Expansion of Human–Phone Interface By Sensing Structure-Borne Sound Propagation

Yu-Chih Tung and Kang G. Shin
The University of Michigan
Email: {yctung,kgshin}@umich.edu

ABSTRACT

ForcePhone is a novel system that enables commodity phones to recognize the force applied to their touch screen and body. Researchers have shown the usefulness and importance of this expressive input interface (especially for the one-hand operation), but this advanced function has not yet been realized and deployed in most state-of-the-art smartphones. Instead of employing or augmenting specialized/proprietary sensors, ForcePhone uses only the phone's built-in sensors to measure the applied force via a physical property called *structure-borne sound propagation*. ForcePhone has been implemented and evaluated on both iOS and Android phones. Multiple demo applications, such as getting the option menu by hard-pressing a button or surfing the previous webpage by squeezing the phone, have been implemented and tested successfully. The estimated force is shown highly correlated to the real applied force and the estimation error is less than 54g when the phone is stationary. Users can easily control the applied force at two different levels with a 97% accuracy. Moreover, ForcePhone can detect the squeeze of the phone body with a higher than 90% accuracy. Most participants in our usability study were able to master the ForcePhone-based apps and find them very useful.

General Terms

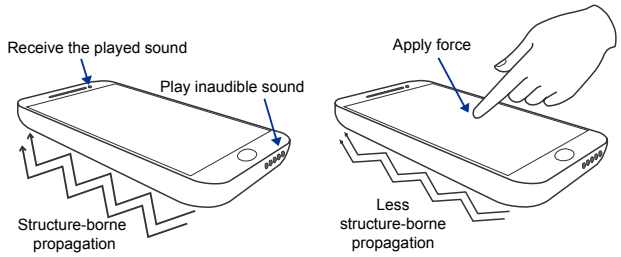
Design, Measurement, Performance, Algorithms

Keywords

Force sensing, Sound, Structure-borne sound, Mobile Phones

1. INTRODUCTION

As smartphones become an essential part of our daily activities, human–phone interactions have become a norm. To enhance the input capability on the severely limited space of a phone's touch screen, researchers and practitioners have been seeking various ways to expand the input dimensions. For example, augmenting a force-sensitive, deformable, or squeezable input is shown to enrich the input vocabulary significantly, especially for the one-hand operations [13, 29, 36]. However, most of those extended input interfaces have not yet been fully developed nor deployed in commodity phones for two reasons. First, they usually require additional hardware, such as capacitive or contact piezoelectric sensors [14, 32, 34], which are usually unavailable in commodity phones, and the requirement of additional cost and space make them less attractive to phone users and manufacturers [9]. Second, systems based only on built-in sensors usually impose unnatural/inconvenient usage restrictions because it is challenging to recognize those interactions without additional sensors. For example, users are required to touch the microphone reception hole or block the camera flash light source for sensing a touch interaction [22, 27], both of which limit the usability of this additional sensing. Other systems require continuous vibration of the phone with a vibration motor, causing significant annoyance to users [21]. To the best of our knowledge, the only commodity phone supporting a force-sensitive touch screen is the latest iPhone 6s, enabled by their proprietary sensors [3]. Unlike these systems, we propose a new, inexpensive solution, called ForcePhone, which provides a force-sensitive input interface to commodity phones without any addition/modification of hardware. Moreover, ForcePhone provides this force-sensing capability to the touch screen as well as the phone's body, called a *squeezable interface*.



(a) No force (b) Force applied

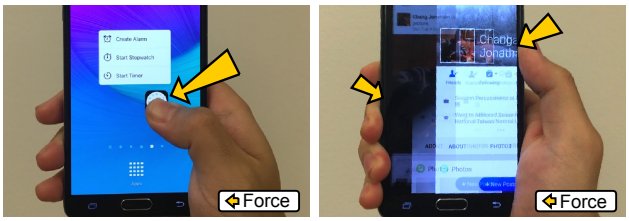
Figure 1—Structure-borne propagation and the applied force. When no force is applied to the phone, the frame and internal components of the phone can vibrate freely, and hence the played inaudible sound can easily propagate through the phone's body.

ForcePhone estimates the user-applied force by utilizing the structure-borne sound propagation, i.e., the sound transmitted through subtle vibrations of the device body. In most designs, such as headphones or pipe-work, this type of propagation is usually considered as a mechanical noise, but ForcePhone uses it in a novel way to estimate the force applied to commodity phones. As shown in Fig. 1, when the phone is left free to vibrate (e.g., the user is not touching/squeezing the phone), the sound sent from the phone's speakers can easily travel through its body to its microphone. However, when force is applied to the phone, it restricts the phone body's vibration with the sound, thus degrading the sound traveling through this structure-borne pathway. ForcePhone estimates the amount of force applied to the phone by monitoring the degree of this degradation.

As mentioned earlier, it is challenging to provide extended input interfaces with only limited built-in phone sensors. For example, in ForcePhone, the recorded audio includes not only the sound

MobiSys'16, June 25-30, 2016, Singapore, Singapore

DOI: http://dx.doi.org/10.1145/2906388.2906394



(a) Hard-pressed option (b) Squeezable back button

Figure 2—Demo apps of `ForcePhone`. Users can reach an option page when a button is pressed hard and can also surf the previous webpage by squeezing the phone.

signal transmitted through the phone's body but also that through air, and other reflections. To achieve reliable force estimation, `ForcePhone` exploits the feature of *active* acoustic sensing to get a proper reference point for fetching the signal traveling through the phone's body. `ForcePhone` also utilizes the information of other sensing materials to enhance sensing accuracy and reduce the false detection rate. For example, the location and the start time of a touch are inferred from the touch screen, and the phone's movement is inferred from accelerometer readings, which are then used to *filter* out unexpected audio signal changes caused by environment and human noises. By integrating readings of these sensors, `ForcePhone` can achieve high force-sensing accuracy while limiting the false positive rate.

To date, `ForcePhone` has been implemented on Android and iOS phones to provide a force-sensitive and squeezable interface. We have realized several apps based on `ForcePhone`. Fig. 2 presents two examples of those apps which have been shown useful: (1) the hard-pressed option, which is similar to the 3D Touch on iPhone 6s where varying options are given to users when applying different amounts of force to buttons, and (2) the squeezable back of a phone, which allows users to surf the previous webpage by just squeezing the phone. Note `ForcePhone` is not limited to these two use-cases, and can extend human–mobile interactions in many ways. For example, it has been shown useful to zoom in/out maps by squeezing phones or type upper-case letters by applying force to keyboards [13, 29, 36]. Other potential apps of `ForcePhone` are discussed in Section 7.

In our experiments, the estimated force is shown greater than 0.9 correlation to the real applied force with 54g errors in stationary environments. The participants in our controlled experiments were able to use `ForcePhone` for applying force at 2 different levels with a higher than 97% accuracy and squeezing the phone body correctly with a greater than 90% accuracy. After trying our demo apps of `ForcePhone` most participants think our current design comparable to state-of-the-art proprietary sensors in accomplishing simple tasks, such as hard-pressing of a button. A demo video of `ForcePhone` can be found from [4].

This paper makes the following four main contributions:

- Design of force-sensitive and squeezable interfaces via structure-borne sound propagation;
- Implementation of `ForcePhone` on both iOS and Android phones;
- Demonstration and evaluation of two popular apps; and
- Achieving a higher than 90% accuracy of using the demo apps in most scenarios.

The rest of this paper is organized as follows. Section 2 discusses the related work on expressive input interfaces. Section 3 describes the principle of structure-borne propagation, and Section 4 details the design of `ForcePhone` based on this principle. The implementation and evaluation of `ForcePhone` are presented in Sections 5 and 6, respectively. Limitations and potential use-cases are discussed in Section 7, and the paper concludes with Section 8.

2. Related Work

Expanding the dimension of touch inputs has been a major research topic for many years owing to the limited-size touch screens of mobile phones. Users are shown to be able to easily and effectively control force-sensitive, deformable, or squeezable input interfaces as they are akin to many natural interfaces, such as applying force to a water knob [13, 29, 36]. A force-sensitive interface can be implemented by adding capacity sensors [36], adding contact piezoelectric sensors [32], checking the flash light source blocked by a human hand [27], monitoring the reduction of sound volume by covering the microphone reception hole [22], and estimating the damped motor vibration with accelerometers [21]. These systems either require additional sensors or impose stringent usage restrictions, such as blocking the flash light source by hand or continuously vibrating the phone.

There have also been systems that extend the user's input by recognizing the materials of tapping or the instantaneous tapping force. For example, TapSense [17] classifies the touch sound to recognize if the user touches the screen by a finger tip or a fist. ForceTap [19] utilizes the accelerometer to learn how hard the user taps or shakes the device. It is also possible to extend the touch surface by other accessible objects. SurfaceLink [14] allows users to perform gestures on a table where the phone is placed, while SkinInput [18] uses the human body as an extension of input interface. Here we focus on how to provide a force-sensitive and squeezable input interface by utilizing the structure-borne sound propagation. Table 1 shows a comparative summary of `ForcePhone` and other related work. Note that instantaneous tapping force is different from the force sensing introduced in this paper; the former only represents the instantaneous force sensed at the time of the user contacting the touch screen, while the latter continuously monitors the force applied to the phone after the user touches the phone. Our system is parallel to these instantaneous-tapping-force systems and can be integrated with others to enrich the user experience further.

