FingerIO: Using Active Sonar for Fine-Grained Finger Tracking

Rajalakshmi Nandakumar[1], **Vikram Iyer**[1], **Desney Tan**[2], **Shyamnath Gollakota**[1]
DUB Group, University of Washington[1] Microsoft Research[2]
{rajaln, vsiyer, gshyam}@cs.washington.edu desney@microsoft.com

ABSTRACT
We present *fingerIO*, a novel fine-grained finger tracking solution for around-device interaction. FingerIO does not require instrumenting the finger with sensors and works even in the presence of occlusions between the finger and the device. We achieve this by transforming the device into an active sonar system that transmits inaudible sound signals and tracks the echoes of the finger at its microphones. To achieve sub-centimeter level tracking accuracies, we present an innovative approach that use a modulation technique commonly used in wireless communication called Orthogonal Frequency Division Multiplexing (OFDM). Our evaluation shows that fingerIO can achieve 2-D finger tracking with an average accuracy of 8 mm using the in-built microphones and speaker of a Samsung Galaxy S4. It also tracks subtle finger motion around the device, even when the phone is inside a pocket. Finally, we prototype a smart watch form-factor fingerIO device and show that it can extend the interaction space to a 0.5×0.25 m^2 region on either side of the device and work even when it is fully occluded from the finger.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): User interfaces—*Input devices and strategies*

Author Keywords
Around device interaction; Finger tracking; Active sonar; Mobile sensing

INTRODUCTION
In this paper, we explore the following question: Can we track the user's finger around the device and can we do this even when they are occluded from each other? A positive answer would allow the user to interact in more expressive ways, utilize the screen fully without hand blockage and also enable new interaction scenarios. For instance, the user can use her finger as a pen to provide input over a much larger surface area than the smartphone. She can also perform subtle finger motion around the device that can be tracked even when the phone is in a pocket. Such a capability would also benefit smaller devices such as smart watches that could track the user's finger, even when fully occluded from it or when the watch is on a different plane from the interaction surface. Existing solutions for finger tracking, however, either instrument the finger with sensors [21, 40, 10, 12] or use cameras/infrared sensors at the device [9, 22, 23]. The former approach is burdensome while the latter does not work with occlusions.

We present *fingerIO*, a fine-grained finger tracking system that does not require instrumenting the finger with sensors and works even with occlusions between the finger and the device. FingerIO tracks finger motion in the region around existing smartphones, and achieves an average 2-D tracking accuracy of 8 mm. It also tracks subtle finger motion around the device, even when the phone is in the pocket. Using fingerIO, we also prototype a smart watch-form factor device that can track the finger, while extending the interaction space to a 0.5×0.25 m^2 region on either side of the device. Further, the watch can continue tracking the finger even when they are fully occluded from each other.

Our key insight is to transform mobile devices (e.g., smartphones) into active sonar systems. At a high level, we transmit inaudible 18-20 kHz sound waves from the device's speaker. These signals get reflected from the finger and can be recorded as an echo at the device's microphones. Finger motion results in changes to the arrival time of the echo at multiple microphones. By tracking the time at which these echoes begin, fingerIO continuously tracks the finger location. The echo from the finger however is noisy and hence estimating the exact time when the echo is received, in the presence of all other reflections, is challenging. To appreciate the challenge, microphones on today's mobile devices have a sampling rates of 48 kHz [27]. Given the speed of sound in air, an error of just 3-4 samples in estimating the start of the echo results in a 2.1-2.8 cm error in the estimated distance from each microphone. Since the finger location is determined by computing the distance from multiple microphones, the finger tracking error would be much higher. In addition, speakers and microphones on mobile devices typically run independently and do not sample at the exact same time. This results in an additional sampling offset, further increasing the tracking error.

FingerIO addresses the above technical challenges by borrowing insights from wireless communication. In wireless systems, the transmitter and the receiver are not synchronized and do not sample at the same time. Wireless receivers however are designed to estimate the sampling offset for every transmission so as to decode the transmitted information. Modern wireless system uses a modulation technique called Orthogonal Frequency Divison Multiplexing (OFDM)

CHI'16, May 7–12, 2016, San Jose, CA, USA.

http://dx.doi.org/10.1145/2858036.2858580

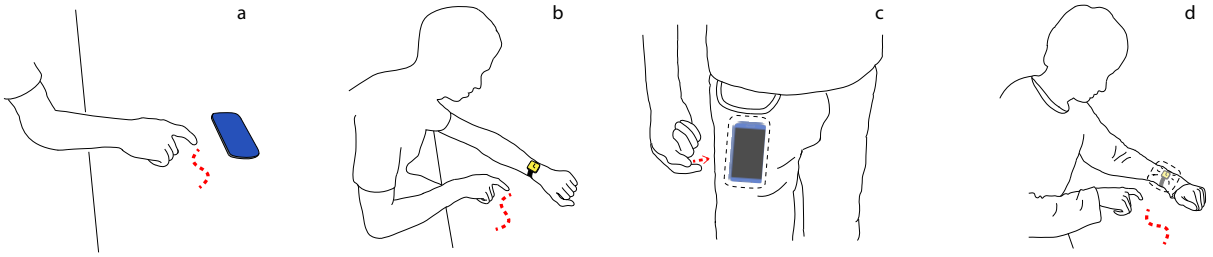

Figure 1: **Applications of fingerIO.** a) Transform any surface into a writing interface; b) provide a new interface for smartwatch form factor devices; c) enable gesture interaction with a phone in a pocket; d) work even when the watch is occluded.

to achieve this goal. Inspired by this, fingerIO achieves accurate finger tracking using OFDM. At a high level, we create 18-20 kHz OFDM symbols by computing the FFT of N random bits and generating N samples. We then compute a cyclic suffix of S samples, as shown in the Fig. 2, which we transmit from the speaker. The key property of OFDM with cyclic suffixes is that, a sample error in identifying the beginning of the symbol, translates linearly into phase changes in the frequency domain; these changes can be extracted at the microphones using an FFT. For instance, an error of E samples translates into the phase linearly increasing from 0 to 2Eπ at the output of the FFT. Further, fractional sampling errors that occur because of sampling drifts between the microphone and the speaker also result in a similar phase change at the output of the FFT. Using this, fingerIO corrects for the sampling errors and achieves fine-grained finger tracking.

While active sonar has been proposed before to perform coarse-level gesture recognition [15, 11], we are not aware of prior attempts to use it to achieve fine-grained finger tracking on existing devices. Hence, our contributions are:

1. We introduce a novel approach to fine-grained finger tracking for around device interaction that does not require instrumenting the finger with sensors and works even in the presence of occlusions between the finger and the device.
2. We propose and develop an active sonar solution to finger tracking. To achieve this goal with high accuracies, we introduce algorithms that use the properties of OFDM to track the changes in echoes caused due to finger motion.
3. We implement our design on a Samsung Galaxy S4 using its in-built speaker and microphones and demonstrate finger tracking around the phone, with no additional hardware. We also built a prototype of our design in a smart watch form factor device using off-the-shelf hardware.
4. We conduct experimental evaluations that show that fingerIO can achieve average 2-D finger tracking accuracies of 8 mm and 1.2 cm at 169 frames/s for the smartphone and smart watch prototypes. Further, it accurately tracks subtle finger motion even when the phone in a pocket as well as with the smart watch fully occluded from the finger.

RELATED WORK

Prior work falls in three key domains.

