2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

EchoWrite: An Acoustic-based Finger Input System Without Training

Yongpan Zou*, Qiang Yang*, Rukhsana Ruby*, Yetong Han*, Sicheng Wu*, Mo Li‡, Kaishun Wu*†
*College of Computer Science and Software engineering, Shenzhen University
†PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China
‡School of Computer Science and Engineering, Nanyang Technological University
{yongpan,wu}@szu.edu.cn; rukhsana.afroz@gmail.com; limo@ntu.edu.sg
{yangqiang2016,wusicheng2016}@email.szu.edu.cn; kynehhh@foxmail.com

***Abstract*—Recently, wearable devices have become increasingly popular in our lives because of their neat features and stylish appearance. However, their tiny sizes bring about new challenges to human-device interaction such as texts input. Although some novel methods have been put forward, they possess different defects and are not applicable to deal with the problem. As a result, we propose an acoustic-based texts-entry system, *i.e.*, EchoWrite, by which texts can be entered with a finger writing in the air without wearing any additional device. More importantly, different from many previous works, EchoWrite runs in a training-free style which reduces the training overhead and improves system scalability. We implement EchoWrite with commercial devices and conduct comprehensive experiments to evaluate its texts-entry performance. Experimental results show that EchoWrite enables users to enter texts at a speed of 7.5 WPM without practice, and 16.6 WPM after about 30-minute practice. This speed is better than touch screen-based method on smartwatches, and comparable with previous related works.**

***Index Terms*—Acoustic signals, Texts input, HCI**

I. Introduction

Due to the limited sizes of screens, wearable devices bring about new challenges for human-device interaction such as texts entry. Researchers have proposed various novel approaches for entering texts on mobile devices. But they possess different shortcomings. Speech recognition enables people to convey commands without touching, but leaks privacy in public, degrades performance in noisy environment and is inconvenient in certain occasions. Radio-frequency (RF) signals and inertial sensors have also been utilized to design texts-input systems [1]–[9]. But they either need specialized hardware, or requiring users to attach/carry devices with them. Prior works have also proposed to use acoustics to track finger motion precisely [10]–[12]. Nevertheless, they require multiple microphone-speaker pairs which are not available for most commercial devices especially tiny smart devices. As a result, we ask such a question: *can we design a novel text-input interface for existing commercial devices that does not require any additional hardware and works in a device-free style?* Besides entering texts on tiny devices, such a system can also handle cases where hands are wet or oil-scalded. Fig. 1 shows the possible application scenarios.

In response to this question, we propose EchoWrite in this paper, a system that enables users to enter texts without touching a device, wearing additional hardware and conducting system training. The high-level idea is to construct a mapping between simple gestures and English letters, and infer texts by recognizing fine-grained finger gestures via pervasive acoustic sensors (*i.e.*, microphone and speaker). However, there are three challenges to deal with in order to transform the idea into a practical system. First, it deserves great effort to design an input scheme mapping English letters to finger gestures with high learnability and efficiency. Second, for the sake of comfort, the designed input gestures should be fine-grained. But this in turn induces challenges to analyze subtle signal changes for accurate finger gesture recognition. Third, considering possible errors in performing and recognizing gestures, it is required to conduct appropriate correction. But how to design an efficient method is not straightforward since there are exponential possible cases.

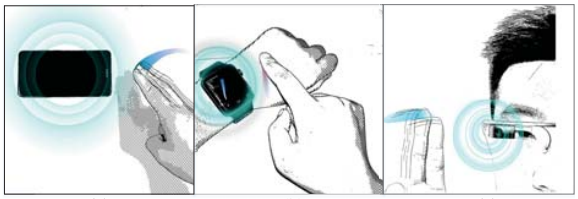


Fig. 1. Possible scenarios where EchoWrite can be applied. EchoWrite cannot only be on wearable devices such as smartwatches and smartglasses to deal with the inconvenience caused by small screens, but also on mobile devices such as smartphones and tablets to handle the cases where hands are wet or oil-scalded.

In the design of EchoWrite, we have taken the following measures to resolve the above challenges. First, we decompose basic letters into six basic strokes and groups them according to their first or second strokes when they are written naturally. Meanwhile, we design a finger gesture for each group by directly utilizing the first stroke or making slight modification. As a result, all letters are classified into different groups and mapped with basic stroke gestures. Since this mapping relationship is constructed based on users' writing habits, it enjoys benefits of favorable learnability, memorability and efficiency. Second, we transform time-domain signals into spectrogram and extract unique Doppler shift profiles for stroke gestures.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00082

By carefully designing signal processing flowchart, we can track frequency shifting along with finger movements. More importantly, the extracted profiles are intrinsically related with gestures themselves, which makes EchoWrite get rid of labor-intensive training overhead. Third, we propose a stroke correction method and adopt reasonable simplification to improve its efficiency based on physical insights and experimental results. By this means, we can achieve a good trade-off between performance and efficiency. We have implemented a prototype of EchoWrite on a Huawei Mate 9 and conduct extensive experiments to evaluate its performance. The results show that, with EchoWrite, users can enter texts at a speed of 7.5 WPM (*i.e.*, word per minute) without repetitive practice, and the speed increases to 16.6 WPM after about 30-minute practice. Compared with existing texts-entry methods for smartwatches, EchoWrite exceeds them not only in speed but also in user experience. In a nutshell, the contributions of our work can be summarized as follows.

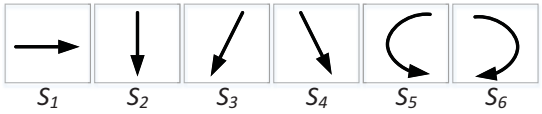
- We propose a device-free texts-input system based on pervasive acoustic sensors in smart devices. It is scalable to devices of different form factors, robust to background noises, and requires no system training process.
- We propose an input scheme that matches basic English letters with simple finger gestures with high learnability. We also develop effective data processing methods to extract minute Doppler shifts caused by fine-grained finger gestures, and design accurate texts inference method.
- We implement a prototype, *i.e.*, EchoWrite, on a smartphone and conduct comprehensive experiments to evaluate its performance. The results demonstrate that our system enables users to enter texts at a favorable speed up to 16.6 WPM.

The remainder of this paper is organized as follows. In Sec. II, we give an overview of EchoWrite. Following that, we give details of system design in Sec. III. We describe the implementation and experiments in Sec. IV, and display performance evaluation in Sec. V. Last in Section VI and Section VIII, we discuss related work and conclude the paper respectively.