Sound has been a widely-used sensing method as it only requires microphones and speakers on commodity phones. For example, knowing the surrounding sounds and reflections can provide accurate indoor localization [28, 38] or tracking human living/sleeping behaviors [30, 31]. `ForcePhone` is also sound-based but designed for providing force-sensitive and squeezable interactions. The closest to `ForcePhone` are Touch & Active [33] and PseudoButton [22]. Touch & Active attaches external contact piezoelectric microphones and speakers to objects to learn how the user is touching the object, which is achieved by fingerprinting the object's resonants. A follow-on study [32] showed that the same fingerprinting technique can also be used to infer the force applied to objects, which aimed mainly to ease product prototyping. Using the built-in speakers and microphones on commodity phones was mentioned as a potential interface for Touch & Active, but no further details were provided. Moreover, `ForcePhone` directly models and interprets the signal changes caused by the applied force, rather than classifying the resonants by using machine learning. The latter usually requires a significantly more effort for periodic training and retraining. On the other hand, PseudoButton utilizes similar sound degradation to estimate the applied force. However,

System	Interaction	Sensing Methods	Additional Sensors	Working Areas
Apple 3D Touch [3]	Force	Proprietary sensors	Additional	Touch screen
Camera and Flash[27]	Force	Camera+Flash light	Existing	Camera lens
Touch & Active [33, 32]	Force	Sound	Additional	Any object (w/ sensors attached)
PseudoButton [22]	Force	Sound	Existing	Microphone reception hole
Expressive Touch [34]	Tapping material	Sound	Existing	Touch screen
ForceTap [19]	Instantaneous tapping force	Accelerometer	Existing	Touch screen+Phone body
VibPress [21]	Force+Squeeze	Actuator+Accelerometer	Existing	Vibrated phone body
Acoustruments [24]	Force+Squeeze	Sound	Additional	External connected objects
ForcePhone	**Force+Squeeze**	**Structure-Borne sound**	**Existing**	**Touch screen+Phone body**

Table 1—Existing touch interfaces to enrich input dimensions.

it focuses on the sound degradation of airborne propagation, so it can only estimate the force changes at the microphone reception hole, e.g., the recorded volume decreases when the user completely blocks the microphone reception hole. In contrast, ForcePhone utilizes the structure-borne sound propagation, which can provide this force-sensitive touch around the entire body of each phone.

3. Structure-Borne Propagation

Sound is a mechanical wave broadcasted by compressions and rarefactions. The most common material for sound to propagate is the air, which is known as *airborne propagation*. Besides the air path, as shown in Fig. 3, when the sound is generated and received by the same device, its body becomes another pathway for the sound to travel, which is called *structure-borne propagation*. Since the structure-borne propagation is usually unrelated to the intended application, it is generally considered as a noise [12, 26, 39]. However, ForcePhone utilizes it in a novel way to estimate the force applied to the phone by monitoring the degradation of structure-borne propagated sound.

As shown in Fig. 4, we model a vibrating phone as a forced, damped mass–spring system, where the phone vibrates up and down with the force F_v, which is caused by the sound played at the phone speaker. When there is no external force applied, the phone can vibrate freely with amplitude A_0, which is captured by the system's effective spring constant, K_0, and also the damping coefficient. Moreover, since the phone is considered as a rigid body, the equilibrium is located at A_0 above the table. Based on Hooke's law [20], when an external force F_h is applied to the screen, the system equilibrium moves downward by F_h/K_0. However, since the phone (a rigid body) is impossible to move downward "into" the table, the applied force also makes the system spring constant increase to K so as to meet this equilibrium change and rigid body constraint. In such a case, if the vibration energy is constant, the potential energy for both systems will be identical, thus leading to:

$$\frac{1}{2}K_0A_0^2 = \frac{1}{2}KA^2 = \frac{F_h}{A_0 - A}A^2 \quad (1)$$

which defines the relation between the applied force and the reduced vibration amplitude. Note that this basic model is designed only for an illustrative purpose as it doesn't consider the horizontal vibration and the increased friction caused by the applied force. Moreover, since the hand and the table also slightly vibrate with the phone, the system's damping coefficient and effective mass will change accordingly, which are not accounted for, either. However, according to our vibration measurements, this simplified model suffices to describe the principle of ForcePhone's operation.

We used the Polytec OFV-303 laser vibrometer [7] to measure the nm-level vibration of a phone, capturing the change of

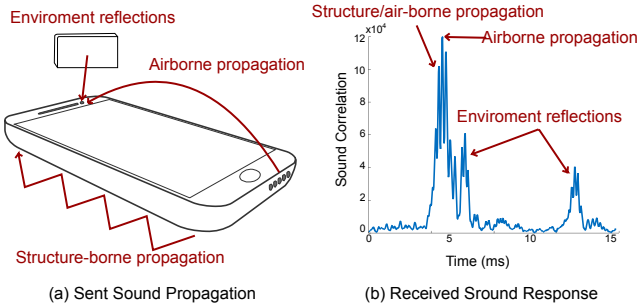


Figure 3—Structure-borne propagation in a phone. The sound received at a phone is a combination of structure- and air-borne propagations as well as the environments' reflections (echo). ForcePhone uses 20 samples before the strongest correlation peak as an indicator for a structure-borne propagation.

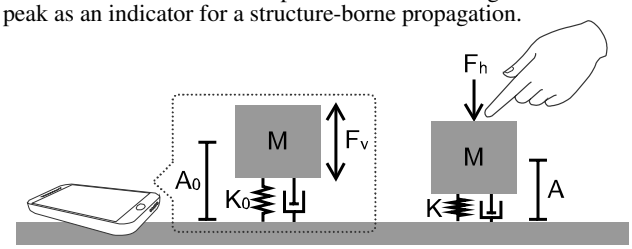
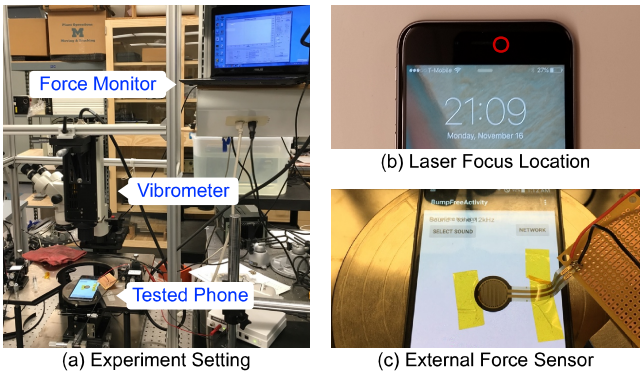


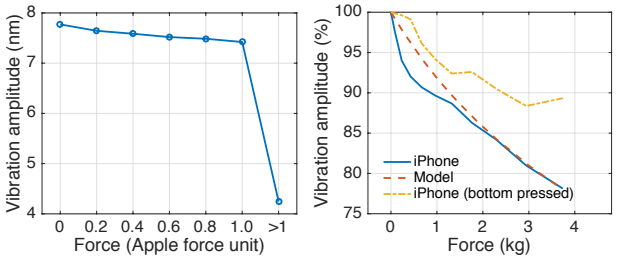
Figure 4—Phone vibration model. Phone vibration is modeled as a forced and damped mass–spring system where the phone vibrates with amplitude A_0 due to the force F_v from the phone speaker. The vibration amplitude is decreased to A and the effective system spring coefficient is increased to K due to the applied force F_h.

structure-borne propagation caused by touching the phone with hand. Compared to capturing the structure-borne propagation via microphones (which ForcePhone is using as described in the next section), this vibration measurement helps neglect the noises caused by the air-propagated sound and the imperfect microphone hardware. Fig. 5 shows our experimental setting where the phone is placed on a metal surface under the laser vibrometer. The laser's focus has been calibrated before the test to ensure that the laser beam will be reflected properly by the phone surface. The latest Apple iPhone 6s is used for these measurements owing to its capability of estimating the applied force. Thumb is used to apply force at the middle of the phone and the iPhone 3D Touch reading (ranging from 0 to 1) is used as a reference of the applied force. From our preliminary test, we found that the Apple 3D Touch sensors can only read the applied force up to about 380g. Hence, we also repeat the same measurements with the Interlink force sensor [5] which can measure force up to 10kg. Its current change caused by the applied force is recorded by the Monsoon power monitor [6].



(a) Experiment Setting (b) Laser Focus Location (c) External Force Sensor

Figure 5—Vibration measurement setting. Vibration is measured by Polytec OFV-303 laser vibrometer when force is applied.



