2017 IEEE 23rd International Conference on Parallel and Distributed Systems

SoundWrite II: Ambient Acoustic Sensing for Noise Tolerant Device-Free Gesture Recognition

Gan Luo, Mingshi Chen
College of Communications Engineering,
PLA Army Engineering University
Nanjing, China
Email:[luogan418,cms603421]@gmail.com

Ping Li, Maotian Zhang
College of Communications Engineering,
PLA Army Engineering University
Nanjing, China
Email: [pingli0112,maotianzhang]@gmail.com

Panlong Yang
School of Computer Science and Technology, University of Science and Technology of China
Hefei, China
Email: panlongyang@gmail.com

***Abstract*—Acoustic sensing has brought forth the advances of prosperous applications such as gesture recognition. Specifically, ambient acoustic sensing has drawn many contentions due to the ease of use property. Unfortunately, the inherent ambient noise is the major reason for unstable gesture recognition. In this work, we propose "*SoundWrite II*", which is an improved version of our previously designed system. Compared with our previous design, we utilize the two threshold values to identify the effective signals from the original noisy input, and leverage the MFCC (Mel frequency cepstral coefficient) to extract the stable features from different gestures. These enhancements could effectively improve the noise tolerant performance for previous design. Implementation on the Android system has realized the real time processing of the feature extraction and gesture recognition. Extensive evaluations have validated our design, where the noise tolerant property is fully tested under different experimental settings and the recognition accuracy could be 91% with 7 typical gestures.**

I. Introduction

Smart devices have advanced to serve as an inseparable tool and aid for daily life. Particularly, microphone is serving as a communication device for voice interactions, which is a regular configuration for smartphones. Acoustic sensing is an innovative extension on the functionality of microphone. In that, the microphone serves as a sensor (passive mode) or radar (active mode), where moving objects nearby could be tracked and localized. Inspired by this technology, acoustic sensing is customized for gesture recognition.

In application domain, for active mode, a customized ultrasound signal is sent for object tracking, which needs more energy consumption for transmission and will possibly lead to healthcare issues because of the ultra-sonic wave usages. While for the passive mode, the acoustic sensing scheme collects the sounds emitting from the objects ambiently, which encourages more inspiring long-term sensing, such as keyboard typing and gaming. By hand stroking and sliding on the table, the emitted sound could be identified and recognized as different gestures.

While in technical domain, for active mode, such as LLAP [1] and FingerIO [2] the phase differences are fully exploited for accurate movement detection. In addition, Strata *et al* [3] can effectively track the phase changes under strong noise. Unfortunately, this solution is not customized for gestures recognition, which needs relatively long time monitoring and consumes large amount of energy if the tracking signals are emitted.

On the other hand, WritingHacker [4] leverages the embedded microphone on the phone to snoop the victim's input gesture but suffered from the relatively poor accuracy. SoundWrite [5] uses the similar physical features of acoustic signal and achieves a higher accuracy performance. However, the position of handwriting and device must be predefined, and its performance is vulnerable to strong ambient noise.

Vision processing technology is widely used in object tracking and gesture recognition. For example, Microsoft Kinect [6] uses depth sensor as well as camera to recognize user's movement, LeapMotion [7] is another vision-based object tracking device which leverages the similar technology with Kinect. However, these approaches are not suitable in smartphones for limited computation power, energy, and sensing capability (i.e., no depth sensors).

Recently, RF-based device-free object tracking algorithm in smart home and office environment has been widely investigated [8] [9] [10] [11] [12]. WiSee [8] uses WiFi signals to recognize 9 predefined gestures in several environments. WiTrack [9] and WiTrack 2.0 [10] apply Frequency Modulated Continuous Wave (FMCW) to track a user's hand in high accuracy. Although these algorithms achieve considerably high performance, it is hardly to apply these algorithms in smartphones. For example, the FMCW signal used in WiTrack [9] and WiTrack 2.0 [10] need customized hardware which can sweep the channel in nearly 2 GHz bandwidth.

In this work, we improve our previously designed system "*SoundWrite*" [5] by incorporating the noise tolerant ability, where the gesture recognition rate could be high in real working scenario. There are two intrinsic challenges need to be formally addressed before this inspiring vision could be achieved.

