FingerSound: Recognizing unistroke thumb gestures using a ring

CHENG ZHANG, Georgia Institute of Technology
ANANDGHAN WAGHMARE, Georgia Institute of Technology
PRANAV KUNDRA, Georgia Institute of Technology
YIMING PU, Georgia Institute of Technology
SCOTT GILLILAND, Georgia Institute of Technology
THOMAS PLOETZ, Georgia Institute of Technology
THAD E. STARNER, Georgia Institute of Technology
OMER T. INAN, Georgia Institute of Technology
GREGORY D. ABOWD, Georgia Institute of Technology

We introduce FingerSound, an input technology to recognize unistroke thumb gestures, which are easy to learn and can be performed through eyes-free interaction. The gestures are performed using a thumb-mounted ring comprising a contact microphone and a gyroscope sensor. A K-Nearest-Neighbor(KNN) model with a distance function of Dynamic Time Warping (DTW) is built to recognize up to 42 common unistroke gestures. A user study, where the real-time classification results were given, shows an accuracy of 92%-98% by a machine learning model built with only 3 training samples per gesture. Based on the user study results, we further discuss the opportunities, challenges and practical limitations of FingerSound when deploying it to real-world applications in the future.

CCS Concepts: • **Human-centered computing** → **Gestural input**; *Text input*;

Additional Key Words and Phrases: Input, Wearable, Ring, Gesture Recognition

ACM Reference format:
Cheng Zhang, Anandghan Waghmare, Pranav Kundra, Yiming Pu, Scott Gilliland, Thomas Ploetz, Thad E. Starner, Omer T. Inan, and Gregory D. Abowd. 2017. FingerSound: Recognizing unistroke thumb gestures using a ring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 120 (September 2017), 19 pages.
DOI: http://doi.org/10.1145/3130985

1 INTRODUCTION

Wearable computing has developed from a niche to a sizable consumer market that has seen substantial uptake in most recent years. Wearable devices – most prominently smart watches and fitness bands, but also mobile virtual reality devices such as the Oculus Rift – can now be considered commodity hardware and large proportions of the population are using them in their everyday lives. With such popularity comes the desire and opportunity

Author's addresses: C. Zhang and A. Waghmare and P. Kundra and Y. Pu and S. Gilliland and T. Ploetz and T. Starner , School of Interactive Computing, Georgia Tech; O. T. Inan,School of Electrical and Computer Engineering, Georgia Tech; G. D. Abowd, School of Interactive Computing, Georgia Tech.

2474-9567/2017/9-ART120 $15.00
DOI: http://doi.org/10.1145/3130985

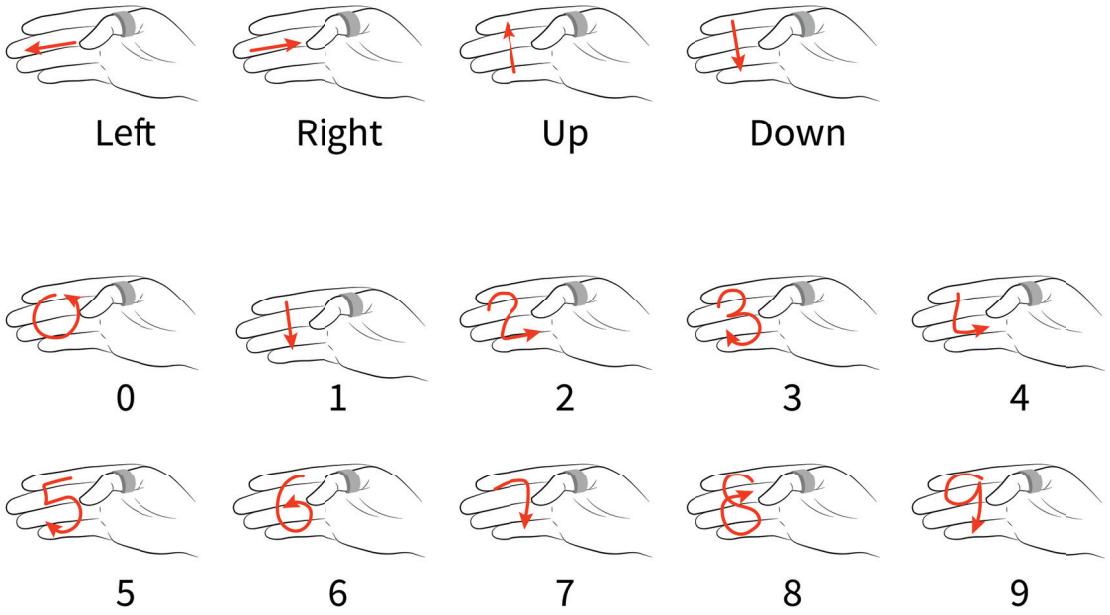

Fig. 1. The top shows the FingerSound prototype and typical placement on the user's thumb. The bottom shows the two unistroke gesture families (directional and digits) we designed, implemented and evaluated.

for streamlining input to wearable and mobile computing devices. The reason for the former is that traditional means of interaction, such as mouse and keyboard, are typically not very well suited for miniaturized mobile and wearable devices. On the other hand, progress in miniaturization and substantially increased sensing and computing capabilities enables innovative and potentially more convenient means of interaction. Furthermore, with the emergence of entirely new device categories and / or application domains effective input means need to be developed that are both convenient for the user and reliable to process automatically.

In the example of mobile virtual reality (VR), users are immersed in a synthetic world where traditional computer interfaces such as the keyboard and mouse may disappear. As such, the demand for novel, effective input modalities is striking. Arguably, text input for short messages may still be necessary in such scenarios for responding to notifications from others, labeling files or objects, or controlling the operating system. The need to input short messages input has led to VR systems that render virtual keyboards and controls over the virtual world where the user can select each letter with head or hand movement. Another option is to render a representation of a physical keyboard in the virtual world so that the user can find it. Both of these options require significant visual and manual attention and can break the sense of immersion in the virtual world. In addition, physical keyboards (and virtual keyboards rendered in a specific location) require the user to move to the interface, which may be awkward or distracting in a virtual world.

In this paper we present FingerSound, a system for unistroke thumb gesture recognition that enables character-based input for wearable computing devices. FingerSound uses a ring with an gyroscope and a contact microphone on the thumb to detect unistroke gestures made against the hand. A user can perform gestures by rubbing/scraping the thumb across fingers. Input can be started virtually at any time and in any position without requiring the

visual attention of the user to select each letter. Similarly, command gestures can be made without requiring the visual attention of the user or causing the user to feel around the physical environment blindly searching for an interface device.

FingerSound may also be useful in certain contexts for wearable computing. Imagine being in a team meeting with a head worn display integrated into the lens of a pair of eyeglasses [18]. Unlike the use of a mobile phone in a meeting, the head-worn display is designed to be subtle and maintain the rapport of the conversation. However, as soon as the user touches the eyeglasses to control them, it draws attention to the wearer and to the use of the system. With FingerSound, the user can place their hand under the table and give commands to the head worn display. Suppose the head worn display shows an incoming phone call. Drawing an *X* with the thumb against the palm sends the call to voice-mail. Similarly, an incoming text message might show options for quick responses that the user selects by drawing a number or letter against his palm. For example, "meet you for dinner¿' might show "K: OK X: Can't make it" and the user selects 'OK' by drawing a *K* against their palm. For situations where a custom and short message needs to be constructed, FingerSound allows all 36 letters and digits to be written by scraping the thumb across the fingers. While FingerSound is limited to writing words character by character, today's auto-completion systems can help speed text input and correct recognition and spelling errors in the future.

