BeyondTouch: Extending the Input Language with Built-in Sensors on Commodity Smartphones

Cheng Zhang, Anhong Guo, Dingtian Zhang, Caleb Southern, Rosa Arriaga, Gregory Abowd
Georgia Institute of Technology
85 Fifth Street NW, Atlanta GA 30332, USA
{chengzhang, guoanhong, dingtianzhang, caleb.southern, abowd}@gatech.edu, arriaga@cc.gatech.edu

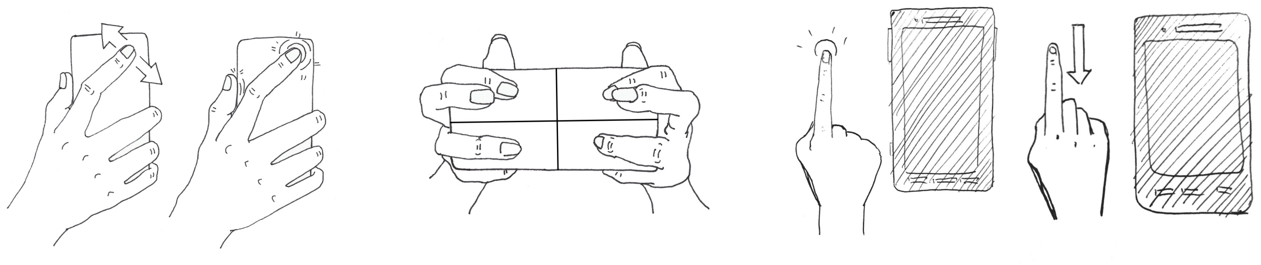


Figure 1: BeyondTouch interaction techniques.

ABSTRACT
While most smartphones today have a rich set of sensors that could be used to infer input (*e.g.*, accelerometer, gyroscope, microphone), the primary mode of interaction is still limited to the front-facing touchscreen 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 tapping and sliding inputs on the case of and the surface adjacent to the smartphone. We outline the implementation of 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.

Author Keywords
Mobile interactions; smartphones; inertial sensors; acoustic sensors.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: User interfaces; Input devices and strategies.

*IUI'*15, March 29 - April 01 2015, Atlanta, GA, USA
Copyright 2015 ACM 978-1-4503-3306-1/15/03…$15.00
http://dx.doi.org/10.1145/2678025.2701374

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 on the 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 *one-handed*, *two-handed* and *on-table*, extending the input language of a smartphone in several interesting ways (see Figure 1).

Contributions and Overview of Paper
While 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 [11, 22]), 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.

While previous work has suggested the possibility of sensing a small set of input events, BeyondTouch is distinguished by the breadth of input events enabled and the discussion on practical issues about applying machine learning on mobile interactions (*e.g.*, personalization on models) and challenges deploying them in real applications.

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

- On-case interactions (one-handed and two-handed), to avoid occluding the limited screen area with the fingers;
- Indirect interaction, by touching the surface (*e.g.*, a table) on which the phone is placed.

We will provide a review of related work next. We then describe the technical approach behind the implementation of BeyondTouch, the machine learning approach to recognize the various input events. We provide an empirical evaluation of the performance of BeyondTouch and discuss the implication of these results for practical use. Our results suggest a hybrid machine learning approach for some scenarios in which a little bit of personalized training data improves performance for some users.

RELATED WORK

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., who instrumented a PDA with a variety of extra sensing capabilities to demonstrate a wide variety of interactions that could augment the user experience [9]. This work inspired much of the related research we have reviewed. While most of the above work has taken the same 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.

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 in order to eliminate occlusion or increase efficiency and accessibility of the device. This can be accomplished by adding additional hardware, such as a keypad [11,14], a touchpad [1,24,25,26,30] or through computer vision [23,29].

Other research has explored the potential gestures supported by the built-in sensing capabilities on the phones. Hinckley et al. explored techniques using only inertial sensors and touch input for hand-held 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 [10]. Tap events have been further investigated by recognizing side taps [15], detecting taps on the back case of a phone using the microphone as a sensor relying on off-device computation to process the input [20] or distinguishing between a single gentle or firm tap on the case [8]. 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 [21], 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 [12]. 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 [5].

