APPLICATIONS OF BEYONDTOUCH

In addition to the applications proposed in Section 1, the study participants also suggested interesting usage scenarios for BeyondTouch. In this section, we discuss how we can apply our techniques to these scenarios by exploring the limitations of BeyondTouch and the opportunities to address these constraints in future work.

8.1. In-Pocket

The in-pocket interaction can help with many scenarios that require a simple and quick response with up to four discrete input events.

8.1.1. Applications. The simplest interaction is a single tap anywhere on the phone, but previous research would handle this simple case. By providing up to four events, BeyondTouch can support many more applications, such as reading aloud SMS/email, controlling a music player, recording audio secretly, or forward/backward events. Google Glass can also utilize the in-pocket interaction as an alternative interaction method. It is natural to map a pair of the corners to navigate the tab in glass menu and use another pair to switch menu levels.

8.1.2. Limitations. In the evaluation, we observed minimal false positives and found that false negatives and classification errors were influenced by the size of a user's pocket. A tight pocket may increase the chance of false negative. A phone inside a loose pocket may easily change the phone's orientation and position, which increases the effort for a user to find the phone. False-positive events were rare when walking at normal speed (as in our evaluation), which should be enough for most applications. However, if the user starts running, there is a higher chance of triggering a false-positive event.

8.1.3. Future Work. In our current implementation, we do not provide any feedback other than audio, which may not be appropriate in some scenarios. For example, when tapping to respond to a phone call, it is necessary to notify the user of the actual response. Several participants suggested vibration as a good candidate for feedback.

Although we rarely observed false positives in the evaluation of the in-pocket interaction, a false positive can cause serious problems under certain conditions (e.g., an emergency call). One feasible way to address this issue in future work is to provide a user-defined unlock gesture with a sequence of tap events. It is unlikely to trigger a continuous false positive with the same sequence. Another observation is that the antidiagonal positions (e.g., top-left and bottom-right) are rarely confused (see Table III). To avoid the misclassification of events, a delay between the feedback and the actual action can be set such that users can have time to undo or correct the classification errors.

8.2. One-Handed

One-handed interactions can support up to five input events. In our evaluation, some gestures were detected with high accuracy, whereas others were confused with other gestures. However, not all applications require all five events. As reported in the results, we can achieve higher accuracy by combining the single and double back taps into a single gesture and by combining the up and down swipes into a single gesture. However, how a new combination of input events influence the usability needs to be further evaluated.

8.2.1. Applications. While a user is holding the phone in one hand on a phone call, a common task is to mute the microphone. Instead of moving the phone from the ear and finding the mute button on the screen, the user can directly use the index finger/thumb to tap on the phone to activate the mute function. Vibration feedback can be provided for this eyes-free interaction. Participants suggested many other applications that only need a single input event from one-handed gestures, such as taking selfies with the front camera, playing Flappy Bird (requires high speed), and a quick check of the time.

The most common scenarios for one-handed interaction with phones are reading Web sites/books/emails or playing videos. These usually require two or three input gestures, such as forward/backward. From the evaluation, we can see that there is little confusion between tap and slide events or tap on back and tap on side events. Therefore, we can map these pairs to backward/forward functions (e.g., map tap with forward and slide with backward). Unfortunately, this mapping is arbitrary and unnatural [Norman 2013]. The natural mapping should be slide up and down, which exhibited some confusion of classification in the evaluation and may require personalization in actual application. These simple tap events can also be used as a shortcut to certain functions/menus, such as mapping tap events to open the task managers and then using the slide up/down or side tap for switching applications.

8.2.2. Limitations. One limitation of one-handed interaction is the potential for false positives while users are adjusting their hands. Based on our observation, even if a user performs gestures while walking around, as long as he is holding the phone steadily, there will be a lower chance of triggering false positives. But if a user shakes the phone and generates sound noise at the same time (e.g., scratching the case), there will be a higher chance of triggering false positives. As mentioned earlier, all one-handed interaction was performed by the right hand of the participants regardless of their dominant hand. Nevertheless, we believe that our method would be generalizable for both hands as long as the hand posture detected, which has been proven possible [McGrath and Li 2014].

8.2.3. Future Work. We are mostly using the rule-based method for segmentation now. However, a machine learning plus rule-based method can be built to provide much better results to minimize false-positive errors. One way is to systematically collect noise input from various users and add them into the machine learning classifier. Another possible way is to apply more signal processing methods and look at the frequency domain of the data. For example, when the user is moving the hands, the relative sound is different from a tap event, which can be separated by a band-pass filter. In addition, we can explore a higher resolution of input events, such as which position of the back case is tapped and sliding up/down on the side case. Our preliminary investigation indicates a high likelihood of detecting these events for one-handed gestures. However, considering the dexterity of fingers while holding a phone in one hand, too many input events may be unnatural and unnecessary. Therefore, identifying a natural mapping is prudent before applying these gestures to any applications.