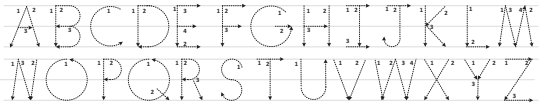
II. System Overview

A. Input Scheme Design

The key idea of our input scheme is similar to T9 soft keyboard in which each key represents a set of English letters and texts can be recognized in a fuzzy way. Inspired by the fact that uppercase English letters can be decomposed into six basic strokes as shown in Fig. 2(a) [5], we design the input scheme by assigning 26 letters to basic strokes following two principles. The first one is high learnability. That means, the input scheme should incur as light mental workload as possible for users. Thus, the matching relationship between letters and strokes should be natural and memorable, such as grouping letters according to their first or second strokes as shown in Fig. 2(b) recommended for kid's learning English [13]. The second principle is the uniqueness of Doppler profiles



(a) Basic strokes of English letters



(b) The stroke order of English letters

Fig. 2. The design of input scheme.

(see Sec. III-B) of different strokes, which means each stroke should induce unique Doppler shift pattern. Taking the above into account, we design an input scheme as shown in Fig. 3.

We also conduct a preliminary user study to evaluate the learnability and efficiency our input scheme in which we assume that our system recognizes strokes with an accuracy of 90% [1]. We recruit 6 participants with 3 females and 3 males from our campus without knowledge of our project before. Before experiments, we spend time to introduce the input scheme until they have fully understood. And then we request them to write out stroke sequences corresponding to 300 most frequent and randomly shuffled words selected from Corpus of Contemporary American English (COCA) [14]. In this process, participants are allowed to see each word for only once, and are reminded but not permitted to make correction when they make errors. Each participant writes words continuously for 15 minutes. After that, we can obtain the accuracy of writing stroke sequences every one minute as shown in Fig. 4. As we can see, after 15 minutes' practice, participants can write out stroke sequences of different words with an average accuracy up to 98%. Moreover, we also investigate participants' words-input speed and accuracy after 15 minutes' practice as shown in Fig. 5 and Fig. 6 It is clear that participants can enter words with our scheme at a speed of 11 WPM and achieve a word-recognition accuracy of 90%[2].

B. The System Flowchart

Fig. 7 displays the flowchart of EchoWrite. When users intend to enter texts, they write a sequence of strokes with a finger near a device. Meanwhile, a built-in speaker emits inaudible single-tone audio signals of 20 KHz and a microphone samples echoes at 44.1 KHz simultaneously. Then we perform short time Fourier transform (STFT) with Hanning window on audio sequence to obtain corresponding spectrogram. To reduce noises in spectrograms and enhance Doppler shifts caused by finger movement, we apply a series of image processing techniques such as smoothing and binarization. After that, we propose a mean value-based contour extraction algorithm (MVCE) to figure out outlines of Doppler shifts

[1]Since we only evaluate the learnability of input scheme instead of the whole system, it is acceptable to make such an assumption.

[2]It is noted that this accuracy is obtained by multiplying assumed stroke-recognition accuracy (90%) by word sequence accuracy.

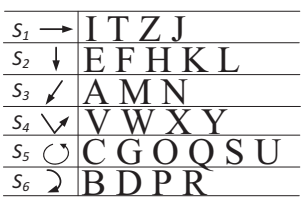


Fig. 3. The design of letter assignment scheme

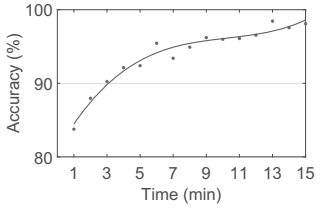


Fig. 4. The accuracy of stroke sequence along with practice time

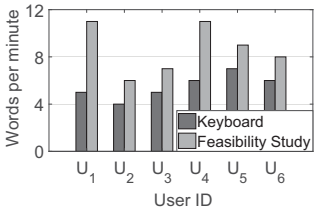


Fig. 5. The words-input speed of different participants in learnability study

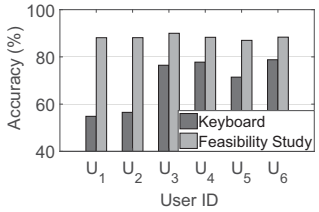


Fig. 6. The stroke-input accuracy of participants in learnability study

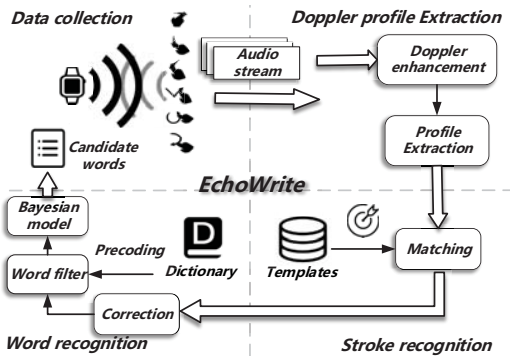


Fig. 7. The work flowchart of EchoWrite.

based on cleaned spectrograms as profiles of different strokes. At last, we perform dynamic time warping (DTW) to recognize strokes and utilize Bayesian language model to infer inputed texts (*i.e.*, words and sentences) by a list of candidates with corresponding probabilities. It is noted that the Doppler profile in our work is different from that in [15], since it is extracted from reflected signals instead of direct propagations.

III. System Design

A. Doppler enhancement

After receiving echoes, we perform short-time Fourier transform (STFT) with a Hanning window on signal sequence in order for framing audio files and extracting Doppler shifts caused by finger movements. To balance the time-frequency resolution and real-time performance of motion analysis, we empirically set the frame length (*i.e.*, FFT size) and window step size to be 8192 and 1024 samples (corresponding to 0.186 s and 0.023 s), respectively. Consequently, we can obtain the spectrogram of raw signals which displays the Doppler shifts of writing strokes as shown in Fig. 8(a). Then we concatenate the STFT results of every 5 frames and obtain the spectrogram of corresponding signal sequence. To reduce following computation overhead, we estimate the frequency range of interest by considering the Doppler shifts calculated by

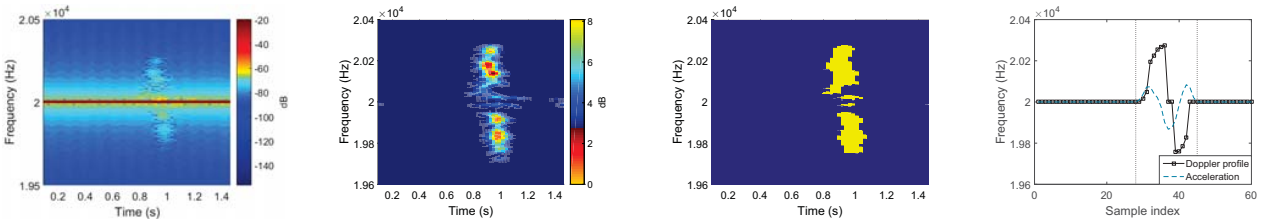
$$\Delta f = f_0 \times |1 - \frac{v_s \pm v_f}{v_s \mp v_f}| \quad (1)$$

where f_0, v_s and v_f represent frequency of emitted signals (20 KHz), speed of sound (340 m/s) and velocity of finger movement (4 m/s at maximum [16]), respectively. Thus the resultant frequency shift is about 470.6 Hz and the effective frequency range should be within $[19530, 20470]$ Hz. In this way, the column size of spectrogram to be processed can be reduced from 8192 to 350.