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 [3]. Others have explored off-the-phone interactions while a phone is placed on a flat surface (*e.g.*, table), requiring customized pickups attached on the surface to capture the acoustic signals on the surface, such as a modified stethoscope [6] or contact piezoelectric sensors [19]. 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 [4]. However, what kind of off-the-phone gestures can be detected without using any additional hardware other than the built-in sensors is still unclear. In addition, we introduce the use of dual microphones available in most modern smartphones for recognizing gestures. Although previous work has explored recognizing gestures by differentiating sounds generated by gestures [7], the on-table interaction in BeyondTouch can detect both sliding and tapping gestures with only built-in sensors.

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 touchscreen [2,16,18,31]. However, most of these techniques rely on off-line inference; it is unclear how they will work for real-time interaction tasks.

MOTIVATION AND APPLICATIONS

BeyondTouch was motivated by many common smartphone usage scenarios, where the touchscreen or on-case buttons limit convenient interactions. For example, when the user is cooking, he may need to clean or dry his hands before using the touchscreen. In other scenarios, such as gaming, the user's fingers occlude the already limited viewing area of the touchscreen. To address these limitations, we implemented three classes of interaction scenarios: one-handed, two-handed, and on-table.

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 1a).

The one-handed tapping interactions can be used for applications such as browsing through images and webpages. They can also be used for camera functions, like 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 on and off the flashlight 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 song on a playlist.

Two-Handed

The 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 the two-handed interaction 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 soft buttons on the back of the case (Figure 1b).

In the current implementation, we offer 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 case rather than covering the touchscreen.

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 is 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 1c). Furthermore, the on-table interactions can be extended for many applications that demand straightforward interactions, such as responding to a phone call or controlling the music player.

IMPLEMENTION OF BEYONDTOUCH

Table 1 describes the complete set of input events and related sensors that BeyondTouch supports. 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 techniques in order to identify the characteristic signature of each input event. The machine learning techniques were implemented using Weka [27], a publicly available tool that provides implementations of various machine-learning algorithms that can be run in real-time on a smartphone.

Table 1: Interaction techniques.

Scenario	Interaction	Input Events	Sensors
One-Handed	Tap	back-single-tap, back-double-tap, side-tap	Gyroscope, accelerometer, the microphone at bottom
	Tap-Slide	back-single-tap, back-double-tap, back-slide-up, back-slide-down	Gyroscope, accelerometer, the microphone at bottom
Two-Handed	6-point-tap	up-left, up-right, top-left, top-right, bottom-left, bottom-right	Gyroscope, accelerometer, the microphone at bottom
On-Table	Slide, Tap	Slide, tap on the surface around a phone	Gyroscope, accelerometer, two microphones

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) a 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. Therefore, we decided to employ machine-learning based methods to explore other interactions potentially supported by the built-in sensors on a smartphone.

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 [12]. In order to detect where the user tapped on the case, we then added the gyroscope, which better reflects the difference on 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.

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

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

Implementation of On-case Interactions

We implemented the *one-handed* (Tap, Tap-Slide) interactions and *two-handed* 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 100 Hz, 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 x-axis/z-axis of the accelerometer and y-axis/z-axis of gyroscope.

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 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 and decided on which features to use and 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 gyroscope and accelerometer, we calculated the seven features over each frame, including root-mean-square, the derivative of 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 the 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 JTransform [13] to calculate the frequency energy on each frequency band. We build 31 bins over the frequency 100 Hz 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 [27], and then use this model to classify gestures.

Implementation of On-table Interaction

The on-table interaction is also implemented using a combination of rule-based and machine learning approach.

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 bottom microphone and the maximum and minimum value from z-axis of the accelerometer or y-axis of the gyroscope.

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

We compared the performance of different machine learning algorithms on our collected training data using ten-fold-cross validation provided by Weka. Both SVM and K Nearest Neighbor (KNN) provided above 97.5% for precision and recall. We elected to use KNN in our implementation.