(a) By Apple Sensors (b) By External Sensors

Figure 6—Phone vibration damped by force. The correlation between the damped vibration and the applied force enables ForcePhone's force-sensitive and squeezable interfaces.

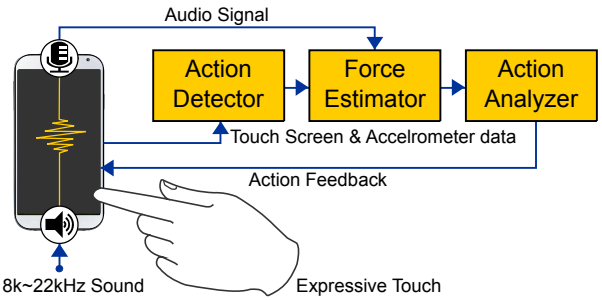


Figure 7—System overview. Force applied to the phone damps the inaudible sound sent from the phone's speaker to its microphone. Accelerometer and gyroscope readings are used to avoid other audio signal noises caused by movements.

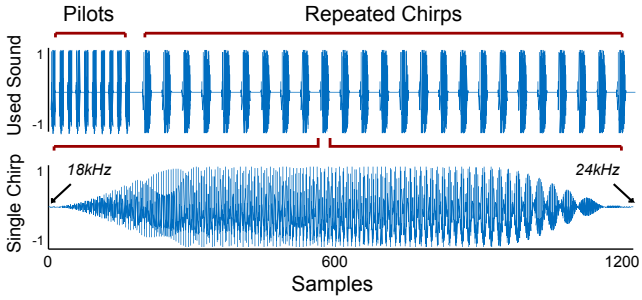


Figure 8—Example of transmitted inaudible sound. The pilots are used to synchronize the phone's microphone and speaker. The subsequent chirps stop for 25ms before playing the next chirp to avoid unexpected noises from environmental reflections. (This figure is rescaled for easy visualization)

Fig. 6 shows the results of phone vibration while applying different amounts of force. As shown in Fig. 6(a), when the Apple 3D Touch reaches its maximum sensitivity, the amplitude of phone vibration decreases about 5%. If more force is applied further (marked as > 1), near 50% of the vibration amplitude is damped by hand. Fig. 6(b) shows the results of measuring the force with external force sensors. The vibration amplitude also decreases about 5% when a 380g force is applied to the phone. Our system model is shown to be able to capture the main trend of vibration amplitude when the initial system spring constant $K_0 = 2.7$(kg/nm) is assumed. The most important property observed in this experiment is the high correlation between the applied force and the decreased vibration amplitude. Based on this property, ForcePhone can provide useful force-sensitive applications as introduced in the following sections.

As mentioned earlier, our model is not perfect. For example, as shown in Fig. 6(b), when the force is applied to the bottom of the phone, it incurs less vibration change since the measured location (i.e., the top of the phone) has less restriction when the phone's bottom is pressed. A similar phenomenon also occurs when different speakers are used to play sound. Thus, it is necessary for ForcePhone to make one-time calibration for unifying different estimations when force is applied at different locations on the screen. The calibration needs to be tuned to the phone models as they have varying vibration patterns depending on the phone material. For example, the vibration of Galaxy Note 4 is found to be 50% larger than the vibration of iPhone 6s, and more sensitive to the applied force since its plastic body is easier to compress. We also found, for some phone models, such as iPhone 6s, this vibration decays too fast when sound is played by the bottom speaker and received by the top microphone, making the force estimation less accurate. In such a case, we used the top speaker to get an adequate signal to noise ratio (SNR) of collected sound even though the collected sound will include a part of airborne propagation components.

4. System Design

As shown in Fig. 7, ForcePhone actively plays an *inaudible* sound with the phone speaker, and then picks up this sound with the phone microphone. The touch screen input (such as the location or the start time of a touch) and the data from the other motion sensors are also recorded. These sensor data are then used to improve the force estimation and reduce the number of false detections. When force is applied to the touch screen or the other parts of the phone body, the action analyzer triggers the pre-designed feedback/behavior based on the monitored (inaudible) sounds and user motions.

4.1 Sound Selection

The design of sound is critical to the system performance. While there are many other possible options, the current design of ForcePhone uses a 1200-sample linear chirp from 18kHz to 24kHz. A hamming window is multiplied to the first and last of 300 samples to eliminate the audible noise caused by spectral leakage [16]. The main frequency range of sound signals used to sense

is about 20kHz–22kHz while the remaining signals close to 18kHz and 24kHz are played with minimal volume which are "stuffed" only to avoid signal loss due to the windowing. ForcePhone samples this chirp at 48kHz and replays it every 50ms. This sound is designed to achieve (i) minimal user annoyance, (ii) high SNR, and (iii) adequate force sensing delay.

According to the field study in [35], humans are shown to have a limited ability to hear sound above 20kHz. Since most modern smartphones only support the sampling rate up to 48kHz, the highest sound frequency to use is 24kHz. This setting is likely to be expanded in the near future; for example, Android 6.0 has announced a plan to support a higher sample rate. Our experiments have shown that playing a high frequency chirp directly is still audible to humans because an abrupt increase/decrease of energy in time domain incurs frequency leakages. To avoid this problem, ForcePhone uses a similar windowing process as in [25] to reduce the noise burst at the beginning and end of signals. This windowing process adds an envelope to the sent signal that controls the signal amplitude to 0 in the beginning and then gradually restores the signal amplitude to the normal range (the same principle is also applied to the end of the signal). This design choice will cause SNR degradation when we estimate the audio correlation of received/sent signals, but it is practically important to avoid any audible sound. Note that the chirp starts at 18kHz but is still inaudible because the beginning of the chirp is played only at the minimal volume (windowing). None of the participants in our user study noticed/heard the sound used in ForcePhone.

Ideally, less stop time between chirps will provide a smaller sensing delay, but the chirps in ForcePhone are designed to be played every 50ms. This parameter setting is chosen to prevent the inclusion of audio signals reflected by environments as an unexpected/unwanted noise. For example, as shown in Fig. 3(b), after the sound transmitted via airborne propagation (i.e., the highest correlation peak) is received, there are multiple following local peaks which indicate the reception of sound reflected by the environment. From our experiments, the received reflections are found to degrade 20dB after 25ms, which is small enough to let ForcePhone play and receive the next round of chirp correctly with only minimal noise due to environmental reflections. Thus, it is necessary for ForcePhone to have a 25ms (i.e., 1200-sample) stop time after playing each chirp.

ForcePhone uses the signal correlation (also known as the *matched filter*) to estimate the reception of the played sound. The SNR of this correlation in the chirp is proportional to the signal length and sweeping frequency [37]. We selected the sweeping frequency as above in order to not annoy users and cope with the hardware limitation while setting the signal length to 25ms. Even though a longer chirp can achieve a higher SNR, it also increases the sensing delay because ForcePhone needs to wait for the completion of the sound being played. To strike a balance between SNR and the sensing delay, ForcePhone sets the chirp length to 25ms, which makes the total delay in sensing each chirp equal to 50ms (i.e., 20 force estimations can be made every second). This setting is found to provide enough SNR for estimation of the applied force and an adequate sensing delay which meets most users' needs.

Besides the design of sound signal, there is another parameter that affects the received SNR: the phone's speaker volume. In the current setting, the audio volume is set to 50% of the maximum to avoid some audible noise due to the imperfect speaker hardware and deal with the coexistence of multiple phones that run

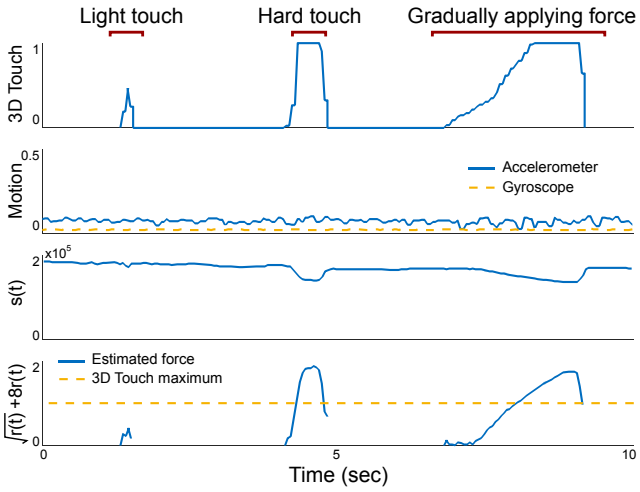


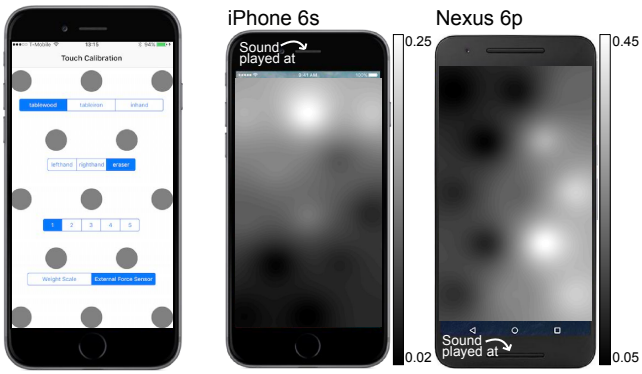
Figure 9—Responses of different amounts of applied force. Motion sensors only capture the initial response of a touch, but the sound response can monitor the subsequent applied force.

