EchoWrite: An Acoustic-Based Finger Input System Without Training

Kaishun Wu, *Member, IEEE*, Qiang Yang, *Student Member, IEEE*, Baojie Yuan, Yongpan Zou, *Member, IEEE*, Rukhsana Ruby, *Member, IEEE*, and Mo Li, *Member, IEEE*

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. Moreover, EchoWrite provides favorable user experience of entering texts.

Index Terms—Acoustic signals, texts input, HCI, mobile device

1 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], [2], [3], [4], [5], [6], [7], [8], [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], [11], [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.

• K. Wu, Q. Yang, B. Yuan, Y. Zou, and R. Ruby are with the College of Computer Science and Software Engineering, Shenzhen University 3688 Nanhai Ave, Shenzhen, Guangdong 518060, China. E-mail: {wu, yongpan, ruby}@szu.edu.cn, yangqiang1201@foxmail.com, itssyuu728@163.com.
• M. Li is with the School of Computer Science and Engineering, Nanyang Technological University50 Nanyang Avenue, Singapore 639798. E-mail: limo@ntu.edu.sg.

Manuscript received 15 Apr. 2019; revised 10 Jan. 2020; accepted 7 Feb. 2020. Date of publication 0 . 0000; date of current version 0 . 0000. (Corresponding author: Yongpan Zou.) Digital Object Identifier no. 10.1109/TMC.2020.2973094

In response to this question, we propose EchoWrite in this paper, a system that enables users to enter texts nearly 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.

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. By carefully designing signal processing flowchart, we can track frequency shifting along with finger

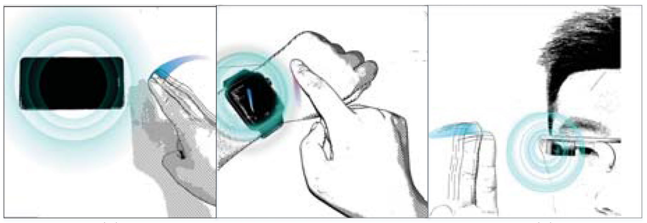


Fig. 1. Possible scenarios where EchoWrite can be applied. EchoWrite cannot only be on wearable devices such as smartwatches and smartglasses to deal with the inconvenience caused by small screens, but also on mobile devices, such as smartphones and tablets, to handle the cases where hands are wet or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

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. What is more, the user study results indicate EchoWrite provides satisfactory user experience.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work. Following that, we give an overview of EchoWrite in Section 3 and details of system design in Section 4, respectively. We describe the implementation and experiments in Section 5, and display performance evaluation in Section 3.3. Last in Section 8, we conclude the paper.

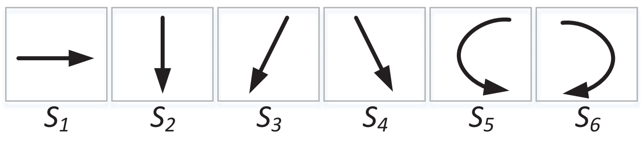
2 Related Work

2.1 Commercial Text-Entry Approaches

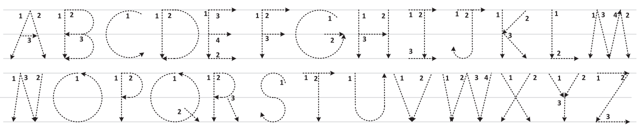
With popularity of mobile devices, efficient texts-entry approaches have attracted increasing academic and industrial attention. Soft keyboard is the most common method for human-device interaction (HCI) on mobile devices such as smartphones, tablets, smartwatches and the like. The advantages of soft keyboard-based interface mainly include low cost, high efficiency and favorable convenience. Nevertheless, it also possesses several shortcomings. For example, soft keyboard requires large screens, otherwise its performance degrades severely. As a result, for devices with small or without screens like smartwatches, smartglasses and head-mounted displays (HMDs), soft keyboard is not an optimal choice to enter texts. Another shortcoming is that this method requires a user to touch on the screen which is not applicable when it is not available. To improve texts-entry performance on tiny devices, researchers have proposed some novel systems in previous work. In early works [13], the authors design an one-dimension handwriting input scheme on smart glasses. However, it is not applicable for other tiny devices such as smartwatches and HMDs. A following work [14] proposes a bezel-based text entry technique while it requires to rotate multiple cursors. In a latest work [15], the authors an eyes-free texts entry method with a handled touchpad. These systems either require touching on the device or additional hardware support (bezels, cursors or touchpad). Another promising technology for texts entry is speech recognition owing to its high accuracy and good experience, based on which speech assistants like Siri [16] and Cortana [17] have been popularly used on mobile devices. However, speech recognition possesses shortcomings of performance degrading in noisy environment, privacy leakage and awkward feelings in public occasions.

2.2 Acoustic Signal-Based HCI

Acoustic sensors have been widely used for human-device interaction with mobile devices in coarse gesture recognition [18], [19], [20], [21], [22], [23] and precise motion tracking [10], [11], [12], [24], [25]. The early works [18], [22], [23] require users to move devices to achieve device-to-device interaction. The works [19], [20], [21] 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 [21], 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 (finger versus hand), techniques developed in AudioGest [21] can not be applied in our work according to our implementation. A key problem is that since the induced Doppler shift is weaker, we design novel techniques to highlight and extract it from original spectrogram accurately, even with external interference such as body/arm movement and others walking. Second, as EchoWrite is a text-entry system, we have also devoted to designing efficient and accurate input scheme, developing texts recognition techniques, and conducting more comprehensive evaluation. Some other works develop high-precision motion tracking systems in a device-based [24], [25] or device-free way [10], [11], [12] which accomplish mm-level 2D motion tracking. But they require multiple normal microphone-speaker pairs which



(a) Basic strokes of English letters



(b) The stroke order of English letters

Fig. 2. The design of input scheme.

are not available for most commercial devices especially tiny smart devices. More related works [26], [27], [28] follow a passive sensing way which recognize the written texts by analyzing the sounds caused by handwriting on a table. Similar works [29], [30], [31] present active audio-based handwriting recognition systems with machine learning approach. Compared with them, our work is training-free which means much lower training overhead and higher scalability to different people and scenarios.

2.3 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], [32] and radio frequency (RF)-based [1], [2], [3], [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], [32], [33]. 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], [34] or coarse inputing gestures [3]. The Soli project [35] uses 60 GHz chip in a mobile device to recognize fine-grained finger gestures. Nevertheless, they require additional RF equipment, such as RFID readers and tags, Wi-Fi/60 GHz 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.

3 System Overview

3.1 Input Scheme Design

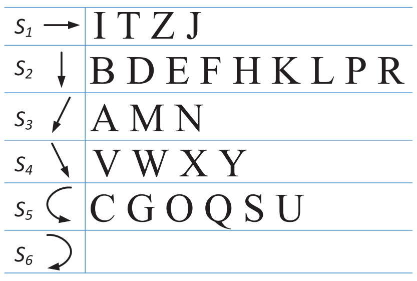
As is well known, uppercase English letters can be decomposed into six basic strokes as shown in Fig. 2a. The key idea of our input scheme is similar to T9 input method which recognizes texts in a fuzzy way. Specifically, each stroke represents a set of certain English letters and a word can be inferred from a sequence of strokes. When users intend to enter texts (letters, words, or sentences), they only need to 'write' a sequence of strokes in the air. As the first step, we need to consider how to assign 26 letters to different strokes. A basic principle of designing an input scheme is learnability, that is, it should incur as light mental workload as possible when they utilize this system. Considering this, we initially group the 26 letters according to their first strokes since it is a natural way for users to remember. Although different people may have varied writing styles, we adopt the stroke orders of upper-case letters[1] as shown in Fig. 2b which is recommended for kid's learning English [36]. As a result, we can obtain the letter assignment scheme as shown in Fig. 3a. However, we can notice that there are no letters corresponding to S_6 while too many letters are assigned to S_2. This shall greatly degrades a user's learnability, memorability and performance. Second, it shall also balance the number of characters assigned to each stroke. Consequently, we rearrange the scheme by moving letters of 'B', 'D', 'P' and 'R' down to S_6 based on their second strokes, and accordingly obtain the scheme shown in Fig. 3b.