After that, we perform a 3×3 median filter to remove random noises. Following this, we subtract STFT of static frames (*i.e.*, without finger movements) from each following frame within a stroke in order to suppress static frequency components of background noises, direct transmission and multipath reflections. Specifically, we first compute the average STFT of initial 5 frames and subtract the corresponding result from each frame within a stroke. The rationale is that background noise and static multipath keep stable within each stroke lasting no more than 1 seconds. Besides static interference, there also exists bursting hardware noise whose power is larger than background noise but lower than echoes reflected from finger, which results in some random noisy points in spectrogram after spectral subtraction. To deal with this problem, we empirically define an energy threshold α below which elements of the matrix are set to be zero, and then apply a Gaussian filter with kernel size of 5 to smooth the spectrogram as shown in Fig. 8(b). We observe that α is closely related to hardware and set to be 8 in our system design. Later on, we sequentially conduct zero-one normalization to mitigate the effect of absolute amplitude and perform binarization with a threshold of 0.15. Further more, we also fill up the "holes" in binary spectrogram by performing a flood-fill operation on background pixels [17]. By the above operations, we can enhance the Doppler shifts of finger movements and obtain a relatively clean spectrogram as shown in Fig. 8(c).

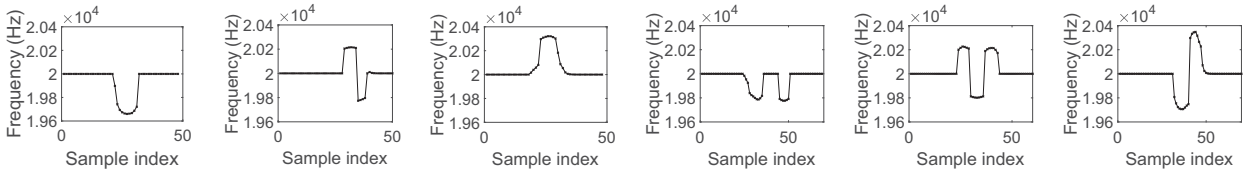
B. Extracting Doppler profile

The next step is to extract the Doppler profile of writing each stroke by figuring out the contour of spectrogram which depicts the overall trend of each stroke. The challenge is that due to multipath propagation, the received echoes are reflected from different parts of a moving hand and other body parts, which induces different Doppler shifts and makes it difficult to pick out the finger component. We observe that reflections from other parts except the finger possess lower frequency shifts due to relatively slow moving velocities. Nevertheless, it fails to work to simply select the frequency bin in a frame with maximum shift value (*i.e.*, Δf) considering random frequency



(a) The spectrogram of the whole signal sequence with STFT (b) After spectral subtraction and Gaussian smoothing (c) After spectrogram normalization and binarization (d) The extracted Doppler profile after performing MVCE

Fig. 8. Different stages of extracting the Doppler shifts profile of writing stroke S_2.



(a) S_1 (b) S_2 (c) S_3 (d) S_4 (e) S_5 (f) S_6

Fig. 9. The extracted Doppler profiles of different strokes according to the final version of input scheme.

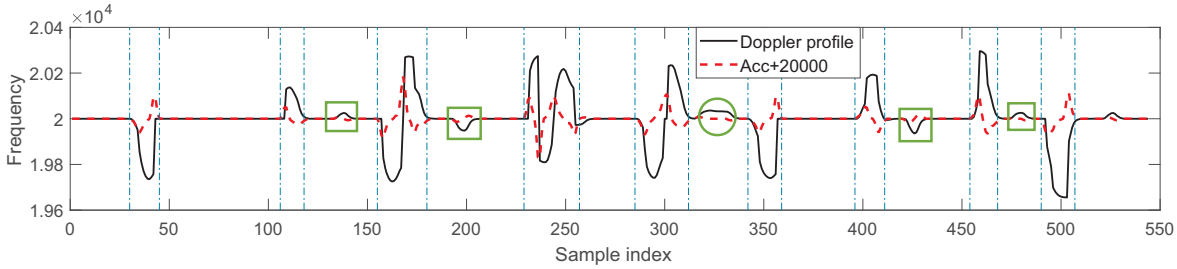


Fig. 10. The segmentation results for a series of writing strokes.

fluctuations. As a result, we propose a mean value-based contour extraction algorithm (MVCE) which first compares the average frequency shift with the center frequency to indicate overall finger movement direction. After that, MVCE picks out maximum frequency shift bin within positive or negative shift range based on the above result. After performing MVCE, we further smooth the raw Doppler profile to remove outliers with a moving average filter with sliding window size of 3 as shown in Fig. 8(d). The pseudocode of extracting Doppler profile is as shown in Algorithm 1.