Building Training Model

Usually, a trained machine-learning model 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

[1] We approximate the derivative of sensor data as the change in value divided by the change in time.

[2] Energy is the square root of the sum of squares for all samples in the frame.

user-independent model in our implementation, which applies a pre-collected training data set and does not require any data collection from each user. Based on our pilot study result, 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 in order to approximate the behavior of different users.

To build the training set for two-handed and one-handed interactions, we collected training data from 10 users, including two authors. Four of them provided examples for both two interactions, while others provided examples for only one of the interaction. As a result, each interaction has 10 trainers and approximately 1000 instance 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 one-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.

After collecting all the data, we manually labeled each input event in the training set. 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.

EVALUATION METHOD

We evaluated the accuracy and usability of BeyondTouch for the interactions listed in Table 1. We presented users with a series of stimuli for the interactions, and recorded the responses interpreted by the pre-trained machine-learning models. All interactions were performed in a lab environment.

Participants and Apparatus

We recruited 12 participants (1 female) for two-handed and one-handed interaction using a Samsung Galaxy S3 phone, and another 11 participants (6 female) for on-table interaction using a Samsung Galaxy S3 (7 participants) and a Samsung Nexus phone (4 participants).

Study Procedure

In the practice phase of the study, 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 ten times in a row.

In the evaluation phase, we presented the participant with blue-colored visual stimuli on the smartphone screen indicating which gesture to make. The user also received visual feedback after responding to the stimuli. When classification result matched the stimuli, the feedback was given in green, otherwise red. Each participant responded to 40 stimuli (2 sessions × 20 stimuli) for each gesture with random order in one-handed and two-handed interactions. For the on-table interaction, each participant responded to 90 stimuli (3 sessions × 30 stimuli), randomly chosen between tap and slide gestures. For each interaction, we asked the users to report any self-made errors in this phase. It took less than one hour for each participant in the user study, and there was no compensation.

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.

Accuracy

For one-handed and two-handed interactions, there were 480 examples (20×2×12) for each gesture, and 990 inputs (30×3×11) for on-table interaction. Table 2 presents the overall accuracy and standard deviation for each interaction.

Table 2: Accuracy of each interaction technique.

Interaction	Two-Handed	One-Handed-Tap	One-Handed-Tap-Slide	On-Table
Accuracy	71.28% (SD=12.89%)	88.47% (SD=3.55%)	72.92% (SD=6.53%)	93.74% (SD=4.64%)

One-handed Interaction

For one-handed-Tap interaction, the overall accuracy is 88.47%. The confusion matrix is reported in Table 3. Side-tap was the most accurate (98.54%), and double-tap was the 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.

Table 3: One-handed-Tap confusion matrix.

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

If we collapse back-tap and double-tap into one gesture, higher accuracy can be achieved (97.92%) (Table 4).

Table 4: One-handed-Tap (back tap combined + side) confusion matrix.

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

For one-handed-tap-slide interaction, the overall accuracy is 72.92%. The confusion matrix is reported in Table 5. 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.

Table 5: One-handed-Tap-Slide confusion matrix.

User Input	Classification			
	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%

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 6).

Table 6: One-handed-Tap-Slide (backtap combined) and slide (combined) confusion matrix.

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

Two-handed Interaction

For two-handed interaction, the overall accuracy is 71.28%. The confusion matrix is reported in Table 7. The up-right corner 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), making them feel tired. This also led to 12 user self-made errors.

Table 7: Two-handed-6-point confusion matrix.

User Input	Classification					
	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%

On-Table Interaction

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

DISCUSSIONS

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.

Accuracy and User Impressions

Some users reported that tapping continuously for so long (tap and slide for over 500 times) made them feel less natural and comfortable. While the experimental procedure allowed us to gather much data under different situations, we recognize that it put unrealistic demands on the 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 some users adjusted their gestures according 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.

We discuss other observations for each scenario that impacted results.

One-handed Interaction

When the 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.

Two-handed 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 a lot. When the users adjust hands positions while holding the phone, some false-positives were introduced.