Nevertheless, another consideration of designing such a scheme is the uniqueness of Doppler profiles of different strokes. As a result, we conduct preliminary experiments and extract Doppler profiles of six strokes following instructions from Sections 4.1 to 4.2, to check whether the new scheme satisfies this requirement. The results demonstrate that S_1 and S_4, S_2 and S_5 have much similar Doppler patterns and are rather difficult to be differentiated (Section 4.3). To tackle this problem to guarantee uniqueness, we make slight adjustments by modifying S_4 and S_5 as shown in Fig. 3c. After adjustment, the corresponding Doppler profiles of different strokes can be guaranteed to be unique, which will be shown in Section 4.2.

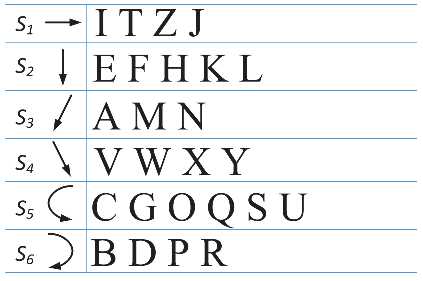
3.2 Demonstration of Words Representation

Until now, a key question to be answered is *whether this stroke-letter mapping scheme can represent English words accurately*. To verify this, we generate the sequence of corresponding strokes for each word in the Corpus of Contemporary American English (COCA) [37]. The original COCA dataset contains a total number of 450 million words in which the same word could appear for multiple times due to its different PoS (part of speech) with corresponding word frequency. We select 5000 most frequent words which can cover most daily usage and merge the dataset by adding the word frequencies of a same word with different PoS together. After this operation, the new dataset contains a total number of 4348 words. We then apply word recognition algorithm, (i.e., Algorithm 2) in Section 4.3 on each generated stroke sequence which returns sorted word candidates with descending probabilities. Note that as we have taken possible errors in performing and recognizing strokes, we add error correction mechanism in Algorithm 2. However, we do not need to consider this here, and thus we remove it from the algorithm. To evaluate how accurately the mapping scheme represent English words, we introduce top-k accuracy which is defined as the ratio of the number of words appeared in the top-k candidates provided by the algorithm over the total number of words in the database. We perform this operation on all the words in the database and then compute the average top-1 to top-5 accuracies. The result shows that these five accuracies are 85.3, 94.1, 97.2, 98.4 and 99.2 percent, respectively. This

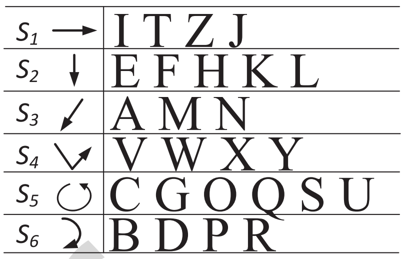
1. As for the more complex lower-case letters, we do not consider this problem in this work. An intuitive way is to add a shift function in the system design, via a certain gesture or touching button.



(a) The first version of letters assignment

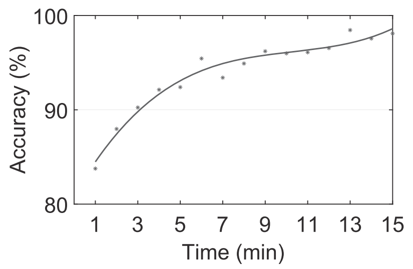


(b) The second version of letters assignment

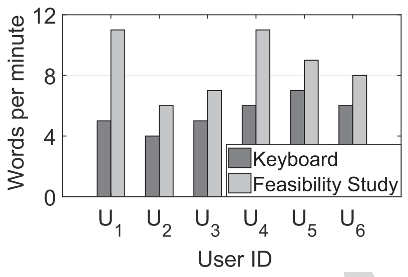


(c) The final version of letters assignment

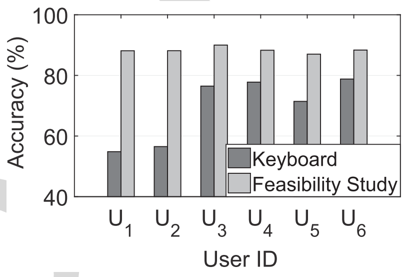
Fig. 3. The design of letter assignment scheme.



(a) The accuracy of stroke sequence along with practice time



(b) The words-input speed of different participants in learnability study



(c) The stroke-input accuracy of different participants in learnability study

Fig. 4. The experimental results of learnability study.

indicates that the mapping scheme in Fig. 3b can represent the common English words by providing no more than five candidates.

3.3 Evaluation of Learnability

Before designing the system, we conduct user study experiments to obtain learning curve and evaluate the learnability of the above input scheme. To do this, we assume that our system recognizes strokes in real time with an accuracy of 90 percent although we have not designed such a system yet.[2] We recruit a total number of 6 participants (3 females and 3 males) from our university who have no any knowledge about our project. Before experiments, we spend 5 minutes to introduce the input scheme and make sure that they have fully understood. After that, we request them to write out stroke sequences corresponding to 300 most frequent and randomly shuffled words which are selected from COCA [37]. In comparison, we also request them to input the same words that they have written via soft keyboard on Huawei smartwatch 2. During this process, the participant is allowed to see each word for only once. When they make errors, they are reminded but not permitted to make correction. Each participant writes words continuously for 15 minutes and every minute we count error rate of letters to get the learning curve as shown in Fig. 4a. As we can see, after 15 minutes' practice, participants can write out stroke sequences of different words with an average accuracy up to 98 percent. Moreover, we also investigate words-input speed and accuracy and obtain results shown in Figs. 4b and 4c after 15 minutes' practice. It is clear that participants can enter words with our scheme at a speed of 11 words per minute and achieve a word-recognition accuracy of 90 percent.[3] These results preliminarily indicate that our scheme is capable of achieving favorable learnability, memorability and efficiency.

3.4 The System Flowchart

Fig. 5 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 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 [38], since it is extracted from reflected signals instead of direct propagations.

4 System Design

4.1 Doppler Enhancement

After receiving echoes, we perform short-time Fourier transform (STFT) with a Hanning window on signal sequence in

2. Since we only evaluate the learnability of input scheme instead of the whole system, it is acceptable to make such assumptions.

3. It is noted that this accuracy is obtained by multiplying assumed stroke-recognition accuracy (90 percent) by word sequence accuracy.