Following that, we proceed to segment the continuous Doppler profile curve. To achieve this, we come up with an acceleration-based segmenting method which localizing start and end points of a stroke by detecting abrupt changes of acceleration of Doppler shifts. The key insight is that writing a stroke is a short-duration and high-acceleration process which induces rapid changes in Doppler shifts. Specifically, we first calculate first-order difference of obtained Doppler profile with a noise-robust differential approach as follows [18].

$$acc(i) = \frac{2 \times [y(i+1) - y(i-1) + y(i+2) - y(i-2)]}{8} \tag{2}$$

where $y(i)$ and $acc(i)$ represent Doppler shifts sequence (*i.e.*, finger moving speed) and its first-order differential (*i.e.*, acceleration), respectively. On the other hand, considering the principle of Doppler effect, we can obtain

$$f_t = \frac{1 \pm \frac{v_f}{v_s}}{1 \mp \frac{v_f}{v_s}} f_0 = \pm \frac{2 f_0 v_f}{v_s \mp v_f} + f_0 \approx \pm \frac{2 f_0 v_f}{v_s} + f_0 \tag{3}$$

since $v_f \ll v_s$. The first-order differential of Doppler shift ($\Delta f_t = |f_t - f_0|$) is as follows:

$$\Delta f_t^{'} = \frac{2 f_0}{v_s} v_f^{'} = \frac{2 f_0}{v_s} a \tag{4}$$

where a is the acceleration of finger movement which is about 0.15 m/s^2. Substituting this into the above equation, we know that the acceleration of Doppler shift in normal cases is about 40. Therefore, we set a acceleration threshold β of detecting stroke movement to be 40 and search the first point P whose value is above this threshold. From P, we search backward until the point whose Doppler shift is closest to zero. This point is then identified as the start point of a stroke, denoted by P_{start}. When a user finishes writing a stroke, he/she withdraws the finger and prepares to enter next stroke. In this process, the speed remains but the acceleration decreases notably. To identify the end point of a stroke, we set another acceleration

ALGORITHM 1: Doppler Profile Extraction

Input: Spectrogram matrix P, centre frequency bin cf
Output: Doppler shift profile $DopShift$

```
colNum=getColumNum(P);
DopShift(1:colNum)=cf; //initialization
for i=1:colNum do
    row=getNonNullRows(col(i)); // get non-null rows
      of the ith col
    if isNotEmpty(row) then
        meanValue=mean(row);
        if meanValue>cf then
            DopShift(i)=max(row);
        else
            DopShift(i)=min(row);
        end
    end
end
DopShift = SMA(DopShift); // SMA represents
  smoothed moving average filter
```

threshold γ to be $\frac{\beta}{2}$, *i.e.*, 20. If a point and its following nine points are detected to be less than γ, it is therefore identified as the end point of a stroke, denoted by P_{end}. By the above method, segments corresponding to stroke movements can be detected as shown in Fig. 10. As we can see, even with interfering Doppler shifts caused by multipath (labeled by green square) and irrelevant hand movement (labeled by circle), our method can effectively detect the start and end points of each stroke.

C. Stroke and Texts Recognition

Following profile extraction, the next step is to recognize strokes. Our insight is that different strokes produce unique Doppler profiles as shown in Fig. 9. More importantly, within a certain range, the Doppler profiles are intrinsically related with strokes themselves, while irrelevant with who performs them and how fast they are performed. Limited by the page limit, we leave out the proof of above statement. For stroke recognition, we make use of dynamic time warping (DTW) to match an extracted Doppler shift profile with templates pre-stored in the system. DTW is a mathematical method to compute similarity of time series which outperforms over other methods by taking stretch and contraction of into consideration [5].

Recognizing a stroke sequence enables us obtain different combinations of letters among which some are feasible words. To infer corresponding words, we make use of posterior probability maximization technique and find out feasible words with highest probability. To be specific, for a stroke sequence denoted by $I = s_1 s_2 ... s_n$, the output is a letter combination denoted by $w = l_1 l_2 ... l_n$, where $s_1, s_2, ..., s_n$ and $l_1, l_2, ..., l_n$ are mutually independent. By Bayesian rule,

$$P(w|I) = \frac{P(w,I)}{P(I)} \approx P(w,I) \tag{5}$$

since $P(I)$ is the same for every possible candidate. Further, we know that

$$\begin{aligned} P(w,I) &= p(w) \cdot P(I|w) \\ &= p(w) \cdot P(s_1 s_2 ... s_n | l_1 l_2 ... l_n) \\ &= P(w) \cdot \prod_{i=1}^{n} P(s_i|l_i) \end{aligned} \tag{6}$$

This is,

$$P(w|I) \approx P(w) \cdot \prod_{i=1}^{n} P(s_i|l_i) \tag{7}$$

where where $P(w)$ represents the prior probability of a candidate word obtained by existent vocabulary statistics, and $P(s_i|l_i)$ can be obtained from confusion matrix in stroke-recognition stage.

A straightforward approach to work out candidate words is directly applying Eq. 7 to calculate probabilities of feasible candidates and select the one with highest probability. However, such an approach ignores possible errors in stroke recognition caused by imperfection of our algorithm and user's writing strokes. Consequently, we propose a stroke correction technique to improve the accuracy of word recognition. In theory, stroke correction can be performed by deleting, inserting and substituting a certain number of letters in different positions within the word, which yet induces exponential increase of computation overhead. However, we notice that our acceleration-based stroke detection method can accurately detect strokes and ignore irrelevant motion interference. It means that we can take no account of deleting and inserting cases without much performance decline. What is more, according to our experiments, we find that there is little possibility for multiple strokes to be wrongly recognized simultaneously in a sequence. Consequently, when we perform correction for a candidate word, we only consider the substitution case with editing distance of 1 at each time.

To further improve computation efficiency, we make use of an experimental observation that errors of stroke recognition are mainly caused by high false positive rate of S_1 and false negative rate of S_5, only consider these possible substitution cases when performing stroke correction. This is because S_2, S_4 and S_6 are likely to be recognized as S_1, and S_5 are likely to be recognized as S_2 and S_6, as indicated by Fig. 9. Thus, we replace S_1 by S_2, S_4 and S_6, and substitute S_2 and S_6 for S_5 in turn . By substituting one stroke each time, we can obtain a set of corrected stroke sequences.

As aforementioned, each stroke sequence corresponds to multiple letter combinations of which a part are feasible words. To check a letter combination, we build up a customized dictionary consisting of 5000 words with top frequencies selected from Corpus of Contemporary American English (COCA), which is one of the largest currently available corpora [14], [19]. Each word is encoded to be an entry with attributes of $\{word, frequency, length, strokeSeq\}$, of which $strokeSeq$ represents its corresponding stroke sequence. When we find a word match in the dictionary, we calculate its corresponding

ALGORITHM 2: Word Recognition Algorithm

Input: Stroke sequence $I = s_1 s_2 ... s_n$
Output: candidate words W

```
CandidateI=correct(I)∪ I;
W= ∅;
for each I in [candidateI] do
    words=findInDictionary(I);
    for each word w in [words] do
        compute P(w|I) = P(w) · ∏_{i=1}^{n} P(s_i|l_i);
    end
    W = W∪words;
end
sort W by word length in ascending order, and
  P(W|I) in descending order;
```

probability by $P(w) \cdot \prod_{i=1}^{n} P(s_i|l_i)$, where $P(w)$ is the attribute of $frequency$ and $P(s_i|l_i)$ can be obtained by stroke recognition. We sort these feasible words by their $P(w|I)$ in a descending order and provide top k (k equals 5 in our implementation) candidates with highest probabilities for a user to choose. The above process is summarized in Algorithm 2. In this way, users can input text with a small screen which enables them to perform sliding and choosing, such as smartwatch screens and touchpads of smart glasses. In our implementation, if a user does not make any choice longer than 1 second, the system will acquiescently pick top 1 candidate as the result. After word recognition, our texts-entry algorithm will predict following words by automatic successive associations by using the 2-gram data of COCA.