Near Device Interaction. Prior work in this domain can be broadly categorized as either requiring instrumenting the

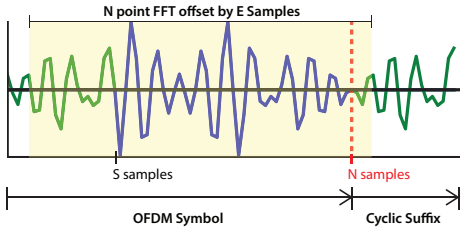

Figure 2: **OFDM signal structure.** The first S samples of the OFDM samples are appended to the end to create the cyclic suffix. Taking an N point FFT of the signal beginning at offset E will include part of the cyclic suffix resulting in a phase difference.

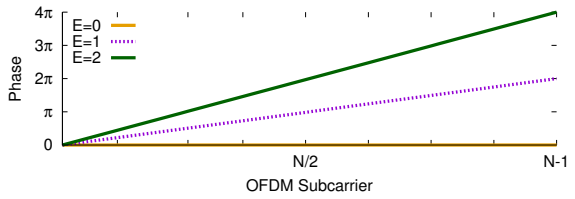

Figure 3: **OFDM phase change due to sample error.** The cyclic suffix introduces a phase change of $2E\pi$ across the OFDM subcarriers when the beginning of the OFDM symbol is estimated incorrectly by E samples. This phase can be used to correct the sample error and improve the accuracy of the system.

human body or vision based systems. iRing [28] uses an infrared sensor to recognize rotation, bending and external force on the finger. LightRing [21] designs a ring-form factor device that consists of a infrared proximity sensor and a 1-axis gyroscope to measure the finger flexion and rotation respectively. Magic finger [40] designs a finger-worn device that uses an optical mouse sensor and a micro RGB camera to sense touch as well as the texture of the surface being touched. Fingerpad [10] is a nail-mounted device that uses the tip of the index finger as a touchpad using magnetic tracking. uTrack [12] proposes to instrument the back of the fingers with magnetometers and the back of the thumb with a magnet to enable a 3D input system using magnetic sensing.

Systems such as Digits [22] and SideSight [9] do not require instrumenting the finger with sensors but use vision/infrared sensors and hence do not work with occlusions. Specifically, Digits [22] uses a wrist-worn 3D infrared camera to recover the full pose of the user hand. SideSight [9] instruments the sides of a mobile device (e.g., smartphone) with an array of infrared proximity sensors which detect the presence and position of the fingers in line-of-sight of the sensors. Hover-

flow [23] uses an array of IR sensors placed along the edges of the phone to detect hand gestures. [35] uses the built-in camera of a phone to detect in-air gestures and tracking of hand parts in the camera's view. [41] attaches an omni-directional mirror to the phone's camera to increase its field of view. In contrast to these systems, fingerIO leverage the microphones and speakers that are common on most mobile devices to design an active sonar system that can track finger motion in the vicinity of the device. Further, fingerIO uses acoustic signals and hence can operate even when the finger is occluded.

Finally, PocketTouch [34] detects finger-strokes through fabric using a custom capacitive sensing hardware. In contrast, fingerIO can track subtle finger motion through pockets with existing smartphones, without the need to touch the device.

Active Acoustic Localization. Device localization systems such as Cricket [29], Doplink [6], Spartacus [37], and Shake and Walk [17] localize and determine the direction of a device movement using acoustic transmissions. AAmouse [42] uses Doppler shifts to track the phone position using anchor devices in the room. Whiteboard technologies such as Mimio [4] use an active stylus with ultrasound and infrared and localize using an anchor device placed at the corner of the board. In contrast, fingerIO is a device-free localization solution that tracks an uninstrumented finger using existing devices; this is achieved using the properties of OFDM. While OFDM has been used in wireless communication and device localization systems [19, 20] due to its resilience to multipath, we are unaware of prior work that uses it for finger tracking.

SoundWave [15] leverage Doppler shifts from acoustic transmissions to recognize gestures such as moving the hand towards or away from the device. Airlink [11] and Surfacelink [13] use Doppler shifts and surface-mounted piezoelectric sensors respectively to detect hand waving gestures from one device towards the other. These design focus on predefined set of hand and arm gestures and are not designed for finger tracking. Finally, ApneaApp [27] tracks the periodic breathing movements in a sleep environment using FMCW reflections of the inaudible transmissions from the phone.

Chirp microsystems [3] designs on-chip ultrasonic rangefinders operating at 217 kHz with a bandwidth of 12 kHz using an array of seven transducers to perform angle-of-arrival techniques and get an angular resolution of 15 degrees [31]. The key motivation was to reduce the power consumption of cameras and instead use an ultrasonic design. Our approach differs from this in three key ways. First, we use existing devices with one to two microphones and do not need any custom chips. Second, we leverage OFDM and show that using just 2 kHz of bandwidth we can achieve centimeter level localization. Third, while the performance of these on-chip designs has not been evaluated for finger tracking, we apply our design in various finger tracking applications and show that it can enable a number of interesting interaction scenarios.

Passive Acoustic Localization. [38, 24] use the audible sounds made when clicking on a keyboard to snoop on the keys typed by the users. [33] localizes taps on solid aluminum and glass surfaces by using a piezoelectric shock sensor to sense the sound propagation through the material. Toffee [39] uses vibro-acoustic piezo sensors to find the direction of the audible sound and vibration waves that propagate as the user taps on a table. It achieves a mean angular resolution of 4.3 and 18.3 degrees using four sensors at the corners of a laptop and smartphone respectively. More recently, [8] uses contact microphones attached to the surface to distinguish between various impact events such as touch, knock and swipe. In contrast to this work, fingerIO uses an active sonar approach that transmits inaudible signals and achieves centimeter level finger tracking both on surfaces as well as in the air.

RF-based Gesture Systems. WiSee [32], AllSee [18] and SideSwipe [43] uses Wi-Fi, TV and cellular transmissions respectively to recognize coarse hand, arm and leg gestures. WiTrack [5] uses custom radar transmissions to detect pointing gestures. WiDraw [36] tracks the arm motion in the vicinity of a Wi-Fi device using transmissions from 20-30 other Wi-Fi devices in the network. Google has reported that project Soli is exploring the use of 60 GHz radar to recognize subtle finger gestures [14]. None of these approaches have been demonstrated on smartphones and require custom sensor hardware. We also believe that the active sonar approach introduced in this paper is more attractive for two reasons: RF signals propagate at the speed of light and so to get a centimeter resolution requires processing GigaHertz of bandwidth. In contrast, the speed of sound is significant lower and hence with 48 kHz, fingerIO could achieve centimeter level accuracies. Further, our approach uses microphones and speakers that are already available on existing mobile devices and hence the bar for adoption is much lower.

FingerIO

FingerIO achieves centimeter level finger tracking by transforming the mobile device into an active sonar system. At a high level, we transmit an inaudible sound signal in the frequency range of 18-20 KHz from the device's speaker. These signal are reflected by all the objects in the environment and can be recorded by the microphone as echoes. When the user moves her finger, the time of arrival for the corresponding echo changes. By comparing the echo profile from one instance to the other, we can extract the echoes that correspond to the moving finger. As described earlier, since these echoes are noisy, the challenge is in accurately identifying the beginning of the echo so that we can achieve finger tracking with high accuracies. FingerIO leverages a modulation technique called OFDM to achieve this goal.

In the rest of this section, we first explain the properties of OFDM. Next, we describe how we generate OFDM transmissions using speakers in the 18-20 kHz range. We then show how fingerIO uses OFDM to measure the distance of a moving finger from a single microphone. Finally, we discuss how to use two microphones to achieve 2D tracking.

