VSkin: Sensing Touch Gestures on Surfaces of Mobile Devices Using Acoustic Signals

Ke Sun
State Key Laboratory for Novel Software Technology
Nanjing University, China, kesun@smail.nju.edu.cn

Ting Zhao
State Key Laboratory for Novel Software Technology
Nanjing University, zhaoting@smail.nju.edu.cn

Wei Wang
State Key Laboratory for Novel Software Technology
Nanjing University, China, ww@nju.edu.cn

Lei Xie
State Key Laboratory for Novel Software Technology
Nanjing University, China, lxie@nju.edu.cn

ABSTRACT

Enabling touch gesture sensing on all surfaces of the mobile device, not limited to the touchscreen area, leads to new user interaction experiences. In this paper, we propose VSkin, a system that supports fine-grained gesture-sensing on the back of mobile devices based on acoustic signals. VSkin utilizes both the structure-borne sounds, *i.e.*, sounds propagating through the structure of the device, and the air-borne sounds, *i.e.*, sounds propagating through the air, to sense finger tapping and movements. By measuring both the amplitude and the phase of each path of sound signals, VSkin detects tapping events with an accuracy of 99.65% and captures finger movements with an accuracy of 3.59 *mm*.

CCS CONCEPTS

• **Human-centered computing → Interface design prototyping**; **Gestural input**;

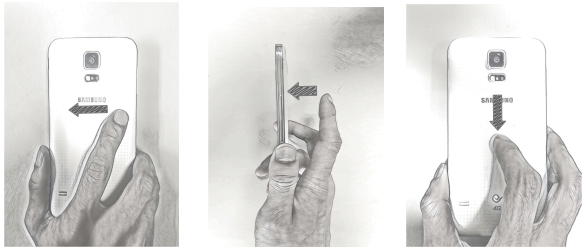
KEYWORDS

Touch gestures; Ultrasound

ACM Reference Format:
Ke Sun, Ting Zhao, Wei Wang, and Lei Xie. 2018. VSkin: Sensing Touch Gestures on Surfaces of Mobile Devices Using Acoustic Signals. In *MobiCom '18: 24th Annual International Conference on Mobile Computing and Networking, October 29–November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3241539.3241568

MobiCom'18, October 29–November 2, 2018, New Delhi, India

ACM ISBN 978-1-4503-5903-0/18/10…$15.00
https://doi.org/10.1145/3241539.3241568



(a) Back-Swiping (b) Back-Tapping (c) Back-Scrolling

Figure 1: Back-of-Device interactions

1 INTRODUCTION

Touch gesture is one of the most important ways for users to interact with mobile devices. With the wide-deployment of touchscreens, a set of user-friendly touch gestures, such as swiping, tapping, and scrolling, have become the *de facto* standard user interface for mobile devices. However, due to the high-cost of the touchscreen hardware, gesture-sensing is usually limited to the front surface of the device. Furthermore, touchscreens combine the function of gesture-sensing with the function of displaying. This leads to the occlusion problem [30], *i.e.*, user fingers often block the content displayed on the screen during the interaction process.

Enabling gesture-sensing on all surfaces of the mobile device, not limited to the touchscreen area, leads to new user interaction experiences. First, new touch gestures solve the occlusion problem of the touchscreen. For example, Back-of-Device (BoD) gestures use tapping or swiping on the back of a smartphone as a supplementary input interface [22, 35]. As shown in Figure 1, the screen is no longer blocked when the back-scrolling gesture is used for scrolling the content. BoD gestures also enrich the user experience of mobile games by allowing players to use the back surface as a touchpad. Second, defining new touch gestures on different surfaces helps the system better understand user intentions. On traditional touchscreens, touching a webpage on the screen could mean that the user wishes to click a hyperlink or the user just wants to scroll down the page. Existing touchscreen schemes

often confuse these two intentions, due to the overloaded actions on gestures that are similar to each other. With the new types of touch gestures performed on different surfaces of the device, these actions can be assigned to distinct gestures, *e.g.*, selecting an item should be performed on the screen while scrolling or switching should be performed on the back or the side of the device. Third, touch sensing on the side of the phone enables virtual side-buttons that could replace physical buttons and improve the waterproof performance of the device. Compared to in-air gestures that also enrich the gesture semantics, touch gestures have a better user experience, due to their accurate touch detection (for confirmation) connected to the useful haptic feedbacks.

Fine-grained gesture movement distance/speed measurements are vital for enabling touch gestures that users are already familiar with, including scrolling and swiping. However, existing accelerometer or structural vibration based touch sensing schemes only recognize coarse-grained activities, such as the tapping events [5, 35]. Extra information on the tapping position or the tapping force levels usually requires intensive training and calibration processes [12, 13, 25] or additional hardware, such as a mirror on the back of the smartphone [31].

In this paper, we propose VSkin, a system that supports fine-grained gesture-sensing on the surfaces of mobile devices based on acoustic signals. Similar to a layer of skin on the surfaces of the mobile device, VSkin can sense both the finger tapping and finger movement distance/direction on the surface of the device. Without modifying the hardware, VSkin utilizes the built-in speakers and microphones to send and receive sound signals for touch-sensing. More specifically, VSkin captures both the structure-borne sounds, *i.e.*, sounds propagating through the structure of the device, and the air-borne sounds, *i.e.*, sounds propagating through the air. As touching the surface can significantly change the structural vibration pattern of the device, the characteristics of structure-borne sounds are reliable features for *touch detection*, *i.e.*, whether the finger contacts the surface or not [12, 13, 25]. While it is difficult to use the structure-borne sounds to sense finger movements, air-borne sounds can measure the movement with mm-level accuracy [14, 28, 34]. Therefore, by analyzing both the structure-borne and the air-borne sounds, it is possible to reliably recognize a rich set of touch gestures as if there is another touchscreen on the back of the phone. Moreover, VSkin does not require intensive training, as it uses the physical properties of the sound propagation to detect touch and measure finger movements.

The key challenge faced by VSkin is to measure both the structure-borne and the air-borne signals with high fidelity while the hand is very close to the mobile device. Given the small form factor of mobile devices, sounds traveling through different mediums and paths arrive at the microphone within a short time interval of 0.13∼0.34 *ms*, which is just 6∼16 sample points at a sampling rate of 48 kHz. With the limited inaudible sound bandwidth (around 6 kHz) available on commercial mobile devices, it is challenging to separate these paths. Moreover, to achieve accurate movement measurement and location independent touch detection, we need to measure both the phase and the magnitude of *each* path. To address this challenge, we design a system that uses the Zadoff-Chu (ZC) sequence to measure different sound paths. With the near-optimal auto-correlation function of the ZC sequence, which has a peak width of 6 samples, we can separate the structure-borne and the air-borne signals when the distance between the speaker and microphone is just 12 *cm*. Furthermore, we develop a new algorithm that measures the phase of each sound path at a rate of 3,000 samples per second. Compared to traditional impulsive signal systems that measure sound paths in a frame by frame manner (with frame rate <170 Hz [14, 34]), the higher sampling rate helps VSkin capture fast swiping and tapping events.

We implement VSkin on commercial smartphones as real-time Android applications. Experimental results show that VSkin achieves a touch detection accuracy of 99.65% and an accuracy of 3.59 *mm* for finger movement distances. Our user study shows that VSkin only slightly increases the movement time used for interaction tasks, *e.g.*, scrolling and swiping, by 34% and 10% when compared to touchscreens.

We made the following contributions in this work:

- We introduce a new approach for touch-sensing on mobile devices by separating the structure-borne and the air-borne sound signals.
- We design an algorithm that performs the phase and magnitude measurement of multiple sound paths at a high sampling rate of 3 kHz.
- We implement our system on the Android platform and perform real-world user studies to verify our design.

2 RELATED WORK

We categorize researches related to VSkin into three classes: Back-of-Device interactions, tapping and force sensing, and sound-based gesture sensing.

Back-of-Device Interactions: Back-of-Device interaction is a popular way to extend the user interface of mobile devices [5, 11, 31, 32, 35]. Gestures performed on the back of the device can be detected by the built-in camera [31, 32] or sensors [5, 35] on the mobile device. LensGesture [32] uses the rear camera to detect finger movements that are performed just above the camera. Back-Mirror [31] uses an additional mirror attached to the rear camera to capture BoD gestures in a larger region. However, due to the limited viewing angle of cameras, these approaches either have limited sensing area or need extra hardware for extending sensing

range. BackTap [35] and βTap[5] use built-in sensors, such as the accelerometer, to sense coarse-grained gestures. However, sensor readings only provide limited information about the gesture, and they cannot quantify the movement speed and distance. Furthermore, accelerometers are sensitive to vibrations caused by hand movements while the user is holding the device. Compared to camera-based and sensor-based schemes, VSkin incurs no additional hardware costs and can perform fine-grained gesture measurements.

Tapping and Force Sensing: Tapping and force applied to the surface can be sensed by different types of sensors [4, 7, 9, 10, 12, 13, 15, 19, 25]. TapSense [7] leverages the tapping sound to recognize whether the user touches the screen with a fingertip or a fist. ForceTap [9] measures the tapping force using the built-in accelerometer. VibWrite [13] and VibSense [12] use the vibration signal instead of the sound signal to sense the tapping position so that the interference in air-borne propagation can be avoided. However, they require pre-trained vibration profiles for tapping localization. ForcePhone [25] uses linear chirp sounds to sense force and touch based on changes in the magnitude of the structure-borne signal. However, fine-grained phase information cannot be measured through chirps and chirps only capture the magnitude of the structure-borne signal at a low sampling rate. In comparison, our system measures both the phase and the magnitude of multiple sound paths with a high sampling rate of 3 kHz so that we can perform robust tap sensing without intensive training.