ForcePhone. An example of the sent sound is shown in Fig. 8. One thing to note is that there are 10 additional chirps sweeping from 24kHz to 10kHz played before the above-mentioned chirps, which are used as the pilot signal for synchronizing the timing of the phone's microphones and speakers. If the synchronization fails, ForcePhone will stop the sensing process and replay the pilot to achieve correct time synchronization. Detailed performance measurements are presented and discussed in Section 6.

4.2 Estimation of Applied Force

ForcePhone utilizes the structure-borne sound propagation to estimate the applied force. As mentioned in Section 3, when force is applied to the phone body, the hand in contact with the phone body damps/degrades the structure-borne sound propagation. Besides the structure-borne propagation, there are other factors that affect the received sound strength. For example, the airborne propagation might be blocked by hand, and the overall sound signal strength may be enhanced by reflections from the environment or the internal resonant. In ForcePhone, these noises are identified and removed by timestamping the received audio signal. For example, the reflection from an object 10cm away will be received 28 samples later than the airborne propagation since it travels a 10cm longer distance. Moreover, sound usually travels 100x faster in a solid phone [23]. Thus, the structure-borne propagation will be received 21 samples ahead of the airborne propagation when the microphone and the speaker are 15cm apart. Based on these observations, ForcePhone uses the signal which is 20 samples ahead of the airborne propagation as the indicator of the structure-borne propagation, thus removing the most undesirable noise.

Note that the reference of airborne propagation is assumed to be the strongest audio correlation because the sound energy decays faster through the solid phone body and absorbed more on the reflection objects than air. Ideally, the temporal -3dB width of audio correlation are 7 samples for our chirp selection, so it would be possible to find a clear peak ahead of the airborne propagation, indicating the reception of structure-borne propagation. However, as shown in Fig. 3(b), there is no such clear peak found before airborne propagation, and the temporal -3dB width of audio correla-



(a) Calibration UI (b) Signal change $\overline{r(t)}$ of 500g force

Figure 10—Touch calibration. The extent of signal changes caused by the applied force varies with the touch location. Thus, a one-time touch calibration is made at the 13 marked locations to compensate the estimated force at different locations.

tion is about 40 samples in this measurement. This phenomenon is caused by the adoption of windowing, which suppresses the frequency-domain signal leakage but incurs the time-domain signal leakage. This 20-sample-ahead sampling heuristic will thus include both air- and structure-borne propagations. To get a reliable estimation of the applied force, `ForcePhone` utilizes the sound strength when the touch begins as a reference to estimate the subsequent change caused by the force applied later. In the rest of this paper, we will denote the sound correction at time t as signal $s(t)$ and the signal at the beginning of a touch as s_{start}. The subsequent force at time t is estimated based on a metric called the *signal changing ratio*, $r(t) = |(s(t) - s_{start})/s_{start}|$. The applied force $f(t)$ at time t is then determined by a linear regression model with $r(t)$. Fig. 9 shows a real-world example of applying force to an iPhone 6s placed on a wooden table. We simply visualize the estimated force as $\sqrt{r(t)} + 8r(t)$; the intuition behind this setting will be discussed later. In this measurement, there are 3 different types of touch, each with a different applied force: (1) light touch, (2) hard touch, and (3) touch with a gradually increasing force. The ground truth of the applied force can be read from the Apple 3D Touch sensors [3]. As shown in the figure, motion sensors can only detect the slight movement at the start/end of a touch but are unable to determine the applied force. Our heuristic based on $r(t)$ captures most of the force characteristics even when the applied force exceeds the maximum sensing range of Apple 3D Touch.

Note the calibration between $f(t)$ and $r(t)$ varies with the location of force applied on the phone as described in Section 3. Thus, a preparatory experiment is needed for each phone model before `ForcePhone` is activated. However, this calibration is just a one-time requirement, which is different from other sound-fingerprinting systems that need laborious training each time before using the application.

Our current implementation of `ForcePhone` is calibrated by using external force sensors. The calibration is done by applying different amounts of force (up to 1.5kg) at 13 selected locations as shown in Fig 10(a). The remaining parts of touch screen are calibrated by interpolating the estimated force at those 13 locations. Fig. 10(b) shows examples of average signal change ratio, $\overline{r(t)}$, when a 500g force is applied to the 13 calibrated locations and the estimated $\overline{r(t)}$ over the remaining parts of the phone. As shown in this figure, $\overline{r(t)}$ varies with location due to different distances

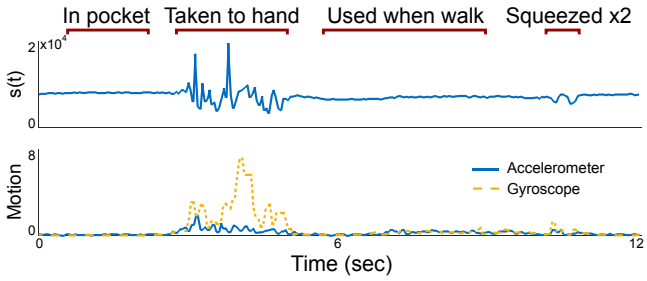
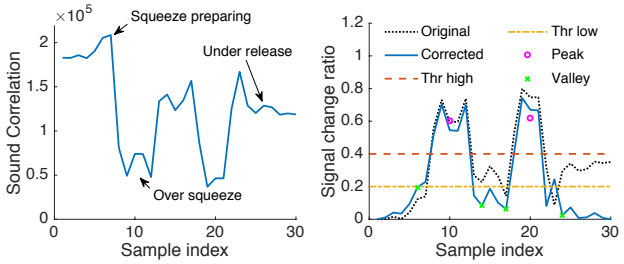


Figure 11—Response of movement and squeeze. Sound correlation changes when the environment changes, such as moving the phone from the pocket to hands, but it becomes stable quickly when people hold phones in their hands.



(a) Squeeze response (b) Squeeze detection

Figure 12—Squeeze detection example. Received signal is first normalized by the start and the end of signal amplitudes. Peak is identified when the corrected signal passing the high threshold and the signal above the low threshold is considered as part of the peak.

from/to the speaker/microphone and also with the structure of the phone. In general, touching near the speaker used to play sounds generates more pronounced signal changes while touching locations far away from the speaker causes less pronounced changes. Note that our current calibration is done when the phone is placed on a static surface, thus ignoring the airborne signal changes due to the movement of the phone during the touch. However, as we will discuss later, the added estimation noise due to the phone movement can be tolerated with proper app design since users are also unaware of the actual amount of force being applied to the phone unless the response from the phone is sensed. Most participants of our usability study felt comfortable with the current setting. Improving this calibration process with more advanced sensors and algorithms is part of our future work.

4.3 Squeeze Detection

It is challenging to detect the squeeze of phone body. Fig. 11 shows an example of the observed signal change when a phone is taken out of the pocket with hand, used while walking, and then squeezed twice. As shown in this figure, even though the signal response of a squeeze is clearly observable, it is also possible to include lots of noise due to the phone's movements. To avoid this noise from large phone movements, we have built a motion detector based on both accelerometer and gyroscope readings, which turns off our squeeze detection when there is a large phone movement and the signal is noisy. We thus set a threshold to turn off the detection during large phone movements (such as taking the phone out of pocket and transferring it to a hand) but keep the squeeze function on during slow/small phone movements, such as walking and using the phone in hand.

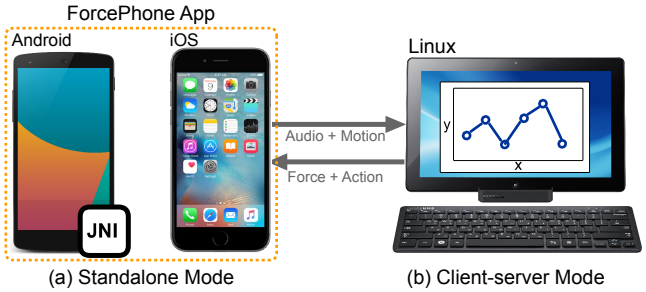


Figure 13—Implementation overview. ForcePhone has been implemented as a standalone app on Android via Java Native Interface (JNI). Our iOS implementation requires the force estimation done at a remote server.

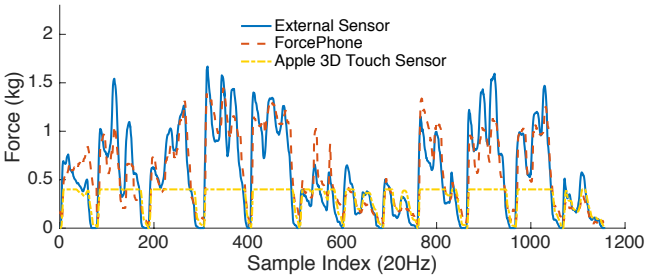


Figure 14—Accuracy of force estimation. 12 touch events with different amounts of applied force are plotted. The force estimated by ForcePhone provides high correlation with the ground truth estimated by using our external force sensors.