Understanding OFDM

Orthogonal frequency division multiplexing (OFDM) is a common modulation technique used in modern wireless communication systems including Wi-Fi and LTE. In this section, we focus on the properties of OFDM that are relevant to our

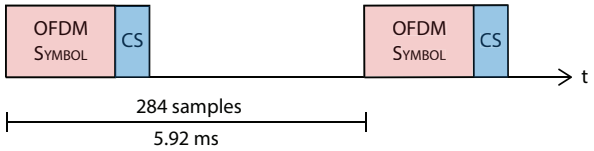

Figure 4: **FingerIO transmissions at the speaker.** The 84 samples for the OFDM symbol and the cyclic suffix are followed by 200 samples of silence. This silence duration is sufficient to receive echoes from all objects within 1 m from the device. Given a 48 kHz sample rate the above transmissions, the above transmissions achieves a frame rate of 169 Hz.

design. See [7, 16, 30] for more extensive discussion about OFDM. OFDM splits up the bandwidth into orthogonal subcarriers and transmits data independently on each of the subcarriers. For example, in Wi-Fi, the 20 MHz bandwidth is divided into 64 subcarriers each with a width of 312.5 kHz. The data is then transmitted on each of these 64-subcarriers. To achieve this, OFDM uses Fourier transforms. Say we divide the bandwidth into N subcarriers and transmit the data bit $\mathbf{X}_n$ on the nth subcarrier. OFDM generates the time-domain samples that are sent at the transmitter by performing an inverse Fast Fourier transform (IFFT) over these data bits, i.e.,

$$\mathbf{x}_k = \sum_{n=0}^{N-1} \mathbf{X}_n e^{i2\pi kn/N} \quad k = 0\ to\ N-1$$

This creates N time-domain samples, $\mathbf{x}_k$, that are then sent by the transmitter. An ideal receiver would receive these time-domain samples and performs a Fast Fourier transform (FFT) to recover the data bits, i.e.,

$$\mathbf{X}_n = \sum_{k=0}^{N-1} \mathbf{x}_k e^{-i2\pi kn/N} \quad n = 0\ to\ N-1$$

In practice since the receiver is not perfectly synchronized, it does not know the exact beginning of this symbol. To help address this problem, the transmitter sends a cyclic suffix which is a repetition of the first S time-domain samples as shown in Fig. 2. To see why this helps, say the receiver has an error of E samples in estimating the beginning of the OFDM symbol. Given this error, it would perform an FFT over the N time-domain samples that are offset by E, as shown in the figure. Since we use a cyclic suffix, these new time-domain samples can be written as $\mathbf{x}_{(k+E)modN}$. Now when the receiver performs an FFT over these samples, we get,

$$\begin{aligned}\mathbf{X^E}_n &= \sum_{k=0}^{N-1} \mathbf{x}_{(k+E)modN} e^{-i2\pi kn/N} \\ &= \mathbf{X}_n e^{i2\pi En/N}\end{aligned}$$

We see that the new frequency-domain data is the same as the original data but with an additional phase that depends on the error in estimating the beginning of the symbol (E). It also linearly increases with the subcarrier number n as shown in the Fig. 3. For example, an error of one sample results in the phase linearly increasing from 0 to 2π across the subcarriers. More generally, an error of E samples, results in the phase increasing from 0 to $2E\pi$ across the N OFDM subcarriers.

To summarize, if the receiver knows the data bits, $\mathbf{X}_n$, that are been transmitted, it can compute the error E in estimating the beginning of the OFDM symbol. Further, the above analysis holds even when there is a fractional time offset between the transmitter and the receiver, allowing us to estimate it. We leverage this OFDM property in our design to achieve centimeter-level finger tracking accuracies.

FingerIO transmissions at the speaker

FingerIO generates OFDM signals in the inaudible frequency range of 18-20 KHz which is then played by the device's speaker. There are however two key subtleties with creating an acoustic OFDM system: i) acoustic devices do not use oscillators to generate and transmit a carrier frequency. This is because the audio sampling rate of 48 kHz is sufficient to cover the entire frequency range of typical speaker and microphones. ii) The input to the speaker is a 16-bit real number and cannot transmit the complex numbers generated by the IFFT. So fingerIO generates a carrier-less real value OFDM symbol. To do this, given a sampling rate of 48 kHz, we first split the operational frequency of 0–24 kHz into 64 subcarriers each spanning a width of 375 Hz. Since we want to operate only in the inaudible frequencies of 18–20 kHz, we set the subcarriers outside this range to zero. For the rest, we set each subcarrier to either +1 or -1. Then, we compute an IFFT that gives us a complex time-domain signal with 64 samples, $\mathbf{x}_k$. We convert these complex numbers into real values by computing the real value of these complex numbers. Specifically, we use the following transformation,

$$\mathbf{real}_k = |\mathbf{x}_k| cos(\angle \mathbf{x}_k)$$

Here $|.|$ and $\angle$ denote the amplitude and phase of the complex number. These 64 real values form the real-valued OFDM symbol that is transmitted by the speaker. We append the first 20 of these values to create a cyclic suffix that together is played repeatedly from the speaker, as shown in Fig. 4.

At a sampling rate of 48 kHz, these 84 samples (including the cyclic suffix) form a pulse that occupies 1.75 ms. We separate these pulses by 200 samples which in turn translates to a separation of 4.17 ms. We pick this duration to ensure that all the echoes from a distance of 1 m can arrive before the beginning of the next pulse. Given these parameters, we transmit an OFDM pulse once every 5.92 ms and achieve a frame rate of 169 Hz.

Measuring the distance from the microphone

The OFDM signal played by the speaker gets reflected off different objects including the hand and is then recorded by the microphone. To find the distance from the finger in the presence of all these reflections, we perform three key steps: (1) generate the echo profile of all the reflections at the microphone, (2) identify the echo corresponding to the moving finger, (3) process the OFDM symbol that is echoed by the finger to fine-tune its distance from the microphone.

Step 1. Generating the echo profile. While processing the recording at the microphone, we first identify the individual echoes that occur due to all objects within a distance. To do that, we perform correlation of the received signal with the

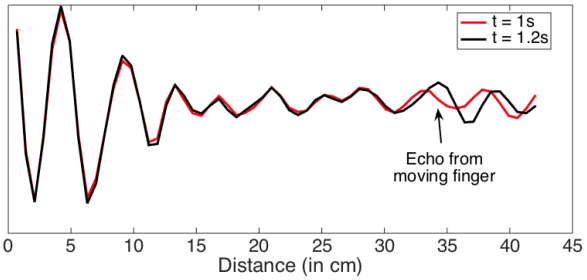

Figure 5: **Echo profile at two time instances.** Each peak indicates the arrival of an echo and the X-axis shows the corresponding distance computed based on the speed of sound. When the finger moves from 34 cm to 35 cm with respect to the device's microphone we can see a shift in the peak due to the change in the arrival time of the echo.