Sound-based Gesture Sensing: Several sound-based gesture recognition systems have been proposed to recognize in-air gestures [1, 3, 6, 16, 17, 21, 23, 33, 37]. Soundwave [6], Multiwave [17], and AudioGest [21] use Doppler effect to recognize predefined gestures. However, Doppler effect only gives coarse-grained movement speeds. Thus, these schemes only recognize a small set of gestures that have distinctive speed characters. Recently, three state-of-the-art schemes (*i.e.*, FingerIO [14], LLAP [28], and Strata [34]) use ultrasound to track fine-grained finger gestures. FingerIO [14] transmits OFDM modulated sound frames and locates the moving finger based on the change of the echo profiles of two consecutive frames. LLAP [28] uses Continuous Wave (CW) signal to track the moving target based on the phase information, which is susceptible to the dynamic multipath caused by other moving objects. Strata [34] combines the frame-based approach and the phase-based approach. Using the 26-bit GSM training sequence that has nice autocorrelation properties, Strata can track phase changes at different time delays so that objects that are more than 8.5 cm apart can be resolved. However, these schemes mainly focus on tracking in-air gestures that are performed at more than 20 *cm* away from the mobile device [14, 23, 28, 34]. In comparison, our system uses both the structure-borne and the air-borne sound signals to sense gestures performed on the surface of the mobile devices, which are very close (*e.g.*, less than 12 *cm*) to both the speakers and the microphones. As the sound reflections at a short distance are often submerged by the Line-of-Sight (LOS) signals, sensing gestures with SNR $\approx$ 2 dB at 5 *cm* is considerably harder than sensing in-air gestures with SNR $\approx$ 12 dB at 30 *cm*.

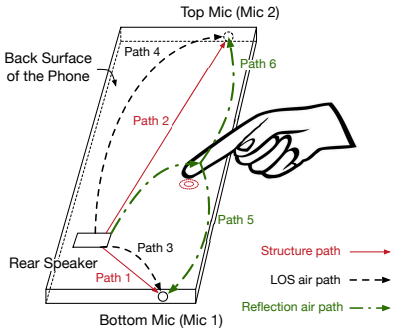


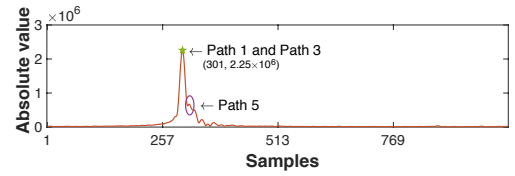
Figure 2: Sound propagation paths on a smartphone

3 SYSTEM OVERVIEW

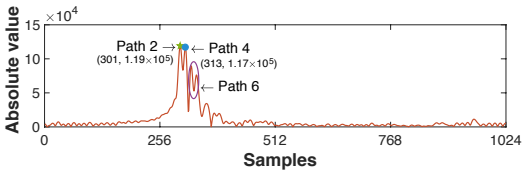
VSkin uses both the structure-borne and the air-borne sound signals to capture gestures performed on the surface of the mobile device. We transmit and record inaudible sounds using the built-in speakers and microphones on commodity mobile devices. As an example illustrated in Figure 2, sound signals transmitted by the rear speaker travel through multiple paths on the back of the phone to the top and bottom microphones. On both microphones, the structure-borne sound that travels through the body structure of the smartphone arrives first. This is because sound wave propagates much faster in the solid (>2,000m/s) than in the air (around 343m/s) [24]. There might be multiple copies of air-borne sounds arriving within a short interval following the structure-borne sound. The air-borne sounds include the LOS sound and the reflection sounds of surrounding objects, *e.g.*, the finger or the table. All these sound signals are mixed at the recording microphones.

VSkin performs gesture-sensing based on the mixture of sound signals recorded by the microphones. The design of VSkin consists of the following four components:

Transmission signal design: We choose to use the Zadoff-Chu (ZC) sequence modulated by a sinusoid carrier as our transmitted sound signal. This transmission signal design meets three key design goals. First, the auto-correlation of ZC sequence has a narrow peak width of 6 samples so that we can separate sound paths arrive with a small time-difference by locating the peaks corresponding to their different delays, see Figure 3. Second, we use interpolation schemes to reduce the bandwidth of the ZC sequence to less than 6 kHz so that it can be fit into the narrow inaudible range of 17 $\sim$ 23 kHz



(a) Bottom microphone (Mic 1)



(b) Top microphone (Mic 2)

Figure 3: IR estimation of dual microphones

provided by commodity speakers and microphones. Third, we choose to modulate the ZC sequence so that we can extract the phase information, which cannot be measured by traditional chirp-like sequences such as FMCW sequences.

Sound path separation and measurement: To separate different sound paths at the receiving end, we first use cross-correlation to estimate the Impulse Response (IR) of the mixed sound. Second, we locate the candidate sound paths using the amplitude of the IR estimation. Third, we identify the structure-borne path, the LOS path, and the reflection path by aligning candidate paths on different microphones based on the known microphone positions. Finally, we use an efficient algorithm to calculate the phase and amplitude of each sound path at a high sampling rate of 48 kHz.

Finger movement measurement: The finger movement measurement is based on the phase of the air-borne path reflected by the finger. To detect the weak reflections of the finger, we first calculate the differential IR estimations so that changes caused by finger movements are amplified. Second, we use an adaptive algorithm to determine the delay of the reflection path so that the phase and amplitude can be measured with high SNR. Third, we use an Extend Kalman Filter to further amplify the sound signal based on the finger movement model. Finally, the finger movement distance is calculated by measuring the phase change of the corresponding reflection path.

Touch measurement: We use the structure-borne path to detect touch events, since the structure-borne path is mainly determined by whether the user's finger is pressing on the surface or not. To detect touch events, we first calculate the differential IR estimations of the structure-borne path. We then use a threshold-based scheme to detect the touch and release events. To locate the touch position, we found that the delay of the changes in structure-borne sound is closely related to the distance from the touch position to the speaker. Using this observation, we classify the touch event into three different regions with an accuracy of 87.8%.

Note that finger movement measurement and touch measurement can use signal captured by the top microphone, the bottom microphone, or both. How these measurements are used in specific gestures, such as scrolling and swiping, depends on both the type of the gestures and the placement of microphones of the given device, see Section 6.5.

4 TRANSMISSION SIGNAL DESIGN

4.1 Baseband Sequence Selection

Sound signals propagating through the structure path, the LOS path, and the reflection path arrive within a very small time interval of less than $0.34ms$, due to the small size of a smartphone ($< 20cm$). One way to separate these paths is to transmit short impulses of sounds so that the reflected impulses do not overlap with each other. However, impulses with short time durations have very low energy so that the received signals, especially those reflected by the finger, are too weak to be reliably measured.

In VSkin, we choose to transmit a periodical high-energy signal and rely on the auto-correlation properties of the signal to separate the sound paths. A continuous periodical signal has higher energy than impulses so that the weak reflections can be reliably measured. The *cyclic auto-correlation* function of the signal $s[n]$ is defined as $R(\tau) = \frac{1}{N}\sum_{n=1}^{N} s[n]s^*[(n-\tau) \mod N]$, where N is the length of the signal, τ is the delay, and $s^*[n]$ is the conjugation of the signal. The cyclic auto-correlation function is maximized around $\tau = 0$ and we define the peak at $\tau = 0$ as the *main lobe* of the auto-correlation function, see Figure 5(b). When the cyclic auto-correlation function has a single narrow peak, *i.e.*, $R(\tau) \approx 0$ for $\tau \neq 0$, we can separate multiple copies of $s[n]$ arrived at different arrival delay τ by performing cross-correlation of the mixed signal with the cyclically shifted $s[n]$. For the cross-correlation results as shown in Figure 3, each delayed copy of $s[n]$ in the mixed signal leads to a peak at its corresponding delay value of τ.

The transmitted sound signal needs to satisfy the following extra requirements to ensure both the *resolution* and *signal-to-noise ratio* of the path estimation:

- **Narrow autocorrelation main lobe width**: The width of the main lobe is the number of points on each side of the lobe where the power has fallen to half (−3 dB) of its maximum value. A narrow main lobe leads to better time resolution in sound propagation paths.
- **Low baseband crest factor**: Baseband crest factor is the ratio of peak values to the effective value of the baseband signal. A signal with a low crest factor has higher energy than a high crest factor signal with the same peak power [2]. Therefore, it produces cross-correlation results with higher signal-to-noise ratio while the peak power is still below the audible power threshold.