Even though the movement noise is removed by the motion detector, determining if the user squeezes the phone is still harder than estimating the force applied to the touch screen. Due to the lack of touch screen input, we don't know when the squeeze starts. Moreover, the location of squeeze is inaccessible, thus making it difficult to perform a proper calibration. To solve these issues, our squeezable back function is made to respond only to a predefined squeeze behavior, such as double squeezes applied to the phone's left and right body frame as shown in Fig. 2, and the entire squeeze process is assumed to complete in 1.5 seconds.

Fig. 12(a) shows an example of this predefined squeeze response, where the applied squeezes cause two significant signal drops at time 10 and 20. Besides the signal response to the squeeze operation, there are three characteristics of human squeezing behavior, which are critical to our design of squeeze detection. First, the the applied force might decrease slightly before a squeeze, probably because the users need to relax their hands before squeezing their phone. After the squeeze, the applied force may also decrease around the expected force peak. We call this phenomenon *over-squeeze*, and think it is caused by the fact that users actually release part of their fingers/palms from the phone body when they try to apply more force with the other part(s) of the hand. At the end of squeeze, the applied force may not return to its initial condition either, because the users may change their position of holding the phone during the squeeze.

Squeezes are also detected based on the signal change ratio $r(t)$. ForcePhone continuously checks each 30-sample received signal to see if the predefined squeeze pattern has occurred. As the hand position is likely to change during a squeeze, it is less accurate to estimate the applied force based only on s_{start}. For example, the larger drop of the second squeeze shown in Fig. 12(a) might be caused by a hand position change rather than a larger squeezing force. To account for this phenomenon, we heuristically set the reference signal based on both the start and end of this 30-sample signal. This new reference signal varies over time and is defined as $s(t)_{reference} = ((t_{end} - t)s_{start} + (t - t_{start})s_{end})/(t_{end} - t_{start})$. The signal change ratio $r(t)$ is calculated based on $s(t)_{reference}$ using the same process as described earlier. Basically, this method estimates $r(t)$ by weighting the reference signal proportional to the difference between t and t_{start}, t_{end}. Fig. 12(b) shows the result of this correction. After estimating $r(t)$, ForcePhone sets two thresholds to identify the force peaks caused by squeezes. A peak occurs when $r(t) > thr_{high}$. The following samples where $r(t) > thr_{low}$ are defined as parts of this identified peak. This setting is designed to avoid false identification due to the *over squeeze* and *squeeze preparation*. Thus, thr_{high} is set larger than the assumed squeeze preparation force change, and thr_{low} is set smaller than the over squeeze force change. A valid squeeze is identified when the number of detected peaks, the peak widths, and the peak-to-valley ratios fit a predefined criterion. The performance of our current setting will be evaluated in Section 6.

5. Implementation

We have implemented ForcePhone on both Android and iOS devices. As shown in Fig. 7, our Android implementation runs as a standalone app; the force estimation is implemented as a library in native code (C++) which we integrate into the app via Java Native Interface (JNI). On the other hand, the current iOS implementation requires the force estimation to be done at a remote server using Matlab. Both implementations provide real-time force estimations. The delay between recording each chirp and receiving a force estimation is 15ms and 61ms at our local and remote implementations, respectively. We are currently porting the force estimation library implemented in C++ to iOS.

One implementation challenge is that ForcePhone needs accurate time synchronization between the phone speaker and microphone to extract valid structure-borne propagations. However, both iOS and Android platforms don't meet this real-time requirement since both take 20–100ms to run the play/record audio command. To overcome this difficulty, ForcePhone adds 10 pilot sequences ahead of the used chirps. In this synchronization, ForcePhone checks if the received signal contains the identical sent pilots, such as having the same stop time among pilots. In our experiments, more than 95% of the received pilots have less than 5-sample jitters in their stop time, and 15% of trials on Android experience a special 960-sample delay before the 4-th pilots. The criterion for ForcePhone to finish this synchronization test is the existence of these 10 pilots, and 50% of received pilots have the exact same stop time, and the last three pilots have less than 5-sample jitters. Once the pilots are identified correctly, the timing of the last pilot is used as a reference to extract the structure-borne propagation from the subsequent chirps as described earlier.

6. Evaluation

We first measure system performance, such as force estimation accuracy, overhead, and robustness to noise. We then measure the user's benefit of using ForcePhone, such as the probability of an assigned request being honored or whether the users think it useful.

6.1 Accuracy of Force Estimation

We evaluate the accuracy of force estimation by using the mean square errors and correlation coefficients where the former represents the ability to estimate exact force while the later indicates the capability to learn how force is changed. Fig. 14 shows an example of 12 different touches when the iPhone 6s is held in left hand and the center of the touch screen is pressed with left thumb. The applied force is estimated by the internal Apple 3D Touch sensors, external force sensors, and ForcePhone. Since Apple 3D Touch can only provide normalized results and no documented interpretation of the sensed values is available, we built our own post-hoc translation based on an external force scale. We found the sensed value is linear in the applied force, and the maximum value is reached when 380g is applied.

As shown in Fig. 14, the proprietary sensors used in Apple 3D Touch captures the force change below its maximum (380g in this case) with high accuracy. Even though ForcePhone estimates the force under 380g with less accuracy, it captures most force-change characteristics as shown in Fig. 14. Meanwhile, ForcePhone can estimate the force above 380g without using any additional sensors.

The mean square error of this in-hand example is 205g and the correlation coefficient to the force estimated by external sensors is 0.87. Compared to the maximum sensed force up to 1.5kg, this error is tolerable for most force-sensitive apps as shown later. When the phone is stationary on a wooden table, the mean square error falls to 54g and the correlation coefficient increases to 0.91. Note that the error in estimating the exact value of the applied force could change if the force is applied in a different way than our calibration (as we don't consider the damping coefficient change in our current model as introduced in Section 3.) For example, there might be an estimation drift, so applying a varying force from 500g to 1000g might be estimated wrongly as changing from 400g to 900g plus the previously-mentioned errors. However, this feature doesn't hurt the experience of using ForcePhone because users are not aware of the exact value of the force applied to a phone unless this value is shown in the user interface [36]. With a proper user interface design, even though the estimated force is 100g less than the real force applied to the phone, users can easily learn to adjust the applied force for getting a correct response. Actually, even the proprietary sensors used for 3D Touch have about 100g errors between the corners of touch screen, and it is suggested not to be used for estimating exact force [1]. In our current setting, when the force is applied to the middle area of the phone, the estimated force (in g) is calculated as $f(t) = 15 + 230\sqrt{(r(t))} + 2800r(t)$ in iPhone 6s and $f(t) = 1900r(t)$ on Galaxy Note 4. As shown in our usability study, this setting is adequate for building useful force-sensitive and squeezable apps. Calibration while considering how users touch the phone is part of our future work, which could be accomplished by other grip detection systems like GripSense [15].

6.2 Noise and Interference

The estimation of force based on sound propagation becomes erroneous if there is a strong audio noise with a similar structure as the chirps used in ForcePhone. For example, the mean square error of ForcePhone can increase from 50g to more than 500g when the same chirps are played by nearby computers's speakers. Even tough this is the natural limitation of any audio-based application, it is not the case in most real-life scenarios. In our measurements, ForcePhone is shown to be resistant to: (1) real-life background audio noise such as music being played in the testing room,

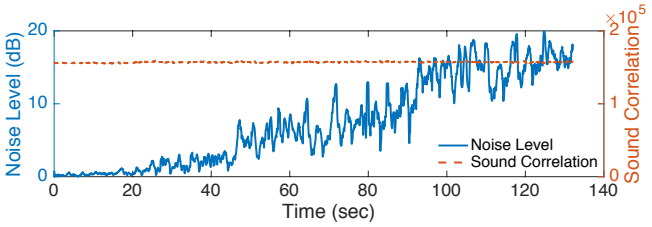
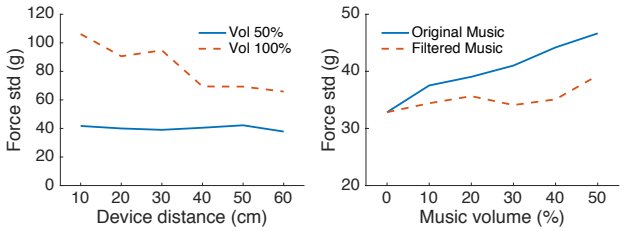


Figure 15—Resistance to background noise. Music (i.e., noise) played by a laptop 20cm away from the device under test has limited effect on the sound correlation even if the noise level is increased to 20dB higher than the used chirps.