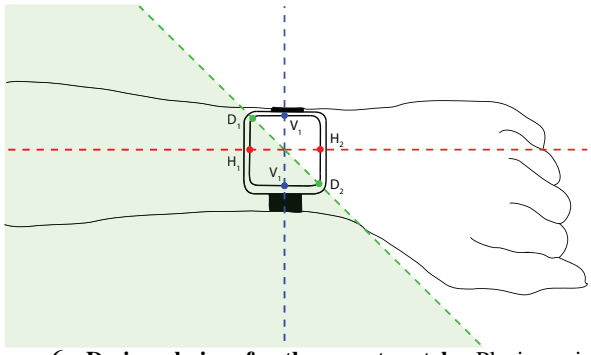

Figure 6: **Design choices for the smart watch.** Placing microphones on opposite sides of a smart watch either limit the user from drawing on the arm or on part of the surface below. Placing microphones along the diagonal could be a reasonable compromise as would allow users to interact both on the arm and a surface.

original OFDM symbol. The output of this correlation process is an echo profile as shown in Fig. 5. Each of the peaks in this profile corresponds to the beginning of an echo from an object around the mobile device. This can be translated to a distance at which the reflecting object is present from the microphone and speaker. While OFDM has good autocorrelation properties, given noise, this step gives us the beginning of each echo with an error of 2–3 samples. This translates to an error in distance estimate of 1.4–2.1 cm.

Step 2. Identifying the echo corresponding to the finger. When the finger moves, the time of arrival for the echo corresponding to the finger changes. Fig. 5 shows the echo from the finger as it moves from a distance of 34 cm to 35 cm from the microphone. We can see from the figure that the position of the echo changes as the finger moves around the device. We identify this change by performing a subtraction between the echo profiles of consecutive OFDM pulses. To recognize if a change has occurred at a specific distance value, we use a threshold based approach. Specifically, if the changes across all the distance values in the echo profile are smaller than a threshold, we conclude that there has been no finger motion in the vicinity of the device. Changes that are greater than the threshold value are designated as those corresponding to motion. We set this threshold value to 15% of the amplitude of the speaker signal as heard directly by the speaker. We pick this relative threshold value to ensure that amplitude fluctuations that occur because of non-linearities in microphones and speakers do not lead to false positives.

The width of a human adult finger is around a centimeter. This is close to the distance resolution (0.7 cm) provided by an active sonar system where the sampling rate if 48 kHz. Hence, when the finger moves, one would expect a change to occur at only one-two samples corresponding to the old and new locations of the finger. However, close examination of Fig. 5 reveals that the changes occur not only at the finger's location but also near its vicinity. This is because a finger motion also causes a hand movement that traces a similar path as the finger. For our design, however, this means that we need to correctly identifying the changes caused by the finger motion in the presence of changes corresponding to other small hand movements. We achieve this by picking the closest sample, where the change crosses the threshold value. The reason why this works is that the finger is always closer to the microphone compared to the rest of the hand. Further, as the user draws using their finger, the maximum displacement (and hence changes) occurs at the tip of the finger.

Step 3. Fine-tuning the distance from the microphone. Finally, once we identified the echo corresponding to the finger, we use the properties of OFDM to accurately estimate the beginning of this echo. Specifically as described earlier, when we perform an FFT over the echo of an OFDM pulse, any error in estimating the beginning of the echo translates to a linear phase shift at the output of the FFT. Thus, at a high level, we first compute a 64-point FFT starting at the approximate location of the echo as estimated by the correlation in step 1. We then use the linear phase shift at the output of the FFT to accurately estimate the beginning of the echo, which in turn gives us the distance from the microphone. We note that since the microphones receive a real OFDM signal, we need to first transform it into a complex signal before performing the FFT. To do this, we use a technique called negative sideband suppression [26] where we obtain a complex representation by setting the negative frequency components of the signal to zero. This overall process allows us to fine-tune the distance estimate of the finger from the microphone.

2D finger tracking using two microphones

To perform 2D tracking, we compute the distances of the finger with respect to two microphones and combine them to measure the 2D location. Note that the distance we are measuring is actually the sum of the distance traveled by the signal from the speaker to the finger and the distance traveled by the echo from the finger to the microphone. This means that given the distance measurements computing on a single microphone, the finger can lie on any point in a 2D space where the sum of its distance from the microphone and the speaker is equal to the measured distance. Said differently, the finger can lie on an ellipse where the speaker and the microphone locations are the two focii of the ellipse and the distance measured is twice the length of the major axis of the ellipse. Given the measurements from two microphones,

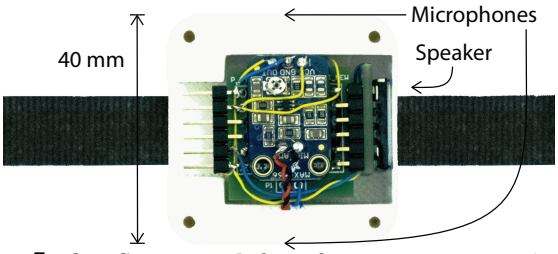

Figure 7: **Our Smart watch form-factor prototype.** The prototype consists of two microphones (embedded in the case) and a speaker mounted in a 3D printed case (shown transparent in figure) with a Velcro strap.

fingerIO can find the intersection of the corresponding two ellipses to narrow down the possibilities for the finger location.

While in general, two ellipses can intersect in up to four points, in our case there can only be two intersection points. This is because, two ellipses that share a focal point (the speaker) can intersect in a maximum of two points [25]. These two points lie on either side of the line joining the two microphones. This means that the system is symmetrical along the line joining the two microphones and the 2-D location can lie on either side of the phone. The implication for our design is that we cannot distinguish between when the user is moving her finger on either side of the line connecting the two microphones. For the smartphone use case, this means that the user could draw on only one side of the phone at any instance. It is acceptable since the user is likely to interact in the region between the phone and themself. This however raises interesting design choices for the smartwatch. If we place the two microphone in positions H_1 and H_2 as shown in Fig. 6 then the user can easily interact on either side of the arm but cannot draw along her arm. Placing them in the vertical positions V_1 and V_1 would allow us to interact on the arm but on a truncated region between the hand and the user. A reasonable compromise could be to place them along the diagonal positions D_1 and D_2. Another solution is to use three microphones and eliminate this symmetry. The implementation in this paper uses the horizontal locations, H_1 and H_2 for the microphone positions.

Using two microphones, our design can track the motion on any 2D plane. Motion along a different plane will be projected to the plane along which the microphones lie. In the case of a smart watch, the user places the wrist on the interaction surface (e.g., table). Note that the width of a user's arm is around a couple of centimeters and the plane along which the smart watch lies can be approximated to be parallel to that the interaction surface. Thus, the projections of the finger-motion along the interaction surface are similar to that of the actual motion. Note that using more than two microphones, e.g., two microphones near the display and an additional one along the strap, will avoid the need for the above approximations.

IMPLEMENTATION

We implement fingerIO prototypes on two platforms.

Off-the-shelf smartphones. We developed a third party Android app and tested it on a Samsung Galaxy S4 smartphone. The phone has one speaker at the bottom and two microphones one at the top right and the other at the bottom right spaced apart by 13.5 cm. The app generates the OFDM symbols and uses the AudioTrack class that is an existing Android API to play it on the phone's speaker. The app is set to record simultaneously from both microphones using the stereo mode using Android's built-in AudioRecord class.

Smart watch form-factor device. While Apple smart watch has effective speakers and microphones that allow them to make calls and use Siri [2], they are currently not as programmable as smart phones and further have only a single microphone. While 1D tracking can be achieved with a single microphone, since we also want to demonstrate 2D tracking, we built a simple prototype consisting of two microphones and a speaker mounted in a 3D printed case with a Velcro strap, as shown in Fig. 7. The electret microphones (CMA-4544PF-W, CUI Inc) are mounted on opposite sides of the watch case spaced apart by 40 mm. The output from the microphones are connected to a Adafruit development board [1] containing a preamplifier (MAX4466, Maxim) which is then connected to the NI MyDAQ for data acquisition. The same OFDM signal used for the smartphone experiments was supplied to a class D amplifier (MAX98306, Maxim) that takes input from the phone's 3.5 mm headphone jack and drives the speaker. The system was powered using the Keysight E3631A DC power supply.