	Interpolation Method	Auto-correlation main lobe width	Baseband crest factor	Auto-correlation gain	Auto-correlation side lobe level
GSM (26 bits)	Time domain	14 samples	8.10 dB	11.80 dB	-4.64 dB
	Frequency domain	8 samples	6.17 dB	11.43 dB	-3.60 dB
Barker (13 bits)	Time domain	16 samples	10.50 dB	11.81 dB	-9.57 dB
	Frequency domain	8 samples	5.12 dB	13.46 dB	-6.50 dB
M-sequence (127 bits)	Time domain	16 samples	5.04 dB	12.04 dB	-11.63 dB
	Frequency domain	8 samples	6.68 dB	13.90 dB	-6.58 dB
ZC (127 bits)	Time domain	16 samples	3.85 dB	12.14 dB	-12.45 dB
	Frequency domain	**6 samples**	**2.56 dB**	**13.93 dB**	**-6.82 dB**

Table 1: Performance of different types of sequences

• **High auto-correlation gain**: The auto-correlation gain is the peak power of the main lobe divided by the average power of the auto-correlation function. A higher auto-correlation gain leads to a higher signal-to-noise ratio in the correlation result. Usually, a longer code sequence has a higher auto-correlation gain.

• **Low auto-correlation side lobe level**: Side lobes are the small peaks (local maxima) other than the main lobe in the auto-correlation function. A large side lobe level will cause interference in the impulse response estimation.

We compare the performance of the transmission signals with different code sequence designs and interpolation methods. For code sequence design, we compare commonly used pseudo-noise (PN) sequences (*i.e.*, GSM training sequence, Barker sequence, and M-sequence) with a chirp-like polyphase sequence (ZC sequence [18]) in Table 1. Note that the longest Barker sequence and GSM training sequence are 13 bits and 26 bits, respectively. For M-sequence and ZC sequence, we use a sequence length of 127 bits.

We interpolate the raw code sequences before transmitting them. The purpose of the interpolation is to reduce the bandwidth of the code sequence so that it can be fit into a narrow transmission band that is inaudible to humans. There are two methods to interpolate the sequence, the time domain method and the frequency domain method. For the time domain method [34], we first upsample the sequences by repeating each sample by k times (usually $k = 6 \sim 8$) and then use a low-pass filter to ensure that the signal occupies the desired bandwidth. For the frequency domain method, we first perform Fast Fourier Transform (FFT) of the raw sequence, perform zero padding in the frequency domain to increase the length of the signal, and then use Inverse Fast Fourier Transform (IFFT) to convert the signal back into the time domain. For both methods, we reduce the bandwidth of all sequences to 6 kHz with a sampling rate of 48 kHz so that the modulated signal can be fit into the $17 \sim 23$ kHz inaudible range supported by commercial devices.

The performance of different sound signals is summarized in Table 1. The ZC sequence has the best baseband crest factor and auto-correlation gain. Although the raw M-sequence has the ideal auto-correlation performance and crest factor, the sharp transitions between "0" and "1" in M-sequence make the interpolated version worse than chirp-like polyphase sequences [2]. In general, frequency domain interpolation is better than the time domain interpolation, due to their narrow main lobe width. While the side lobe level of frequency domain interpolation is higher than the time domain interpolation, the side lobe level of -6.82 dB provided by the ZC sequence gives enough attenuation on side lobes for our system.

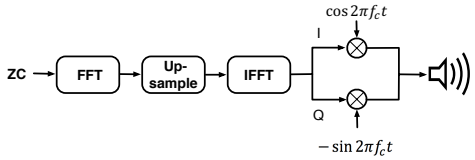


Figure 4: Sound signal modulation structure

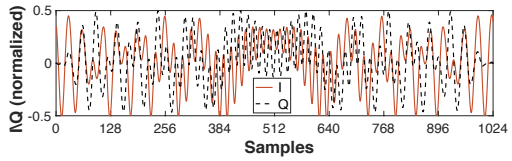
Based on above considerations, we choose to use the frequency domain interpolated ZC sequence as our transmitted signal. The root ZC sequence parametrized by u is given by:

$$ZC[n] = e^{-j\frac{\pi u n(n+1+2q)}{N_{ZC}}}, \quad (1)$$

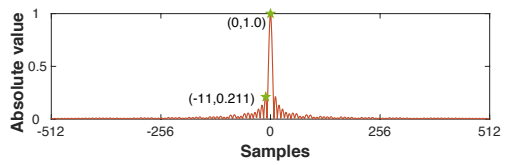
where $0 \leqslant n < N_{ZC}$, q is a constant integer, and N_{ZC} is the length of sequence. The parameter u is an integer with $0 < u < N_{ZC}$ and $gcd(N_{ZC}, u) = 1$. The ZC sequence has several nice properties [18] that are useful for sound signal modulation. For example, the ZC sequences have constant magnitudes. Therefore, the power of the transmitted sound is constant so that we can measure its phase at high sampling rates as shown in later sections. Note that compared to the single frequency scheme [28], the disadvantages of modulated signals including using ZC sequence are that they have to occupy the larger bandwidth and therefore require stable frequency response for the microphone.

4.2 Modulation and Demodulation

We use a two-step modulation scheme to convert the raw ZC sequence into an inaudible sound signal, as illustrated in Figure 4. The first step is to use the frequency domain interpolation to reduce the bandwidth of the sequence. We first perform N_{ZC}-points FFT on the raw complex valued ZC sequence, where N_{ZC} is the length of the sequence. We then zero-pad the FFT result into $N'_{ZC} = N_{ZC} f_s / B$ points by



(a) Baseband signal in the time domain



(b) Autocorrelation of baseband signal

Figure 5: Baseband signal of the ZC sequence

inserting zeros *after* the positive frequency components and *before* the negative frequency components, where B is targeting signal bandwidth (*e.g.*, 6 kHz) and f_s is the sampling rate of the sound (*e.g.*, 48 kHz). In this way, the interpolated ZC sequence only occupies a small bandwidth of B in the frequency domain. Finally, we use IFFT to convert the interpolated signal back into the time domain.

In VSkin, we choose a ZC sequence length of 127 points with a parameter of $u = 63$. We pad the 127-point ZC sequence into 1024 points. Therefore, we have $B = 5.953$ kHz at the sampling rate of $f_s = 48$ kHz. The interpolated ZC sequence is a periodical complex valued signal with a period of 1024 sample points (21.3ms) as shown in Figure 5(a).

The second step of the modulation process is to up-convert the signal into the passband. In the up-convert step, the interpolated ZC sequence is multiplied with a carrier frequency of f_c as shown in Figure 4. The transmitted passband signal is $T(t) = \cos(2\pi f_c t)ZC_T^I(t) - \sin(2\pi f_c t)ZC_T^Q(t)$, where $ZC_T^I(t)$ and $ZC_T^Q(t)$ are the real part and imaginary part of the time domain ZC sequence, respectively. We set f_c as 20.25 kHz so that the transmitted signal occupies the bandwidth from 17.297 kHz to 23.25 kHz. This is because of frequencies higher than 17 kHz are inaudible to most people [20].

The signal is transmitted through the speaker on the mobile device and recorded by the microphones using the same sampling frequency of 48 kHz. After receiving the sound signal, VSkin first demodulates the signal by down-converting the passband signal back into the complex valued baseband signal.

5 SOUND PATH SEPARATION AND MEASUREMENT

5.1 Multipath Propagation Model

The received baseband signal is a superposition of multiple copies of the transmitted signals with different delays due to multipath propagation. Suppose that the transmitted baseband signal is $ZC_T(t)$ and the system is a Linear Time-Invariant (LTI) system, then the received baseband signal can be represented as:

$$ZC_R(t) = \sum_{i=1}^{L} A_i e^{-j\phi_i} ZC_T(t - \tau_i) = h(t) * ZC_T(t), \quad (2)$$

where L is the number of propagation paths, τ_i is the delay of the i^{th} propagation path and $A_i e^{-j\phi_i}$ represents the complex path coefficient (*i.e.*, amplitude and phase) of the i^{th} propagation path, respectively. The received signal can be viewed as a circular convolution, $h(t) * ZC_T(t)$, of the *Impulse Response* $h(t)$ and the periodical transmitted signal $ZC_T(t)$. The Impulse Response (IR) function of the multipath propagation model is given by

$$h(t) = \sum_{i=1}^{L} A_i e^{-j\phi_i} \delta(t - \tau_i), \quad (3)$$

where $\delta(t)$ is Dirac's delta function.

We use the cross-correlation, $\hat{h}(t) = ZC_R^*(-t) * ZC_T(t)$, of the received baseband signal $ZC_R(t)$, with the transmitted ZC sequence $ZC_T(t)$ as the estimation of the impulse response. Due to the ideal periodic auto-correlation property of ZC code, where the auto-correlation of ZC sequence is non-zero only at the point with a delay τ of zero, the estimation $\hat{h}(t)$ provides a good approximation for the IR function.

In our system, $\hat{h}(t)$ is sampled with an interval of $T_s = 1/f_s = 0.021$ ms, which corresponds to 0.7 cm (343 $m/s \times 0.021$ ms) of the propagation distance. The sampled version of IR estimation, $\hat{h}[n]$, has 1024 taps with $n = 0 \sim 1023$. Therefore, the maximum unambiguous range of our system is $1024 \times 0.7/2 = 358$ cm, which is enough to avoid interferences from nearby objects. Using the cross-correlation, we obtain one *frame* of IR estimation $\hat{h}[n]$ for each period of 1,024 sound samples (21.33 ms), as shown in Figure 3. Each peak in the IR estimation indicates one propagation path at the corresponding delay, *i.e.*, a path with a delay of τ_i will lead to a peak at the $n_i = \tau_i/T_s$ sample point.