IV. Implementation and Experiments

A. Implementation

We implement EchoWrite on a Huawei Mate 9 equipped with a 2.4 GHz Hisilicon CPU and 6 GB RAM. The Android application is developed with Java and C codes, of which the former are responsible for high-level logic and user interface designing, while the latter are utilized for low-level data processing algorithms. When the system runs, a process controls the speaker to emit 20 KHz sinusoidal modulated audio signals continuously, and another process manages a microphone to sample echoes at 44.1 KHz and stores collected data in buffer with a size of 5 frames. When the buffer is full, data are fed to the following processing flowchart.

Limited by CPU capacity, current smartwatches can not afford the data-processing workload, which makes it unfeasible to implement real-time EchoWrite on them. Even though, to verify capabilities of other hardware, we implement the function of emitting and receiving audio signals simultaneously on a Huawei smartwatch 2. The received echoes are processed offline following the same routine as shown in Fig. 7.

B. Experiments

In order to evaluate EchoWrite, we have conducted comprehensive experiments under three settings as follows.

- **Meeting room** During experiments, the air conditioners are turned on and windows are closed as a usual case in daily life. The average noise level is measured to be $60 \sim 70$ db with a sound level meter.
- **Lab area** The room size is 8 m $\times$ 9 m in which twenty students are working on workdays. During experiments, non-participants are unaware of on-going experiments. They keep a usual state such as working with computers, chatting casually, and walking around occasionally.
- **Resting zone** It is an open area in the CSE building spared for discussing problems and conversing casually. Moreover, since it is very close to a corridor, students usually walk around or talk with each other near our experiments site. To test EchoWrite's robustness to irrelevant movements, we also request a participant to walk around near the site with a distance of $30 \sim 40$ cm.

We recruit 6 participants for our experiments. Overall, our experiments can be divided into three parts, namely, stroke recognition, words input and phrases entry. As participants have no idea about our design scheme, we explain rules to them until they have totally understand how experiments should be conducted. Before experiments, we let them do some texts entry practice according to the rules in order to make sure they have actually understand. For stroke recognition, we request each participant to perform each stroke for 30 times in each experimental setting. As a result, we can obtain a total number of 3240 ($3(settings) \times 6(participants) \times 6(strokes) \times 30(repetitations)$) testing instances. For the evaluation of words input, we select 10 words of short, medium and long lengths from COCA as shown in Table I. These words are commonly used in our communications such as short messages, brief memos, and *etc.*. And also words of different lengths cover all six strokes. In this set of experiments, participants are requested to write each word for a total number of 30 repetitions.

V. Evaluation

In this section, we demonstrate system evaluation. Several metrics are explained as follows.

- *Top k accuracy:* For a word entered for N repetitions, it occurs n times in the lists with top k candidates. Then top k accuracy is defined as $\frac{n}{N}$.
- *WPM:* It is short for *word(s) per minute* which measures the average number of words entered per minute. This metric is commonly used for evaluating the efficiency of an text-entry system.
- *LPM:* It is short for *letter(s) per minute* which measures the average number of letters entered per minute. This metric takes the length of words into account when evaluating texts-entry efficiency.

A. Stroke Recognition Performance

1) Overall performance on different devices: As mentioned in Section IV-B, although we have not implemented real-time EchoWrite on a smartwatch, we test the performance of recognizing strokes on a Huawei smartwatch by off-line data

TABLE I
THE SELECTED WORDS FOR OUR EXPERIMENTS FROM COCA

Properties \ Words	OK	YES	YOU	CALL	SOON	WAIT	LATER	THANKS	MEETING	RECEIVE
ID	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Length	2	3	3	4	4	4	5	6	7	7
Frequency ($\times 10^4$)	5.5	15.7	308	36.7	7.7	10.2	14.2	2.3	5.0	8.0

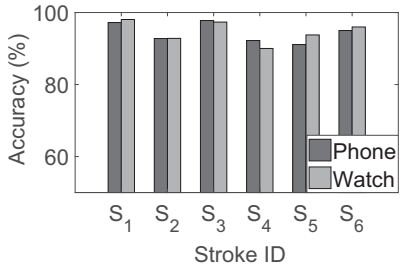


Fig. 11. The performance comparison of EchoWrite with smartphone and smartwatch.

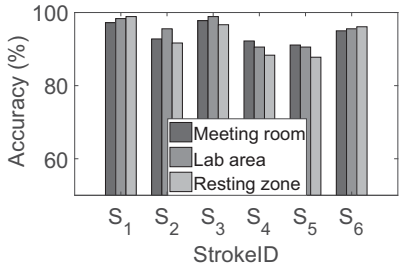


Fig. 12. The recognition accuracy of different strokes in three experimental environment.

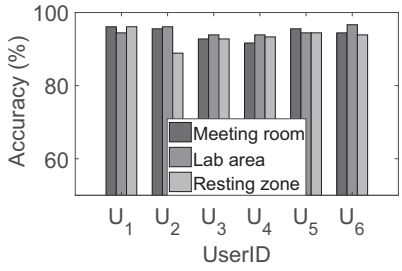


Fig. 13. The accuracy of recognizing strokes for different participants in experiments.

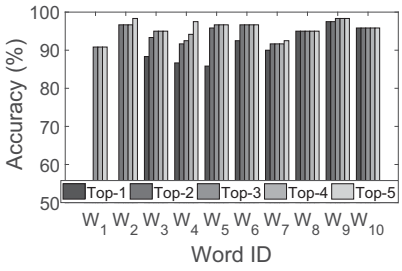


Fig. 14. The top 5 accuracies with stroke correction for different words in experiments.

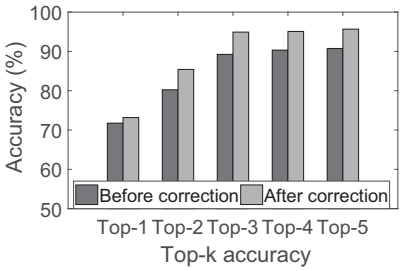


Fig. 15. The average top 5 accuracies of words recognition with and without stroke correction.