- First, the background noise imposes negative effect on identifying the start and ending points of a basic gesture. Strong noise will trigger wrong gesture recognition and lead to more contagious errors in consequent recognition.
- Second, the noise will interfere the feature extraction and gesture recognition. Strong background noise is bearing the wide spectrum range, which will inevitably interfere

978-1-5386-2129-5/17/31.00 ©2017 IEEE
DOI 10.1109/ICPADS.2017.00027

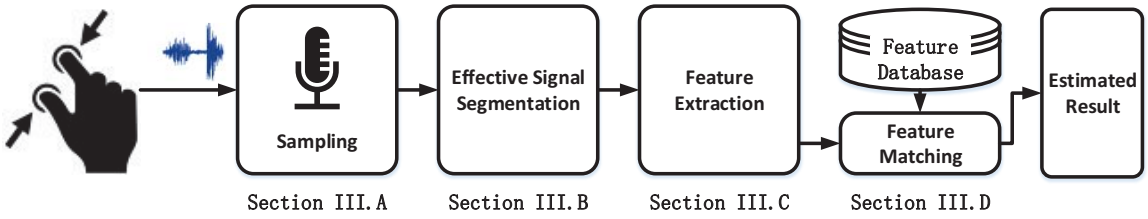


Fig. 1: Algorithm Diagram

with the sound emitting from the gestures in spectrum domain.

There are three contributions in our work.

- We present a dual-threshold scheme to deal with the strong background noise for segmentation. The threshold values are carefully selected according to various tests and show satisfiable performance across those scenarios.
- We incorporate MFCC [13] scheme into our previous design, which effectively solves the intangible feature extraction problem when leveraging the features using ASD only. Futhermore, we carefully select the parameters when MFCC and ASD are jointly used, and achieve satisfiable performance.
- We conduct extensive evaluations to validate our design. Evaluations are made across different volunteers in various scenarios. Specifically, we show the noise tolerant property with little training overhead.

The rest parts of this paper are organized as follows: First of all, in Section II, we outline the basic design principle and components with working flow illustrations, and then present a comprehensive introduction with technical details for system design and implementation. Secondly, we validate our design with experimental results and analysis in Section III. Finally, we conclude the paper in Section IV.

II. System Design and Implementation

A. System Overview

In this section, we briefly introduce the workflow of our system. As Fig.1 shows, our system consists of four components: sampling, effective signal segmentation, feature extraction, and feature matching. Specifically, when user inputs gesture on the desktop, the acoustic signal caused by the friction between fingers and desktop will be captured by the imbedded microphone on the phone. Then, this acoustic signal will be fed into the "Effective Signal Segmentation" module to extract the valid parts. In the next step, an MFCC (Mel Frequency Cepstrum Coefficient) analysis will be performed to extract the unique feature of such acoustic signal. Finally, classification algorithm such as KNN (K-Nearest Neighbors) [14] will be applied to estimate the input gesture by comparing the extracted acoustic feature with the previously learned feature which is stored in the feature database. We will give a detailed descriptions in Section II.

B. Sampling Process

As depicted in Fig. 2, most of the significant information is retained under 2kHz. Meanwhile, high-frequency signals contain more noise. To this end, we set the sampling frequency at 8kHz, which is sufficient to extract gesture features according to Shannon theorem. For this purpose, we call the *AudioRecorder* function of android studio to collect the acoustic signal.