EVALUATION

We recruited ten participants (5 female and 5 male) between the ages of 20–25; none of them were provided any monetary benefits. The participants were asked to draw any pattern they wanted using their finger and we evaluate fingerIO's accuracy in various scenarios for both the smartphone and smart watch implementations. Since the participants could draw any pattern, we use a second touch-based mobile device that collects the ground truth data. Specifically, we place an Android smartphone at different locations around our fingerIO-enabled smartphone/smart watch and ask the participants to draw freely using their finger on this device. We use the Android OnTouch Event Listener to obtain the pixel locations that the participants touch. We also extended the draw API to simultaneously display the path traversed by the user's finger. The OnTouch API only provides the locations as pixels in the screen space. We convert this into a screen location (in cm) by scaling the pixel value with the number of pixels per centimeter. We then offset this screen location with the distance between the fingerIO-enabled smart watch/smartphone and the smartphone used for ground truth data collection.

In this rest of this section, we first present 2D finger tracking accuracy for both the smartphone and the smart watch prototype implementations in line-of-sight and occluded scenarios. We then evaluate fingerIO's interaction surface (i.e., 2D-range) for both the prototypes. Finally, we address unintentional motion tracking with fingerIO.

FingerIO's Finger Tracking Accuracy

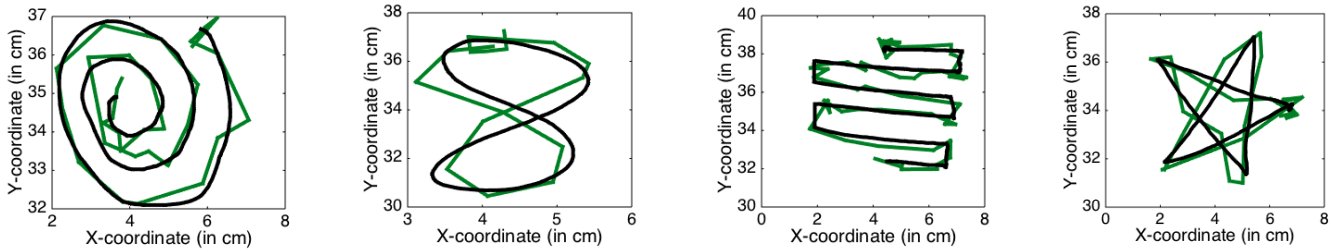

Figure 8: **Traces computed using fingerIO for smartphone setup.** The figures show both the ground truth trace (black lines) as well as fingerIO's estimated trace (green lines) for four of our participants.

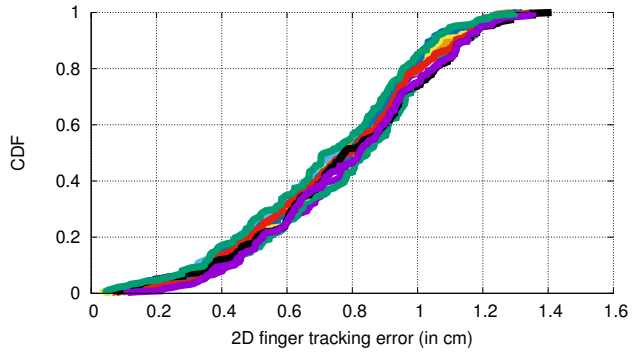

Figure 9: **Finger tracking accuracies with smartphone.** Cumulative distribution functions (CDFs) for the 2D tracking errors for each of the ten participants. The median tracking error across all the participants is 8 mm.

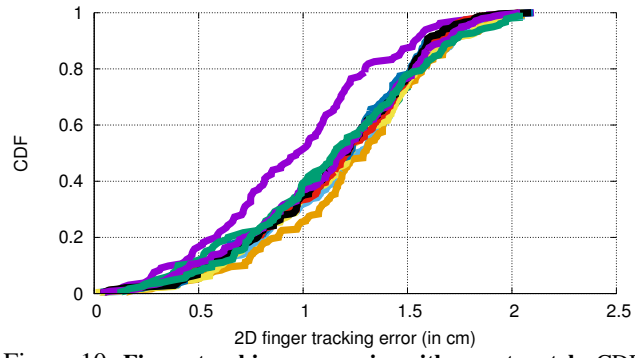

Figure 10: **Finger tracking accuracies with smart watch.** CDFs for the 2D tracking errors for each of the ten participants. The mean tracking error was 1.2 cm.

We run experiments in one of the offices in our organization. We place a smartphone running fingerIO on a table, lengthwise in front of the participants. A separate smartphone that is used to collect the ground truth was placed 25 cm from the fingerIO-enabled smartphone. The participants were given a demonstration of the system and were allowed to practice with it. Once the participants became familiar with the setup, they were instructed to draw a pattern of their choice that consisted a single continuous finger movement. The participants were asked to repeat their patterns thrice. All the participants placed their palm on the table while using their finger to draw. We process all this data and compute the average tracking error from the ground truth data. To compute the finger tracking error, we measure the average least perpendicular distance of each point along the trace computed by fingerIO with the ground truth.

Fig. 9 shows the CDF of the tracking error for all ten participants. The figure shows that the tracking error is similar across all participants. Further, the median error across all participants was 0.8 cm. This demonstrates that fingerIO achieves its goal of centimeter-level finger tracking in practice. Fig. 8 shows four of the most complex patterns picked by the participants. The black line in these traces is the ground truth trace of the participants and the green traces are the ones computed by fingerIO, subsampled by a factor of five. The figures show that the two traces are close to each other and that fingerIOs algorithm can also deal with intricate motion such as those shown in the traces.

We repeat the above set of experiments with our smart watch prototype. In particular, we ask the participants to wear the smart watch on their arm and place it on a table. The participants drew patterns on the table at a distance of around 25 cm from the smart watch. As before, we compute the finger tracking error by measured the least perpendicular distance between the traces for fingerIO and the ground truth.

Fig. 10 shows the CDF of the tracking error for the smart watch experiments. As with the smartphone, the tracking error is similar for most participants. The median error across all participants was around 1.2 cm. We observe that the accuracy with the smart watch setup is lower than with the smartphone scenario. This is because we see higher noise levels at the microphones we use in our smart watch prototype, even in the absence of any finger motion. This is likely because of insufficient isolation between the speaker and the microphone in our hardware setup. Better isolation, could in principle improve the accuracies further.

FingerIO's Interaction Surface

We evaluate how fingerIO extends the interaction surface around our prototype smart watch and smartphone.