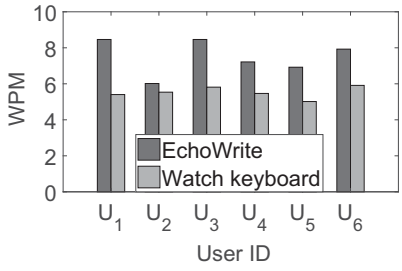


Fig. 16. The comparison of words-entry speed between EchoWrite and Soft keyboard on a smartwatch.

processing. Fig. 11 shows the results. The average accuracy of recognizing user-defined stroke gestures with a smartwatch is about 94.4% which is very close to that of EchoWrite implemented on Mate 9 (*i.e.*, 94.7%). The negligible difference indicates that acoustic sensors on a smartwatch are capable of sensing finger gestures, which verifies the feasibility of extending EchoWrite to smartwatches in the near future.

2) Accuracy of different strokes: Fig. 12 shows average accuracies of recognizing different strokes under three settings. Overall, the accuracy can be up to 98.9% (S_3, lab area) and is no lower than 87.8% (S_5, resting zone), which indicates EchoWrite's high performance of recognizing strokes. Moreover, compared with other strokes, S_4 and S_5 possess worst performance since they are more complex which makes it more difficult to write them well. In addition, we can see that the average accuracies over all strokes are 94.4%, 94.9% and 93.2% in meeting room, lab area and resting zone, respectively. We can obtain that EchoWrite is robust to noises from different sources such as air conditioner, human talking and typing keyboard, and irrelevant human motions. The reasons are two-fold. On the one hand, the frequency range of received echoes shares few overlaps with common nosies in daily environments. On the other hand, compared with finger strokes, normal human motions are with lower accelerations, and thus are filtered by our acceleration-based stroke detection method. However, due to bursting noise that span whole frequency band like rubbing interference, accuracy in resting zone decreases slightly.

3) Accuracy of different participants: Fig. 13 displays stroke recognition performance for different participants in order to evaluate the impact of user diversity. The average accuracies of different participants over tested settings are 95.6%, 93.5%, 93.1%, 93.0%, 94.8% and 95%, respectively. Due to different proficiencies in performing finger gestures, their performances deviate from each other by a maximum gap of 2.6%. However, considering the minute standard deviation (*i.e.*, 1.1%), we can claim that with same training, participants can achieve nearly the same performance.

B. Texts Recognition Performance

1) Accuracies of different words: Fig. 14 shows the top 5 accuracies of recognizing words shown in Table I. For $k = 1, 2, ..., 5$, the average top k accuracy over different words is 73.2%, 85.4%, 94.9%, 95.1% and 95.7%, respectively. By

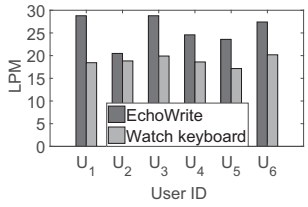


Fig. 17. Letter-entry speed between EchoWrite and smartwatch keyboard.

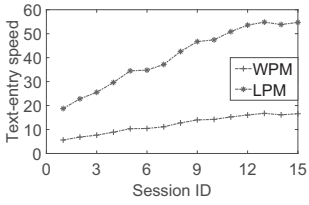


Fig. 18. Speed of text input after different numbers of training sessions.

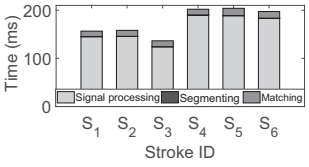


Fig. 19. The running time of different parts of data processing.

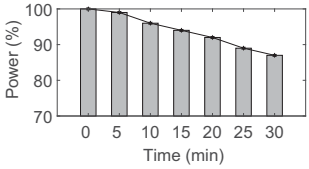


Fig. 20. The energy consumption when running the application.

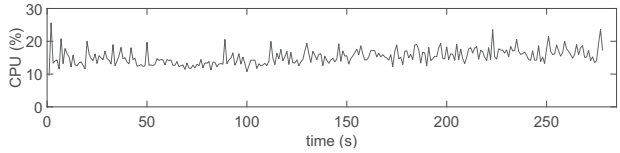


Fig. 21. The CPU overhead during the process of recognizing words.

providing 3 candidates, EchoWrite can infer words correctly with a probability of 94.9% which is favorable for texts entry. When k exceeds 3, the accuracy slightly increases by 0.8% that can also be verified by Fig. 15. The above results mean that EchoWrite can achieve high words-recognition performance by providing only 3 candidates. Moreover, different words possess varied absolute top k accuracies and overall trend. For words such as W_6, W_8 and W_{10}, when k is 2, the accuracy reaches the maximum value; while for W_4, the accuracy increases with k.

2) Impact of stroke correction: To evaluate the impact of stroke correction, we show the average top 5 accuracies for cases with and without stroke correction in Fig. 15. Overall, the average top 5 accuracies for both cases are 84.5% and 88.9%, respectively. For each k, the corresponding top-k accuracy with stroke correction is higher than that of without stroke correction. This indicates that stroke correction indeed improves the performance of words recognition. The reason is that with stroke correction, the algorithm provides more candidates for inferring correct words. What is more, we can clearly see that when k exceeds 3, the top-k accuracy nearly keeps stable as aforementioned.

3) Speed of texts entry: Fig. 16 displays the average speeds of entering given paragraphs randomly selected in Fry Instant Phrases [20] with EchoWrite and touch screen on a smartwatch. The paragraphs are grouped in five blocks, each of which contains two paragraphs. The average texts-entry speeds over all participants with EchoWrite and smartwatch are 7.5 WPM and 5.5 WPM, respectively. Moreover, by providing more candidates, users can input texts with a higher speed up to 8 words per second in a fuzzy way. Although this speed is not comparable with soft keyboard on mobile devices with large screens, it is yet sufficient for most texts-entry applications with wearable devices such as writing a memo, making simple notes, giving short reply and the like. Considering the differences of words' lengths, we also compare the texts-entry speed by LPM (*i.e.*, letter per minute) in Fig. 17. We can clearly see that the average speed is 25.6 LPM for EchoWrite which is higher than that of smartwatch by about 18.8 LPM.