	Path	Speed	Distance	Delay	Amplitude
1	Structure (Mic 1)	>2,000 m/s	4.5 cm	≪0.13 ms	Large
2	Structure (Mic 2)	>2,000 m/s	12 cm	≪0.13 ms	Medium
3	LOS (Mic 1)	343 m/s	4.5 cm	0.13 ms	Large
4	LOS (Mic 2)	343 m/s	12 cm	0.34 ms	Medium
5	Reflection (Mic 1)	343 m/s	>4.5 cm	>0.13 ms	Small
6	Reflection (Mic 2)	343 m/s	>12 cm	>0.34 ms	Small

Table 2: Different propagation paths

5.2 Sound Propagation Model

In our system, there are three different kinds of propagation paths: the structure path, the LOS air path and the reflection air path, see Figure 2.

Theoretically, we can estimate the delay and amplitude of different paths based on the speed and attenuation of

sound in different materials and the propagation distance. Table 2 lists the theoretical propagation delays and amplitude for the six different paths between the speaker and the two microphones on the example shown in Figure 2. Given the high speed of sound for the structure-borne sound, the two structure sound paths (Path 1 and Path 2) have similar delays even if their path lengths are slightly different. Since the acoustic attenuation coefficient of metal is close to air [26], the amplitude of structure sound path is close to the amplitude of the LOS air path. The LOS air paths (Path 3 and Path 4) have longer delays than the structure paths due to the slower speed of sound in the air. The reflection air paths (Path 5 and Path 6) arrive after the LOS air paths due to the longer path length. The amplitudes of reflection air paths are smaller than other two types of paths due to the attenuation along the reflection and propagation process.

5.3 Sound Propagation Separation

Typical impulse response estimations of the two microphones are shown in Figure 3. Although the theoretical delay difference between Path 1 and Path 3 is 0.13 *ms* (6 samples), the time resolution of the interpolated ZC sequence is not enough to separate Path 1 and Path 3 on Mic 1. Thus, the first peak in the IR estimation of the Mic 1 represents the combination of Path 1 and Path 3. Due to the longer distance from the speaker to Mic 2, the theoretical delay difference between Path 2 and Path 4 is 0.34 *ms* (17 samples). As a result, the Mic 2 has two peaks with similar amplitude, which correspond to the structure path (the first peak) and the LOS air path (the second peak), respectively. By locating the peaks of the IR estimation of the two microphones, we are able to separate different propagation paths.

We use the IR estimation of both microphones to identify different propagation paths. On commercial mobile devices, the starting point of the auto-correlation function is random due to the randomness in the hardware/system delay of sound playback and recording. The peaks corresponding to the structure propagation may appear at random positions every time when the system restarts. Therefore, we need to first locate the structure paths in the IR estimations. Our key observation is that the two microphones are strictly synchronized so that their structure paths should appear at the same position in the IR estimations. Based on this observation, we first locate the highest peak of Mic 1, which corresponds to the combination of both Path 1 and Path 3. Then, we can locate the peaks of Path 2 and Path 4 in the IR estimation of Mic 2 as the position of Path 2 should be aligned with Path 1/Path 3. Since we focus on the movement around the mobile devices, the reflection air path is $5 \sim 15$ samples ($3.5 \sim 10.7$ *cm*) away from LOS path for both microphones. In this way, we get the delays of (i) combination of Path 1 and Path 3, (ii) Path 2, (iii) Path 4, and (iv) the range of reflection air path (Path 5 and Path 6), respectively. We call this process as path delay calibration, which is performed once when the system starts transmitting and recording the sound signal. The path delay calibration is based on the first ten data segments (213 *ms*) of IR estimation. We use an 1-nearest neighbor algorithm to confirm the path delays based on the results of the ten segments.

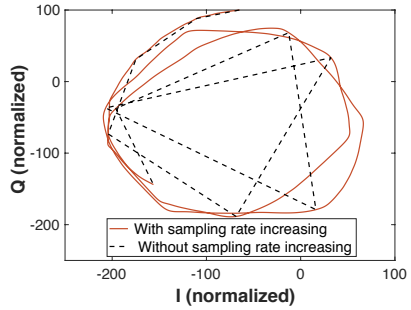


Figure 6: Path coefficient at different sampling rate

Note that the calibration time is 14.95 *ms* for one segment (21.3 *ms*). Thus, we can perform calibration for each segment in real-time. To save the computational cost, we only calibrate the LOS path and structure-borne path delays for the first ten segments (213 *ms*). The path delay calibration is only performed once after the system initialization because holding styles hardly change delays of the structure-borne path and the LOS path. For the reflection path delay, we adaptively estimate it as shown in Section 6.2 so that our system will be robust to different holding styles.

5.4 Path Coefficient Measurement

After finding the delay of each propagation path, we measure the path coefficient of each path. For a path i with a delay of n_i samples in the IR estimation, the path coefficient is the complex value of $\hat{h}[n_i]$ on the corresponding microphone. The path coefficient indicates how the amplitude and phase of the given path change with time. Both the amplitude and the phase of the path coefficient are important for later movement measurement and touch detection algorithms.

One key challenge in path coefficient measurement is that cross-correlations are measured at low sampling rates. The basic cross-correlation algorithm presented in Section 5.1 produces one IR estimation per frame of 1,024 samples. This converts to a sampling rate of $48,000/1,024 = 46.875$ Hz. The low sampling rate may lead to ambiguity in fast movements where the path coefficient changes quickly. Figure 6 shows the path coefficient of a finger movement with a speed of 10 cm/s. We observe that there are only 2~3 samples in each phase cycle of 2π. As a phase difference of π can be caused either by a phase increases of π or a phase decreased by π, the direction of phase changing cannot be determined by such low rate measurements.

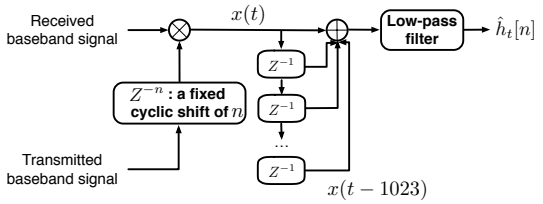


Figure 7: Path coefficient measurement for delay n

We use the property of the circular cross-correlation to upsample the path coefficient measurements. For a given delay of n samples, the IR estimation at time t is given by the circular cross-correlation of the received signal and the transmitted sequence:

$$\hat{h}_t[n] = \sum_{l=0}^{N'_{ZC}-1} ZC_R[t+l] \times ZC_T^*[(l-n) \mod N'_{ZC}] \quad (4)$$

This is equivalent to take the summation of N'_{ZC} point of the received signal multiplied by a conjugated ZC sequence cyclically shifted by n points. The key observation is that ZC sequence has constant power, *i.e.*, $ZC[n] \times ZC^*[n] = 1, \forall n$. Thus, each point in the N'_{ZC} multiplication results in Eq. (4) contributes equally to the estimation of $\hat{h}_t[n]$. In consequence, the summation over a window with a size of N'_{ZC} can start from any value of t. Instead of advancing the value t by a full frame of 1,024 sample points as in ordinary cross-correlation operations, we can advance t one sample each time. In this way, we can obtain the path coefficient with a sampling rate of 48 kHz, which gives the details of changes in path coefficient as shown in Figure 6.

The above upsampling scheme incurs high computational cost. To obtain all path coefficients $\hat{h}_t[n]$ for delay n ($n = 0 \sim 1023$), it requires 48,000 dot productions per second and each dot product is performed with two vectors of 1,024 samples. This cannot be easily carried out by mobile devices. To reduce the computational cost, we observe that not all taps in $\hat{h}_t[n]$ are useful. We are only interested in the taps corresponding to the structure propagation paths and the reflection air paths within a distance of 15 cm. Therefore, instead of calculating the cross-correlation, we just calculate the path coefficients at given delays using a fixed cyclic shift of n. Figure 7 shows the process of measuring the path coefficient at a given delay. First, we synchronize the transmitted signal and received signal by cyclically shifting the transmitted signal with a fixed offset of n_i corresponding to the delay of the given path. Second, we multiply each sample of the received baseband signal with the conjugation of the shifted transmitted sample. Third, we use a moving average with a window size of 1,024 to sum the complex values and get the path coefficients. Note that the moving average can be carried out by just two additions per sample. Fourth, we use low-pass filter to remove high frequency noises caused by imperfections of the interpolated ZC sequence. Finally, we get the path coefficient at 48 kHz sampling rate. After the optimization, measuring the path coefficient at a given delay only incurs one multiplication and two additions for each sample.

6 MOVEMENT MEASUREMENT

6.1 Finger Movement Model

Finger movements incur both magnitude and phase changes in path coefficients. First, the delay for the peak corresponding to the reflection path of the finger changes when the finger moves. Figure 8(a) shows the magnitude of the IR estimations when the finger first moves away from the microphone and then moves back. The movement distance is 10 cm on the surface of the mobile device. A "hot" region indicates a peak at the corresponding distance in the IR estimation. While we can observe there are several peaks in the raw IR estimation and they change with the movement, it is hard to discern the reflection path as it is much weaker than the LOS path or the structure path. To amplify the changes, we take the difference of the IR estimation along the time axis to remove these static paths. Figure 8(b) shows the resulting differential IR estimations. We observe that the finger moves away from the microphone during 0.7 to 1.3 seconds and moves towards to the microphone from 3 to 3.5 seconds. The path length changes about 20 cm (10×2) during the movement. In theory, we can track the position of the peak corresponding to the reflection path and measure the finger movement. However, the position of the peak is measured in terms of the number of samples, which gives a low resolution of around 0.7 *cm* per sample. Furthermore, estimation of the peak position is susceptible to noises, which leads to large errors in distance measurements.