In this paper, we demonstrate different sets of easily learned unistroke gestures (e.g., directional controls, the digits $0-9$, and Graffiti characters) as Figure 1 shows, that can be performed by the wearer subtly and without the need to look at the device. The arrow shows the movement of the thumb. Moreover, it does not cause any social awkwardness that may result from using other wearable devices, like Google Glass. In summary, FingerSound presents the following contributions:

- The design of a ring with a built-in contact microphone and a gyroscope sensor, that captures the sound and movement of a thumb drawing gestures along the fingers.
- The demonstration of three sets of unistroke-based thumb gestures.
- A data processing pipeline that uses K-Nearest-Neighbors for classification and Dynamic Time Warping as a temporal distance metric for gesture classification.
- A user study to validate the effectiveness of FingerSound in recognizing three sets of unistroke gestures using only three training samples per gesture.
- A discussion of the opportunities and challenges for real-world deployment.

2 RELATED WORK

Wearable devices are typically much smaller than traditional computing devices. Therefore input on such tiny devices is often challenging, which has been a research theme in the HCI community for years. In this section, we compare FingerSound to other wearable input technology, especially the ones that also use the ring as the form factor.

Many novel wearable input technologies are based on some form of armbands to facilitate user input. Various sensing modalities have been explored to capture signals on arms, such as acoustic signals [6, 15] and Electromyography signals [17]. Even though these armbands provide a rich set of input functions, the user may find it inconvenient and socially awkward to wear these devices during daily activities.

Different from an armband, it is easier to convince a user to wear a wrist-mounted devices, since people have already worn watches for years. Therefore, many projects built wrist-mounted devices for input. These devices were embedded with different sensing modalities to recognize finger or hand gestures, such as force sensing [4], acoustic sensing [16, 22], electrical sensing [26], static electric field sensing [3], proximity sensing [5], camera [12] and motion sensing [11, 25]. The input vocabularies for these are all relatively small, such that text input is not supported. The Twiddler [13] allows the user to input text by wearing a keyboard in hand. However, wearing

a device in hand may be cumbersome in daily activities. Moreover, it has a steeper learning curve, so much so that a user needs to spend over 20 hours to learn how to effectively use the device for input.

Similar to FingerSound, previous projects have already explored building rings for input, as the rings are relatively small and light. A ring can capture the finger movement by applying appropriate sensing modalities. Some rings were built to support finger interaction on surfaces [10, 26] or interaction with other objects [21]. Other rings were designed to capture the finger movement in a 3D space such as uTrack [2, 24] and CyclopsRing [1]. DigitSpace [7] has explored the design space of using thumb for input and find it is possible to input Graffiti letters using the thumb. However only 6 graffiti letters were evaluated in an off-line cross-validation analysis. Subtle finger movements can also be detected with a ring made of printed electrodes as being demonstrated in [20]. The most recent work FingOrbits used the similar sensing modalities but only recognized a smaller set of finger gestures[23].

Though using the thumb for input has already been explored, the number of gestures that has been demonstrated is relatively small, which limits the scope of potential applications. For instance, we are not aware of any work using thumb movments to recognize the whole set of Graffiti letters. FingerSound can recognize up to 42 unistroke gestures.

3 FINGERSOUND

3.1 Technology Description

The motivation for FingerSound was to build a system which would be always available, easy to use, and provide a rich input set. We realized this with a ring form factor designed to be worn on the thumb. The ring as a form factor augments the finger in a way which is non-obstructive with daily life activities and is socially acceptable unlike other wearable input devices [13]. The reason for choosing the thumb as the location of the ring is because of the reachability of the thumb to most parts of the back of the fingers. This allows the user to use the fingers as the gesture canvas. Other existing ring input devices (e.g., [2]), require the user to perform gestures in the air without haptic feedback and clear signals of the start and end of a gesture. We designed FingerSound to allow the user to perform thumb gestures by scraping across the fingers and palm, providing a natural haptic feedback and clear indication of when a gesture begins and ends (which is the contact of the user's thumb with the gesture canvas). Users can use this feedback to guide their thumb to perform the gesture smoothly. Furthermore, by continuously sensing when the thumb makes contact with the hand, we provide an always-available input modality.

To perform a gesture, the users are required to scrape the ring-bearing thumb along the palm or fingers, so as to make a preset pattern. This scraping action of rubbing the thumb against the skin creates a small, but easily perceived sound as well as subtle motion movement of the thumb. Compared with only using the movement to detect a gesture event, using both sound and movement can help recognize gestures from noise. For instance, the body movement (e.g. walking) may look very similar to the thumb gestures on gyroscope. But based on our investigation, the sound caused by these daily activities is quite different from that caused by the rubbing of the thumb while performing gestures. Therefore we pass the sound captured by the contact microphone along with the motion data captured by the gyroscope sensor through multiple filtering mechanisms and then analyze it to determine whether a gesture was performed or whether it was simply noise from other finger-related activity. Given the location of the device on the finger, it is very easy to get input (sounds and motion) which was not produced while performing a gesture pattern but was generated by any kind of touching of the fingers to another surface. This is like the Midas touch problem [8] and our system needs to handle it. We designed a dedicated machine learning pipeline to recognize valid gestures while rejecting noise. The detailed technical implementation is explained in the following sections.

3.2 Hardware Design

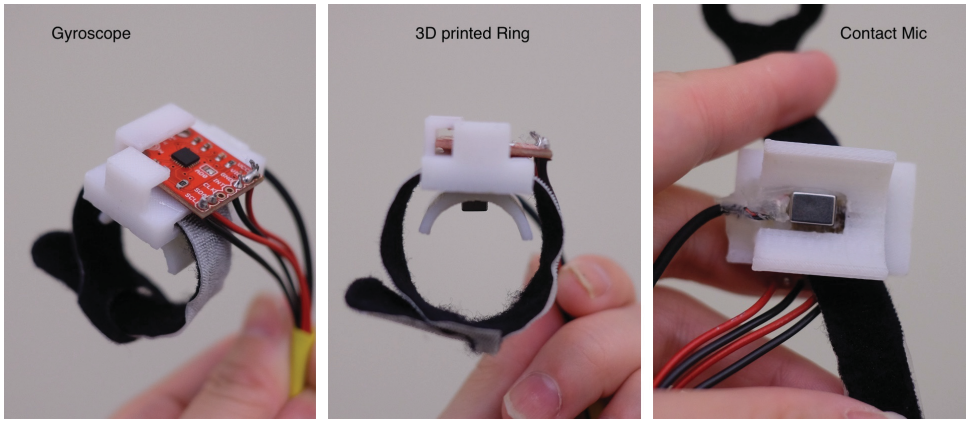

Fig. 2. The ring with a contact microphone and a gyroscope

As mentioned above, we designed a ring with a built-in contact microphone and gyroscope to capture the relevant acoustic and motion data for the purpose of capturing thumb gestures. Compared to a normal airborne microphone, the contact microphone allows the system to maximize the quality of finger scratching sound while minimizing the impact of environmental noise. The contact microphone we used is a Knowles BU-21771, which is 7.92 mm by 5.59 mm by 4.14 mm in size and provides a low noise floor and very flat frequency response, but also a low output voltage. In order to capture the signal effectively, we designed a pre-amplifier board which amplifies the signal by over 100 times, and then filters the audio signal before passing it off board. The design of the preamp is shown in Figure 3. The output of the preamp is passed on to a laptop (a 2013 MacBook Pro) via a USB sound card and sampled at 44, 100Hz.