(a) Inter-device interference (b) Self interference

Figure 16—Resistance to inter-device and self interferences. The variation of sound correlation for each second is used to indicate the error when another device is running ForcePhone or a music is played on the same device.

(2) inter-device interference from ForcePhone running on another nearby device, and (3) self-interference from the audio played on the same device. To study the performance degradation caused only by the audio noise, rather than the force instability of human hands or other movement noises, we kept the test phone stationary on a wooden table and no force is applied. The standard deviation of force estimations is recorded and reported. This estimated force variation also represents errors when there is a force applied to the phone if the received noise is modeled as an additive noise.

Particularly, we use an Apple MAC Air or an iPhone 6s placed on the same surface as test devices. Multiple sets of noises were played, but no significant differences were observed, so we only report the results of music-playing noise, i.e., rock music "Jump – Van Halen" or gaming background music "Spot On". Since we place both the test device and the noise source on the same table, the results account for the vibration coupling and the noise transmitted through the surface.

We found ForcePhone to be resistant to most external real-life noises. As shown in Fig. 15, even when the rock music is played at full-volume by a laptop 20cm away from the test device, the sound correlation of the used chirps doesn't change much. This is because most human (singing or chatting) noises have limited signals in the 18kHz – 24kHz range. Moreover, this natural noise usually has only a limited correlation to our 1200-sample chirps. In this measurement, the increased estimation errors are less than 10g, which does not affect most of ForcePhone's functionality, even when the noise level is 20dB higher than the used chirps.

Compared to the uncorrelated noises, ForcePhone is more sensitive to the noise that has a similar structure as the chirp it uses. Solutions of this issue are critical in allowing multiple phones to run ForcePhone in the same environment. To measure this, we had another device playing the same chirps used by ForcePhone, but with random stop time between consecutive chirps. This is

Setting	Idle	Backlight	ForcePhone	Website	Game
Power (mean)	33	856	1160	2564	3692

Table 2—Power consumption (mW). The additional power consumption by ForcePhone is about 304mW, which is small relative to that of normal phone usage.

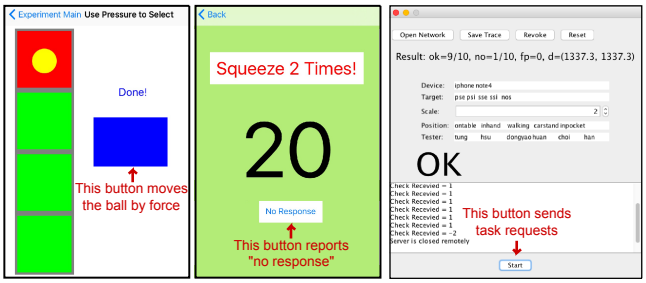
designed to learn the average performance when two devices run ForcePhone simultaneously. As shown in Fig. 16(a), the average estimated error increases to 110g when the correlated noise is played at full-volume by another device. This trend is mitigated when the noise volume is reduced. Since ForcePhone only requires the designed chirps to be played at 50% of full-volume, multiple devices can run ForcePhone if they are at least 10cm apart from each other.

In the last scenario, we evaluate whether other audio signals can be played on the same device when ForcePhone is running. This capability is critical to the user experience, especially when using ForcePhone for a game control. We play a game background music from the same device when ForcePhone is running, and record the increased estimation error. As shown in Fig. 16(b), when the music is played by the same speaker at 50% of full-volume, ForcePhone experiences about 46g of additional estimation error, which does not affect the core functionality of ForcePhone. Moreover, when the played music is processed further by using a 15kHz low-pass filter, the imposed error decreased further to 37g. Note that the current design of ForcePhone allows game music to be played at most 50% of full-volume, as the other 50% is dedicated for ForcePhone's functionality. This limitation can be addressed when the phone development kit supports streaming music/chirps to different speakers (devices usually have 2 or more speakers, but only one of them can be used at a time).

6.3 Power Consumption

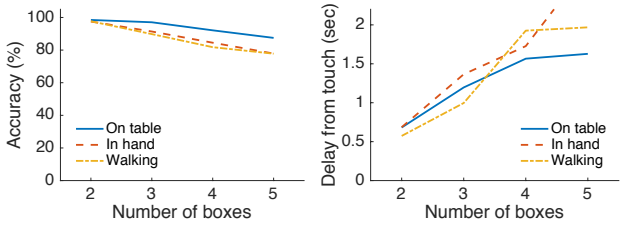
We measure the power consumption of ForcePhone with the Monsoon Power Monitor [6] on Galaxy Note 4. For the baseline measurements, we turn off the phone's radios and estimate the force through our JNI/C++ local implementation. We ran each scenario for 20 seconds in 6 different states: 1) idle, with the screen off, 2) backlight, with the screen displaying a white background, 3) ForcePhone, with the screen on, 4) website surfing, when networking is enabled, and 5) video game, "Puzzle&Dragon". These measurement results are summarized in Table 2.

Activating the naive implementation of ForcePhone consumes an additional power of 304mW compared to the case when nothing is executed and background is on. More than 90% of this power consumption comes from using the hardware of speaker, microphone, and accelerometer — this is consistent with the results in [10]. The computation cost of ForcePhone is minimal since it builds a model to estimate the applied force rather than fingerprinting the sound changes. Note that the ratio of ForcePhone's power consumption is dependent on its usage. For example, though ForcePhone incurs 26% power overhead (compared to the blacklight case) when it is used to select app options at home screen, users won't press this option for a long time and ForcePhone can be turned off once the selection is done. If the proposed squeezable back function is used for web surfing, activating ForcePhone incurs only an 11% power-consumption overhead, which reduces the life of Note 4's 12 Wh battery by about 30min. The percentage overhead is reduced further if ForcePhone is used for game control or when microphones or speakers have been activated for other purposes.



(a) Experiment App (b) Experiment Controller

Figure 17—User interface for experiments. Users are requested to move a ball to the marked red box by applying different amounts of force to the blue button and squeeze the phone twice for surfing the previous web page. Action requests are sent, and the results are recorded by the controller.



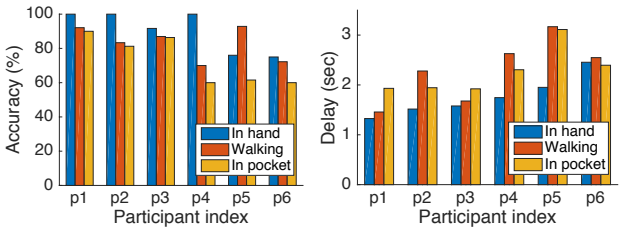
(a) Accuracy (b) Delay

Figure 18—Result of controlling a ball with ForcePhone. Results are averaged over 6 participants. Delay is estimated as the time between the user presses/releases the button.

6.4 Usability Test

We recruited 6 participants (4 males and 2 females) to test the usability of ForcePhone by asking them to perform a set of tasks as shown in Fig. 17(a). Since these 6 participants are students who are aware of this project, we only tested their ability to complete the assigned tasks. (The results of the other users' opinions on ForcePhone will be provided in the next subsection.) In the first task, users are asked to move a ball by applying force and stop it at a randomly-selected red box; the highest box is reached by applying a 500g force and the ball located at the bottom of boxes (0g) when the task starts. This experiment follows a similar design principle as that in [36] to test if the user can effectively control the force at different levels. Experiments start from stopping the ball at one of the two big boxes (i.e., only two levels) to one of the five small boxes. We asked users to perform these tasks at different positions, such as controlling the phone while it is on a table, while holding it in hand, or while walking and using it simultaneously. A remote controller sends requests and records the accuracy and delay of users' reaction as shown in Fig. 17(b).

Fig. 18 shows the accuracy (the success rate of moving the ball into the selected box) and the delay between the user touching the blue button and finishing the task. The plots show that most participants can effectively control ball movement with ForcePhone when there are only 2 – 3 boxes. This result supports our hard-pressed option app since the users can easily control the applied force at two levels with higher than 97% accuracy. When more than 3 boxes (levels) are provided, users can still achieve higher than 90% accuracy when the phone is stationary on the table. This is consistent with our earlier evaluation of estimation accuracy. Although different touch positions might incur additional errors, the



(a) Accuracy (b) Delay

Figure 19—Results of squeeze detection. The accuracy of the last three participants is increased to more than 90% when the detection criterion is adjusted after this test.

Questions	Strongly disagree /Disagree	No option	Strongly agree /Agree
Hard-pressed option is helpful	0	0	21
Hard-pressing(3D Touch) is responsive	1	1	19
Hard-pressing is responsive	0	1	20
Ball-moving game is interesting	0	3	18
Moving ball is responsive	2	4	15
Squeezable back is helpful	1	2	18
Squeezing is responsive	2	1	18
False detection is acceptable	4	7	10

Table 3—App study results.

high correlation between the force estimated by ForcePhone and the real force is sufficient for users to control the phone effectively. It is worth noting that moving the ball toward the top/bottom boxes is easier than the box in the middle; moving the ball to the bottom or top box has 13% higher accuracy than moving the ball into the middle box. A similar property was also reported in [36].