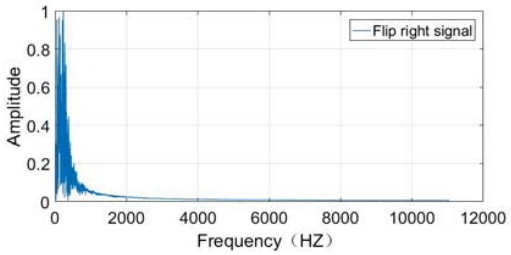


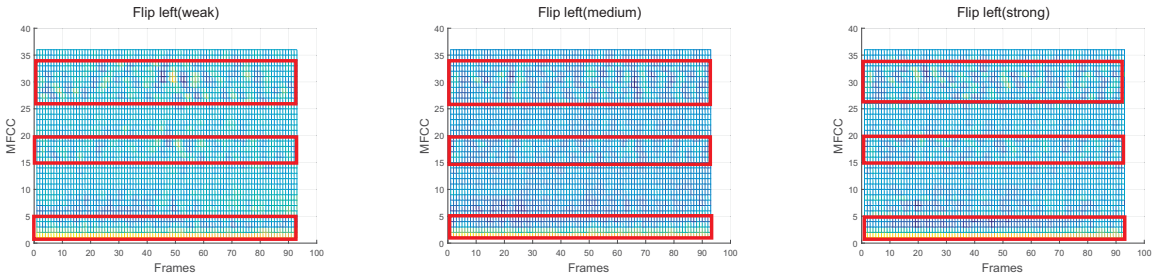
Fig. 2: Amplitude spectrum density of acoustic signal

The main subject of our sampling module is to call the function of Audio System of Android, except for the hardware and audio driver, the entire implementation of android audio can be regarded as user. At the APP level, we get the permissions of the microphone recording and storage device writing by "android.permission.RECORD_AUDIO" and "android.permission.WRITE_EXTERNAL_STORAGE" statement. Then on the framework layer, we call *AudioTrack* function to collect audio and *AudioRecorder* to record audio data. The Lib layer is an interface transition from the upper framework layer class written by Java language to HAL (hardware abstraction layer) which manages audio equipment.

C. Effective Signal Segmentation

We use the moving average window to eliminate the interference of ambient noise. After moving average filtering, the noise of the primitive acoustic signal is reduced, while the significant characteristics of the original signal are retained. Moreover, it is helpful to decrease the complexity of the subsequently feature extraction.

Fig.4 shows the amplitude of one gesture input in time domain. For each acoustic signal we use the short time energy and zero-crossing rate to detect the effective start point and end point. We first normalize the acoustic signal and set the threshold values for the short time energy and zero-crossing rate respectively. Only when both parameters have reached their given threshold values can the acoustic signal be



(a) Flip left (weak) (b) Flip left (medium) (c) Flip left (strong)

Fig. 3: The MFCC of one gesture in different strength

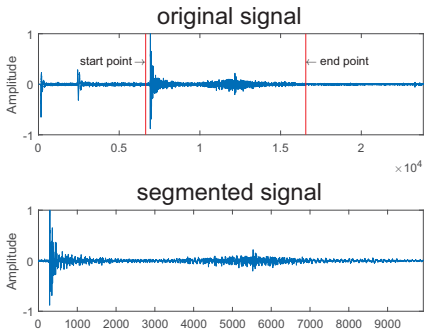


Fig. 4: Illustration of Effective Signal Segmentation

regarded as an effective gesture signal. Otherwise, the acoustic signals will be treated as noise. Next, we set "max-silence" as the maximum mute length allowed in the effective signal segment and "min-len" as the shortest length of the effective signal segment. We can easily find the start point and end point of the effective signal via the two threshold values. At last, we separate the effective signal from the original signal for the next step analysis. As for android design, we call *System.arraycopy()* function to extract the effective signals from the originals, which is the build-in method of Android system.

D. Feature Extraction

In this section, we describe how Soundwrite II identifies different gestures and the features we have used. In our previous work, Zhang *et al* [5] has recognized different gesture type by using amplitude spectrum density (ASD) features which present the frequency domain profiles of gestures. For each acoustic signal of the input gesture, "Soundwrite" calculates the FFT and finds the peak of spectrum, since different gestures exhibit distinct values across frequencies, so different gestures can be recognized and distinguished. However, this method is not robust. When we slide on the table with different strength, the ASD of one gesture will change differently.

To avoid the disadvantage of ASD, we adopt the most commonly used acoustic feature named MFCC. MFCC takes human perception sensitivity with respect to frequencies into consideration, and therefore are best for speech recognition.

Fig.3 shows the MFCC of gesture flip left with different strength. We reuse the data of the same gesture tested before, and calculate their MFCC feature. As Fig.3 shows, the MFCC feature changes are not obviously. The areas marked with red boxes are the place where the main sub-features we consider in MFCC, from all three sub-figures, we can see that the change of MFCC is very slight, which shows a great advantage in feature extraction.