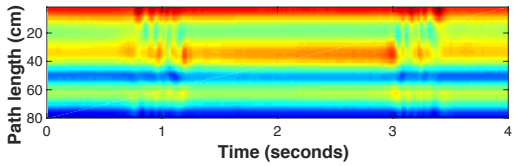
We utilize phase changes in the path coefficient to measure movement distance so that we can achieve mm-level distance accuracy. Consider the case the reflection path of the finger is path i and its path coefficient is:

$$\hat{h}_t[n_i] = A_i e^{-j(\phi_i + 2\pi \frac{d_i(t)}{\lambda_c})}, \quad (5)$$

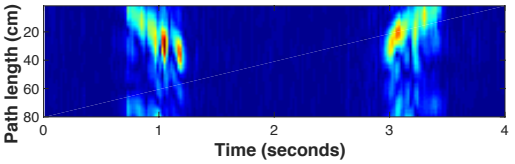
where $d_i(t)$ is the path length at time t. The phase for path i is $\phi_i(t) = \phi_i + 2\pi \frac{d_i(t)}{\lambda_c}$, which changes by 2π when $d_i(t)$ changes by the amount of sound wavelength $\lambda_c = c/f_c$ ($\approx$1.69 cm) [28]. Therefore, we can measure the phase change of the reflection path to obtain mm-level accuracy in the path length $d_i(t)$.

6.2 Reflection Path Delay Estimation

The first step for measuring the finger movement is to estimate the delay of the reflection path. Due to the non-negligible main lobe width of the auto-correlation function, multiple IR estimations that are close to the reflection path have similar changes when the finger moves. We need to



(a) Magnitude of the raw IR estimations



(b) Magnitude of differential IR estimations

Figure 8: IR estimations for finger movement.

adaptively select one of these IR estimations to represent the reflection path so that noises introduced by side lobes of other paths can be reduced.

Our heuristic to determine the delay of the reflection path is based on the observation that the reflection path will have the largest change of magnitude compared to other paths. Consider the changes of magnitude in $\hat{h}_t[n_i]$: $\left|\hat{h}_t[n_i] - \hat{h}_{t-\Delta t}[n_i]\right| = \left|A_i\left(e^{-j(\phi_i+2\pi\frac{d_i(t)}{\lambda_c})} - e^{-j(\phi_i+2\pi\frac{d_i(t-\Delta t)}{\lambda_c})}\right)\right|$. Here we assume that A_i does not change during the short period of Δt. When the delay n_i is exactly the same as of the reflection path, the magnitude of $\left|\hat{h}_t[n_i] - \hat{h}_{t-\Delta t}[n_i]\right|$ is maximized. This is because the magnitude of $|A_i|$ is maximized at the peak corresponds to the auto-correlation of the reflection path, and the magnitude of $\left|e^{-j(\phi_i+2\pi\frac{d_i(t)}{\lambda_c})} - e^{-j(\phi_i+2\pi\frac{d_i(t-\Delta t)}{\lambda_c})}\right|$ is maximized due to the largest path length change at the reflection path delay.

In our implementation, we select l path coefficients with an interval of three samples between each other as the candidate of reflection paths. The distance between these candidate reflection paths and the structure path is determined by size of the phone, *e.g.*, $5 \sim 15$ samples for the bottom Mic. We keep monitoring the candidate path coefficients and select the path with the maximum magnitude in the time differential IR estimations as the reflection path. When the finger is static, our system still keeps track of the reflection path . In this way, we can use the changes in the selected reflection path to detect whether the finger moves or not.

6.3 Additive Noise Mitigation

Although the adaptive reflection path selection scheme gives high SNR measurements on path coefficients, the additive noises from other paths still interfere with the measured path coefficients. Figure 9 shows the result of the trace of the complex path coefficient with a finger movement. In the ideal case, the path coefficients is $\hat{h}_t[n_i] = A_i e^{-j(\phi_i+2\pi d_i(t)/\lambda_c)}$

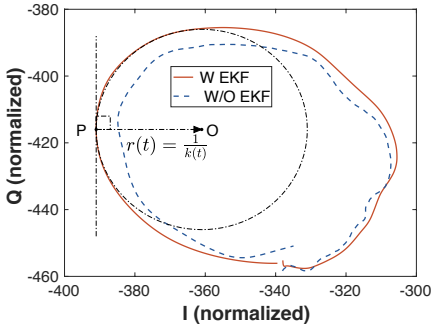


Figure 9: Path coefficients for finger reflection path.

with a constant attenuation of A_i in a short period. Therefore, the trace of path coefficients should be a circle in the complex plane. However, due to additive noises, the trace in Figure 9 is not smooth enough for later phase measurements.

We propose to use the Extended Kalman Filter (EKF), a non-linear filter, to track the path coefficient and reduce the additive noises. The goal is to make the resulting path coefficient closer to the theoretical model so that the phase change incurred by the movement can be measured with higher accuracy. We use the sinusoid model to predict and update the signal of both I/Q components [8]. To save the computational resources, we first detect whether the finger is moving or not as shown in Section 6.2. When we find that the finger is moving, we initialize the parameters of the EKF and perform EKF. We also downsample the path coefficient to 3 kHz to make the EKF affordable for mobile devices. Figure 9 shows that results after EKF are much smoother than the original signal.

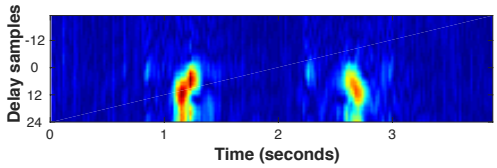
6.4 Phase Based Movement Measurement

We use a curvature-based estimation scheme to measure the phase change of the path coefficient. Our estimation scheme assumes that the path coefficient is a superposition of a circularly changing dynamical component, which is caused by the moving finger, and a quasi-static component, which is caused by nearby static objects [28, 29, 34]. The algorithm estimates the phase of the dynamic component by measuring the curvature of the trace on the complex plane. The curvature-based scheme avoids the error-prone process of estimating the quasi-static component in LEVD [28] and is robust to noise interferences.

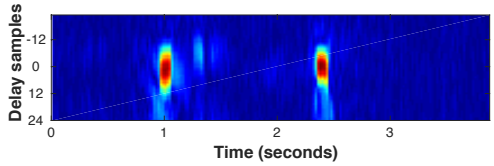
Suppose that we use a trace in the two-dimensional plane $\boldsymbol{y}(t) = (I_{\hat{h}_t}, Q_{\hat{h}_t})$ to represent the path coefficient of the reflection. As shown in Figure 9, the instantaneous signed curvature can be estimated as:

$$k(t) = \frac{\det(\boldsymbol{y}'(t), \boldsymbol{y}''(t))}{\|\boldsymbol{y}'(t)\|^3}, \tag{6}$$

where $\boldsymbol{y}'(t) = d\boldsymbol{y}(t)/dt$ is the first derivative of $\boldsymbol{y}(t)$ with respect to the parameter t, and det is taking the determinant



(a) Magnitude of differential IR estimations when touch and release at 7 cm away from speaker



(b) Magnitude of differential IR estimations when touch and release at 1 cm away from speaker

Figure 10: Touching on different locations

of the given matrix. We assume that the instantaneous curvature remains constant during the time period $t-1 \sim t$ and the phase change of the dynamic component is:

$$\Delta\theta_{t-1}^{t} = 2\arcsin\frac{k(t)\left|\boldsymbol{y}(t)-\boldsymbol{y}(t-1)\right|}{2}. \tag{7}$$

The path length change for the time period $0 \sim t$ is:

$$d_i(t) - d_i(0) = -\frac{\sum_{i=1}^{t}\Delta\theta_{i-1}^{i}}{2\pi}\times\lambda_c, \tag{8}$$

where $d_i(t)$ is the path length from the speaker reflected through the finger to the microphone.

6.5 From Path Length to Movements

The path length change for the reflection air path can be measured on both microphones. Depending on the type of gestures and the placement of the microphones, we can use the path length change to derive the actual movement distance. For example, for the phone in Figure 2, we can use the path length change of the reflection air path on the *bottom* microphone to measure the finger movement distance for the scrolling gesture (up/down movement). This is because the length of the reflection path on the bottom microphone changes significantly when the finger moves up/down on the back of the phone. The actual movement distance can be calculated by multiplying the path length change with a compensating factor as described in Section 8. For the gesture of swiping left/right, we can use path length changes of two microphones to determine the swiping direction, as swiping left and right will introduce the same path length change pattern on the bottom microphone but different path length change directions on the top microphone.