The other sensing modality we used on the ring is a gyroscope sensor, the InvenSense ITG-3200. We connected this sensor to a Teensy 3.2 microcontroller board, which sends the data to the same laptop via USB.The initial sampling rate available on this gyroscope was around 200 Hz.To achieve a higher sampling rate,we optimized the I^2C[1] communication between the sensor to the Teensy and overclocked the CPU of Teensy at 120MHz. As a result, we are able to sample the gyroscope sensor at about 3, 800Hz. Using the highest sample rate can help provide a baseline for the highest achievable recognition accuracy using the similar hardware set, algorithms and training set.

As figure 2 shows, the ring has two parts: a 3D printed model and a band made of Velcro, allowing us to fit the ring on most users' thumbs without modifying the size of the ring. Another challenge while designing the ring was how to fit the contact microphone well between the 3D printed model and the skin. If the contact microphone does not sit firmly in the ring, much noise can be introduced while the user is performing gestures. To address this issue, we glued the microphone inside the ring while allowing one of its surfaces to protrude out.

3.3 Data processing pipeline

Our data processing pipeline allows the FingerSound system to capture and analyze data in real-time. The hardware components—a microphone and gyroscope—send data to a MacBook Pro laptop over its USB ports separately. A Java program reads both inputs simultaneously and stores them in a readily accessible data structure.

[1]https://en.wikipedia.org/wiki/I%C2%B2C

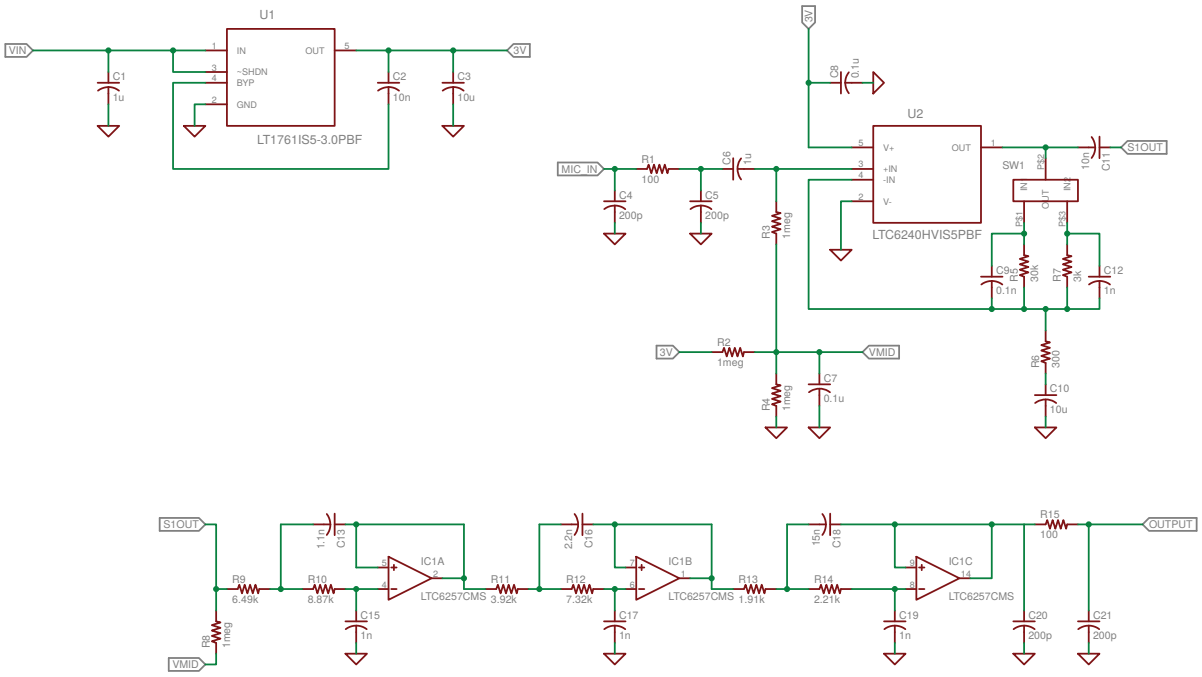

Fig. 3. The amplifier circuit

On another parallel thread, the input sound stream is continuously analyzed to detect an input gesture activity. This is done using an energy-based sliding window segmentation technique as explained in Section 3.4. If this algorithm detects a possible input, it segments that part of sound data and the corresponding gyroscope data and saves it for further processing. While segmenting the gyroscope data, we extend the segment in both directions to acquire some extra gyroscope data as a buffer. We do this to prevent clipping any gyroscope data in the gesture and also to accommodate for any data receiving delays. This segmented sound and gyroscope data is then passed through a support-vector machine (SVM) classifier to detect if the data represents a genuine gesture or noise. The details about the classifier are provided in Section 3.5. If the data is recognized as a gesture by the SVM classifier, then the gyroscope data is sent through a low-pass-filter and finally to our classifier which recognizes the input gesture pattern as described in Section 3.6. Figure 4 highlights the major components of the data processing pipeline.

3.4 Energy-based gesture segmentation

To detect the start and end of a gesture, we analyze the sound produced by the grazing of the thumb on the fingers or palm. Our analysis is based on the short-term energy representation of the microphone signal. This energy is calculated as the square root of the sum of the Euclidean norm of the microphone signal over a short analysis window (frame). This analysis window is 4, 410 samples (0.1s) long and shifted along the raw sensor data.

The resulting energy signal is the basis for detecting onset and offset of relevant thumb gestures. For gesture segmentation, that is determination of start and end points of relevant thumb gestures within the continuous stream of sensor data, we employ a two-stage filtering approach.

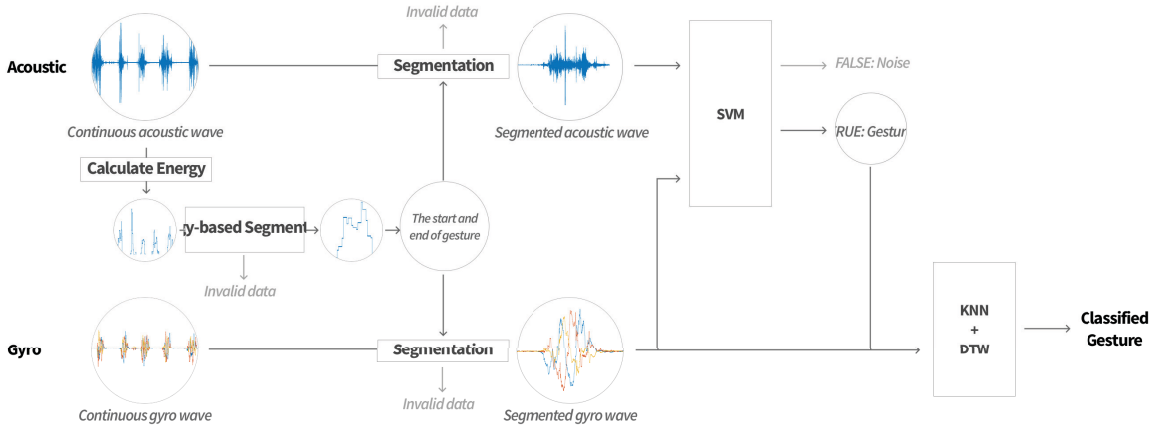

Fig. 4. Data processing pipeline for FingerSound

First, using another sliding window procedure we extract analysis windows that cover two consecutive seconds of audio (energy) data. Through our empirical studies we found out that relevant thumb gestures typically last no longer than two seconds, which determined the window length. In an intermediate processing step we first eliminate windows where the signal energy is below a certain noise threshold thus effectively skipping "silent" phases.