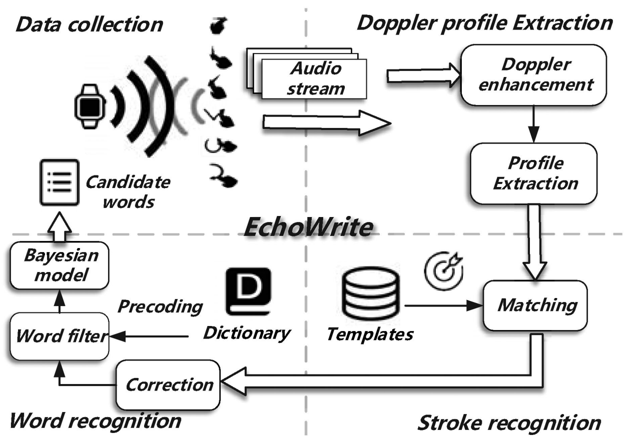


Fig. 5. The work flowchart of EchoWrite.

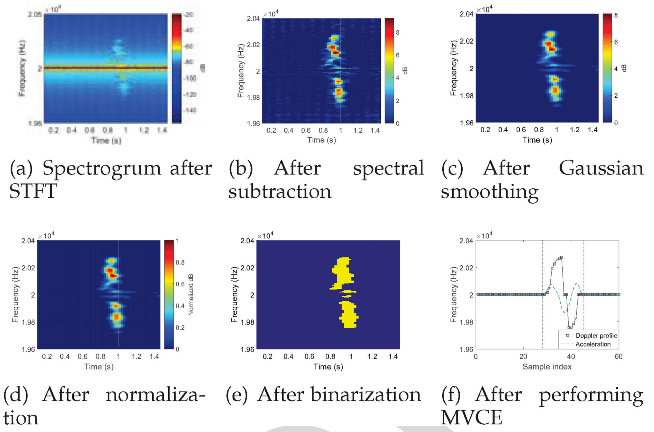

(a) Spectrogrum after STFT (b) After spectral subtraction (c) After Gaussian smoothing (d) After normalization (e) After binarization (f) After performing MVCE

Fig. 6. Different stages of extracting the Doppler shifts profile of writing stroke S_2.

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 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. 6a. 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

$$\Delta f = f_0 \times \left|1 - \frac{v_s \pm v_f}{v_s \mp v_f}\right|, \tag{1}$$

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 [20]), 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. 6c. The Gaussian filter is a 2D convolution smoothing operator that is used to blur images and remove detail and noise. Here we borrow this method from image processing to remove random noises in spectrograms. 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 [39]. By the above operations, we can enhance the Doppler shifts of finger movements and obtain a relatively clean spectrogram as shown in Fig. 6e.

4.2 Extracting Doppler Profile

Based on the processed spectrogram, we further proceed to extract the Doppler profile by figuring out the contour of spectrogram. As we can see, due to reflections from multiple parts, there exist multiple Doppler shifts in a single frame. To depict finger motion clearly, it is theoretically required to separate each dynamic multipath and pick out the one corresponding to user's finger. However, it is impossible and unnecessary for our system design. For stroke recognition, we only need to get Doppler shift that can describes overall trend of finger motion, instead of precise moving details. As a result, we extract the contour of spectrogram with a mean value-based contour extraction algorithm (MVCE). The key idea of our method is that for each frame, we regard the mean value of Doppler shifts as the real Doppler shift caused by finger motion. By determining Doppler shift of finger motion in each frame, we can obtain the Doppler profile corresponding to each stroke (see Fig. 6f).

However, there are two more steps besides MVCE. First, before extracting contour, we need to perform a bit more processing on the spectrogram. We fill up the "holes" appeared in the spectrogram with an existing image processing technique. Second, we further smooth the obtained Doppler profile to remove outliers with a moving average method of which the sliding window size is 3.

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,

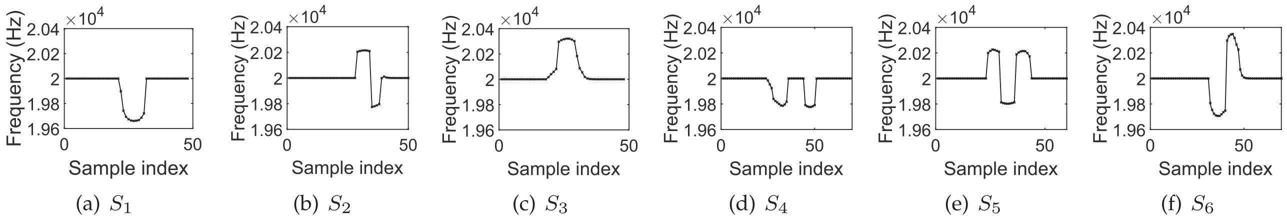


Fig. 7. The extracted Doppler profiles of different strokes according to the final version of input scheme.

we first calculate first-order difference of obtained Doppler profile with a noise-robust differential approach as follows [40]:

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

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

$$f_t = \frac{1 \pm \frac{v_f}{v_s}}{1 \mp \frac{v_f}{v_s}} f_0 = \pm \frac{2 f_0 v_f}{v_s \mp v_f} + f_0 \approx \pm \frac{2 f_0 v_f}{v_s} + f_0, \quad (3)$$

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

$$\Delta f_t^{'} = \frac{2 f_0}{v_s} v_f^{'} = \frac{2 f_0}{v_s} a, \quad (4)$$

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 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. 8. 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. We summarize the above mean value-based Doppler profile extraction process as shown in Algorithm 1.

Algorithm 1. Doppler Profile Extraction

```
   Input: Spectrogram matrix $P$, centre frequency bin $cf$
   Output: Doppler shift profile $DopShift$
 1 colNum=getColumnNum($P$);
 2 $DopShift$(1:colNum)=$cf$; //initialization
 3 for $i$=1:colNum do
 4   row=getNonNullRows(col(i)); // get non-null rows of the
     ith col
 5   if isNotEmpty(row) then
 6     meanValue=mean(row);
 7     if meanValue>cf then
 8       $DopShift$(i)=max(row);
 9     else
10       $DopShift$(i)=min(row);
11     end
12   end
13 end
14 $DopShift$ = SMA($DopShift$); // SMA represents smoothed
   moving average filter
```

4.3 Stroke and Texts Recognition

After profile extraction, the following step is to recognize different strokes. Our insight is that writing different strokes produces unique Doppler shift profiles as shown in Fig. 7. More importantly, within a certain area, the induced Doppler shift profiles are intrinsically related with strokes themselves and irrelevant with the person who performs them and how fast they are performed, which can be deduced by the Doppler principle. For stroke recognition, we make use DTW to match extracted Doppler shift profile with templates that have been stored in the system. Dynamic time wrapping is a mathematical method to compute similarity of time series. Compared with other methods, DTW exceeds in taking stretch and contraction of time series into consideration [41].

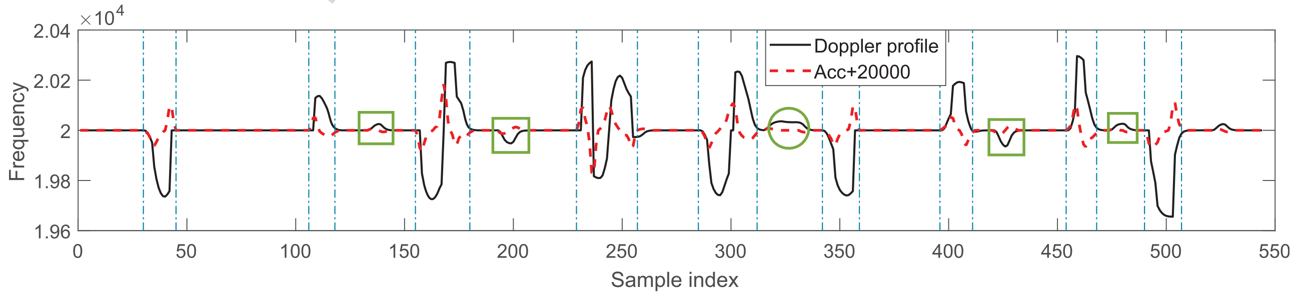


Fig. 8. The segmentation results for a series of writing strokes.