On average, users require only 0.7 second to stop the ball at the correct position when there are two boxes. This delay increases to 2 seconds when the users attempt moving the ball among 5 boxes while walking. Interestingly, controlling the ball when the phone is stationary is considered the easiest task, but the participants use a slightly less time to complete the 2-box task while walking. We conjecture this phenomenon is caused by the order of tests we perform, where the walking test is executed after testing the phone–table and in-hand cases. This property implies that performance can be improved further once the users become familiar with ForcePhone. The average accuracy of repeating the same task on the Galaxy Note 4 is increased by 3%. We think this minimal improvement is due to the users' familiarity with this task rather than the force estimation accuracy. In summary, users can effectively use ForcePhone to control different levels of applied force for various scenarios.

In the second experiment, we asked the participants to squeeze the phone's body. Once ForcePhone detects the squeeze, the testing app switches to the previous UI page with a different background and indicates the page number as shown in Fig. 17(a). Due to the lack of ground truth (no touch event exists), we asked the users to click the *no response* button once the user observes the applied squeeze not detected. We count the number of times the squeeze is detected without any instruction from the remote controller as a false positive.

Fig. 19 shows the accuracy and delay for each user. Participant 1 has the highest accuracy and the lowest delay because the detection setting is tuned according to his own preliminary test and since he practiced this task more than the other users. The average accuracy when the phone is held in hand is 90%. The accuracy drops to 82% when the participants use their phones while walking. Due to the lack of timing information on when the user applies and stops the squeeze, we only record the delay between the instruction and the squeeze feedback (either success or fail). As shown in Fig. 19(b), the average delay of each squeeze is about 2 seconds. Users experience low accuracy usually experience a long delay since they have to wait and then click the "no response" button if the squeeze is not detected. From the measurements of our previous moving-ball test, we found users usually need 0.8 second to react to the displayed instruction, and hence the delay of squeeze detection is expected no longer than 1.5 seconds.

It is possible to detect a squeeze in users' pocket, which can be used to turn off alarms or notifications when the phone is in pocket. However, our measurement shows that its performance depends not only on how the phone is squeezed but also on the clothing material. Tight jeans (e.g., worn by p1 to p3) generally have a better detection accuracy than loose pants. During this controlled experiment, only 4 false detections are observed. Users' feedback on the false positive rate will be presented in the next section.

Based on more than 600 squeeze profiles collected from the 6 participants, we tuned our final squeeze detection setting as follows. We set the high and low thresholds of squeeze detection to 310g and 175g, respectively. We identify signals as a peak only when the peak width lies between 2 and 8 samples and the peak-to-valley ratio is greater than 2.5. By applying these new criteria to the collected traces, the overall detection accuracies for the last three participants increase to higher than 90% for both in-hand and walking scenarios. We used the same setting in our subsequent application tests for other participants.

6.5 Users Study of Proposed Apps

For a users study of ForcePhone-based apps, we randomly chose 21 participants (7 females and 14 males) among a large student population at the University of Michigan. All of recruited participants own smartphones; only 3 of them have the latest iPhone 6s and knew how to use 3D Touch prior to this test. In contrast to our previous usability test, these participants were unaware of ForcePhone and our research group (so they have no initial bias). We first ask the participants to use the built-in hard-pressed option of iPhone 6s and then try the same function implemented by ForcePhone on Galaxy Note 4. We built a fake UI (as shown in Fig. 2(a)) to make them feel if they were triggering the real option on Android. We set the testing threshold for enabling options to 340g, which is similar to the iPhone's. We chose not to make a blind test by using our iPhone implementation because the current iOS disables the vibration service when audio is being recorded. According to our preliminary test, vibrating the phone when the hard-pressed option is being activated is critical to the user experience. After trying the hard-pressed option, the users were instructed to play a simple game with the moving-ball test app. This allows us to record the users' feedback on using ForcePhone as a continuous UI input. While playing the game, users have to move the ball into the red box, and the number of boxes was increased when the users completed the task. Last, we asked the participants to test our squeezable back app that automatically navigates the previous UI page when they squeeze the phone twice. The threshold is set based on the results of our previous usability study. It took about 15 minutes on average, to complete the entire test.

After testing our app, a survey form was filled out by the participants, and the results are summarized in Table 3. The ques-

tion mark by "3D Touch" in Table 3 refers to the hard-pressed option implemented by Apple 3D Touch, while the others refer to ForcePhone. Most users are positive of the proposed apps and think them helpful. For the hard-press option, most users think ForcePhone has a comparable performance to Apple 3D Touch. One of them said ForcePhone is better than 3D Touch because the vibration in Android is much clearer (stronger) than iPhone's, which is not related to the force detection. Only 3 users think iPhone's performance is better, but still acknowledge ForcePhone is responsive enough for the hard-pressed option app. This indicates that ForcePhone can handle simple tasks with a comparable performance as adding proprietary sensors. Moreover, most users feel the squeezable back app is helpful, which is a unique capability of ForcePhone.

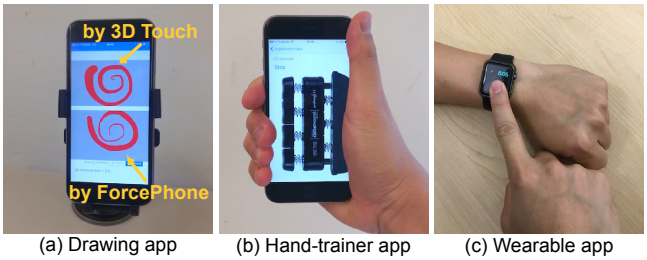
Most users regard that controlling the ball based on the applied force is relatively difficult, but still were able to control the ball. Two users think our test setting is too sensitive, making it difficult to move the ball. We also discovered some errors caused by applying a large initial force and releasing the button immediately, which was not the intended case for ForcePhone. After the users are instructed to move the ball by gradually applying force, they are able to control the ball with ForcePhone. The squeezable back app received a similar rating as the ball-moving game. In our other survey, 16 users indicated difficulty in clicking the app back button when operating the phone with one hand which supports the design of our squeezable back app. Most users think our current parameter setting tuned by previous 6 participants is responsive and acceptable. A half of users experienced false detections when they moved the phone from one hand to the other. During the test, none of users heard the sound used in ForcePhone, so no user annoyance. The test locations were close to a cafe crowded with students, but ForcePhone was robust to human noises. Most common comments from the users are "cool idea" and "useful'. We plan to have extensive and large-scale tests before releasing ForcePhone to public.

7. Discussion

ForcePhone has been shown to be able to expand user input interfaces by using only built-in sensors. Several demonstrative apps have been developed and tested, but there are many other use-cases of ForcePhone. Discussed below are the limitations of current ForcePhone and some of possible directions of our future work.

7.1 Limitations

As our evaluation results show, ForcePhone's force estimation includes noise from object contacts and human movements, calling for an app design to eliminate this noise. An extreme example that breaks ForcePhone's functionality is to activate the force estimation by placing the phone on a table then quickly moving and holding it in hand. However, this limitation of ForcePhone can be avoided by a proper app design. For example, the hard-pressed option app only needs force measurements within a short period of time (e.g., 2 seconds) and is inactivated if the phone is being moved. Apps like drawing lines with different line styles/thickness by applying different amounts of force is not suitable for the current ForcePhone as it is more probable for the user to (unintentionally) change the environment during the line drawing. But selection of thickness by applying different amounts of force is possible since it is akin to our ball-moving test. In our measurement, to achieve accurate and long-lasting force sensing, ForcePhone needs the phone to be fixed at a certain location. For example, Fig. 20(a) demonstrates the above-mentioned force-sensitive drawing app by attaching the phone on a stand.



(a) Drawing app (b) Hand-trainer app (c) Wearable app

Figure 20—Potential usage of ForcePhone.

In our squeezable back app, missed detections occur when the detection is intentionally turned off upon identification of extensive movements, such as sudden turns or stops. This is a safety mechanism for ForcePhone to avoid false detections, but the users are not aware of this. One way to avoid this misunderstanding and improve the user experience is to provide a feedback when the squeeze detection is temporarily turned off. For example, the background of website or the title bar can be dimmed slightly when the squeezable back app is deactivated. Studying the users' reaction to this new UI design is part of our future work.

While the touch location near the speaker/microphone used to play/record sounds is found more sensitive to the applied force (i.e., making more pronounced signal changes), the middle area of touch screen usually provides more reliable force estimation. For example, when more than 1.2kg force is applied to the top area of iPhone 6s, the received signals jump quickly and the relation between $f(t)$ and $r(t)$ breaks down. We suspect this phenomenon is due to the pressuring of phone's microphone and other internal components. So, we suggest to limit the estimated force to 500g for consistent performance, even though certain locations can sense the force up to 3kg. This phenomenon also causes the estimated signal strength $s(t)$ at certain locations to rise when the force is applied. Most of these situations caused by pressing the phone can be compensated by our calibration of $r(t)$ and the estimation constraints caused by this problem will likely be addressed in the near future since commodity phones start to increase the number and fidelity of speakers/microphones. For example, when multiple speaker–microphone pairs are used to sense force interactively, the locations yielding erroneous estimations for a single microphone–speaker pair can be corrected by the other microphones and speakers. We should also note that, as shown in our measurements, these minor issues incur minimal annoyance to users when ForcePhone is properly applied.