FingerIO's interaction surface with the smart watch prototype. The participants were asked to wear the watch on their hand and place the hand on the table. We divide the area in front of the hand into 5×5 cm grids as shown in Fig. 11. Within each grid, the participants draw a straight line with a length of 4 cm twice. In each trial, we compute the trajectory of the finger with fingerIO. We compare this with the ground truth collected from our experiments. To compute the finger tracking error, we measure the average least perpendicular distance of each point along the trace computed by

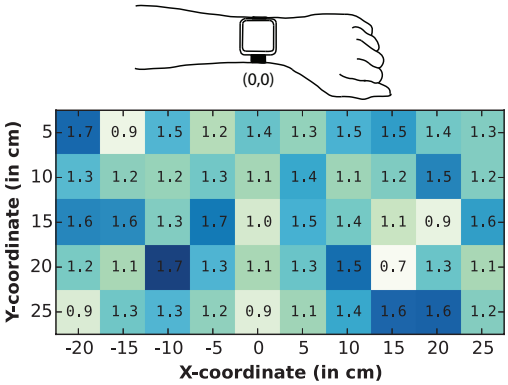

Figure 11: **Interaction surface for smart watch.** The surface around the hand was divided into 5×5 cm grids and the average finger tracking accuracy was computed for each grid. fingerIO enables an $0.5 \times 0.25\ m^2$ interaction surface with an average finger tracking accuracy of 1.2 cm.

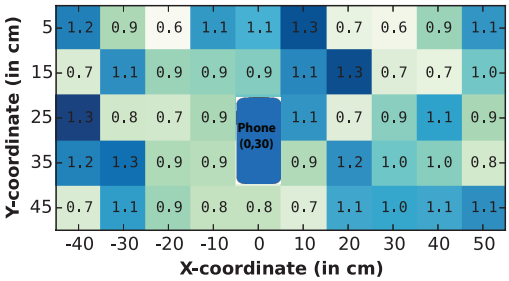

Figure 12: **Interaction surface with smartphone.** The surface around the device was divided into 10×10 cm grids and the average tracking accuracy was computed for each grid.

fingerIO with the ground truth. We compute the average error in each grid by averaging across trials in that grid. The figure shows that the interaction surface with the smart watch is a 0.5×0.25 m^2 region on one side of the arm. Since the performance is symmetric across the line joining the two microphones, the actual interaction surface is double is region. The average error is about 1.2 cm and is uniform across the region. We note that beyond the grids shown in the figure, the accuracies quickly drop off with distance. As observed before, because of insufficient isolation between the speaker and the microphone in our hardware prototype, we see increased noise even in the absence of any motion. This contributes to slightly higher tracking errors than a smartphone scenario.

FingerIO's interaction space with the smartphone. In contrast to our smartwatch prototype, the output power of the smartphone speaker is set to the maximum value of 15 allowed by the Android API. This is around 10 dB greater than that in our smart watch implementation. Further, commercial devices such as smartphones have better isolation between microphones and speakers. So next we evaluate the interaction space for fingerIO using a off-the-shelf smartphone. To measure this, we place the smartphone on a table and divide the area around the phone into 10×10 cm grids. Since we would like to see what the maximum interaction surface area can be

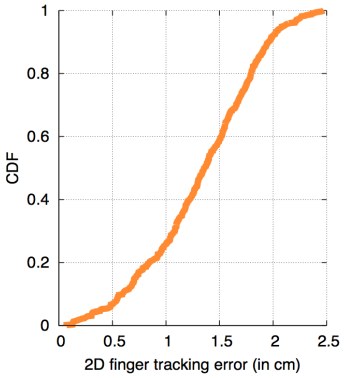

Figure 13: **Smart watch occluded behind a jacket.** The figure plots the CDF for the 2D tracking errors for five participants when the smart watch was occluded behind a jacket. The median tracking error across all the participants is 1.35 cm compared to 1.2 cm when the smart watch was not occluded.

with good isolation and higher transmission power, we place the phone screen down. This ensures that the transmissions on the speaker that is on the back of the phone can reach much further distances and imitates a better version of the smart watch prototype. We perform the same experiments as in the previous scenario and measure the tracking error in each grid locations. Fig. 12 shows the smartphone and the finger tracking accuracy for the grids around it. The figure shows that, fingerIO expands the interactive surface to about a 0.5 m^2 region around the smartphone. The average error within this range is less than 1 cm and is fairly uniform; beyond this distance however this error increases to 3 cm. Further, these accuracies are similar on all four sides of the smartphone, demonstrating that fingerIO can perform well for different phone orientations from the user. The increased interaction area demonstrates that with better isolation and higher power, we can achieve a larger interaction space and better tracking accuracies. We believe however, a tracking range of less than a meter is sufficient for some interaction applications when the user is interacting with their smartphones or watches.

FingerIO in Occluded Scenarios

We evaluate two specific occlusion scenarios.

Smart watch occluded behind a jacket. We ask the participants to wear a polyester jacket that fully covers the smart watch. We use a similar setup as before to compute the tracking error. The participants were first asked to wear the smart watch and place their hand on the table as shown in fig. 13(a). We then asked the participants to draw any pattern with their finger. We run these experiments with five of our ten participants where each of them was asked to repeat the pattern they drew 3 times. Fig. 13(b) shows the CDF of the tracking errors between fingerIO and the ground truth across all five participants. The median error across all the participants was 1.35 cm. This is slightly greater than the error in the absence of the occlusion. This demonstrates that fingerIO operates even in the presence of occlusions.

Smartphone in the pocket. Next, we run experiments with a smartphone placed inside the pocket of a pair of jeans,

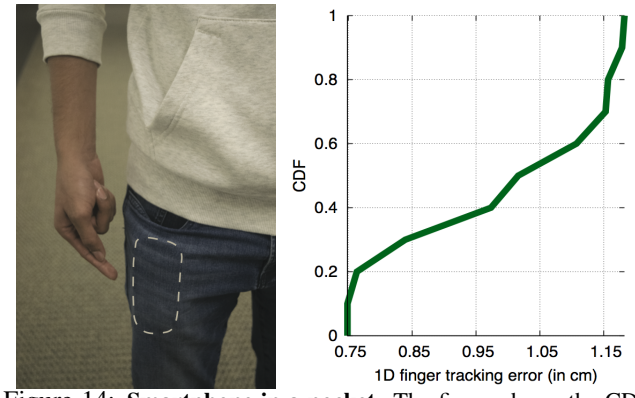

Figure 14: **Smartphone in a pocket.** The figure shows the CDF for the 1D tracking errors for five participants when the smart phone was inside the pocket. The median tracking error across all the participants is 1 cm compared to 8 mm when the smartphone was not occluded.

with the back of the smartphone facing outward as shown in fig. 14(a). Five of the participants were instructed to perform a finger swipe in the air in front of the pocket. The swipe motion consists of the thumb moving over the index finger. We configure the fingerIO algorithm to compute the distance moved by the finger by processing the data from a single microphone. The participants were allowed to perform the above swipe finger motion from any angle to the smartphone. We ask the participants to only move their fingers at a resolution of the prominent lines on their finger. This provides us with the ground truth data. On average, the participants moved their index finger by around 5 cm. We compute the error as the difference in the distance estimated by fingerIO and the ground truth motion. Fig. 14(b) shows the CDF of the errors in this computed distance. The plots show an average error of 1 cm across all participants. In all cases, our algorithm correctly identifies the direction of the finger motion, i.e., either towards or away from the phone. This demonstrates the feasibility of through-the-pocket finger motion tracking.

Addressing Unintended Motion with FingerIO

In a system like fingerIO, we need a mechanism to inform the device the beginning and end times of when it should track the finger. This would prevent random motion in the vicinity of the device from being confused for finger motion for the purpose of interaction. To do this, we introduce a double swipe as a start and stop motion. A swipe is defined as a finger motion in a straight line for a length of at least 4 cm. A double swipe requires the user to perform a swipe motion in two opposite directions; we detect this by looking for distance values linearly increasing for 4 cm and then decreases for at least 4 cm. We consider a double swipe that is performed within a range of 5 cm from our device to be our start/stop motion. We pick the 5 cm range to ensure that similar finger motion that occurs at a farther distance is not confused for the start/stop motion.