Recognizing a stroke sequence enables us obtain different sets of candidate letters whose combinations consist of multiple 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 $I = s_1 s_2 \dots s_n$, the output is a letter combination denoted by $w = l_1 l_2 \dots l_n$, where $s_1, s_2, \dots, s_n$ and $l_1, l_2, \dots, l_n$ 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

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

This is

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

where $P(w)$ represents the prior probability of a candidate word obtained by vocabulary statistics, $t(l_i)$ is the stroke template of letter l_i, and $P(s_i|t(l_i))$ denotes the probability of recognizing strokes which 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.[4] 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, we find that there is little possibility for multiple strokes to be wrongly recognized simultaneously in a sequence according to our experiments. 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 is likely to be recognized as S_2 and S_3, which can be also inferred from Fig. 7. By substituting one stroke each time, we can obtain a set of corrected stroke sequences.

4. It is noted that although correction is not performed on candidate words, the principle and effect are much similar, that is, improving accuracy by allowing for possible errors.

For each obtained stroke sequence, there exist different corresponding letter sequences but only part of them are feasible words. To verify a letter sequence, we build up a customized dictionary consisting of 5000 words with top frequencies selected from COCA, which is one of the largest currently available corpora [37], [42]. To improve searching efficiency, we follow the steps below to construct a tree-structure dictionary. First, each word is encoded to be an entry with attributes of $\{word, frequency, length, strokeSeq\}$, of which $strokeSeq$ represents its corresponding stroke sequence. Second, words are stored with a tree data structure. Specifically, words are first arranged according to $length$ in the first layer. Then in the following layers, words are further arranged by the type of first stroke, second stroke, ..., until the last one. As for words belonging to the same leaf node, we arrange them with a descending order of $frequency$. As a result, to find out the word matching with a letter sequence, we only need to follow the tree structure within a limited number of steps in such a way. If there is no word match, it indicates that the letter sequence is not a feasible word.

When we find a word match in the dictionary, we calculate its corresponding probability by $P(w) \cdot \prod_{i=1}^{n} P(s_i|t(l_i))$, where $P(w)$ is the attribute of $frequency$ and $P(s_i|t(l_i))$ can be obtained by stroke recognition. For a stroke sequence recognized by our algorithm, there are multiple candidates after correction, each of which has several possible words with different probabilities. 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 word recognition method is summarized in Algorithm 2. In the implementation, we also configure the system to pick the first candidate as the final result when a user does not make any choice for a thresholding time. Without specification, experimenters are instructed to adopt the first approach, namely, making selection among candidates, in experiments of performance evaluation in Section 5.2. After word recognition, our texts-entry algorithm will predict following words by automatic associations. In our system design, we make use of the 2-gram data of COCA. Specifically, the system will search following words in the 2-gram database according to automatic associations and provide the five words with top frequencies as potential choices. If a user makes a choice among these five words, the system will continue automatic associations.

Algorithm 2. Word Recognition Algorithm

```
   Input: stroke sequence $I = s_1 s_2 \dots s_n$
   Output: candidate words W
 1 CandidateI=correct(I)∪ I;
 2 W= ∅;
 3 for each I in [candidateI] do
 4     words=I.findIn(Dictionary);
 5     for each word w in [words] do
 6         compute $P(w|I) = P(w) \cdot \prod_{i=1}^{n} P(s_i|l_i)$;
 7     end
 8     W = W∪ words;
 9 end
10 sort W by $length$ in ascending order, and $P(W|I)$ in descending order;
```

TABLE 1
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

5 Implementation and Experiments

5.1 Implementation

We first implement EchoType on Android platform with Huawei Mate 9 which is equipped with a 2.4 GHz Hisilicon CPU and 6 GB RAM. Although Mate 9 has two normal microphones, we only need to make use of a single one of them in our system design. The whole application is programmed with both Java and C codes. The Java codes are responsible for high-level logic and user interface design, while the C codes are utilized for low-level signal processing and algorithms including signal preprocessing, Doppler shifts detection, profile extraction, and Bayesian-based texts inference. This hybrid programming is useful for improving running efficiency of the application. When the system is running, a process takes responsibility for controlling the speaker to emit 20 KHz audio signals continuously. At the same time, a parallel process manages a microphone to receive echoes at a 44.1 KHz sampling rate and temporarily stores data in a buffer. The buffer size is set to be 5 frames corresponding to 40906 samples which is optimized according to the efficiency of data processing. When the buffer is full, data are pushed to following data processing flowchart. To optimize system performance, we adopt two key tricks in specific implementation. First, since STFT costs too much time, we balance the computational overhead by parallelizing the whole computing process to improve efficiency of whole application. Second, we optimize several key parameters such as sizes of buffer, FFT and window step which affect system performance to a great extent.

It is noted that, limited by the power of CPU, current commercial smartwatches can not afford the data-processing workload of this application. As a result, it is unfeasible to implement EchoType on existing smartwatches. However, considering the vast improvement of CPU on commercial devices, we envision that it is feasible to implement this application on smartwatches in the near future. We shall conduct a detailed discussion about the scalability to smartwatches in Section 7. Even so, to verify this capability of other hardware, we implement part of functions of EchoType on a Huawei smartwatch 2 which include emitting and receiving audio signals simultaneously. The received echo signals are further processed in an offline style following the same routine as shown in Fig. 5.

5.2 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. Moreover, to evaluate the impact of different levels of noises, we also test the system under $70 \sim 80$ and $80 \sim 90$ dB noises, by controlling the volume of playing music with another device in the meeting room.
- *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.
- *Mobile scenarios.* We also test the performance of EchoWrite in recognizing strokes when it is used in mobile scenarios. That is, a user perform stroke gestures while he/she is walking with the device held in another hand. This part of experiments are conducted in the meeting room with $60 \sim 70$ dB noise.

We recruit 10 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 10800 ($6(settings) \times 10(participants) \times 6(strokes) \times 30(repetitations)$) testing instances. For the evaluation of words input, we select 10 words of short, medium and long lengths from COCA as shown in Table 1. 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, each participant is requested to write each word for a total number of 30 repetitions in each of three basic experimental settings, namely, meeting room, lab area and resting zone. The last part is user-study experiments to evaluate the user experience of using EchoWrite and soft keyboard on a smartwatch.

6 Evaluation

In this section, we demonstrate the evaluation of EchoType from different aspects. The main metrics that we utilized for evaluation are explained in a detail as follows.

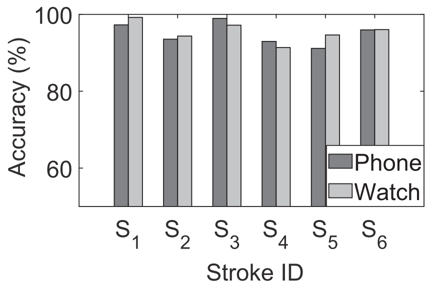


Fig. 9. The performance comparison of EchoWrite with Mate 9 and a smartwatch.