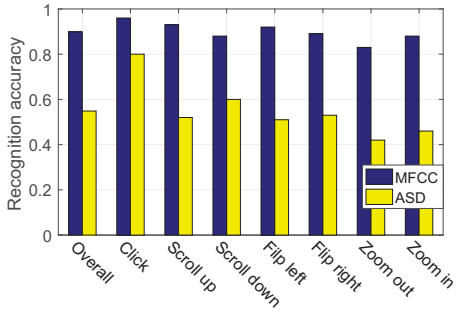


Fig. 5: Average accuracy with MFCC and ASD

Fig.5 further demonstrates the advantage of gesture recognition by using MFCC as the feature vector. From the figure we can see that, for different gestures, MFCC based algorithm can achieve a higher recognition accuracy than ASD based algorithm, which has an nearly 30% improvement.

To obtain the MFCC feature, each sound segment is divided into frames with length Nc, and the overlap of two adjacent frames is Nf. For each frame, the number of cepstrum coefficients we adopt is Nm. So for each frames i=1,2,3...N, we calculate a static vector of MFCCs: $[c_1, c_2, c_3...c_t]$, and

$$c_i = \frac{1}{N}\sum_{j=1}^{N} c_j^i, i = 1, ...Nm$$

In our prototype, the sampling rate of audio signal is 8KHz, and we set Nc=256, Nf=80 and Nm=36, N is depending on the length of the effectively acoustic signal and c_j^i denotes the ith MFCC values of jth frame. The corresponding frame length is 32 ms and the frame offset is 10 ms, the reason we set the parameters is based on such observation, with the longer frame length, the short-term stability of the acoustic signal can hardly be guaranteed, while with the shorter frame length, it will increase the computational complexity and cause a larger processing time.

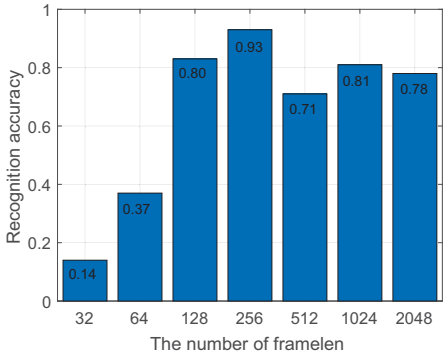


Fig. 6: average accuracy with different frame length

Fig.6 further shows the reason why we set Nc=256, Nf=80. In our experiments, we set the frame with different number of sampling points, and the frame offset is about one-third of the frame length. We can see clearly that the average accuracy achieves the highest value when we set the frame point at 256 sampling points and the frame offset is 80, the corresponding length of time is 256/8000*1000 = 32ms.

For each sound fragment, we extract the following features, which are commonly used to classify acoustic signal.

- mfcc: Which contains 12 MFCCs values, 12 first-order difference parameters, and 12 second-order difference parameters. The difference vector is calculated using the following equation:

$$d_t = \begin{cases} c_{t+1} - c_t, t < K \\ \frac{\sum_{k=1}^{K} k(c_{t+k} - c_{t-k})}{\sqrt{2\sum_{k=1}^{K} k^2}}, others \\ c_t - c_{t-1}, t \geq Q - K \end{cases}$$

  Where d_t denotes the tth first order difference, c_t denotes the tth cepstrum coefficient, Q denotes the order of the cepstrum coefficient, and K denotes the time difference of the first-order derivative, which is set at 1. The second-order difference vector can be obtained by replacing c_t with d_t and recalculate the above equation.
- amp: Which shows the short time energy, denoted as

$$\text{amp} = \sum_{\text{n}=1}^{N} |X(n)|$$

  Where $X(n)$ is the signal amplitude and N is the frame length.
- zcr: The zero-crossing rate, denoted as

$$zcr = \sum_{n=1}^{N-1} |X(n) - X(n+1)|$$

We denote the feature vector F for each acoustic signal fragment by:

$$F = [c_1, c_2 ....., c_{N_m}, amp, zcr]$$

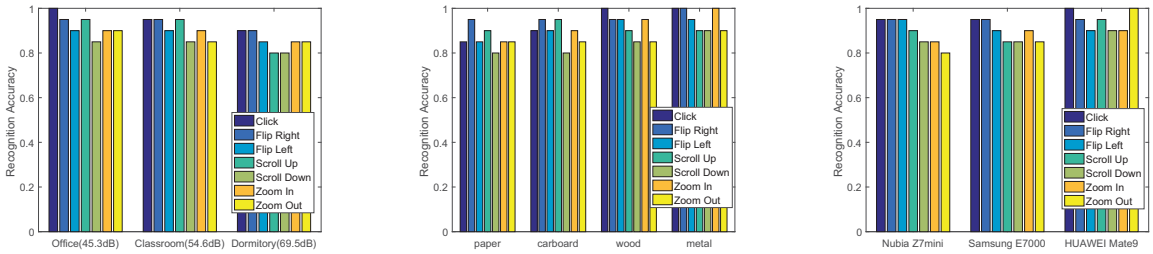
Here we introduce the android implementation of feature extraction module. We input the effective acoustic gesture signal into the MFCC class. Firstly, the signal is divided into a series of frames, each frame lasts for 30ms and takes about 256 points, which can effectively avoid excessive variation between frames. To extract the whole acoustic features, the overlay between adjacent frames is set at $1/3$. Next, we utilize the *FFT.java* class for FFT transformation in each frame of the acoustic signal, the information in time domain will be converted to the energy distribution in frequency domain. Then, we call the *calculateMelBasedFilterBank()* function in the *MFCC.java* class to get the Mel filter banks. In order to eliminate the effect of harmonics and highlight the resonant peak of the original acoustic signal, we smooth the signal through 20 sets of Mel-scale filter banks. The Mel filters protect the MFCC from the effect of the signal strength variation and reduce the computational complexity as well. After that, we call the *Math.log* function to calculate the logarithmic energy output for each filter bank. Finally, the above mentioned logarithmic energy is multiplied by the DCT matrix calculated by the *initializeDCTMatrix()* function of the *MFCC.java* class, and the discrete cosine transform is performed to obtain the 12 MFCCs values. We get the static values from the *getParameters()* function of the *MFCC.java* class.

E. Feature Matching

KNN(k-nearest neighbors) is a non-parametric classification method commonly used in pattern recognition because of its effectivity and simplicity. The basic idea of KNN algorithm is that, an object should be assigned to the most possible category among its k nearest neighbors, which were trained and labeled previously. The output of KNN is a class label and usually uses Euclidean distance to determine the neighbors.

We input the feature vector of undetermined gesture and feature database of the training gesture into the feature matching module as is shown in the Fig.1. Firstly, we call the *Distancecalculate()* function to calculate the distance between the target gestures with the training gestures. Then, we call the *mindiatance()* function to find k nearest neighbors. Finally, we use the *maxcount()* function to count the maximum number label of k nearest neighbors and output its gesture label.

In SoundWrite II, for the sake of convincible demonstration on our algorithm, we train 7 gestures and each gesture was repeated 20 times. With the increased parameter k, we conduct



(a) with different scenes (b) with different materials (c) with different smartphones

Fig. 7: Average accuracy evaluation over different scenarios

the gesture recognition experiments respectively and establish the training set. In our gesture recognition algorithm, the statistical accuracy rate has shown the optimal setting of parameter k is 7, and the corresponding recognition accuracy could be 91.4%.

III. Performance Evaluation

To evaluate the performance of our proposed algorithm, we conduct a series of experiments in real environment. We make extensive evaluations on macro benchmark for system level performance and micro benchmark for component based evaluation with various impact factors.

A. Experiment Setup

We set our test environment in office, classroom and dormitory, respectively, and the ambient noise in each scenario gradually increases. Our proposed application is installed in three different smartphones, and ten volunteers are invited to evaluate the gesture recognition accuracy in different environment configurations. We set the number of training instances per gesture at 10, and for each experiment, the volunteers are required to input seven different gestures for 20 times. Then, we count the recognition accuracy of each gesture separately.