Second, within every extracted two-second window we then search for potential start and end points of thumb gestures. A start point is thereby characterised as being the first one within the two second window that lies on a positive flank, that is when the signal energy increases from 'zero'[2] to positive values. Correspondingly, an end point is determined as the last one within a two second window that is on a negative flank (from positive values to 'zero'). By means of this onset and offset detection procedure we can segment very effectively gesture candidates.

If the length of an extracted gesture candidate exceeds a preset threshold of minimum gesture length, we consider it to be a genuine input. The extracted start and end points of such gestures are then used as index points for segmenting the actual sound (not energy) and gyroscope data from the continuous data stream and to pass this data to the subsequent stage in our processing pipeline, that is feature extraction and classification.

3.5 Feature Extraction & SVM-Based Noise classification

The acoustic energy-based gesture detection scheme is likely to over-segment the underlying signal, that is to produce false positive predictions. The main reason for this is that at this stage of the processing pipeline only relatively general analysis rules have been applied that analyse the acoustic signal rather coarsely. Up to this point no actual classification has been performed.

In the next step we eliminate false positive gesture predictions by applying a binary SVM classifier for every extracted segment. This classifier effectively filters out those portions of data that do not correspond to thumb gestures but rather to noise. Note that the classifier does not operate on raw signals but rather on their feature representation (see below). We use the sequential minimal optimization (SMO) implementation of SVM provided by Weka[3].

[2]In fact, 'zero' values correspond to minimal (epsilon) as filtered through our first stage of the processing.
[3]http://www.cs.waikato.ac.nz/ml/weka/

Both gyroscope and sound data are used to calculate meaningful features. The features we used were introduced in [22] and are as follows. For each axis of gyroscope data, we extract a virtual sensor by calculating the derivative of each axis data.For each axis of the raw sensor and its derived virtual sensor, we extract a set of statistical features including min, max, standard deviation, zero-crossing rate, root-mean-square(RMS), the values of peaks and the differences between the peaks. We also calculate the values of the first and second peaks, the ratio and differences of the energy peaks, and correlation between different axes on raw gyro and derived virtual sensor. For acoustic data, we extracts a set of common features in the frequency domain, including 26 Mel-frequency cepstral coefficients (MFCC) and the lower 30 bins of the Fast Fourier Transform (FFT). We choose these features because it is shown that these frequency range are the most informative ones [22]. Connecting the features extracted from the gyroscope and the acoustic data together, we have feature vector with 154 components, which is used to train a SVM to classify gesture vs. noise.

3.6 Gesture Recognition Algorithm

In the final stage of the processing pipeline every extracted segment that has previously been classified as a gesture is now analyzed by a dedicated recognizer, which classifies the type of gestures. As our system shall serve as input modality for real-world applications, (near) real-time performance is mandatory. This constraint rules out a number of recognition techniques because they are simply to demanding with regards to computational resources.

We employ Dynamic Time Warping based classification, which has been widely used for the analysis of time-series data in general and for gesture recognition in particular [19]. Dynamic Time Warping is essentially an implementation of Dynamic Programming where two temporal patterns are compared using specific edit distances. DTW quantifies the dissimilarity between two sequential input patterns by finding the minimal set of operations –insert, delete, match, substitute– that map one sequence to the other thereby using cost factors for every single operation. Through minimizing the overall edit costs the procedure finds the optimal alignment and quantifies the error. The advantage of DTW based analysis is that it accounts for input patterns of different lengths and is very efficient.

We combine DTW based sequence matching with a standard k-NN classifier (k=3) for classification. Effectively this procedure translates into very effective and efficient template matching. Our template database consists of representative examples of all relevant thumb gestures. The implementation of DTW was provided by Java machine learning library [4].

4 EVALUATION

4.1 Procedure

To demonstrate the capability of recognizing unistroke thumb gestures and evaluate the interaction experience with FingerSound, we conducted a user study with 9 participants with an average age of 26 (3 male) on two sets of unistroke gestures—the digits 0-9 and directional swipes (see Figure 1)—under two settings. All participants were recruited from a university campus. The study was conducted in a lab-based environment. Each user study lasted about one hour. Before the study, two researchers provided about 100 gestures and 100 noise samples as the basic training data for building the SVM noise classifier. At the beginning of the study, one researcher helped the participant to put on the ring and demonstrated how to perform each gesture. The participant was allowed to practice each gesture until she felt comfortable to proceed to the actual test. The actual study consisted of 2 training sessions and 6 testing sessions.

In the first two training sessions, the participants were asked to put hands and arms on the table. Each unistroke gesture was performed 3 times in a random sequence during each session. Visual stimuli on the screen and an

[4] http://java-ml.sourceforge.net/

audio cue were used to remind the participant of the gesture to be performed. The gesture segmentation pipeline is running continuously to detect and segment a gesture instance. If the system failed to detect a gesture, the participant was advised to repeat the gesture until successfully detected. We treated the first session as a practice session, which helped the participants get familiar with the unistroke gesture sets as well as our experimental real-time system. The second session was used as the training data collection session for building machine learning models of gesture segmentation (SVM) and gesture classification (KNN with DTW distance function). In total, 30 (3×10 gestures) and 12 (3×4 gestures) gesture samples were collected as the training data set for unistroke digit gesture and directional swipe gesture, respectively, for each participant. The collected gesture data was combined with pre-collected data from researchers to train the SVM-based noise classifier for each participant.

After the first two sessions, each participant was required to provide 30 test instances per gesture with their hand in two different locations. Within each session, each participant provided 5 instances per gesture in a random sequence. The gesture recognition results were presented to the participants in real-time on the screen. If the classification result matched the stimuli gesture, the background was marked in green. Otherwise, it turned to red. Furthermore, if the participant performed a gesture, but the system failed to detect it or labeled it as noise, the gesture was labeled as a false-negative error.

To investigate whether the user can perform gestures in an eyes-free fashion and in a different hands posture, we divided these 6 test sessions into two groups. In the first 4 testing sessions, the participants placed hands on a table, similar to the training session. In total, 200 samples (5×10 gestures $\times$ 4 sessions) for unistroke digits and 80 samples (5×4 gestures $\times$ 4 sessions) for directional swipe were tested in these 4 testing sessions for each participant. In the last two sessions, the participants were required to hold the hands under the table to perform the gestures. These two sessions were designed to simulate the real-world scenarios where the user would likely perform gestures in an eyes-free fashion with various hand postures. In total, 100 samples (5×10 gestures $\times$ 2 sessions) for unistroke digits and 40 samples (5×4 gestures $\times$ 4 sessions) for directional swipe were tested in the last two sessions.

All real-time recognition results and the raw sensor data were saved for later analysis.