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 loud enough sound it would increase the difficulty of detecting the occurrence of an event. From our observation, most users can learn very fast during the practice 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.

Personalization

In the user study, the one-handed and two-handed machine learning based interactions showed an obvious drop in terms of accuracies compared with the result we computed on the training data set using 10-fold cross validation and 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 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.

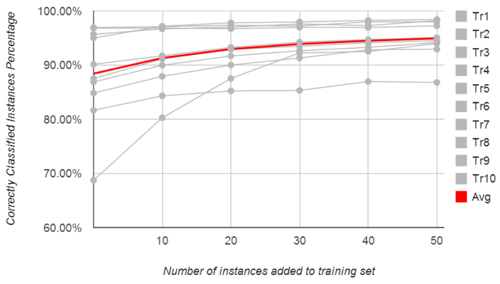
Our approach to personalization is a revision on the leave-one-participant-out method. Instead of leaving one trainer's[3] complete training set out, we methodically incorporated a stratified and randomly selected subset of that person's training data into the model that was then used to test the remainder 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 interactions, we have 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 data set. To avoid bias, we ran the sample selection process 10 times, then built and evaluated the model for each of them. Then we averaged results from the 10 sample runs and report those below. Note that though 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.

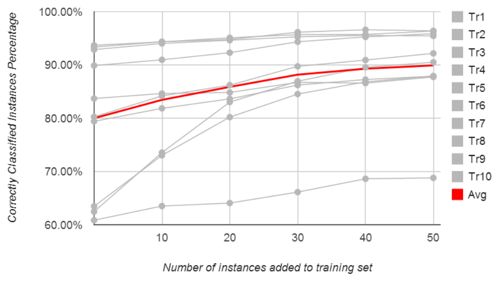
To evaluate this method, we compare the accuracy results for different sizes of the trainers' input set (randomly selected subsets of size 10, 20, 30, 40, 50 of the approximately 100 overall events) against the baseline of using none of that person's training set to build the model (*i.e.,* using 0 of the overall events).

[3] We refer to these individuals as "trainers" in the following discussion, distinct from user study participants.

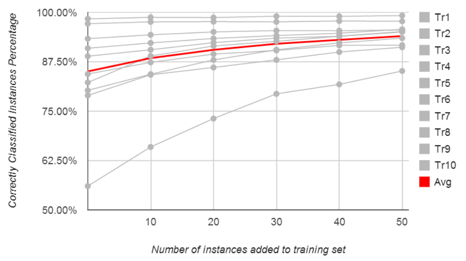
In Figure 2, we show the personalization results for the three usage scenarios. For each graph, the red/bolder line represents the averaged result for the "leave-one-participant-partially-out" method across all trainers in our training set. The x-axis indicates how many of the trainer's input events were added to build the personalized ML model. For example, the one-handed-tap interaction scenario provides three input events: back-tap, back-double-tap, and side-tap. Each user provided 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 data set. Since the one-handed-tap-slide scenario has three distinct input events, a total of 10×3=30 instances were incorporated into the personalized training set.



a. One-handed-Tap



b. One-handed-Tap-Slide



c. Two-handed-6-point

Figure 2: Personalization results

Figure 2 reveals an interesting insight. There is a pretty significant spread of accuracy results across the trainers, with each scenario having at least one trainer whose classification accuracy is much poorer than the rest of the trainers with no personalization (x-axis value of 0). 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% while the standard deviations decrease to 0.09, 0.04, and 0.06. Figure 2 demonstrates these positive trends for all three of the machine learning scenarios.

Furthermore, beyond using 30 examples of each event from the trainer's, there is no appreciable improvement of the personalization. This is also good, as it means 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 trainer's 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 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, and 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.

Challenges for Practical Deployment

There are a number of issues beyond recognition rates that impact the practical utility of BeyondTouch.

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 did not test our technique on a large variety of phones. However, since the sensors we used are most common ones and we applied 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 above. While this data collection may be time-consuming, it does not directly impact the user of the device, until and unless personalization is warranted.