8.3. Two-Handed

With two-handed interaction techniques, even one or two input events on the back of the phone can be useful. For example, we may hold the phone in two hands to take a picture in landscape orientation. With BeyondTouch, the user can tap the side case with the thumb (up-left/right gesture) to take a picture, which is a natural mapping.

8.3.1. Applications. The two-handed interactions may also support other common scenarios where users hold the phone in two hands, such as gaming, watching videos, browsing Web sites, or reading books. For example, it is natural to map the top-left and top-right event to backward/forward events to navigate a video or Web site, mapping the remaining events to other functions like stop/start. One of the most common two-handed usage scenarios is gaming, such as playing car-racing games or Whac-A-Mole. For car racing, we can map top-left to brake and top-right to gas, and up-left/right for directions, which avoids the fingers occluding the screen.

8.3.2. Limitations. One limitation of the two-handed interaction is the input detection speed, as one input event usually lasts more than 0.5 seconds. Any event that occurs too fast may influence the classification result. Another limitation is similar to the one-handed one—that is, the user's additional hand movement may introduce false positives.

8.3.3. Future Work. The solutions to the false-positive problems are similar to what we propose for the one-handed interactions: collecting more noise data and applying other signal processing methods. The variance of the user's tap positions requires personalization. Another possible solution is to add simple physical tactile feedback on the back of phone (e.g., a plastic button) such that each user can tap on the same position.

8.4. On-Table

The on-table interaction can help with many scenarios that require a simple and quick response without touching the phone when the phone is placed on a surface.

8.4.1. Applications. Because on-table only recognizes two-input events (slide and tap), it is best suited for applications using simple input events. The best applications are the ones where a user has to interact with the phone without touching it, such as controlling a music player or a phone call. One motivation of the on-table interaction is to provide interaction gestures, which can be performed when one's hands are dirty (e.g., while cooking). Applying on-table to a music player, we can use the tap events to play/pause a song and the slide event to switch to another song. This is also a natural mapping between on-table gestures and touchscreen gestures. Similarly, we can map the tap and slide event to answer or reject a phone call.

8.4.2. Limitations. In the study, if the nail of the participants' finger were too long, they felt uncomfortable using their fingers and nails to conduct the slide gesture. In addition, some users spent longer time to learn how to perform these gestures, especially if their initial tap gestures were light. In the study, the surface below the phone was either wood or plastic, making detection simpler. Although we have not tested it yet, we expect that the on-table interaction may not work well on surfaces that do not conduct sound well. Another limitation for the current implementation of on-table is that the classifier may be influenced when there is a significant noise source nearby.

8.4.3. Future Work. To address the ambient noise issue, one obvious next step is to introduce frequency-domain features to distinguish different sound sources. We can apply a band-pass filter in the segmentation part to filter out noise. The system must also be tuned for the varying sound propagation properties of different surface materials. In addition, by utilizing the frequency difference of the sound generated from performing gestures, we expect to further increase the variety of input events [Harrison et al. 2011].

9. CONCLUSION

We presented the implementation and evaluation of BeyondTouch, a family of interaction techniques that extend the input language to areas off the touchscreen of a smartphone while using only built-in sensors. BeyondTouch can be used in contexts when the touchscreen is not readily available, such as when the phone is inside a pocket, when it is on a table and the user's hands are dirty or wet, when touching the screen occludes the display, or when the user is holding the phone in one hand while the other hand is occupied with another task. We developed the interactions by machine learning (one-handed, two-handed-6-point, on-table) and rule-based (two-handed-4-point, in-pocket) methods. We also conducted experiments to compare the two implementation methods, which showed that the rule-based method can outperform the machine learning implementation without the need to collect any training data for two-handed-4-point interaction. Our evaluation shows that for machine learning interactions, user-independent recognition results for various events range from just over 70% to greater than 90%, with significant improvements possible with extra personalization, and rule-based interactions can achieve greater than 92% accuracy. We also explored the space of practical applications of BeyondTouch, given its current recognition rates, which we plan to review in the future.

ACKNOWLEDGMENTS

We would like to thank all of the participants for their time in the user study and the reviewers for their insightful feedbacks.

REFERENCES

Patrick Baudisch and Gerry Chu. 2009. Back-of-device interaction allows creating very small touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 1923–1932.

Liang Cai and Hao Chen. 2011. TouchLogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security (HotSec'11)*. 9.

Ke-Yu Chen, Daniel Ashbrook, Mayank Goel, Sung-Hyuck Lee, and Shwetak Patel. 2014. AirLink: sharing files between multiple devices using in-air gestures. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, New York, NY, 565–569.

Mayank Goel, Brendan Lee, Md. Tanvir Islam Aumi, Shwetak Patel, Gaetano Borriello, Stacie Hibino, and Bo Begole. 2014. SurfaceLink: Using inertial and acoustic sensing to enable multi-device interaction on a surface. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, 1387–1396.