7 TOUCH MEASUREMENT

7.1 Touch Signal Pattern

Touching the surface with fingers will change both the air-borne propagation and structure-borne propagation of the sound. When performing the tapping action, the finger movement in the air will change the air-borne propagation of the sound. Meanwhile, when the finger contacts the surface of the phone, the force applied on the surface will change the vibration pattern of the structure of the phone, which leads to changes in the structure-borne signal [25]. In other words, the structure-borne sound is able to distinguish whether the finger is hovering above the surface with a mm-level gap or is pressing on the surface. In VSkin, we mainly use the changes in the structure-borne signal to sense the finger touching, as they provide distinctive information about whether the finger touches the surface or not. However, when force is applied at different locations on the surface, the changes of the structure-borne sound caused by touching will be different in magnitude and phase. Existing schemes only use the magnitude of the structure-borne sound [25], which has different change rates at different touch positions. They rely on the touchscreen to determine the position and the accurate time of the touching to measure the force-level of touching [25]. However, neither the location nor the time of the touching is available for VSkin. Therefore, the key challenge in touching sensing for VSkin is to perform joint touch detection and touch localization.

Touching events lead to unique patterns in the differential IR estimation. As an example, Figure 10 shows the differential IR estimations that are close to the structure-borne path of the *top* microphone in Figure 2, when the user touches the back of the phone. The y-axis is the number of samples to the structure-borne path, where the structure-borne path (Path 2 in Section 5.3) is at $y = 0$. When force is applied on the surface, the width of the peak corresponding to the structure-borne path increases. This leads to a small deviation in the peak position in the path coefficient changes from the original peak of the structure-borne propagation. Figure 10(a) shows the resulting differential IR estimations when user's finger touches/leaves the surface of the mobile device at a position that is 7 cm away from the rear speaker. We observe that the "hottest" region is not at the original peak of the structure-borne propagation. This is due to the force applied on the surface changes the path of the structure-borne signal. To further explore the change of the structure-borne propagation, we ask the user to perform finger tapping on eleven different positions on the back of the device and measure the position of peaks in the path coefficient changes. Figure 11 shows the relationship between the touching position and the resulting peak position in coefficient changes, where the peak position is measured by the number of samples to the original structure-borne path. We observe that the larger the distance between the touching position and the speaker, the larger the delay in coefficient changes to the original structure-borne path (darker color means a larger delay). Thus, we utilize the magnitude and delay of differential IR estimations to detect and localize touch events. Note

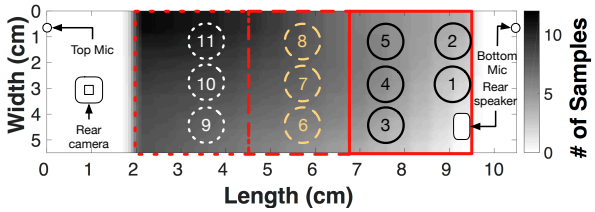


Figure 11: Touching position clustering.

that the differential IR estimations are based on complex-valued path coefficients. If we ignore the phase and only use the magnitude of path coefficients, there are some locations where the phase change caused by the touch event incurs little magnitude change so that the touch event cannot be reliably detected. Similar phenomenon also appears in the case of using the magnitude of WiFi signals to detect small movements, such as human respiration [27].

7.2 Touch Detection and Localization

We perform joint touch detection and localization using the differential IR estimation around the structure path. Since the structure-borne sound and air-borne sound are mixed on the bottom microphone as shown in Section 5.3, we only use the path coefficients of the *top* microphone to sense touching. To detect touch events, we first calculate the time difference of the IR estimation in a similar way as in Section 6.2. We then identify the delay with the maximum magnitude of the time differential IR estimation and use the maximum magnitude as the indicator of the touch event. We use a threshold based scheme to detect touch and release events, *i.e.*, once the magnitude of differential IR estimation exceeds the threshold, we determine that the user either touches the surface or releases the finger. The detection threshold is dynamically calculated based on the background noise level. Our touch detection scheme keeps the state of touching and toggles between touch and release based on the detected events. Touch detection can work when the user holds the phone with his/her hand. Given that the pose of the holding hand does not change, we can still reliably detect touches using the differential IR estimation.

To determine the position of the touch, we use the delay (calculated in terms of samples) of the peak in differential IR estimation. We divide the back surface of the phone into three regions based on the distance to the speaker. The points in different regions are marked with different colors in Figure 11. Using the delay of the peak in differential IR estimation, we can identify the region that the user touches with an accuracy of 87.8%.

8 SYSTEM EVALUATION

8.1 Implementation

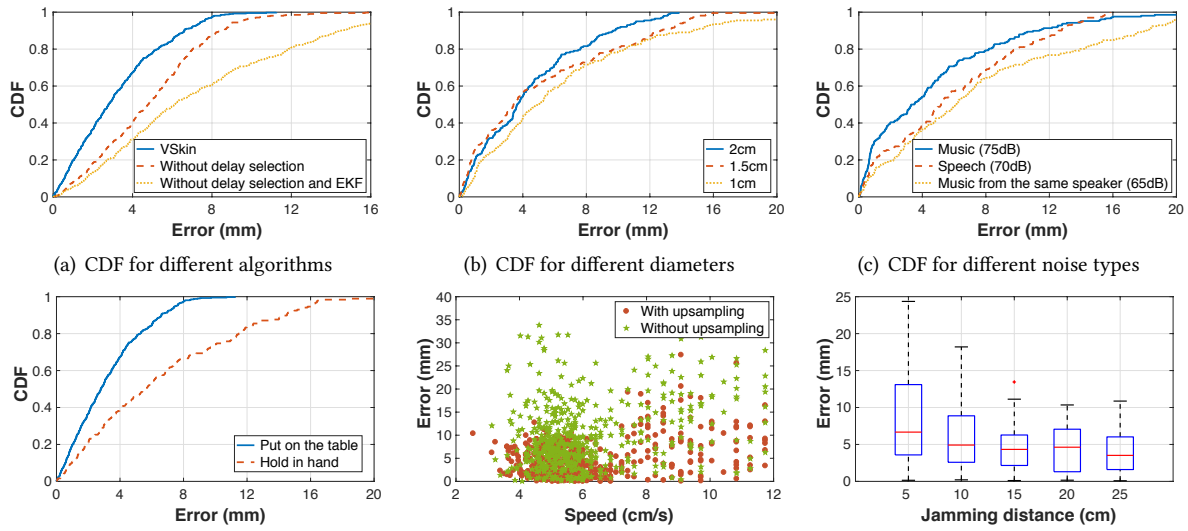
We implemented VSkin on the Android platform. Our system works as a real time APP that allows user to perform touch gestures, *e.g.*, scrolling, swiping, and tapping, on the surfaces of Android phones. Our implementation and evaluation mainly focused on Back-of-Device operations. To achieve better efficiency, we implement most signal processing algorithms as C functions using Android NDK and the signal processing is performed on data segments with a size of 1,024 samples, which is identical to the length of interpolated ZC sequence. We conducted experiments on Samsung Galaxy S5 using its rear speaker, top microphone, and bottom microphone in typical office and home environments. In the experiments, the users interacted with the phone using their bare hands without wearing any accessory.

8.2 Evaluations on Finger Movements

VSkin achieves an average movement distance error of 3.59 *mm when the finger moves for* 6 *cm on the back of the phone.* We attached a tape with a length of 6 *cm* on the back of the phone and asked the users to move their fingers up/down along the tape while touching the surface of the phone. We determine the ground truth of the path length change using a ruler, which is 10 *cm* for the 6 *cm* movement. Our system measures the movement distance by the bottom microphone and the rear speaker, using the compensation factor of 0.6 to convert the measured path length change into the movement distance. Our simulation results show that the compensation factor is in the range of $0.54 \sim 0.6$ for different positions on the back of the phone. Thus, fixing the factor to 0.6 will not significantly influence the accuracy. Figure 12(a) shows the Cumulative Distribution Function (CDF) of the distance measurement error for 400 movements. The average movement distance errors of VSkin, without delay selection and without delay selection and EKF are 3.59 *mm*, 4.25 *mm*, and 7.35 *mm*, respectively. The algorithm for delay selection and EKF reduces the measurement error by half. The standard deviation of the error is 2.66 *mm* and the 90th percentile measurement error is 7.70 *mm*, as shown in Figure 12(a).

VSkin is robust for objects with different diameters from 1 *cm to* 2 *cm.* Since user fingers have different diameters and introduce different reflection amplitude in sound signals, we use pens with three different diameters to measure the robustness of VSkin. Figure 12(b) shows the CDF of the movement distance error averaged by 200 movements of 6 *cm*. The average distance errors for pens with 1 *cm*, 1.5 *cm*, and 2 *cm* diameters are 6.64 *mm*, 5.14 *mm*, and 4.40 *mm*, respectively. Objects with a small diameter of 1 *cm* only incur a small increase in the distance error of 2.24 *mm*.

VSkin is robust for different holding styles. We evaluated our system under two different use cases: holding the phone with their hands and putting it on the table. We asked the users to use their own holding styles during the experiments. The average distance error for different users is 6.64 *mm* when putting the phone on the table. Holding the phone in



(a) CDF for different algorithms (b) CDF for different diameters (c) CDF for different noise types

(d) CDF for different use cases (e) Error for different speeds (f) Error for different jamming distances

Figure 12: Micro benchmark results for movements

hand only increases the average distance error by 3.42 *mm*, as shown in Figure 12(d).

VSkin can reliably measure the movement distance with speeds from 2 *cm/s to* 12 *cm/s*. We asked the user to move his finger at different speeds for a distance of 6 *cm*. Figure 12(e) shows the distribution of the movement distance errors with respect to the movement speeds. The average measurement error decreases from 11.00 *mm* to 4.64 *mm* when using upsampling. Especially, when the moving speed is higher than 8 *cm/s*, the average distance error decreases by about half, from 17.57 *mm* to 8.29 *mm*, when applying upsampling. This shows our upsampling scheme significantly improves the accuracy and robustness when the object is moving at high speeds.