Experiments: To evaluate how well this start/stop motion works, we ask our ten participants to perform the double swipe motion with both our smartphone and smart watch setups within 5 cm from the devices. Each user performs the finger motion twice for both the devices. For each of these motions, if our algorithm fails to identify it as a start/stop motion, we consider it to be a false negative. To compute false positives, we ask our participants to draw random patterns other than the double swipe within 10 cm from the device for a period of 30 seconds. Each user performed it twice and then we look for the double swipe gesture over a total duration of 10 minutes across all the participants. The start/stop motion detected by our algorithm during this duration, are considered to be false positives.

False negatives: Our algorithm detected the double swipe start/stop motion 19 out of the 20 times the participants performed it with the smartphone. Similarly, we detected this motion 18 out of 20 times when it was performed with the smart watch. The undetected motions were because during a double swipe gesture, the participants move their whole hand along with the finger. While our algorithm tracks only the motion at the closest distance, i.e., the finger, for the three start/stop motions that were missed, the participants forcefully moved their entire arm in a different direction than the finger. Since our current implementation can only deal with a single motion direction, it was confused for this. This is however less likely to be the case if we use the active sonar approach to track multiple concurrent motions. We also note that the current false negative rate is still acceptable and could likely become more reliable as the users get accustomed to our system.

False positives: With the smart watch prototype, we did not detect any start/stop motion during the 10 min duration, i.e., the number of false positives during this duration was zero. This is because our algorithm requires a strict double swipe pattern within a small range from the device. With the smartphone, however we detected two start/stop motions during the 10 min duration. Further analysis of the data showed that the two false positives came from a single participant who drew a wiggly pattern tracing 4 cm distance in both directions. This triggered our algorithm to classify this as a start/stop motion. Given that the rest of the participants did not have any false positives, we believe that the double-swipe motion is sufficient in most scenarios.

Addressing Random Motion in the Surroundings

The maximum operational range of fingerIO is less than a meter. This is a key advantage, as it remains unaffected by motion in the surroundings. In this section, we evaluate this property by measuring the finger tracking accuracy of a participant while another participant creates random motion in the surroundings. We conduct experiments with one subject drawing a straight line of distance 4 cm. A phone running fingerIO is placed around 25 cm from the finger location. While the subject performs finger motion, another interfering subject continuously waves their hand toward the first subject. We repeat the experiments with two different distances for the second subject. We compute the finger tracking accuracy for each of these distance values.

Fig. 15 plots these accuracies as a function of distance of the interfering subject from the phone. The plot shows that when the interfering subject is within 50 cm from the phone, the

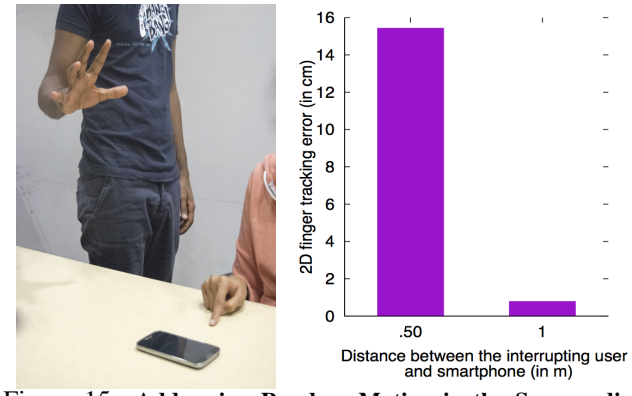

Figure 15: **Addressing Random Motion in the Surrounding.** The figure shoes the 2D tracking errors when there was a second interrupting user in the environment. The accuracy decreases when there is a stronger motion within 50 cm of the device. However, the accuracies remain consistently high when the interrupting user is beyond one meter.

finger tracking accuracies significantly suffer. This is not a fundamental limit of our sonar-based design. Rather this is because our current algorithm is designed only to track a single motion. In principle, one may design algorithms to track independent motions concurrently from multiple distance values since the echoes arrive at different times for each of these distances. The key observation however is that for distances greater than a meter for the interfering subject, the finger tracking accuracies are again high. This is because, at these distances, the reflections caused due to the motion from the interfering subject are significantly attenuated and hence are weaker compared to the echoes from the finger motion that is performed at a closer distance.

LIMITATIONS AND FUTURE DIRECTIONS

We discuss the limitations of our current design as well as opportunities to improve it.

Tracking 3-D motion and non-cursive writing. Our current implementation uses two microphones and cannot achieve 3-D tracking of finger motion. This is however not a fundamental limitation of our approach and can be addressed by using a third microphone. Specifically, three microphones can be used to triangulate the position of the finger in the 3-D space enabling 3-D finger tracking. A similar problem occurs with non-cursive writing, where the user could slightly lift her finger from the 2D surface to move it across different points on the surface. Using only two microphones, this would be tracked as a continuous motion and our algorithm will project this motion on the 2-D drawing plane as part of the input. We note that this is a similar issue faced by camera-based systems where the user draws with her finger in front of a camera. One direction worth exploring is to incorporate a third microphone on a different plane (which can be done on the smart watch setup) and use it to identify this 3-D motion.

Tracking multiple concurrent motions. While this paper focuses on tracking a single finger, in principle, the algorithms presented could track concurrent changes from multiple fingers as long as they occur at different distances from the microphones. This can be used to detect pitch, zoom out and zoom in gestures that require multiple fingers moving at the same time. It can also be used to detect and separate the finger/body motion from other people near the device. We expect the algorithms for doing so to be similar to radar based approaches such as Google Soli. Exploring how well this works in practice is not in the scope of this paper.

FingerIO's power consumption. A full-charged Samsung Galaxy S4 running fingerIO lasts around four hours. As with other always-on mobile gesture sensing techniques, there are a number of power-accuracy tradeoffs that could be made. For instance, we transmit OFDM pulses once every 5.92 ms which translated to a frame rate of 169 frames/s. Since human motion is unlikely to change at this rate, we can operate fingerIO with a much lower frame rate. Further, as shown in Fig. 8, subsampling by a factor of five, still gives us values that look similar to the ground truth. Optimizing this further and reducing the power consumption, would be a worthwhile future direction.

Finger tracking with a moving device. The finger tracking algorithms developed in this paper work under the assumption that the phone or the smart watch is static. To address mobility of the devices, we envision using the accelerometer/gyro, already present in the devices we imagine operating on, to compensate for the motion in our algorithms. Intuitively this would be similar to imitating a synthetic aperture radar system (SAR). We leave the development of such algorithms for future work.

CONCLUSION

We introduce a novel active sonar design for fine-grained finger tracking that does not require instrumenting the finger with sensors and works even in the presence of occlusions between the finger and the device. We validate our design on a Samsung Galaxy S4 using its in-built speaker and microphone and demonstrate finger tracking around the phone. We also built a prototype in a smart watch form factor device using off-the-shelf hardware.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their feedback. We also thank Bryce Kellogg and Vamsi Talla for helpful discussions about the hardware design. This work was funded in part by a Google Faculty Award and the National Science Foundation under award CNS-1420654.