7.2 Potential Applications of ForcePhone

The participants of our usability study suggested many potential uses of ForcePhone after trying the proposed apps. Of them, two most promising and interesting uses are mobile health apps and force-sensitive wearable devices.

Mobile phones or wearables are good candidate platforms for mobile health apps since people carry them all time. For example, systems of notifying/requesting users to walk or excise are now built in most commodity phones [2, 8]. Researchers have also shown the benefits of drinking more water via mobile apps [11]. ForcePhone can provide a new function/capability of these systems because it can determine how hard a user squeezes/touches the phone. As shown in Fig. 20(b), this functionality can be used

as a replacement of force-finger-trainer that is helpful for the people with disability and those who use hands and fingers excessively for their work, e.g., computer programmers.

ForcePhone can also be implemented in wearable devices that have even less space for user inputs than smartphones. Unfortunately, current wearable devices usually have less sensing capabilities than smartphones, especially for the microphone sensitivity and sample rate. For example, Samsung Gear S (watch) can only support an audio sample rate up to 32kHz, which is inadequate for the current design of ForcePhone. Apple Watch hardware is found to be the best to implement ForcePhone, but its current API does not yet support data reading from the audio queue and process audio data in real time. In future, we expect the sensing capability of wearables to improve so that ForcePhone can be implemented and used for numerous apps.

8. CONCLUSION

We have proposed ForcePhone, an inexpensive solution that adds a force-sensitive and squeezable interface to commodity phones without any hardware modification/addition. ForcePhone has been implemented on iOS and Android platforms, and several apps based on its functionality have been developed and tested. Our evaluation has shown ForcePhone to provide comparable performance as augmented proprietary force sensors and to be robust to most real-life noises. The proposed apps are easy to use with higher than 90% accuracy and minimal overheads. In future, we plan to test ForcePhone on a wide range of devices and scenarios.

ACKNOWLEDGEMENTS

The authors would like to thank Karl Grosh, Chuming Zhao, Kassem Fawaz, Lung-Pan Cheng, Chuan-Che Huang, Ming-Yuan Yu, Chen-Ming Chang, Meng-Ting Chung and the anonymous reviewers and the shepherd for constructive comments on the earlier versions of this paper.

9. REFERENCES

[1] Apple doesn't want you weighing things with your iPhone just yet. http://www.theverge.com/2015/10/28/9625340/iphone-6s-gravity-app-digital-scales.

[2] Apple Health App. http://www.apple.com/ios/health/.

[3] Apple iPhone 6s 3D Touch. http://www.apple.com/iphone-6s/3d-touch/.

[4] ForcePhone Demo Video. https://youtu.be/cYxr2wnQVMU.

[5] Interlink 402 FSR. http://www.interlinkelectronics.com/FSR402.php.

[6] Monsoon Power Monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.

[7] Polytec OFV-303 Laser Vibrometer. http://www.polytec.com/us/products/vibration-sensors/.

[8] Samsung S Health App. http://shealth.samsung.com/.

[9] Why Apple's iPhone SE lacks 3D Touch technology. http://appleinsider.com/articles/16/03/23/why-apples-iphone-se-lacks-3d-touch-technology.

[10] F. Ben Abdesslem, A. Phillips, and T. Henderson. Less is more: Energy-efficient mobile sensing with senseless. In *Proceedings of ACM MobiHeld '09*, pages 61–62.

[11] M.-C. Chiu, S.-P. Chang, Y.-C. Chang, H.-H. Chu, C. C.-H. Chen, F.-H. Hsiao, and J.-C. Ko. Playful bottle: A mobile social persuasion system to motivate healthy water intake. In *Proceedings of ACM UbiComp '09*, pages 185–194.

[12] S. Elliott. Active control of structure-borne noise. *Journal of Sound and Vibration*, 177(5):651 – 673, 1994.

[13] A. Girouard, J. Lo, M. Riyadh, F. Daliri, A. K. Eady, and J. Pasquero. One-handed bend interactions with deformable smartphones. In *Proceedings of ACM CHI '15*, pages 1509–1518, 2015.

[14] M. Goel, B. Lee, M. T. Islam Aumi, S. Patel, G. Borriello, S. Hibino, and B. Begole. Surfacelink: Using inertial and acoustic sensing to enable multi-device interaction on a surface. In *Proceedings of ACM CHI '14*, pages 1387–1396.

[15] M. Goel, J. Wobbrock, and S. Patel. Gripsense: Using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of ACM UIST '12*, pages 545–554.

[16] F. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[17] C. Harrison, J. Schwarz, and S. E. Hudson. Tapsense: Enhancing finger interaction on touch surfaces. In *Proceedings of ACM UIST '11*, pages 627–636.

[18] C. Harrison, D. Tan, and D. Morris. Skinput: Appropriating the body as an input surface. In *Proceedings of ACM CHI '10*, pages 453–462.

[19] S. Heo and G. Lee. Forcetap: Extending the input vocabulary of mobile touch screens by adding tap gestures. In *Proceedings of ACM MobileHCI '11*, pages 113–122.

[20] R. Hooke and J. Yonge. *Lectures de Potentia Restitutiva, Or of Spring Explaining the Power of Springing Bodies...* John Martyn, 1931.

[21] S. Hwang, A. Bianchi, and K.-y. Wohn. Vibpress: Estimating pressure input using vibration absorption on mobile devices. In *Proceedings of ACM MobileHCI '13*, pages 31–34, 2013.

[22] S. Hwang and K.-y. Wohn. Pseudobutton: Enabling pressure-sensitive interaction by repurposing microphone on mobile device. In *ACM CHI '12 Extended Abstracts*, pages 1565–1570.

[23] Y.-H. Kim. Sound propagation: An impedance based approach. Wiley, 2010.

[24] G. Laput, E. Brockmeyer, M. Mahler, S. E. Hudson, and C. Harrison. Acoustruments: Passive, acoustically-driven, interactive controls for handheld devices. In *Proceedings ACM CHI '15*.

[25] P. Lazik and A. Rowe. Indoor pseudo-ranging of mobile devices using ultrasonic chirps. In *Proceedings of ACM SenSys '12*, pages 391–392.

[26] Q. Li, J. long Han, and D. jun Wu. Survey on predicting and controlling of structure-borne noise from rail transit bridges. In *Electric Technology and Civil Engineering (ICETCE), 2011 International Conference on*, pages 4559–4563.

[27] S. Low, Y. Sugiura, D. Lo, and M. Inami. Pressure detection on mobile phone by camera and flash. In *Proceedings of ACM AH '14*, pages 11:1–11:4.

[28] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones. In *Proceedings of ACM MobiSys '09*, pages 165–178.

[29] P. Marti and I. Iacono. Evaluating the experience of use of a squeezable interface. In *Proceedings of CHItaly '15*, pages

42–49.

[30] R. Nandakumar, S. Gollakota, and N. Watson. Contactless sleep apnea detection on smartphones. In *Proceedings of ACM MobiSys '15*, pages 45–57.

[31] S. Nirjon, R. F. Dickerson, P. Asare, Q. Li, D. Hong, J. A. Stankovic, P. Hu, G. Shen, and X. Jiang. Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones. In *Proceeding of ACM MobiSys '13*, pages 403–416.

[32] M. Ono, B. Shizuki, and J. Tanaka. Sensing touch force using active acoustic sensing. In *Proceedings of ACM TEI '15*, pages 355–358.

[33] M. Ono, B. Shizuki, and J. Tanaka. Touch & activate: Adding interactivity to existing objects using active acoustic sensing. In *Proceedings of ACM UIST '13*, pages 31–40.

[34] E. W. Pedersen and K. Hornbæk. Expressive touch: Studying tapping force on tabletops. In *Proceedings of ACM CHI '14*, pages 421–430.

[35] C. J. Plack. The sense of hearing. Lawrence Erlbaum Associates, Inc., 2005.

[36] G. Ramos, M. Boulos, and R. Balakrishnan. Pressure widgets. In *Proceedings of CHI '04*, pages 487–494.

[37] S. Salemian, M. Jamshihi, and A. Rafiee. Radar pulse compression techniques. In *Proceedings of WSEAS AEE'05*, pages 203–209.

[38] Y.-C. Tung and K. G. Shin. Echotag: Accurate infrastructure-free indoor location tagging with smartphones. In *Proceedings of ACM MobiCom '15*, pages 525–536.

[39] H. Zheng, J. Mou, W. Lin, and E. Ong. Modeling and prediction of structure-borne seek noise of hard disk drives. *Magnetics, IEEE Transactions on*, 45(11):4933–4936, 2009.