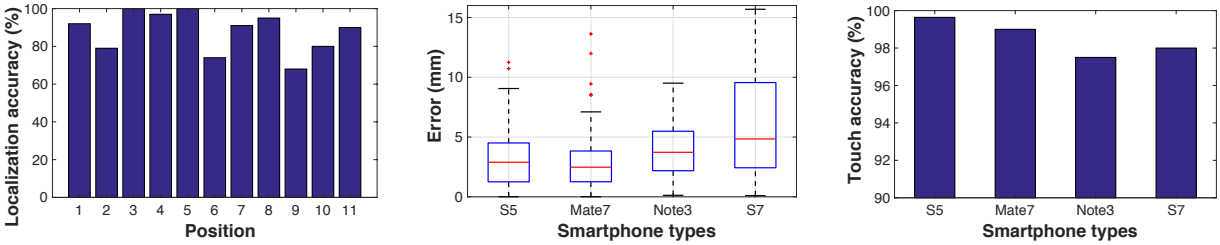
VSkin is robust to interfering movements that are 5 *cm away from the phone.* To evaluate the anti-jamming capacity of VSkin, we asked other people to perform jamming movements, *i.e.*, pushing and pulling hand repeatedly at different distances, while the user is performing the movement. As shown in Figure 12(f), VSkin achieves an average movement distance error of 9.19 *mm* and 3.98 *mm* under jamming movements that are 5 *cm* and 25 *cm* away from the device, respectively. Jamming movements introduce only a small increase in the measurement error, due to the nice auto-correlation property of the ZC sequence that can reliably separate activities at different distances.

VSkin is robust to background audible acoustic noises and achieves an average movement distance error of 6.22 *mm under noise interferences.* We conducted our experiments in three different environments with audible acoustic noises: i) an indoor environment with pop music being played (75 dB on average); ii) a room with people talking being played (70 dB on average); iii) playing music from the same speaker that used by VSkin (65 dB on average). As shown in Figure 12(c), the average movement distance errors are 4.64 *mm*, 5.93 *mm* and 8.08 *mm*, respectively. Note that VSkin does not block the playback functions of the speaker.

8.3 Evaluations on Touch Measurements

VSkin achieves a touch detection rate of 99.64% *for different positions at the back of the mobile phone.* We asked users to touch the back of the mobile phone for 100 times at the 11 different positions in Figure 11. VSkin missed only four touches among the 1100 tests. This gives a false negative rate of merely 0.36%. Since touching on the position close to the speaker causes more significant changes in the structure-borne signal, these four false detections are all at the position 10 and 11 in Figure 11. VSkin also has low false positive ratios. When placed in a silent environment, VSkin made no false detection of touching for 10 minutes. When performing jamming movements 5 *cm* away from the device, VSkin only made three false detections of touching for 10 minutes. Note that VSkin detects exactly the contact event, as users only moved their fingertip for a negligible distance in the touching experiments (measured air path length change of only 0.3 *mm*). In comparison, touch detection that only uses the magnitude of path coefficients has a lower detection rate of 81.27% as discussed in Section 7.1.

VSkin achieves an average accuracy of 87.82% *for classifying touches to three different regions of the phone.* We divide the 11 different positions into three different classes as shown by different colors in Figure 11. We asked users to touch the back of the phone at these 11 different positions for 100 times in each position. VSkin uses the delay of the structure path



(a) Accuracy for different positions (b) Movement error for different phones (c) Touch accuracy for different phones

Figure 13: Micro benchmark results for touching and generalization

(a) Path delay calibration

	Down conversion	Cross-correlation	Total
Time	0.363 *ms*	14.582 *ms*	14.945 *ms*

(b) Movement and touch sensing

	Down conversion	Upsampling	Phase measurement	Total
Time	0.363 *ms*	3.249 *ms*	0.324 *ms*	3.939 *ms*

Table 3: Processing time

(a) Power consumption

	CPU	Audio	Total
Power	121.3 *mW*	370 *mW*	491.3 *mW*

(b) Power consumption overhead

	Backlight	Web Browsing	Gaming
Power overhead	47.8%	25.4%	15.4%

Table 4: Power consumption

change to classify the positions into three different classes and the results are shown in Figure 13(a). The localization accuracies of position 2, 6, and 9 are lower than other positions because these three positions are not on the path of propagation from the rear speaker to the top microphone.

8.4 Latency and Power Consumption

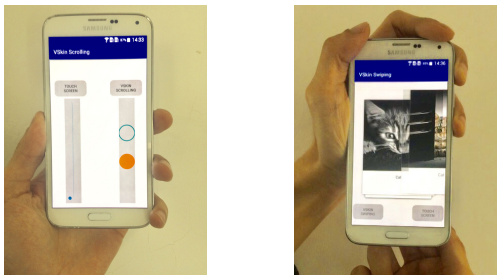
VSkin achieves a latency of 4.83 *ms on commercial smartphones.* We measured the processing time for a Samsung S5 with Qualcomm Snapdragon 2.5GHz quad-core CPU. Our system processes sound segments with a size of 1,024 samples (time duration of 21.3 *ms* at 48 kHz sampling rate). To reduce the processing time, we only perform the path delay calibration on the first ten data segments and later processing does not require recalibration. Furthermore, we use FFT to accelerate the cross-correlation. The processing time of one segments is 14.58 *ms* and 3.93 *ms* for the computational heavy path delay calibration process and the light-weight movement/touch-sensing algorithm. With processing time for other operations, the overall latency for VSkin to process 21.3 *ms* of data is 4.832 *ms* on average. Therefore, VSkin can perform realtime movement and touch sensing on commodity mobile devices.

VSkin incurs a power consumption of 491.3 *mW on commercial smartphones.* We use Powertutor [36] to measure the power consumption of our system on Samsung Galaxy S5. Without considering the LCD power consumption, the power consumptions of CPU and Audio are 121.3 *mW* and 370 *mW*, respectively. To measure the power consumption overhead of VSkin, we measured the average power consumption in three different states with VSkin: 1) Backlight, with the screen displaying the results, 2) Web Browsing, surfing the Internet with the WiFi on, 3) Gaming, playing mobile games with the WiFi on. The power consumption overheads for these states are 47.8%, 25.4%, and 15.4%, respectively. More than 74.2% additional power consumption comes from the speaker hardware. One possible solution is to design low-power speakers that are specialized for emitting ultrasounds.

8.5 Discussions

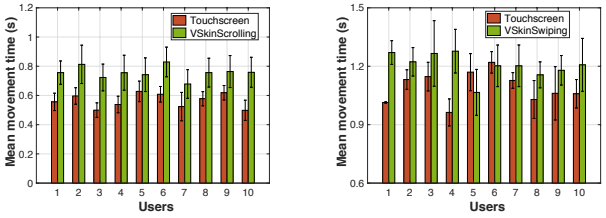
Different phone setups: *VSkin can work on different types of smartphones.* We conducted our experiments on four different smartphones, Samsung S5, Huawei Mate7, Samsung Note3, and Samsung S7, with parameters based on the locations of speaker and microphones. As shown in Figure 13(b), VSkin achieves an average movement distance error of 3.59 *cm*, 2.96 *cm*, 4.02 *cm* and 6.05 *cm* on the four models, respectively. VSkin also achieves more than 98% touch accuracy for all types of smartphones, as shown in Figure 13(c).

Locations of speakers and microphones: The speakers on Samsung S5 and Huawei Mate7 are on the back of the smartphones, while the speakers on Samsung S7 and Samsung Note3 are at the bottom of the smartphones. Our experimental results show that VSkin achieves higher accuracy on S5/Mate7 than S7/Note3. Therefore, the locations of the speaker and microphones are critical to the VSkin performance. The current design of VSkin requires one speaker and two microphones, with one microphone close to the speaker to measure the movement and the other at the opposite side of the speaker to measure the touch. Further generalization of VSkin to different speaker/microphone setups is left for future study.



(a) VSkinScrolling (b) VSkinSwiping

Figure 14: User interface for case study



(a) VSkinScrolling (b) VSkinSwiping

Figure 15: Movement time for different APPs

Privacy concerns: Since VSkin uses microphones to record the sound signal, our system may lead to potential privacy leakage issues. One possible solution is to keep the recorded sound signal within the operating system and only provide touch events to applications.

8.6 Case Study

We developed two VSkin-based APPs, called VSkinScrolling and VSkinSwiping, to further evaluate the performance of VSkin. We invited ten graduate students (eight males and two females with ages in the range of 22 to 27) to hold the phone with their hands and use our APPs. None of them had ever used BoD interactions before the case study.

8.6.1 *VSkinScrolling: Scrolling gesture.*

Application usage: VSkinScrolling enables scrolling gesture on the back surface of the device, as shown in Figure 14(a). Users hold the phone with their hands, first touch the back of the device as the trigger of VSkinScrolling and then drag their finger up/down to control the scrollbar. To improve the user experience, the ball on the scrollbar will change color once the user touches the back surface. We use the top microphone for touch detection and the bottom microphone to measure the scrolling distance and direction. The left scroll bar is controlled by the touchscreen, and the right scroll bar is controlled by VSkinScrolling.