4.2 Results

We report the real-time classification results. The average accuracies for the first four sessions and the last two sessions are 92% and 89% respectively for the 10 unistroke digits. On average, 2.58 false-negative errors were captured in each session. The confusion matrix is presented in figure 5. The most accurate gestures are the digits '1','7','8', and the least accurate digits are '0', '6' and '4'. The '0' and '6' were the most mutually confusing gesture pair, because of their very similar gesture patterns. The only difference is '6' ends a bit lower than '0'. Interestingly, '4' was misclassified with '1', while '1' received the highest precision. To perform a '1', it is easy to find that to draw '4' on the fingers, the participant needs to first drag the finger down first, which is the same as '1', and then turn the thumb to the right.

The average accuracies for the four directional swipes are high in general, 98.19% and 96.94% in the first four sessions and the last two sessions (eyes-free), respectively. Only 'down' and 'left' caused a few confusions when the hands were held below the table. On average, 2.74 false-negative errors were observed in each session.

The current results indicate that the accuracies were slightly lower when participants performed the gestures under the table. There are two factors that influence this accuracy. The first factor is that the decreased accuracies were caused by the lack of visual observation of the hands while performing the gestures in the last two sessions. However, all participants started performing the gestures without looking at their hands once they mastered the gesture, even in the first four sessions based on our observation. Another hypothesis is that the training data was collected while the hands were placed on the table. However, when the hand was held below the table, the posture of hands were different, which may influence how a gesture was performed.

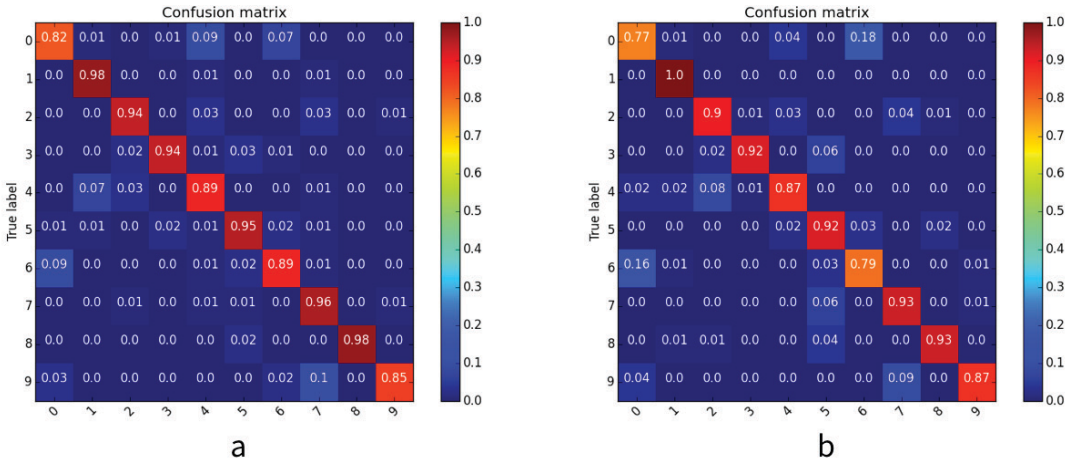

Fig. 5. Confusion matrix for 10 digits: a. first 4 sessions where participants‘ the hands were on the table. b.last 2 sessions where the gestures were performed undertable in an eyes-free fashion

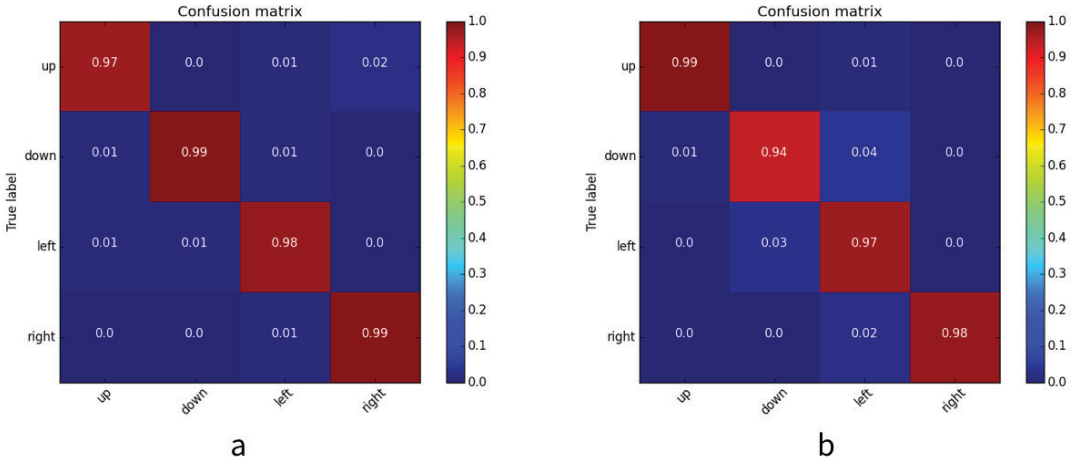

Fig. 6. Confusion matrix for 4 directional slides: a. first 4 sessions where participants‘ the hands were on the table. b.last 2 sessions where the gestures were performed under table in an eyes-free fashion

Figure 7 shows the accuracy for each participant, where P1 and P8 provided the highest accuracies and P6 and the lowest accuracy. The accuracies for most participants decreased in the eye-free evaluation, except for P1. Our observation was that the way P1 perform the gestures was very consistent across all sessions.

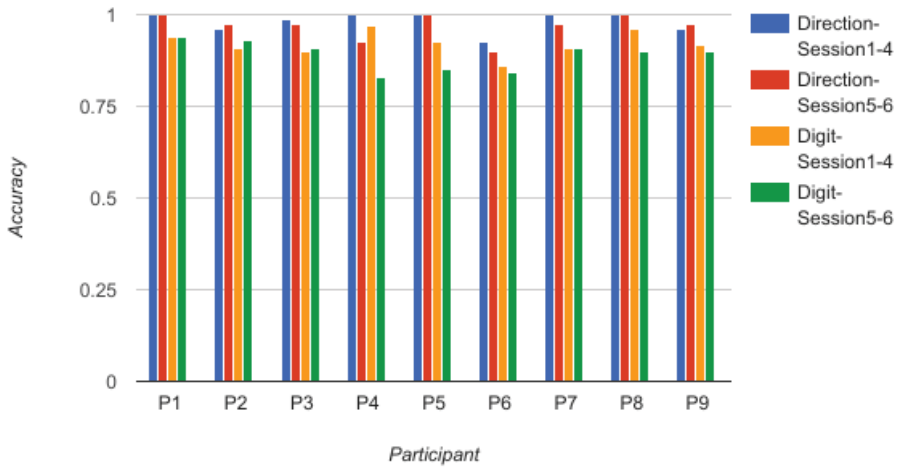

Fig. 7. Accuracy for each participant on unistroke digits and directional swipes

4.3 Input with Graffiti gestures

Fig. 8. Additional 28 unistroke gestures used in the followup study

The results of recognizing 10 digits and 4 directional swipes are encouraging. To further understand the richness of input vocabulary that can be supported by FingerSound, we conducted a follow-up study to recognize a larger set of 28 unistroke gestures including 26 Graffiti-style letters as shown in figure 8. Graffiti is a gesture set that was created by Palm, Inc.[5] to provide text input on PDA. Each Graffiti gesture resembles the uppercase form of the English alphabet, such that it is easy to learn and use. A previous study already has shown that the participants can achieve an accuracy of 97% after five minutes of practice [14]. Showing that our system is able to recognize the Graffiti gestures not only demonstrates the power of our technology to recognize a rich set of unistroke gestures, but also examines the possibility of using this technique as an alternative text input method for short messages in the future.