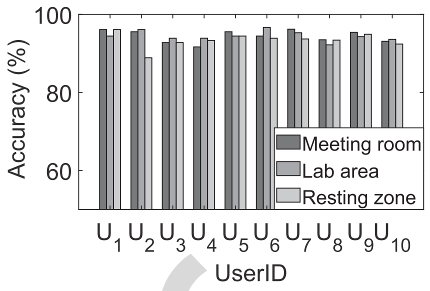


Fig. 11. The accuracy of recognizing strokes for different participants recognizing strokes for different participants.

- *Top k accuracy:* For a word entered for N repetitions, there are n times occurring in the lists with k candidates given by our system. Then the top k accuracy is defined as the ratio of $\frac{n}{N}$.
- *Stroke proportion:* To recognize a word with M letters, the number of strokes required to be entered is m. Then the stroke proportion for this word with the texts-entry system is defined as $\frac{m}{M}$.
- *Top k stroke proportion:* To recognize a M-letter word within k candidates, the number of strokes required to be entered is m. The the top k stroke proportion is defined as $\frac{m}{M}$.
- *WPM:* It is short for *word per minute* which measures the number of words entered in one minute. This metric is commonly used for evaluating the efficiency of an text-entry system.
- *LPM:* It is short for *letter per minute* which measures the number of letters entered in one minute. This metric takes the length of words into account when evaluating texts-entry efficiency.
- *NASA-TLX:* It is short for NASA-TLX Load Index, which is a widely used tool to assess user experience. The obtained score indicates how the user experience is. The lower it is, the better experience a user feels. In our work, we use it for the comparison of user experience between EchoWrite and smartwatch soft keyboard.

6.1 Stroke Recognition Performance

6.1.1 Overall Performance on Different Devices

As mentioned in Section 5.2, 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 processing. Fig. 9 shows the results. The average accuracy with a smartwatch is about 95.4 percent which is very close to that of EchoWrite on Mate 9 (i.e., 94.8 percent). 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.

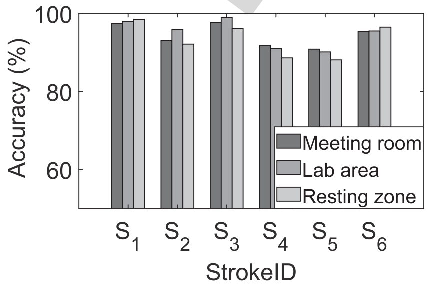


Fig. 10. The recognition accuracy of different strokes in three experimental environment.

6.1.2 Accuracy of Different Strokes

Fig. 10 shows average accuracies of recognizing different strokes under three settings. Overall, the accuracy can be up to 98.9 percent (S_3, lab area) and is no lower than 87.8 percent (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 94.1, 94.5 and 93.3 percent 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.

6.1.3 Accuracy of Different Participants

Fig. 11 displays stroke recognition performance for different participants in order to evaluate the impact of user diversity. The highest and lowest average accuracy in three environments among the 10 participants are 95.5 and 92.8 percent, respectively. Due to different proficiencies in performing finger gestures, their performances deviate from each other by a maximum gap of 2.7 percent. However, considering the minute standard deviation (i.e., 1.4 percent), we can claim that with same training, participants can achieve nearly the same performance.

6.1.4 Performance in Mobile Scenarios

Fig. 12 shows the accuracy of recognizing each stroke in mobile scenario. As we can see, the average accuracies of recognizing strokes in resting zone and mobile scenario are 92.6 and 90.8 percent, respectively. When people use EchoWrite during walking, the stroke recognition performance

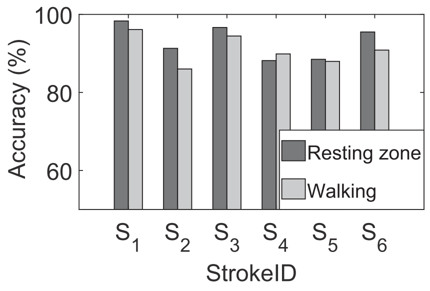


Fig. 12. The performance of stroke recognition in mobile scenarios.

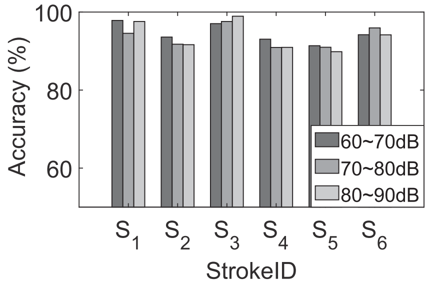


Fig. 13. The performance of stroke recognition under different noise levels.

slightly decreases by about 1.8 percent. The minute gap indicates that EchoWrite has relatively high scalability to normal walking scenario. The reason is that when a user walks steadily, there is no notable relative movement caused by walking between his/her writing finger and another hand that holds the device. As a result, the whole-body movement induces negligible frequency shifts in received echoes.

6.1.5 Performance Under Different Noise Levels

Fig. 13 shows EchoWrite's performance of recognizing strokes under three different noise levels. The average accuracies over different strokes under low, mediate and high noise level are 94.5, 93.6 and 93.3 percent, respectively. As we can see, external noise has negative effect on system performance. But when the noise level increases over 70 dB, purely increasing the sound volume nearly does not decrease the recognition performance. This result is owing to the use of near-ultrasound signals, which makes it difficult to be interfered by common noises.

6.1.6 Performance at Different Distances

We also evaluate the performance of EchoWrite under different distances between the finger and device in the meeting

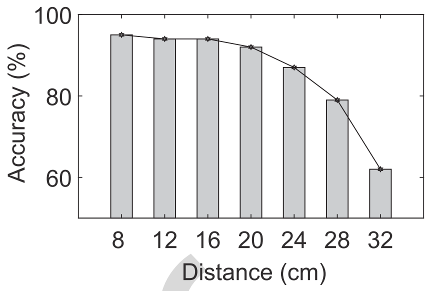


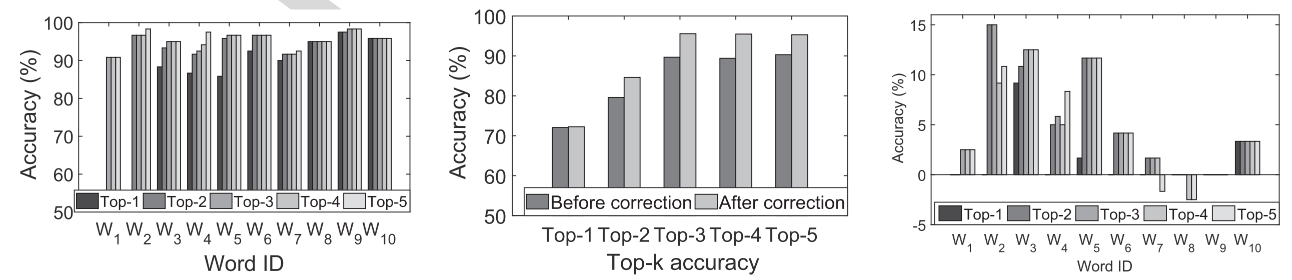
Fig. 14. The performance of stroke recognition at different distances.

room scenario. As it is difficult to measure the distance precisely in texts entry process, we define a 8×8 cm square region within which the finger writes strokes. As a result, the distance is defined between centers of the square and the device which varies from 8 cm to 32 cm. At each distance, we request participants to perform each strokes for 10 times and thus collects 600 trials in total. We calculate the overall accuracies of recognizing strokes at different distances as shown in Fig. 14. As we can see, within 20 cm, even though the accuracy decreases with distance, it remains relatively stable between 95 and 92 percent. However, when the distance exceeds 20 cm, the performance degrades sharply to 62 percent at a distance of 32 cm. This is because the power of received echoes decreases with the distance which results in increasing difficulty of detecting and segmenting strokes, especially when the distance exceeds 20 cm. Considering the difficulty of regulating precise distance in real-world experiments, we only request participants to do experiments within 20 cm without specification.