Rule-based Methods vs. Machine-learning Methods

BeyondTouch employed a combination rule-based (segmentation) and machine learning (classification) techniques for detecting input events. However, in our early exploration, we found that rule-based techniques are also appropriate for classifying less complex input gestures with easy implementation (*e.g.*, tap events on the four corners). Machine-learning methods can recognize more complicated gestures at the price of collecting training data and personalization as well as a higher requirement for computation. A potential mixture between these two methods can be an effective practical method.

Usage of Multiple Microphones

Most smartphones now 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 the environment 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 richer potential interaction events can be detected by combing the data from multiple microphones and inertial sensors.

Interactions by Contexts

In practice, many of the interactions 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, while 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, would increase the false-positives and false-negatives. Hence, an automatic detection of context about the position and orientation of the phone [28] is a necessary step for applying BeyondTouch in real life.

Energy Consumption

BeyondTouch interactions are largely dependent on sensors. However, recent 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 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.

CRITIQUE OF APPLICATIONS OF BEYONDTOUCH

In addition to the applications we proposed in the introduction, the study participants also suggested interesting usage scenarios for BeyondTouch. In this section, we will 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.

One-handed

One-handed interactions can support up to five input events. In our evaluation, some gestures were detected with high accuracy while others were confused with other gestures. However, not all the applications require all of the 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 would new combination of input events influence the usability still needs to be further evaluated.

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/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 which only need a single input event from one-handed gestures, such as taking selfies with the front-camera, playing flappy bird (requires high speed), quick check of the time, and so forth.

The most common scenarios for one-handed interaction with phones are reading websites/books/emails or playing videos. These usually require two or three input gestures, such as forward/backward. From the evaluation, we can see 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 [17]. 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 shortcut to certain functions/menus, for example, mapping tap events to open the task managers, and then using the slide up/down or side tap for switching applications.

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 generate sound noise at the same time (*e.g.*, scratching the case), there will be a higher chance of triggering false-positives.

Future work

We are mostly using 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 method 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.

Two-handed

With the 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 very natural mapping.

The two-handed interactions may also support other common scenarios where users hold the phone in two hands, such as gaming, watching videos, browsing websites, or reading books. For example, it is very natural to map the top-left and top-right event to backward/forward events to navigate a video or website, 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.

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.

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 method. The variance of user's tap positions requires personalization. And 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.

On-table

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

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 one 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 your 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 very natural mapping between on-table gestures and the touchscreen gestures. Similarly, we can map the tap and slide event to answer or reject a phone call.

Limitations

In the study, if the nail of the participants' finger were too long, they felt uncomfortable to use their fingers and nails to conduct the slide gesture. Also 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 yet, we expect the on-table interaction may not work well on other 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.

Future work

To address the ambient noise issue, one obvious next step is to introduce frequency-domain features in order 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 [7].

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 touching the screen occludes the display, when the user is holding the phone in one hand and performing another task with the other hand, or when the user's hand is dirty or wet. Our evaluation shows that user-independent recognition results for various events range from just over 70% to over 90%, with significant improvements possible with extra personalization. We explored the space of practical applications of BeyondTouch, given its current recognition rates, which we plan to review in the future.

ACKOWLEDGEMENT

We would like to thank Thomas Ploetz, Agata Rozga and the Ubicomp Lab at Georgia Institute of Technology for their feedback. We also want to thank all reviewers for their suggestions and the user study participants. This material is based in part upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1148903.

REFERENCES

1. Baudisch, P., & Chu, G. (2009, April). Back-of-device interaction allows creating very small touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1923-1932). ACM.
2. Cai, L., & Chen, H. (2011, August). TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. In *HotSec*.
3. Chen, K. Y., Ashbrook, D., Goel, M., Lee, S. H., & Patel, S. (2014, September). AirLink: sharing files between multiple devices using in-air gestures. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 565-569). ACM.
4. Goel, M., Lee, B., Islam Aumi, M. T., Patel, S., Borriello, G., Hibino, S., & Begole, B. (2014, April). 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* (pp. 1387-1396). ACM.
5. Goel, M., Wobbrock, J., & Patel, S. (2012, October). 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* (pp. 545-554). ACM.
6. Harrison, C., & Hudson, S. E. (2008, October). 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* (pp. 205-208). ACM.
7. Harrison, C., Schwarz, J., & Hudson, S. E. (2011, October). TapSense: enhancing finger interaction on touch surfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 627-636). ACM.
8. Heo, S., & Lee, G. (2011, August). 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* (pp. 113-122). ACM.
9. Hinckley, K., Pierce, J., Sinclair, M., & Horvitz, E. (2000, November). Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM*