In this study, we reduced the number of testing sessions (given a large number of samples) but added one more practice session compared with the previous user study, to give the users more time to learn the larger gesture set. Overall, we had 5 sessions in this study. The first two sessions were practice sessions (3 samples per gesture per session), the third session (3 samples per gesture per session) is the training data collection session and the last two sessions (5 samples per gesture per session, hands on table only) were the testing sessions. As before, the real-time classification result was presented to the participant and recorded.

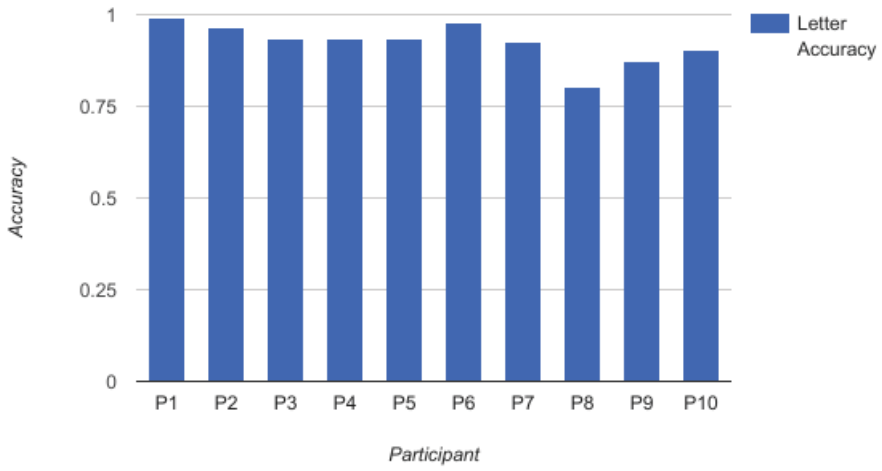

Fig. 9. Accuracy for each participant on Graffiti input

In total, 10 participants (including four researchers, 2 female) with an average age of 27 participated in this study. None of participants were involved in the first study. All session were completed in one hour. The real-time classification results for recognizing the 28 unistroke gestures resulted in an average accuracy of 92.46%. There were 5.9 false-negative errors observed on average for each session.

Figure 10 shows the confusion matrix for this gesture set. The most accurate gestures are the letter ‘X‘ and ‘Z‘ whose precision are 100%. The least accurate gestures are the letter ‘D‘ and ‘P‘ with the precision of 69% and 74%. The reason for confusion between these two letters is visually apparent on Figure 8 as they look very similar.

[5]http://www.palm.com

The only difference between them is where the gesture ends. The “D“ ends at a lower position than where “P“ ends, which appears harder to be distinguishable by intuition compared to other letters.

The accuracy for each participant is presented in Figure 9. P1 and P8 provide the highest accuracy of 98.93% and the lowest accuracy of 80.36%, respectively. Our observation is that P8 did not develop a consistent pattern during the practice and training sessions. As a result, every time a gesture was misclassified, P8 tended to adjust the way that gesture was performed, which lead to more false-positive errors in the end. It indicates that certain users may demand a longer time to master the thumb gestures, or a reinforcement learning methods should be deployed in the future.

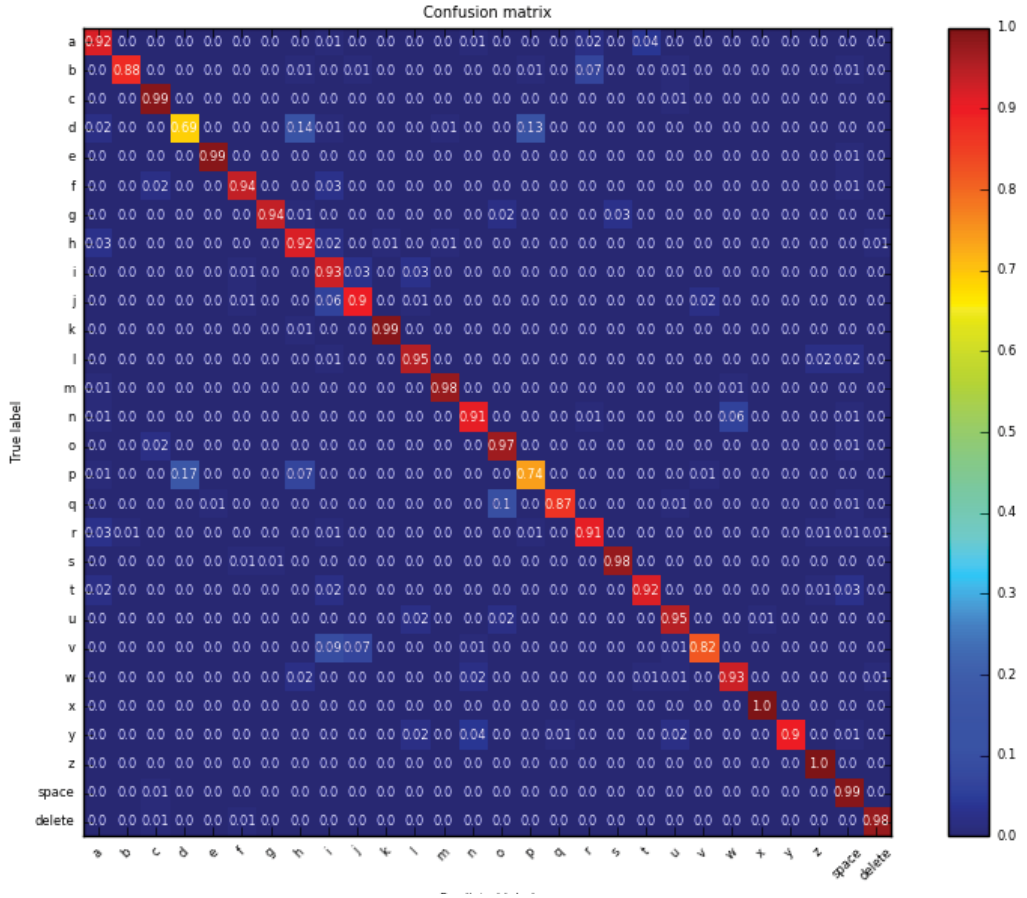

Fig. 10. Confusion matrix for Graffiti Unistroke Gestures

5 DISCUSSION

5.1 Evaluating FingerSound in a noisy environment

In order to understand how FingerSound performs in a noisy environment where both motion and acoustic background noise exist, we conducted a follow-up study with 5 participants (two experienced users, three novice

users, an average age of 31, 1 female). The participants were asked to perform 10 unistroke digits while walking in a noisy environment. There are three sessions: the practice session, training session, and testing session. Only the novice participants were asked to complete the practice session by repeating each gesture three times in a random sequence.