4) Impact of training time: Intuitively, a user's proficiency has great impact on text-entry speed. To evaluate this quantitatively, we request participants to 'write' a block for 15 times (*i.e.*, sessions). We display the WPM and LPM of each session in Fig. 18. As we can see, with increasing number of practice sessions, the speed of user's texts input also increases especially in initial sessions. When this number reaches 13, the WPM and LPM are stable and can be up to 16.6 and 55.3, respectively. Thus, we can conclude that with acceptable training overhead, users can achieve much higher texts-input efficiency.

C. System Running Performance

1) Running time of each part: The time consumption of different data processing parts in EchoWrite is shown in Fig. 19. As we can see, the total processing time of recognizing a stroke is less than 200 ms which indicates favorable real-time performance of EchoWrite. In other words, the response time of our system is no more than 0.2 s. We can also see that signal processing including Doppler enhancement and profile extraction occupies over 90% running time. The underlying reason is that performing STFT is rather time-consuming which can be optimized by downsampling technique in the future work. Further more, we notice that S_4, S_5 and S_6 cost more time than other strokes as they last longer and consist of more samples than other strokes.

2) Energy consumption: To obtain knowledge of energy consumption of EchoWrite, we leave EchoWrite in the backstage only and kill other Apps, monitor power levels of a mobile device which continuously runs EchoWrite to recognize texts every 5 minutes. As is shown in Fig. 20, the power level decreases from 100% to 87% after 30 minutes, which means about 3% power is consumed every 5 minutes. As a result, a smartphone can last for about 2.8 hours if we run EchoWrite continuously. Since entering texts such as short messages, quick replies and *etc.* are not frequent, this power consumption rate is acceptable for mobile devices. By optimizing STFT as aforementioned, we can further reduce the power consumption of EchoWrite.

3) CPU occupation: We also evaluate the occupancy of CPU resources when a mobile device runs EchoWrite. During evaluation, we turn off all other applications except EchoWrite and monitor CPU consumption with Android API. We continuously enter texts to test the maximum CPU consumption with EchoWrite. Fig. 21 shows real-time CPU proportion consumed

during recognizing words with EchoWrite. Even though the CPU proportion varies from 9.5% to 25.6%, the average is about 15.2% with a standard deviation of 2.3% which is acceptable for an application. Indicated by Fig. 19, we can further decrease CPU resources consumption by accelerating image and matrix processing using the GPU.

VI. Related Work

A. Acoustic signal-based HCI

Acoustic sensors have been widely used for human-device interaction with mobile devices in coarse gesture recognition [16], [21]–[25], digits input [26] and precise motion tracking [10]–[12], [27], [28]. The early works [21], [24], [25] require users to move devices to achieve device-to-device interaction. The works [16], [22], [23] makes use of Doppler effect caused by relative motion between one's hand and a device to recognize coarse hand gestures such as "Pull", "Flick" and *etc.*. In contrast, our work focuses on designing a texts-entry system based on more fine-grained finger movements instead of solely recognizing hand gestures, which brings about unique challenges. Among the above works, AudioGest [23], which achieves accurate hand gesture recognition, is most related to ours from technical perspective. However, the differences are notable as well. First, due to finer granularity of input gestures, techniques developed in [23] can not be applied in our work according to our implementation. Second, as EchoWrite is a text-entry system, we have also devoted to designing efficient and accurate input scheme and texts recognition techniques. Some other works develop high-precision motion tracking systems in a device-based [27], [28] or device-free way [10]–[12] which accomplish mm-level 2D motion tracking. But they require multiple normal microphone-speaker pairs which are not available for most commercial devices especially tiny smart devices.

B. Motion sensor- and RF-based input systems

Besides acoustics, other text-entry systems can be mainly divided into two categories, namely, sensor-based [5], [6], [9], [29] and radio frequency (RF)-based [1]–[4]. Motion sensor-based systems mainly utilize inertial sensors (*i.e.*, accelerometer, gyroscope) or with proximity and distance sensors to recognize characters, digits and texts [5], [6], [9], [29], [30]. Compared with EchoWrite, they have the following shortcomings: 1) needing additional hardware such as a ring or glove except the interactive device; 2) requiring users to wear or carry devices during interaction; 3) being sensitive to irrelevant body movements. RF-based systems utilize Wi-Fi, RFID, 60 GHz signals and visible light to achieve high-precision finger input [1], [2], [4], [31] or coarse inputing gestures [3]. Nevertheless, they require additional RF equipment, such as RFID readers and tags, Wi-Fi transceivers, and LEDs, and thus are not applicable for most mobile devices. In comparison, EchoWrite provides a novel texts-input method without any additional device/hardware, and works in a device-free, touch-free and training-free style.

VII. Discussion and Future Work

As an prototype, EchoWrite still has several limitations and needs to be optimized in the future work.

A. Scalability to smartwatches

As aforementioned, EchoWrite can not run on existing smartwatches at present due to the limitation of CPU capacity. However, according to results with a Huawei watch 2 as shown in Fig. 11, the offline performance of strokes recognition is close to that of a Huawei mate 9. This indicates that the hardware of a smartwatch is capable of implementing EchoWrite, provided that the CPU workload is affordable. We are optimistic about tackling this problem considering two-fold reasons. On the one hand, obtaining the spectrogram by continuous STFT costs a high percentage of CPU resources. To decrease computing overhead, a possible approach is to utilize down-sampling technique to reduce the number of FFT points, according to bandpass sampling theorem [32]. More importantly, this operation does not need to modify main methods proposed in this work. On the other hand, we can expect that wearables' CPUs are able to support EchoWrite in the near future considering the rapid hardware improvement.

B. Robustness to bursting noises

Although EchoWrite has exhibited robustness to external noises, it is yet sensitive to certain kinds of burst noises such as knocking tables and striking objects which usually cover a wide frequency range overlapping with signals utilized in EchoWrite. Due to the frequency overlap, the noises cause interference to spectrograms of strokes, which makes it rather difficult to work out corresponding Doppler profiles and degrades the performance of stroke recognition. As a result, it is desirable to further enhance system robustness by tackling this problem. In our opinion, there are two possible approaches to handle this problem. The first one is to improve denoising techniques by making use of properties of such noises like short duration. Another one is to use classification methods to classify texts-entry behaviors and irrelevant behaviors. By this approach, we can discard signal segments only containing bursting noises caused by irrelevant events.