6.2 Texts Recognition Performance

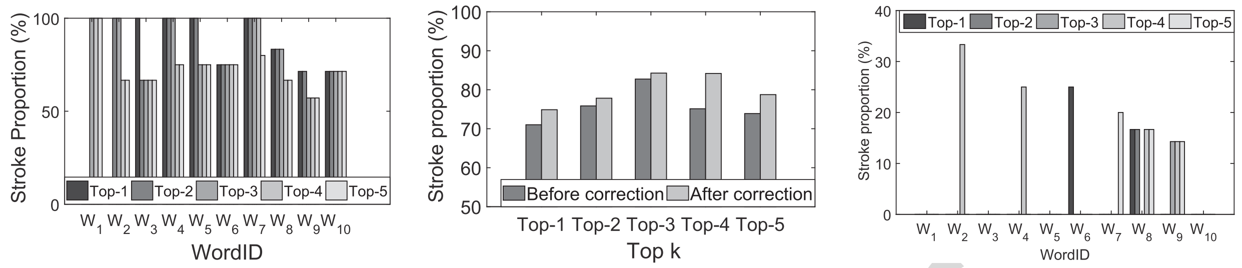
6.2.1 Accuracies of Different Words

Fig. 15a shows the top 5 accuracies of recognizing words shown in Table 1. For $k = 1, 2, \ldots, 5$, the average top k accuracy over different words is 73.2, 85.4, 94.9, 95.1 and 95.7 percent, respectively. By providing 3 candidates, EchoWrite can infer words correctly with a probability of 94.9 percent which is favorable for texts entry. When k exceeds 3, the accuracy slightly increases by 0.8 percent that can also be verified by Fig. 15b. 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.



(a) The top 5 accuracies with stroke correction for different words. (b) The average top 5 accuracies with and without stroke correction (c) Differences of top 5 accuracies beween with and without stroke correction

Fig. 15. The top 5 accuracies of recognizing words w/o and with stroke correction for different words.



(a) The stroke proportion with stroke correction for different words

(b) The comparison of stroke proportion without and with stroke correction

(c) The difference of stroke proportion for different words w/o and with stroke correction

Fig. 16. The top 5 accuracies of recognizing words w/o and with stroke correction for different words.

6.2.2 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. 15b. Overall, the average top 5 accuracies for both cases are 84.2 and 88.6 percent, 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. To figure out the impact more specifically, we also show the differences of top 5 accuracies for each tested word in Fig. 15c. It verifies that for most tested words, stroke correction technique improves their top k accuracies for varied percentages. Further, we can notice that for words with medium length, the performance improvement is more noticeable. This is because, for short words, correction adds more candidate words and the exact word may be listed beyond top 5 candidates; while for long words, there exist more error cases to be corrected. But our current correction method only consider substitution and ignore other cases. Improving the correction method is also one of our future work.

6.2.3 Stroke Proportion

As an indicator of efficiency, we compute stroke proportion by calculating how many strokes are needed for EchoType to recognize a word within top k candidates. The results are displayed in Fig. 16a. As we can see, contrary to recognition accuracy, stroke proportion decreases with k on the whole. This is because with less candidates, it is more difficult to recognize words accurately and requires inputting more strokes. Moreover, for more than half of the selected words, only about 60 percent strokes are needed when more than 3 candidates are provided. However, when only one candidate is considered, full strokes are required to precisely recognize corresponding word. To further evaluate the impact of stroke correction on letter proportion, we also show the average statistics of top k stroke proportions when stroke correction is not applied in Fig. 16b. It is obvious that when stroke correction is used, it requires to enter more strokes in order to recognize target word within k candidates, since there are more stroke sequences and corresponding words after correction. Fig. 16c shows the differences of stroke proportion for different words with various candidates. Overall, stroke proportions of longer words are more likely to increase after correction. Compared with short words, correction adds more candidates for long words, which requires to input more strokes to provide top k candidates.

6.2.4 Speed of Texts Entry

Fig. 17 displays 'the average speeds of entering given paragraphs randomly selected in Fry Instant Phrases with EchoWrite and touch screen on a smartwatch [43]. 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 7.7 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. 18. We can clearly see that the average speed is 25.5 LPM for EchoWrite which is higher than that of smartwatch by about 18.9 LPM.

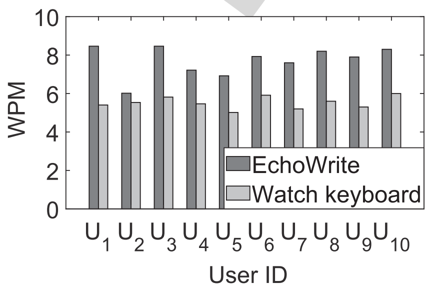


Fig. 17. The comparison of words-entry speed between EchoType and Soft keyboard on a smartwatch.

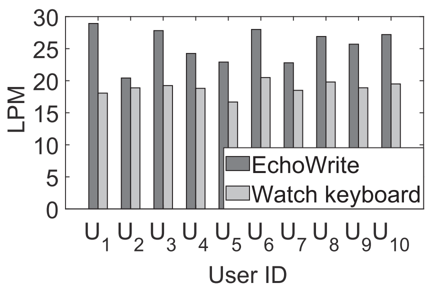


Fig. 18. The comparison of letter-entry speed between EchoType and Soft keyboard on a smartwatch.

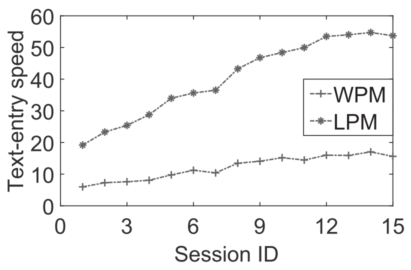
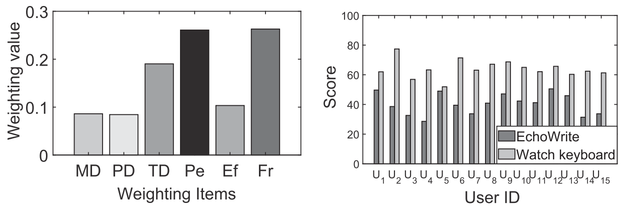


Fig. 19. The performance of text input after different numbers of training sessions.



(a) NASA weight (b) NASA score

Fig. 20. The results of user study experiment.

6.2.5 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) of which each lasts for about 2 minutes. We display the WPM and LPM of each session in Fig. 19. 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 16.0 and 54.0, respectively. With less practice time, the texts-entry speed of EchoWrite is higher than the state-of-the-art related works COMPASS [14] which achieves a speed of 12.5 WPM after 90-minutes practice.