The training session is the same as in the previous studies. The participants provided three training samples for each unistroke digit in the training session while sitting in front of a table. Then one testing session was conducted for each participant, where each gesture was tested 5 times in a random sequence. Differing from the previous studies, to simulate the real world scenarios, the participants were asked to keep walking around a large table (approximately 4 x 3 meters) during the whole testing session. A laptop speaker was placed at the center of the table to play pre-recorded street/crowd Gaussian noise at 80 db to simulate the acoustic noise the user may encounter in daily activities. The ring was connected to a laptop placed on a cart. A researcher moved the cart to follow the participant during each session. The participants were asked to walk at their normal walking speed and to not stop while performing the gestures. Similar to the previous studies, both audio and visual stimuli, as well as the real-time classification results were presented on the laptop. A researcher observed the study and recorded the false-negative/positive errors. The average accuracy of classifying 10 unistroke digits across all participants was 92.8%, which is similar to our previous studies. The lowest accuracy was 90%, which was a novice user. We did not observe any false positive error on detecting an gesture event in this study. However, the average number of false-negative error across all participants increased to 5 in the testing session. We attributed this to the lack of data collected from noisy environment in the final training data set for the gesture/noise binary classifier (the SVM). Collecting training data in noisy environment can potentially increase the recognition accuracy and reduce false-negative errors. However, in order to present a baseline to help the readers to understand the generalizability of the our proposed system as well as compare the current results directly to the results from the previous study, we conducted this additional user study in a more challenging setting, where the training data was collected in a controlled setting (no noise) but tested in a noisy environment.

5.2 Lowering the sampling rate of gyroscope

We sampled the gyroscope at the highest rate (3800 Hz) supported by the current hardware set. Higher sampling rate does not hurt the classification accuracy but can cause higher energy consumption [9] and more response time. To investigate the influence of sampling rate on accuracy, we down-sampled the data collected from the study to 100Hz and reprocessed the data with the same classification pipeline as used in the study. Surprisingly, the accuracy was comparable with the previous study, which were 96.85%, 89.96%, and 93.21% respectively for recognizing 4 directional swipes, 10 unistroke digits and 28 Graffiti unistroke gestures. The comparable accuracy obtained at lower frequency implies that overclocking was unnecessary. This indicates it is possible to reproduce the performance with a much lower sampling rate in the future, which requires low processing power and energy consumption.

5.3 Building User Independent models

In the user study, each participant was asked to provide three training samples before testing the gestures. In real-world scenarios, from the prospective of providing a better user experience, it is ideal to use the system without the need for providing calibration or training data first. Therefore, we performed an user-independent analysis on the data collected from the user study. The training models were built using the training data provided by other participants (3 instance per gesture per participant). We ran our classification pipeline for each participant for each gesture set. The average accuracy across all participants of recognizing 4 directional swipes was 87%, but were under 70% for unistroke digits and Graffiti unistrokes gestures. This indicates the more gestures in the recognition system, the more training is required. However, we do notice a huge variance on the accuracy for participants on 4 directional swipes gestures, where the accuracy of four participants was over 96% while

the accuracy of two participant was around 65%. For the two outliers, the confusion matrix shows that certain gestures were completely misclassified. This suggests with a small set of gestures (E.g., 4 directional swipes), it is possible for some users to use the system without collecting any training data. However, some users may need to provide calibration gestures for the gestures that can not be recognized for them. This study was only conducted in a relatively small set of training data. Further investigation with a much larger set of training data is needed to derive certain conclusions.

5.4 Redesign of unistroke gestures for thumb input

In the study, we evaluated FingerSound using three common unistroke gesture sets. However, these gesture sets were specifically designed for input with a stylus, which may be inappropriate for thumb-based gestures. The thumb has relatively limited freedom of movement compared with using a stylus for input. Gestures that can be easily distinguished when written by stylus, may raise confusion when performed by thumb, because our system is not directly measuring the movement trajectory of the thumb, but rather capturing the rotation motion on the ring. Some gestures that are visually distinguishable may be challenging to classify, such as “D“ and “P“, “V“ and “J“. However, we also noticed that some other gestures that are visually similar can be easily recognized by our system. such as “U“ and “V“ or the four directional swipes. These gestures all have different starting positions. Our observation is that our system can easily distinguish the different starting positions between gestures. In addition, participants reported that some gestures are uncomfortable to be performed with their thumbs. For instance, “Y“ requires the user to finish the gesture on the palm by moving the thumb towards the top-right direction, which was challenging because of the limited freedom of movement with the thumb in that area.

Although our system can recognize unistroke gestures with over 92% accuracy, only a few gestures caused most of the classification errors such as “D“ and “P“. Those gestures would be much easier to be recognized if the design of these gestures was revised to accommodate thumb gestures. Therefore, we summarize our observation of the study as guidelines for future thumb gesture design practice:

(1) Visually similar gestures can be distinguished if the direction or the starting point is different.
(2) Avoid designing gestures whose trajectory is similar and only differ based on length.
(3) Avoid designing gestures that move the thumb towards the edge of the hand, as is ergonomically uncomfortable.

5.5 Hardware limitation and improvement

One limitation of the current system design is that the ring is only serving as a sensing unit and is connecting to a laptop where the recognition algorithm runs. It cannot run the whole algorithm exclusively. However, we expect this limitation can be addressed in one of two ways in the future. One is as technology advances, the processor and battery can be made smaller than it is today. Therefore, all the hardware can be potentially fit into the ring itself, though the limits of Moore‘s Law make this less likely. The other is to add a wireless transmission unit (e.g., Bluetooth) on the ring. Then, the data processing and recognition can be conducted on another portable device, such as a smartphone. We expect the second method to be more pervasive as it fits into the current trend of interconnected smart devices that we are observing in the industry.

Sensor drift is another issue that needs to be addressed before it can be deployed widely in real-world applications. In the current system, we calibrate the sensor once before each participant starts the system. The ring was placed on the table for 2.5 seconds and the program recorded the accumulated offsets, which was used to calibrate all the sensor values afterward. We did not observe any significant influence on the recognition system caused by sensor drift during the one-hour user study for each participant. However, if the system is deployed for a longer time, the sensor may keep drifting, influencing the performance of the gesture recognition system. A popular solution to this problem is to use other sensors in the IMU sensor sets to compensate for the drift.

5.6 Improving the speed of recognition

To classify a new gesture instance, our current implementation computes the DTW distances between the unclassified instance with every instance in the training sets. As the number of gestures increases in the training set, the system response time also increases, which can influence the interaction experience. Since computing the DTW distances is the most time-consuming computation, reducing the numbers of DTW distance calculations would reduce the response time. Therefore, instead of computing the DTW distance with every training sample, the real-time classification system should only compute the distances with a few pre-selected template samples from each gesture set. These representative samples can be selected based on their DTW distances with other training samples for the same gesture. Usually, the samples that have least DTW distances to others are chosen as the templates. For instance, if we choose 1 sample per gesture as the template, the response time would reduce by two-thirds compared with our current implementation. Furthermore, the real-time response time would not change even if the size of training samples increases. A lower sample rate can also potentially reduce the time required for DTW calculation, which can also improve the speed of recognition, as we have discussed the previous section.

5.7 Customized thumb gestures

Though we only demonstrate the recognition of three unistroke gesture sets (of size 4, 10 and 28 gestures), the system has the potential to recognize a wider range of thumb gestures, including the ones designed by the users themselves. These customized gestures can be mapped to different functions by the user, such as an unlocking gesture for other connected devices.