symposium on User interface software and technology (pp. 91-100). ACM.

10. Hinckley, K., & Song, H. (2011, May). Sensor synaesthesia: touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 801-810). ACM.

11. Hiraoka, S., Miyamoto, I., & Tomimatsu, K. (2003). Behind Touch, a Text Input Method for Mobile Phones by The Back and Tactile Sense Interface.*Information Processing Society of Japan, Interaction 2003*, 131-138.

12. Hudson, S. E., Harrison, C., Harrison, B. L., & LaMarca, A. (2010, January). Whack gestures: inexact and inattentive interaction with mobile devices. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction* (pp. 109-112). ACM.

13. JTransform:https://sites.google.com/site/piotrwendykier/software/jtransforms

14. Li, K. A., Baudisch, P., & Hinckley, K. (2008, April). Blindsight: eyes-free access to mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1389-1398). ACM.

15. McGrath, W., & Li, Y. (2014, October). 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*(pp. 215-219). ACM.

16. Miluzzo, E., Varshavsky, A., Balakrishnan, S., & Choudhury, R. R. (2012, June). Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (pp. 323-336). ACM.

17. Norman, D. A. (2002). *The design of everyday things*. Basic books.

18. Owusu, E., Han, J., Das, S., Perrig, A., & Zhang, J. (2012, February). Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications* (p. 9). ACM.

19. Paradiso, J. A., Leo, C. K., Checka, N., & Hsiao, K. (2002, April). Passive acoustic knock tracking for interactive windows. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems* (pp. 732-733). ACM.

20. Robinson, S., Rajput, N., Jones, M., Jain, A., Sahay, S., & Nanavati, A. (2011, May). TapBack: towards richer mobile interfaces in impoverished contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2733-2736). ACM.

21. Ronkainen, S., Häkkilä, J., Kaleva, S., Colley, A., & Linjama, J. (2007, February). Tap input as an embedded interaction method for mobile devices. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (pp. 263-270). ACM.

22. Saponas, T. S., Harrison, C., & Benko, H. (2011, October). PocketTouch: through-fabric capacitive touch input. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 303-308). ACM.

23. Schmieder, P., Hosking, J., Luxton-Reilly, A., & Plimmer, B. (2013). Thumbs Up: 3D Gesture Input on Mobile Phones Using the Front Facing Camera. InHuman-Computer Interaction–INTERACT 2013 (pp. 318-336). Springer Berlin Heidelberg.

24. Schwesig, C., Poupyrev, I., & Mori, E. (2004, April). Gummi: a bendable computer. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 263-270). ACM.

25. Scott, J., Izadi, S., Rezai, L. S., Ruszkowski, D., Bi, X., & Balakrishnan, R. (2010, September). 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* (pp. 171-180). ACM.

26. Sugimoto, M., & Hiroki, K. (2006, September). 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* (pp. 137-140). ACM.

27. Weka. DOI: http://www.cs.waikato.ac.nz/ml/weka/

28. Wiese, J., Saponas, T. S., & Brush, A. J. (2013, April). Phoneprioception: enabling mobile phones to infer where they are kept. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2157-2166). ACM.

29. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., & Shen, C. (2007, October). Lucid touch: a see-through mobile device. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (pp. 269-278). ACM.

30. Wobbrock, J. O., Myers, B. A., & Kembel, J. A. (2003, November). 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* (pp. 61-70). ACM.

31. Xu, Z., Bai, K., & Zhu, S. (2012, April). Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks* (pp. 113-124). ACM.