6.2.6 User Experience Assessment

As aforementioned, we make use of NASA-TLX as our evaluation tool. NASA-TLX is a widely used subjective, multidimensional assessment tool that rates perceived workload in order to assess a task, system, or *etc*.. It divides the total workload (i.e., entering texts in our work) into six subscales which include mental demand (MD), physical demand (PD), temporal demand (TD), performance (Pe), effort (Ef), and frustration (Fr). We recruit 15 volunteers to participate in the user study experiments. Before experiments, we first investigate the weights that participants assign to different subscales for the task of entering texts, following the approach suggested in [44]. The investigation result is shown in Fig. 20a which has been averaged over different participants. We can see that performance (Pe), frustration (Fr) and temporal demand (TD) are the top 3 factors affecting users' assessment, while mental demand (MD) and physical demand (PD) weight less. This is because entering texts is a relatively short period which requires little mental and physical effort. After entering texts with both methods, we request each participant to give scores to each factor in NASA-TLX questionnaire, and then sum up weighted scores of all factors.[5] Fig. 20b shows the overall score of each participant. The overall score of EchoWrite is lower than that of smartwatch, which indicates that our system provides better user experience.

5. It is noted that for different participants, weights assigned for each factor are not the same. When we calculate overall score of each participant, we make use of his/her weights instead of the average weights.

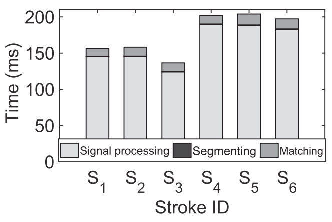


Fig. 21. The running time of different parts of data processing.

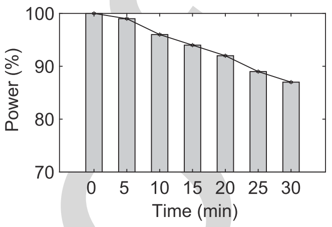


Fig. 22. The energy consumption when running the application.

6.3 System Running Performance

6.3.1 Running Time of Each Part

The time consumption of different data processing parts in EchoWrite is shown in Fig. 21. 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. We can also see that signal processing including Doppler enhancement and profile extraction occupies over 90 percent 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.

6.3.2 Energy Consumption

To obtain knowledge of energy consumption of EchoWrite, We also monitor power consumption of a Mate 9 running EchoWrite to recognize texts continuously and obtain results as shown in Fig. 22. During experiments, the EchoWrite application keeps running and experimenter 'writes' texts continuously. At the same time, the screen is turned on for the experimenter to make selection among candidate words. As is shown, the power level decreases from 100 to 87 percent after 30 minutes, which means about 3 percent 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.

6.3.3 CPU Occupation

We also evaluate the occupancy of CPU resources when a mobile device runs EchoWrite. During evaluation, we turn

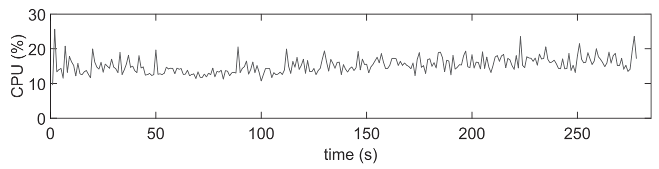


Fig. 23. The CPU overhead during the process of recognizing words.

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. 23 shows real-time CPU proportion consumed during recognizing words with EchoWrite. Even though the CPU proportion varies from 9.5 to 25.6 percent, the average is about 15.2 percent with a standard deviation of 2.3 percent which is acceptable for an application. Indicated by Fig. 21, we can further decrease CPU resources consumption by accelerating image and matrix processing using the GPU.

7 Discussion and Future Work

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

7.1 Scalability to Smartwatches

As aforementioned, EchoWrite can not run on existing smartwatches at present due to the limitation of CPU capacity. The results shown in Fig. 9 indicates that the hardware of a smartwatch can support to implement EchoWrite provided the CPU workload is affordable. We are optimistic about tackling this problem based on 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 [45]. 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. Another concern of scalability is the impact of inaudible sounds on common use of speakers and microphones, which actually also exists in previous works such as [10], [11]. However, as EchoWrite only uses one single speaker-microphone pair, we can control other microphone and speaker to emit and receive ordinary acoustic signals, if the device has multiple pairs. By utilizing signal processing techniques such as filtering, it is feasible to remove inaudible components and reduce its impact.

7.2 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.

7.3 User-Defined Input Scheme

EchoWrite works in a training-free way, more precisely, without collecting training data and training models, which reduces intensive labor workload and increases its scalability to different people and environments. But it needs a ’training’ process for users to learn the writing style of English alphabets and remember the mapping relation between stroke gestures and alphabets. Even though this overhead is acceptable according to our experiments, a possible way to further reduce the practice overhead for users is to take user's own writing habits and preference into account when designing input scheme in Section 3.1. 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.

8 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. It also provides more favorable user experience than touch screen-based method on smartwatches. Although EchoWrite has still some limitations, we envision that it has great potential to be applied on various smart devices.

Acknowledgments

The authors really appreciate the kind effort of the reviewers and AE for giving precious suggestions to revise and improve the paper. This work was supported in part by the China NSFC Grant (61802264, 61872248, U1736207), Guangdong NSF 2017A030312008, Guangdong Provincial Key R&D Program (2019B111103001), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education

Institutions of China (Grant No. 161064), Shenzhen Science and Technology Foundation (No. JCYJ20180305124807337, No. ZDSYS20190902092853047), GDUPS (2015), and Natural Science Foundation of SZU (No. 860-000002110537).

References

[1] J. Wang, D. Vasisht, and D. Katabi, "RF-IDraw: Virtual touch screen in the air using RF signals," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 235–246.

[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 *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 237–248.

[3] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim, "WiDraw: Enabling hands-free drawing in the air on commodity WiFi devices," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 77–89.

[4] T. Wei and X. Zhang, "mTrack: High-precision passive tracking using millimeter wave radios," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 117–129.

[5] S. Agrawal, I. Constandache, S. Gaonkar, R. R. Choudhury, K. Caves, and F. DeRuyter, "Using mobile phones to write in air," in *Proc. 9th Int. Conf. Mobile Syst. Appl. Serv.*, 2011, pp. 15–28.

[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 *Proc. IEEE 16th Int. Symp. Wearable Comput.*, 2012, pp. 52–59.

[7] M. Goel, L. Findlater, and J. Wobbrock, "WalkType: Using accelerometer data to accomodate situational impairments in mobile touch screen text entry," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2012, pp. 2687–2696.

[8] T. Ni, D. Bowman, and C. North, "AirStroke: Bringing unistroke text entry to freehand gesture interfaces," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2011, pp. 2473–2476.

[9] S. Nirjon, J. Gummeson, D. Gelb, and K.-H. Kim, "TypingRing: A wearable ring platform for text input," in *Proc. 13th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2015, pp. 227–239.

[10] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "FingerIO: Using active sonar for fine-grained finger tracking," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2016, pp. 1515–1525.

[11] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, 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 *Proc. 15th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2017, pp. 15–28.

[13] C. Yu, K. Sun, M. Zhong, X. Li, P. Zhao, and Y. Shi, "One-dimensional handwriting: Inputting letters and words on smart glasses," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2016, pp. 71–82.

[14] X. Yi, C. Yu, W. Xu, X. Bi, and Y. Shi, "COMPASS: Rotational keyboard on non-touch smartwatches," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2017, pp. 705–715.