C. User-defined input scheme

In the design of input scheme, we have taken learnability, memorability and efficiency into consideration. But we have not considered users' preferences to input gestures. For example, some people may think certain gestures to be difficult and would like to re-define them by themselves. The current version of EchoWrite can not support users to customize the input gestures due to two reasons. First, we have not designed an module to automatically check whether the customized gestures set are appropriate. For example, some gestures may have the same Doppler profile which are not permitted according to our approach. Another reason is that some parameters are empirically determined according to gestures. When users redefine certain gestures, the corresponding parameters need to be adjusted automatically. We leave adding self-adjusting module into EchoWrite as one of our future work.

VIII. Conclusion

Motivated by limitations of existing texts-entry approaches for mobile devices, we propose a novel system named EchoWrite that enables users to input texts with a finger writing strokes in the air. EchoWrite relies on pervasive acoustic sensors to sense writing gestures and further recognize entered texts. To design this system, we propose a natural and efficient input scheme, develop fine-grained Doppler profile extraction method, design stroke-correction and texts inference algorithm. To evaluate performance of our texts-entry approach, we implement real-time system on a commercial device and conduct comprehensive experiments. Results show that our approach enables users to enter texts at a comparable or even higher speed compared with related works. Although EchoWrite has still some limitations, we envision that it has great potential to be applied on various smart devices.

Acknowledgment

This research was supported in part by the China NSFC Grant (61802264, 61872248, U1736207), Guangdong NSF 2017A030312008, Shenzhen Science and Technology Foundation (JCYJ20180305124807337, JCYJ20170302140946299, JCYJ20170412110753954, GRCK2017082111300325), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (Grant No.161064), GDUPS (2015), and Tencent Rhinoceros Birds- Scientific Research Foundation for Young Teachers of Shenzhen University. Kaishun Wu is the corresponding author.

References

[1] J. Wang, D. Vasisht, and D. Katabi, "RF-IDraw: virtual touch screen in the air using rf signals," in *ACM SIGCOMM*, 2014.

[2] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *ACM MobiCom*, 2014.

[3] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim, "Widraw: Enabling hands-free drawing in the air on commodity wifi devices," in *ACM MobiSys*, 2015.

[4] T. Wei and X. Zhang, "mtrack: High-precision passive tracking using millimeter wave radios," in *ACM Mobicom*, 2015.

[5] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[6] C. Amma, M. Georgi, and T. Schultz, "Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors," in *IEEE ISWC*, 2012.

[7] M. Goel, L. Findlater, and J. Wobbrock, "Walktype: using accelerometer data to accomodate situational impairments in mobile touch screen text entry," in *ACM CHI*, 2012.

[14] M. Davies, "Corpus of contemporary american english," https://corpus.byu.edu/coca/, 2008, accessed on 2018-07-31.

[8] T. Ni, D. Bowman, and C. North, "Airstroke: bringing unistroke text entry to freehand gesture interfaces," in *ACM CHI*, 2011.

[9] S. Nirjon, J. Gummeson, D. Gelb, and K.-H. Kim, "Typingring: A wearable ring platform for text input," in *ACM MobiSys*, 2015.

[10] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio: Using active sonar for fine-grained finger tracking," in *ACM CHI*, 2016.

[11] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *ACM Mobicom*, 2016, pp. 82–94.

[12] S. Yun, Y.-C. Chen, H. Zheng, L. Qiu, and W. Mao, "Strata: Fine-grained acoustic-based device-free tracking," in *ACM MobiSys*, 2017, pp. 15–28.

[13] Super English Kid, "Super English Kid," http://www.superenglishkid.com/2014/11/stroke-order-worksheet-for-teaching-how.html, 2014, accessed on 2018-07-31.

[15] H. Zhang, W. Du, P. Zhou, M. Li, and P. Mohapatra, "Dopenc: acoustic-based encounter profiling using smartphones," in *ACM Mobicom*, 2016.

[16] S. Gupta, D. Morris, S. Patel, and D. Tan, "Soundwave: using the doppler effect to sense gestures," in *ACM CHI*, 2012.

[17] P. Soille, *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[18] P. Holoborodko, "Applied mathematics and beyond," http://www. holoborodko. com/ pavel/, 2015, accessed on 2018-01-16.

[19] MarkDavies, "Word frequency data," https:// www. wordfrequency. info/, 2010, accessed on 2018-01-08.

[20] T. Rasinski, "Fry instant phrases," http://www. timrasinski. com/ presentations/, 2018, accessed on 2018-01-08.

[21] M. T. I. Aumi, S. Gupta, M. Goel, E. Larson, and S. Patel, "DopLink: Using the doppler effect for multi-device interaction," in *ACM UbiComp*, 2013, pp. 583–586.

[22] K.-Y. Chen, D. Ashbrook, M. Goel, S.-H. Lee, and S. Patel, "AirLink: sharing files between multiple devices using in-air gestures," in *ACM UbiComp*, 2014.

[23] W. Ruan, Q. Z. Sheng, L. Yang, T. Gu, P. Xu, and L. Shangguan, "Audiogest: enabling fine-grained hand gesture detection by decoding echo signal," in *ACM Ubicomp*, 2016, pp. 474–485.

[24] Z. Sun, A. Purohit, R. Bose, and P. Zhang, "Spartacus: spatially-aware interaction for mobile devices through energy-efficient audio sensing," in *ACM MobiSys*, 2013, pp. 263–276.

[25] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda, "Swordfight: Enabling a new class of phone-to-phone action games on commodity phones," in *ACM Mobisys*, 2012, pp. 1–14.

[26] Y. Zou, Q. Yang, Y. Han, D. Wang, J. Cao, and K. Wu, "Acoudigits: Enabling users to input digits in the air," in *IEEE PerCom*, 2019, pp. 313–321.

[27] S. Yun, Y.-C. Chen, and L. Qiu, "Turning a mobile device into a mouse in the air," in *ACM MobiSys*, 2015, pp. 15–29.

[28] W. Mao, J. He, and L. Qiu, "Cat: high-precision acoustic motion tracking," in *ACM Mobicom*, 2016, pp. 69–81.

[29] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and i can track my user's arm," in *ACM MobiSys*, 2016, pp. 85–96.

[30] C. Zhang, A. Waghmare, P. Kundra, Y. Pu, S. Gilliland, T. Ploetz, T. E. Starner, O. T. Inan, and G. D. Abowd, "Fingersound: Recognizing unistroke thumb gestures using a ring," *ACM Ubicomp*, 2017.

[31] C. Zhang, J. Tabor, J. Zhang, and X. Zhang, "Extending mobile interaction through near-field visible light sensing," in *ACM Mobicom*, 2015, pp. 345–357.

[32] A. V. Oppenheim, *Discrete-time signal processing (3rd Edition)*. Prentice Hall Press, 1999.