B. Macro-Benchmark

Average accuracy in different scenes: To investigate the performance of our algorithm in different scenes, three typical scenes, office, classroom and dormitory are considered, and the corresponding ambient noise levels are $45.3dBm$, $54.6dBm$ and $69.5dBm$, respectively. In each scenario, volunteers are required to input all 7 gestures and repeat each gestures up to 20 times.

As depicted in Figure.7a, the average gesture recognition accuracy decreases with the increased ambient noise.

Average accuracy with different materials: Fingers slide different type of surface materials will produce different acoustic signals. To investigate the performance of our algorithm among different materials, series of experiments are conducted and four common materials, rubber, cardboard, wood and metal are considered. volunteers are required to input gestures on each material, and count the recognition accuracy of each gesture.

As Figure.7b illustrates, gesture recognition accuracy on different surface material has a different performance. Specifically, when the surface materials are metal and wood, the accuracy is relatively high, while the accuracy is lower when the surface materials are rubber and cardboard. The main reason is that, the surface of metal and wood are rough and solid which make the acoustic signal more stronger, while the surface of rubber and cardboard are smooth and softly which make the sound more weaker. This difference of acoustic signal power affects the gesture recognition accuracy between different material.

Average accuracy with different kinds of smartphones: We also test the stability of our algorithm between different kinds of smartphones, since different smartphone has various hardware configurations. Three common smartphones, Nubia Z7mini, Samsung E7000 and HUAWEI Mate9 are used, and the experimental results are shown in Figure.7c.

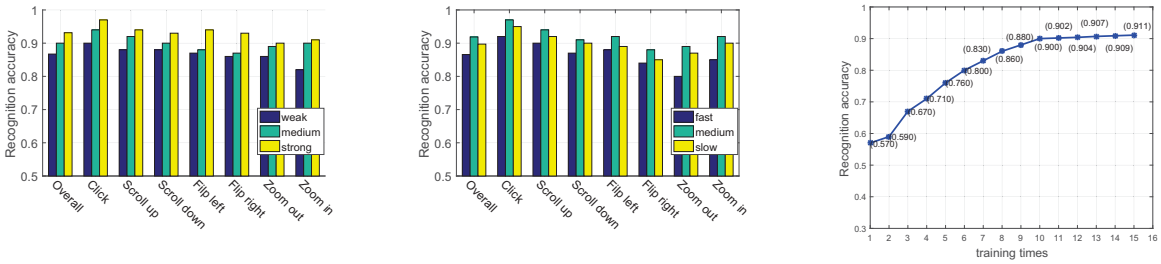
As this figure shows, different hardware configuration can impose a slight impact on accuracy performance. Specifically, the accuracy performance on HUAWEI Mate 9 achieves the highest value, while Nubia Z7mini and Samsung E7000 achieve slightly lower performance. This is because Mate 9 is equipped with a better microphone that can capture the more slighter sound produced by the finger sliding, which is helpful for the signal processing.

C. Impact Factors

To have a deep understanding of SoundWrite II, we conduct extensive experiments to evaluate the impact of some key factors on our algorithm performance.

The impact of touch strengthes on average accuracy: Different users may input the same gesture with different touch strengths. In this section, we estimate the impact of different touch strengths on gesture recognition accuracy. We require volunteers to input the same gesture with different touch strengthes, and gather the gesture estimation results. We employ a digital weighting scale with 0.01g resolution to measure the input strength. The average strength of strong, medium and weak input are 3.56N, 1.76N and 0.75N, respectively. We note that these results are derived by F = mg, where m is the measured weight, g is the acceleration of gravity, and N is the unit of force.

As Figure.8a illustrates, the greater the touch strength, the higher recognition accuracy SoundWrite II can achieve. The reason for the phenomenon is that, with a greater touch strength, the generated acoustic signal is louder, making the



(a) with different strength (b) with different sliding speed (c) with different training instances

Fig. 8: Average accuracy evaluation over different input behaviours

features more easily to be extracted, therefore achieve a better performance.

The impact of sliding speed on average accuracy: The second factor we evaluate is the sliding speed of the input gesture. In this experiment, the volunteers are invited to input the same gesture with different sliding speeds. For example, volunteers are required to input the same gesture "flip right" at the sliding distance at $15cm$ within $0.2s$, $0.48s$ and $1s$, respectively. The collected results are shown in Figure.8b.