[15] Y. Lu, C. Yu, X. Yi, Y. Shi, and S. Zhao, "BlindType: Eyes-free text entry on handheld touchpad by leveraging thumb's muscle memory," in *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, 2017, Art. no. 18.

[16] Apple Inc., "Siri," [Online]. Available: https://www.apple.com/ios/siri/. Accessed: Oct. 20, 2019, 2018.

[17] Microsoft Inc., "Cortana," [Online]. Available: https://www.microsoft.com/en-us/cortana/. Accessed: Oct. 20, 2019, 2018.

[18] M. T. I. Aumi, S. Gupta, M. Goel, E. Larson, and S. Patel, "DopLink: Using the doppler effect for multi-device interaction," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2013, pp. 583–586.

[19] K.-Y. Chen, D. Ashbrook, M. Goel, S.-H. Lee, and S. Patel, "AirLink: Sharing files between multiple devices using in-air gestures," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 565–569.

[20] S. Gupta, D. Morris, S. Patel, and D. Tan, "SoundWave: Using the doppler effect to sense gestures," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2012, pp. 1911–1914.

[21] 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 *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 474–485.

[22] Z. Sun, A. Purohit, R. Bose, and P. Zhang, "Spartacus: Spatially-aware interaction for mobile devices through energy-efficient audio sensing," in *Proc. 11th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2013, pp. 263–276.

[23] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda, "SwordFight: Enabling a new class of phone-to-phone action games on commodity phones," in *Proc. 10th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2012, pp. 1–14.

[24] S. Yun, Y.-C. Chen, and L. Qiu, "Turning a mobile device into a mouse in the air," in *Proc. 13th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2015, pp. 15–29.

[25] W. Mao, J. He, and L. Qiu, "Cat: High-precision acoustic motion tracking," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, 2016, pp. 69–81.

[26] T. Yu, H. Jin, and K. Nahrstedt, "WritingHacker: Audio based eavesdropping of handwriting via mobile devices," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 463–473.

[27] H. Du, P. Li, H. Zhou, W. Gong, G. Luo, and P. Yang, "WordRecorder: Accurate acoustic-based handwriting recognition using deep learning," in *Proc. IEEE INFOCOM*, 2018, pp. 1448–1456.

[28] M. Chen *et al.*, "Your table can be an input panel: Acoustic-based device-free interaction recognition," in *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, 2019, Art. no. 3.

[29] T. Yu, H. Jin, and K. Nahrstedt, "Mobile devices based eavesdropping of handwriting," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2019.2912747.

[30] T. Yu, H. Jin, and K. Nahrstedt, "Audio based handwriting input for tiny mobile devices," in *Proc. IEEE Conf. Multimedia Inf. Process. Retrieval*, 2018, pp. 130–135.

[31] Y. Zou, Q. Yang, Y. Han, D. Wang, J. Cao, and K. Wu, "AcouDigits: Enabling users to input digits in the air," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2019, pp. 313–321.

[32] S. Shen, H. Wang, and R. R. Choudhury, "I am a smartwatch and I can track my user's arm," in *Proc. 14th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2016, pp. 85–96.

[33] C. Zhang *et al.*, "FingerSound: Recognizing unistroke thumb gestures using a ring," in *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, 2017, Art. no. 120.

[34] C. Zhang, J. Tabor, J. Zhang, and X. Zhang, "Extending mobile interaction through near-field visible light sensing," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 345–357.

[35] Google Inc., "Google soli project," [Online]. Available: https://atap.google.com/soli/. Accessed: Oct. 20, 2019, 2015.

[36] Super English Kid, "Super English Kid," [Online]. Available: http://www.superenglishkid.com/2014/11/stroke-order-worksheet-for-teaching-how.html. Accessed: Oct. 20, 2019, 2014.

[37] M. Davies, "Corpus of contemporary american english." Accessed: Apr. 03, 2019, 2008. [Online]. Available: https://corpus.byu.edu/coca/

[38] H. Zhang, W. Du, P. Zhou, M. Li, and P. Mohapatra, "DopEnc: Acoustic-based encounter profiling using smartphones," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, 2016, pp. 294–307.

[39] P. Soille, *Morphological Image Analysis: Principles and Applications*. Berlin, Germany: Springer, 2013.

[40] P. Holoborodko, "Applied mathematics and beyond." Accessed: Jan. 16, 2018, 2015. [Online]. Available: http://www. holoborodko. com/ pavel/

[41] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. 3rd Int. Conf. Knowl. Discovery Data Mining*, 1994, pp. 359–370.

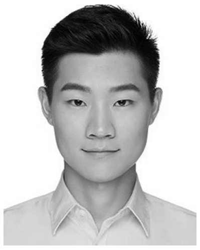
[42] M. Davies, "Word frequency data," [Online]. Available: https://www.wordfrequency. info/. Accessed: Oct. 20, 2019, 2010.

[43] T. Rasinski, "Fry instant phrases," [Online]. Available: http://www.timrasinski.com/presentations/. Accessed: Oct. 20, 2019, 2018.

[44] S. Rubio, E. Díaz, J. Martín, and J. M. Puente, "Evaluation of subjective mental workload: A comparison of SWAT, NASA-TLX, and workload profile methods," *Appl. Psychol.*, vol. 53, no. 1, pp. 61–86, 2004.

[45] A. V. Oppenheim, *Discrete-Time Signal Processing*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice Hall Press, 1999.

Kaishun Wu (Member, IEEE) received the BEng degree from Sun Yat-sen University, Guangzhou, China, in 2007, and the PhD degree from CSE Department, Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2011. He is currently a distinguished professor with the College of Computer Science and Software Engineering, Shenzhen University. His research interests include wireless communications and mobile computing. He won the best paper awards of several international conferences, such as IEEE Globecom 2012 and IEEE ICPADS 2012.



Qiang Yang (Student Member, IEEE) received the MEng degree in computer technology from Shenzhen University, Shenzhen, China, in 2019. He is currently working toward the PhD degree in the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include mobile sensing, human-computer interaction, and Internet of Things (IoT).

Baojie Yuan is working toward the postgraduate degree in the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. Her research interest include mobile computing.

Yongpan Zou (Member, IEEE) received the BEng degree in chemical machinery from Xi'an Jiaotong University, Xi'an, China, in 2013, and the PhD degree from the CSE Department, Hong Kong University of Science and Technology (HKUST), in 2017. He is currently an assistant professor with the College of Computer Science and Software Engineering, Shenzhen University, since September 2017. His research interests mainly include wearable/mobile/ubiquitous computing and HCI. For more information, please visit: https://yongpanzou.github.io/

Rukhsana Ruby (Member, IEEE) received the master's degree from the University of Victoria, Victoria, Canada, in 2009, and the PhD degree from the University of British Columbia, Vancouver, Canada, in 2015. Her research interests include the management and optimization of next generation wireless networks. She has authored nearly 50 technical papers of well-recognized journals and conferences.

Mo Li (Member, IEEE) received the BS degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, and the PhD degree from CSE Department, Hong Kong University of Science and Technology (HKUST), Hong Kong. He is currently an associate professor with the School of Computer Science and Engineering, Nanyang Technological University. He is heading Wireless And Networked Distributed Sensing (WANDS) System Group, Parallel and Distributed Computing Centre (PDCC). He is on the editorial board of the *ACM Transactions on Internet of Things*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Mobile Computing*, and *IEEE/ACM Transactions on Networking*.