REFERENCES

1. Adafruit. `https://www.adafruit.com/products/1063`.
2. Apple Watch - Guided Tour: Phone Calls. `https://www.youtube.com/watch?v=_Zj5KisMVv8`.
3. Chirp Microsystems. `http://www.chirpmicro.com/technology.html`.
4. A MimioTeach Interaction Whiteboard. `http://www.mimio.com/en-NA/Products/MimioTeach-Interactive-Whiteboard.aspx`.
5. Adib, F., Kabelac, Z., Katabi, D., and Miller, R. C. 3D Tracking via Body Radio Reflections. *NSDI 2014*, 317-329.
6. Aumi, M. T. I., Gupta, S., Goel, M., Larson, E., and Patel, S. DopLink: Using the Doppler Effect for Multi-device Interaction. *UbiComp 2013*, 583-586.

7. Boleskei, H. Principles of MIMO-OFDM wireless systems. *2004*.
8. Braun, A., Krepp, S., and Kuijper, A. Acoustic Tracking of Hand Activities on Surfaces. *WOAR 2015*, 1-5.
9. Butler, A., Izadi, S., and Hodges, S. SideSight: Multi-"Touch" Interaction Around Small Devices. *UIST 2008*, 201-204.
10. Chan, L., Liang, R.-H., Tsai, M.-C., Cheng, K.-Y., Su, C.-H., Chen, M. Y., Cheng, W.-H., and Chen, B.-Y. FingerPad: Private and Subtle Interaction Using Fingertips. *UIST 2013*, 255-260.
11. Chen, K.-Y., Ashbrook, D., Goel, M., Lee, S.-H., and Patel, S. AirLink: Sharing Files Between Multiple Devices Using In-air Gestures. *UbiComp 2014*, 565-569.
12. Chen, K.-Y., Lyons, K., White, S., and Patel, S. uTrack: 3D Input Using Two Magnetic Sensors. *UIST 2013*, 237-244.
13. Goel, M., Lee, B., Islam Aumi, M. T., Patel, S., Borriello, G., Hibino, S., and Begole, B. SurfaceLink: Using Inertial and Acoustic Sensing to Enable Multi-device Interaction on a Surface. *CHI 2014*, 1387-1396.
14. Google. Project Soli. `https://www.youtube.com/watch?v=_Zj5KisMVv8`.
15. Gupta, S., Morris, D., Patel, S., and Tan, D. SoundWave: Using the Doppler Effect to Sense Gestures. *CHI 2012*, 1911-1914.
16. Heiskala, J., and Terry, J. OFDM Wireless LANs: A Theoretical and Practical Guide. *Sams publishing, 2001*.
17. Huang, W., Xiong, Y., Li, X.-Y., Lin, H., Mao, X., Yang, P., and Liu, Y. Shake and walk: Acoustic direction finding and fine-grained indoor localization using smartphones. *INFOCOM 2014*, 370-278.
18. Kellogg, B., Talla, V., and Gollakota, S. Bringing Gesture Recognition to All Devices. *NSDI 2014*, 303-316.
19. Khyam, M., Alam, M., Lambert, A., Benson, C., and Pickering, M. High precision multiple ultrasonic transducer positioning using a robust optimization approach. *ISSPIT 2013*, 192-197.
20. Khyam, M., Alam, M., and Pickering, M. OFDM based low-complexity time of arrival estimation in active sonar. *OCEANS 2014*, 1-5.
21. Kienzle, W., and Hinckley, K. LightRing: Always-available 2D Input on Any Surface. *UIST 2014*, 157-160.
22. Kim, D., Hilliges, O., Izadi, S., Butler, A. D., Chen, J., Oikonomidis, I., and Olivier, P. Digits: Freehand 3D Interactions Anywhere Using a Wrist-worn Gloveless Sensor. *UIST 2012*, 167-176.
23. Kratz, S., and Rohs, M. HoverFlow: Expanding the Design Space of Around-device Interaction. *MobileHCI 2009*, 1-8.
24. Liu, J., Wang, Y., Kar, G., Chen, Y., Yang, J., and Gruteser, M. Snooping Keystrokes with Mm-level Audio Ranging on a Single Phone. *MobiCom 2015*, 142-154.
25. MacNeish. The Intersections of Two Conic Sections with a Common Focus. *The American Mathematical Monthly 28*, 6/7, 260–262.
26. Nandakumar, R., Chinatalapudi, K., Padmanaban, V., and Venkatesan, R. Dhwani : Secure Peer-to-Peer Acoustic NFC. *Sigcomm 2013* 2013.
27. Nandakumar, R., Gollakota, S., and Watson, N. Contactless Sleep Apnea Detection on Smartphones. *Mobisys 2015*, 45-57.
28. Ogata, M., Sugiura, Y., Osawa, H., and Imai, M. iRing: Intelligent Ring Using Infrared Reflection. *UIST 2012*, 131-136.
29. Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. The Cricket Location-support System. *Mobicom 2000*, 32-43.
30. Proakis, J., and Salehi, M. Digital Communications. *McGraw-hill, 2007*.
31. Przybyla, R., Tang, H.-Y., Guedes, A., Shelton, S., Horsley, D., and Boser, B. 3D Ultrasonic Rangefinder on a Chip. *IEEE Journal of Solid-State Circuits 2015*, 320-334.
32. Pu, Q., Gupta, S., Gollakota, S., and Patel, S. Whole-home Gesture Recognition Using Wireless Signals. *Mobicom 2013*, 27-38.
33. Reju, V., Khong, A., and Sulaiman, A. Localization of Taps on Solid Surfaces for Human-Computer Touch Interfaces. *IEEE Trans. on Multimedia 2013*, 1365-1376.
34. Saponas, T. S., Harrison, C., and Benko, H. PocketTouch: Through-fabric Capacitive Touch Input. *UIST 2011*, 303-308.
35. Song, J., Sörös, G., Pece, F., Fanello, S. R., Izadi, S., Keskin, C., and Hilliges, O. In-air Gestures Around Unmodified Mobile Devices. *UIST 2014*, 319-329.
36. Sun, L., Sen, S., Koutsonikolas, D., and Kim, K.-H. WiDraw: Enabling Hands-free Drawing in the Air on Commodity WiFi Devices. *Mobicom 2015*, 77-89.
37. Sun, Z., Purohit, A., Bose, R., and Zhang, P. Spartacus: Spatially-aware Interaction for Mobile Devices Through Energy-efficient Audio Sensing. *MobiSys 2013*, 263-276.
38. Wang, J., Zhao, K., Zhang, X., and Peng, C. Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-grained Keystroke Localization. *MobiSys 2014*, 14-27.
39. Xiao, R., Lew, G., Marsanico, J., Hariharan, D., Hudson, S., and Harrison, C. Toffee: Enabling Ad Hoc, Around-device Interaction with Acoustic Time-of-arrival Correlation. *MobileHCI 2014*, 67-76.
40. Yang, X.-D., Grossman, T., Wigdor, D., and Fitzmaurice, G. Magic Finger: Always-available Input Through Finger Instrumentation. *UIST 2012*, 147-156.
41. Yang, X.-D., Hasan, K., Bruce, N., and Irani, P. Surround-see: Enabling Peripheral Vision on Smartphones During Active Use. *UIST 2013*, 291-300.
42. Yun, S., Chen, Y.-C., and Qiu, L. Turning a Mobile Device into a Mouse in the Air. *Mobisys 2015*, 15-29.
43. Zhao, C., Chen, K.-Y., Aumi, M. T. I., Patel, S., and Reynolds, M. S. SideSwipe: Detecting In-air Gestures Around Mobile Devices Using Actual GSM Signal. *UIST 2014*, 527-534.