Performance evaluation: *VSkinScrolling achieves usability comparable to the touchscreen for the scrolling gesture.* In the experiments, we first taught users the usage of VSkinScrolling and let them practice for five minutes. We then asked the users to perform the task of moving the scrollbar to a given position within an error range of ±5%. We compare the movement time (from touching the surface to successfully moving to the target) of VSkinScrolling and the front touchscreen. Each participant performed the task for 20 times using VSkinScrolling and the touchscreen. As shown in Figure 15(a), the mean movement time for VSkinScrolling and touchscreen are 757.6 *ms* and 564.5 *ms*, respectively. VSkinScrolling is only 193.1 *ms* slower than the touchscreen. Most of the participants were surprised that they could perform the scrolling gestures on the back of the device without any hardware modification.

8.6.2 *VSkinSwiping: Swiping gesture.*

Application usage: VSkinSwiping enables swiping gesture on the back of the mobile device, as shown in Figure 14(b). Users hold the phone with their hands, first touch the back of the device as the trigger of VSkinSwiping and then perform the swiping gesture to classify pictures. We use the top microphone for touch detection and both microphones to measure the swiping direction.

Performance evaluation: *VSkinSwiping achieves usability comparable to the touchscreen for the swiping gesture.* We performed the same practicing step before the test as in the case of VSkinScrolling. We asked the users to use the left/right swiping gesture to classify random pictures of cats/dogs (ten pictures per task), *i.e.*, swipe left when saw a cat and swipe right when saw a dog. The mean movement time is defined as the average time used for classifying one picture using the swiping gesture. Each participant performed the task for 20 times using VSkinSwiping and the touchscreen. As shown in Figure 15(b), the mean movement time of one swiping gesture for VSkinSwiping and touchscreen are 1205 *ms* and 1092 *ms*, respectively. On average, VSkinSwiping is only 112.7 *ms* slower than the touchscreen. The average accuracy of swiping direction recognition of VSkinSwiping is 94.5%.

9 CONCLUSIONS

In this paper, we develop VSkin, a new gesture-sensing scheme that can perform touch sensing on the surface of mobile devices. The key insight of VSkin is that we can measure the touch gestures with a high accuracy using both the structure-borne and the air-borne acoustic signals. One of our future work direction is to extend VSkin to flat surfaces near the device, *e.g.*, sensing touch gestures performed on the table by placing a smartphone on it.

ACKNOWLEDGMENTS

We would like to thank our anonymous shepherd and reviewers for their valuable comments. This work is partially supported by National Natural Science Foundation of China under Numbers 61472185, 61373129, and 61321491, JiangSu Natural Science Foundation No. BK20151390, and Collaborative Innovation Center of Novel Software Technology.

REFERENCES

[1] Md Tanvir Islam Aumi, Sidhant Gupta, Mayank Goel, Eric Larson, and Shwetak Patel. 2013. Doplink: Using the doppler effect for multi-device interaction. In *Proceedings of ACM UbiComp*.

[2] Ian H Chan. 2010. Swept Sine Chirps for Measuring Impulse Response. Stanford Research Systems.

[3] Ke-Yu Chen, Daniel Ashbrook, Mayank Goel, Sung-Hyuck Lee, and Shwetak Patel. 2014. AirLink: sharing files between multiple devices using in-air gestures. In *Proceedings of ACM UbiComp*.

[4] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. GripSense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of ACM UIST*.

[5] Emilio Granell and Luis A Leiva. 2017. βTap: back-of-device tap input with built-in sensors. In *Proceedings of of MobileHCI*.

[6] Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. Soundwave: using the doppler effect to sense gestures. In *Proceedings of ACM CHI*.

[7] Chris Harrison, Julia Schwarz, and Scott E Hudson. 2011. TapSense: enhancing finger interaction on touch surfaces. In *Proceedings of ACM UIST*.

[8] Jouni Hartikainen, Arno Solin, and Simo Särkkä. 2011. *Optimal filtering with Kalman filters and smoothers.* Aalto University School of Science.

[9] Seongkook Heo and Geehyuk Lee. 2011. Forcetap: extending the input vocabulary of mobile touch screens by adding tap gestures. In *Proceedings of ACM MobileHCI*.

[10] Sungjae Hwang, Andrea Bianchi, and Kwang-yun Wohn. 2013. VibPress: estimating pressure input using vibration absorption on mobile devices. In *Proceedings of ACM MobileHCI*.

[11] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2017. A smartphone prototype for touch interaction on the whole device surface. In *Proceedings of of MobileHCI*.

[12] Jian Liu, Yingying Chen, Marco Gruteser, and Yan Wang. 2017. VibSense: Sensing Touches on Ubiquitous Surfaces through Vibration. In *Proceedings of IEEE SECON*.

[13] Jian Liu, Chen Wang, Yingying Chen, and Nitesh Saxena. 2017. VibWrite: Towards Finger-input Authentication on Ubiquitous Surfaces via Physical Vibration. In *Proceedings of ACM CCS*.

[14] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of ACM CHI*.

[15] Makoto Ono, Buntarou Shizuki, and Jiro Tanaka. 2015. Sensing touch force using active acoustic sensing. In *Proceedings of ACM TEI*.

[16] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. 2007. Beepbeep: a high accuracy acoustic ranging system using COTS mobile devices. In *Proceedings of ACM SenSys*.

[17] Corey R Pittman and Joseph J LaViola Jr. 2017. Multiwave: Complex Hand Gesture Recognition Using the Doppler Effect. In *Proceedings of ACM GI*.

[18] Branislav M Popovic. 1992. Generalized chirp-like polyphase sequences with optimum correlation properties. *IEEE Transactions on Information Theory* 38, 4 (1992), 1406–1409.

[19] Swadhin Pradhan, Eugene Chai, Karthikeyan Sundaresan, Lili Qiu, Mohammad A Khojastepour, and Sampath Rangarajan. 2017. RIO: A Pervasive RFID-based Touch Gesture Interface. In *Proceedings of ACM MobiCom*.

[20] A Rodrìguez Valiente, A Trinidad, JR García Berrocal, C Górriz, and R Ramírez Camacho. 2014. Extended high-frequency (9–20 kHz) audiometry reference thresholds in 645 healthy subjects. *International journal of audiology* 53, 8 (2014), 531–545.

[21] Wenjie Ruan, Quan Z Sheng, Lei Yang, Tao Gu, Peipei Xu, and Longfei Shangguan. 2016. AudioGest: enabling fine-grained hand gesture detection by decoding echo signal. In *Proceedings of ACM Ubicomp*.

[22] Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. 2015. Exploring user-defined back-of-device gestures for mobile devices. In *Proceedings of ACM MobileHCI*.

[23] Ke Sun, Wei Wang, Alex X. Liu, and Haipeng Dai. 2018. Depth aware finger tapping on virutal displays. In *Proceedings of ACM MobiSys*.

[24] Thomas L Szabo and Junru Wu. 2000. A model for longitudinal and shear wave propagation in viscoelastic media. *The Journal of the Acoustical Society of America* 107, 5 (2000), 2437–2446.

[25] Yu-Chih Tung and Kang G Shin. 2016. Expansion of human-phone interface by sensing structure-borne sound propagation. In *Proceedings of ACM MobiSys*.

[26] Michael Vorländer. 2007. *Auralization: fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality.* Springer Science & Business Media.

[27] Hao Wang, Daqing Zhang, Junyi Ma, Yasha Wang, Yuxiang Wang, Dan Wu, Tao Gu, and Bing Xie. 2016. Human respiration detection with commodity WiFi devices: do user location and body orientation matter?. In *Proceedings of ACM Ubicomp*.

[28] Wei Wang, Alex X. Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of ACM MobiCom*.

[29] Teng Wei and Xinyu Zhang. 2015. mTrack: High-Precision Passive Tracking Using Millimeter Wave Radios. In *Proceedings of ACM MobiCom*.

[30] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid touch: a see-through mobile device. In *Proceedings of ACM UIST*.

[31] Pui Chung Wong, Hongbo Fu, and Kening Zhu. 2016. Back-Mirror: back-of-device one-handed interaction on smartphones. In *Proceedings of Symposium on Mobile Graphics and Interactive Applications, SIGGRAPH ASIA*.

[32] Xiang Xiao, Teng Han, and Jingtao Wang. 2013. LensGesture: augmenting mobile interactions with back-of-device finger gestures. In *Proceedings of ACM ICMI*.

[33] Sangki Yun, Yi-Chao Chen, and Lili Qiu. 2015. Turning a Mobile Device into a Mouse in the Air. In *Proceedings of ACM MobiSys*.

[34] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. 2017. Strata: Fine-Grained Acoustic-based Device-Free Tracking. In *Proceedings of ACM MobiSys*.

[35] Cheng Zhang, Aman Parnami, Caleb Southern, Edison Thomaz, Gabriel Reyes, Rosa Arriaga, and Gregory D Abowd. 2013. BackTap: robust four-point tapping on the back of an off-the-shelf smartphone. In *Proceedings of ACM UIST*.

[36] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. 2010. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of IEEE CODES+ISSS*.

[37] Zengbin Zhang, David Chu, Xiaomeng Chen, and Thomas Moscibroda. 2012. Swordfight: Enabling a new class of phone-to-phone action games on commodity phones. In *Proceedings of ACM MobiSys*.