As Figure.8b illustrates, when user inputs the gesture with a medium sliding speed, SoundWrite II achieves the highest performance. The main reason of this phenomenon is that, since the length of each divided frame in MFCC is constant, the faster the slide speed, the lower the number of frame sequence can be divided, thus the less information can be leveraged for the feature extraction, and the accuracy is weaker consequently. On the other hand, with the over-slow slide speed, the amount of frame sequence becomes larger, which lead to a lot of ambient noise to be regarded as the signal feature, thus affect the final recognition accuracy performance.

The impact of training instances on average accuracy: The number of training instances plays an important role on gesture recognition accuracy. Intuitively, the more number of training instances, the higher the accuracy of gesture recognition. Figure.8c shows the impact of the number of training instances on recognition accuracy.

As such figure illustrates, with one initial training, the recognition accuracy is around 57%, however, as the number of training instance escalates to 10, the recognition accuracy increases to 90%.

IV. Conclusion and Future Work

In this paper, we improve our previous work "SoundWrite" and proposed a new algorithm named "SoundWrite II". Comparing with "SoundWrite" [5], "SoundWrite II" applies a robustness signal segmentation scheme and uses a more advanced acoustic feature "MFCC" to characterize the input audio signal. We have implemented "SoundWrite II" on COTS(Commercial Off-The-Shelf) smartphones and validated its effectiveness and robustness via comprehensive experiments. The results show that, "SoundWrite II" can achieve 91% gesture recognition accuracy with 7 typical gestures in average, and it is robust in the presence of ambient noise.

In future work, we will incorporate more gestures and improve the accuracy with more advanced algorithms. Moreover, we will consider to leverage the advantages of active mode, we are to investigate how to improve the recogntion accuracy and decrease the energy consumption at the same time.

Acknowledgment

This research is partially supported by 2017YFB0801702, National key research and development plan, NSFC with No. 61772546, 61632010, 61232018, 61371118, 61402009, 61672038, 61520106007, China National Funds for Distinguished Young Scientists with No.61625205, Key Research Program of Frontier Sciences, CAS, No. QYZDY-SSW-JSC002, and NSF OF Jiangsu For Distinguished Young Scientist: BK20150030.

References

[1] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016, pp. 82–94.
[2] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio:using active sonar for fine-grained finger tracking," in *CHI Conference on Human Factors in Computing Systems*, 2016, pp. 1515–1525.
[3] S. Yun, Y.-C. Chen, H. Zheng, L. Qiu, and W. Mao, "Strata: Fine-grained acoustic-based device-free tracking."
[4] T. Yu, H. Jin, and K. Nahrstedt, "Writinghacker: audio based eavesdropping of handwriting via mobile devices," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 463–473.
[5] M. Zhang, P. Yang, C. Tian, L. Shi, S. Tang, and F. Xiao, "Soundwrite: Text input on surfaces through mobile acoustic sensing," in *International Workshop on Experiences with the Design and Implementation of Smart Objects*, 2015, pp. 13–17.
[6] *Microsoft X-box Kinect*, http://xbox.com.
[7] *Leap motion*, http://www.leapmotion.com/.
[8] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," in *International Conference on Mobile Computing & NETWORKING*, 2013, pp. 27–38.
[9] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller, "3d tracking via body radio reflections," pp. 317–329, 2013.
[10] F. Adib, Z. Kabelac, and D. Katabi, "Multi-person localization via rf body reflections," in *Usenix Conference on Networked Systems Design and Implementation*, 2015, pp. 279–292.
[11] K. Joshi, D. Bharadia, M. Kotaru, and S. Katti, "Wideo: fine-grained device-free motion tracing using rf backscatter," in *Usenix Conference on Networked Systems Design and Implementation*, 2015, pp. 189–204.
[12] I. Poupyrev, "Soli: ubiquitous gesture sensing with millimeter wave radar," *Acm Transactions on Graphics*, vol. 35, no. 4, p. 142, 2016.
[13] *MFCC*, https://en.wikipedia.org/wiki/Mel-frequency_cepstrum.
[14] *KNN*, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.