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*. ACM, New York, NY, 545–554.

Chris Harrison and Scott E. Hudson. 2008. Scratch input: Creating large, inexpensive, unpowered and mobile finger input surfaces. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology.* ACM, New York, NY, 205–208.

Chris Harrison, Julia Schwarz, and Scott E. Hudson. 2011. TapSense: Enhancing finger interaction on touch surfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology.* ACM, New York, NY, 627–636.

Seongkook Heo and Geehyuk Lee. 2011. Forcetap: Extending the input vocabulary of mobile touch screens by adding tap gestures. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, New York, NY, 113–122.

Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. 2000. Sensing techniques for mobile interaction. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 91–100.

Ken Hinckley and Hyunyoung Song. 2011. Sensor synaesthesia: Touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 801–810.

Shigeo Hiraoka, Isshin Miyamoto, and Kiyoshi Tomimatsu. 2003. Behind Touch: A text input method for mobile phones by the back and tactile sense interface. In *Proceedings of the 9th IFIP TC13 International Conference on Human-Computer Interaction (Interact'03)*. 131–138.

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 4th International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, New York, NY, 109–112.

Kevin A. Li, Patrick Baudisch, and Ken Hinckley. 2008. Blindsight: Eyes-free access to mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 1389–1398.

William McGrath and Yang Li. 2014. Detecting tapping motion on the side of mobile devices by probabilistically combining hand postures. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 215–219.

Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. 2012. TapPrints: Your finger taps have fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. ACM, New York, NY, 323–336.

Donald A. Norman. 2013. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, New York, NY.

Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: Password inference using accelerometers on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. ACM, New York, NY, 9.

Joseph A. Paradiso, Che King Leo, Nisha Checka, and Kaijen Hsiao. 2002. Passive acoustic knock tracking for interactive windows. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*. ACM, New York, NY, 732–733.

Simon Robinson, Nitendra Rajput, Matt Jones, Anupam Jain, Shrey Sahay, and Amit Nanavati. 2011. TapBack: Towards richer mobile interfaces in impoverished contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 2733–2736.

Sami Ronkainen, Jonna Häkkilä, Saana Kaleva, Ashley Colley, and Jukka Linjama. 2007. Tap input as an embedded interaction method for mobile devices. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. ACM, New York, NY, 263–270.

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*. ACM, New York, NY, 303–308.

Paul Schmieder, John Hosking, Andrew Luxton-Reilly, and Beryl Plimmer. 2013. Thumbs up: 3D gesture input on mobile phones using the front facing camera. In *Human-Computer Interaction—INTERACT 2013*. Lecture Notes in Computer Science, Vol. 8118. Springer, 318–336.

Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. 2004. Gummi: A bendable computer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 263–270.

James Scott, Shahram Izadi, Leila Sadat Rezai, Dominika Ruszkowski, Xiaojun Bi, and Ravin Balakrishnan. 2010. RearType: Text entry using keys on the back of a device. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, New York, NY, 171–180.

Masanori Sugimoto and Keiichi Hiroki. 2006. HybridTouch: An intuitive manipulation technique for PDAs using their front and rear surfaces. In *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, New York, NY, 137–140.

Weka. n.d. Weka 3: Data Mining Software in Java. Retrieved June 17, 2016, from http://www.cs.waikato.ac.nz/ml/weka/.

Piotr Wendykier. n.d. JTransforms. Retrieved June 17, 2016, from https://sites.google.com/site/piotrwendykier/software/jtransforms.

J. Wiese, T. S. Saponas, and A. J. Brush. 2013. Phoneprioception: Enabling mobile phones to infer where they are kept. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 2157–2166.

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*. ACM, New York, NY, 269–278.

Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. 2003. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 61–70.

Zhi Xu, Kun Bai, and Sencun Zhu. 2012. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, New York, NY, 113–124.

Cheng Zhang. 2013. BeyondTouch: A framework for extending input on commodity smartphones. In *Proceedings of the Adjunct Publication of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13 Adjunct Publication)*.

Cheng Zhang, Aman Parnami, Caleb Southern, Edison Thomaz, Gabriel Reyes, Rosa Arriaga, and Gregory D. Abowd. 2013. BackTap: Robust four-point tapping on the back of an off-the-shelf smartphone. In *Proceedings of the Adjunct Publication of the 26th Annual ACM Symposium on User Interface Software and Technology.* ACM, New York, NY, 111–112.

Cheng Zhang, Anhong Guo, Dingtian Zhang, Caleb Southern, Rosa Arriaga, and Gregory Abowd. 2015. BeyondTouch: Extending the input language with built-in sensors on commodity smartphones. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. ACM, New York, NY, 67–77.

Received July 2015; revised December 2015; accepted March 2016