5.8 Activation gestures

The current system was only evaluated in a lab environment, where zero false-positive were observed in detecting gesture events during all sessions. However, more false-positive errors may occur if the users are involved in other activities that have heavy body movements. This is a hard challenge that every gesture recognition system must address before practical deployment. Our solution is to design an activation gesture to start the whole system. That means, only when an activation gesture is detected, the system starts recognizing the full set of gestures. The activation gesture can be chosen from the ones that have the least confusion with other gestures. For example, we could choose “X“ as the activation gesture of our system because it has 100% precision and 99% recall as shown in Figure 10.

5.9 Applications

However, FingerSound is not designed for input on all tasks. It may not be appropriate for writing long text because input with FingerSound is less efficient in terms of speed and accuracy than using traditional keyboards. Some participants reported it was physically demanding to use the thumb to input for a long time (one hour in the study). In addition, it may not be desirable for applications that require extremely fast responses, such as first person point-of-view shooting or car racing games.

Given its current design and performance, FingerSound is more appropriate to be used to input on other devices or applications that demand short responses. For instance, home entertainment systems use a remote control for input. However, to input text, the user has to use cumbersome input techniques, such as directional buttons navigating a QWERTY keyboard on the screen. By using FingerSound, text input can be completed by simply scratching the thumb on the hands.

FingerSound can also be used as an alternative input device for smart watches. Because the watch screen is relatively small, input of short text or digits is very challenging. The fingers may occlude the content. FingerSound

would allow the smartwatch users to input on their watches using the same hand or a different hand without occluding the screen.

Similar as other gesture-based input technique, FingerSound does not provide perfect recognition accuracy, which means the user may encounter recognition errors while using the system. To provide the optimal user experience, these errors need to be compensated. One obvious way is to further improve the system accuracy, which we will discuss in the next section. The other equally important solution is to design the interactions in the context of applications appropriately. For instance, the designer should consider use the most accurate and efficient gesture to access the "delete" function, which help correct the errors.

5.10 Limitations and Future Work

5.10.1 Improving the accuracy and efficiency. The system was only tested in a lab-based environment. However, it may be challenging to achieve similar recognition performance using only three training samples while the system is deployed in the wild. One solution is to collect more training data in a noisy environment. The other is to adopt more advanced algorithm such as Hidden-Markov Model to improve the recognition efficiency and accuracy. Furthermore, the current training data for noise was collected in a lab environment, which may not best represent the noise in the daily activities. Collecting noise data during daily activities can potentially help improve the performance. For instance, our current noise samples were collected in a lab-based environment. Although it worked in the studies, we plan to collect the noise data in daily activities of under a naturalistic environment in the future.

Another part of future work we plan to conduct is to improve the input accuracy and efficiency. As we have discussed, applying more advanced machine learning technique(E.g., HMM) can potentially improve the accuracy given a larger training data set. Furthermore, adopting an auto-completion system would further improve the input efficiency. We plan to further investigate these issues in the future.

5.10.2 Others. We also plan to replace the gyroscope sensor with a 9-degree IMU sensor. Based on the new sensor, a sensor drifting compensation algorithm can be implemented to increase the stability of the sensor data.

Another issue in the current system is that the ring is tethered with cables, which may limits its freedom of movement and pull down the recognition accuracy. The orientation of the ring can also influence the performance of the system. To address these issues, we plan to make the whole system portable by adding a wireless communication module in the future, such that we can be consistent on the orientation of the ring and deploy the system in the wild.

6 CONCLUSION

FingerSound is an input technology that recognizes unistroke thumb gestures using a ring, consisting of a contact microphone and a gyroscope sensor. A KNN model with the distance function of DTW is implemented to recognize gestures using only three training samples for each gesture. A user study with 19 participants shows that FingerSound was able to recognize 4 directional swipes, 10 unistroke digits, and 28 Graffiti letters with an average accuracy of 92%, 98.19% and 92.46%, respectively. We discussed the potential applications, the opportunities, and the challenges need to be addressed before it can be deployed in real-world applications.

7 ACKNOWLEDGMENTS

We appreciate the reviewers for their detailed feedback and the participants involved in the user study.

REFERENCES

[1] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. 2015. CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &*

Technology (UIST '15). ACM, New York, NY, USA, 549–556. https://doi.org/10.1145/2807442.2807450

[2] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. 2013. uTrack: 3D Input Using Two Magnetic Sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 237–244. https://doi.org/10.1145/2501988.2502035

[3] Gabe Cohn, Sidhant Gupta, Tien-Jui Lee, Dan Morris, Joshua R Smith, Matthew S Reynolds, Desney S Tan, and Shwetak N Patel. 2012. An ultra-low-power human body motion sensor using static electric field sensing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 99–102.

[4] Artem Dementyev and Joseph A. Paradiso. 2014. WristFlex: Low-power Gesture Input with Wrist-worn Pressure Sensors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. Association for Computing Machinery (ACM). https://doi.org/10.1145/2642918.2647396

[5] Jun Gong, Xing-Dong Yang, and Pourang Irani. 2016. WristWhirl: One-handed Continuous Smartwatch Input Using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 861–872. https://doi.org/10.1145/2984511.2984563

[6] Chris Harrison, Desney Tan, and Dan Morris. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 453–462.

[7] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. 2016. DigitSpace: Designing Thumb-to-Fingers Touch Interfaces for One-Handed and Eyes-Free Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1526–1537. https://doi.org/10.1145/2858036.2858483

[8] Howell Istance, Richard Bates, Aulikki Hyrskykari, and Stephen Vickers. 2008. Snap Clutch, a Moded Approach to Solving the Midas Touch Problem. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications (ETRA '08)*. ACM, New York, NY, USA, 221–228. https://doi.org/10.1145/1344471.1344523

[9] Aftab Khan, Nils Hammerla, Sebastian Mellor, and Thomas Plötz. 2016. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters* 73 (2016), 33–40.

[10] Wolf Kienzle and Ken Hinckley. 2014. LightRing: Always-available 2D Input on Any Surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 157–160. https://doi.org/10.1145/2642918.2647376

[11] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 321–333. https://doi.org/10.1145/2984511.2984582

[12] Christian Loclair, Sean Gustafson, and Patrick Baudisch. 2010. PinchWatch: a wearable device for one-handed microinteractions. In *Proc. MobileHCI*, Vol. 10.

[13] Kent Lyons, Thad Starner, Daniel Plaisted, James Fusia, Amanda Lyons, Aaron Drew, and E. W. Looney. 2004. Twiddler typing. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*. Association for Computing Machinery (ACM). https://doi.org/10.1145/985692.985777

[14] I Scott MacKenzie and Shawn X Zhang. 1997. The immediate usability of Graffiti. In *Graphics Interface*, Vol. 97. 129–137.

[15] Adiyan Mujibiya, Xiang Cao, Desney S. Tan, Dan Morris, Shwetak N. Patel, and Jun Rekimoto. 2013. The Sound of Touch: On-body Touch and Gesture Sensing Based on Transdermal Ultrasound Propagation. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS '13)*. ACM, New York, NY, USA, 189–198. https://doi.org/10.1145/2512349.2512821

[16] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1515–1525. https://doi.org/10.1145/2858036.2858580

[17] T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A. Landay. 2009. Enabling Always-available Input with Muscle-computer Interfaces. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 167–176. https://doi.org/10.1145/1622176.1622208

[18] Thad Starner, Steve Mann, Bradley Rhodes, Jeffrey Levine, Jennifer Healey, Dana Kirsch, Rosalind W Picard, and Alex Pentland. 1997. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments* 6, 4 (1997), 386–398.

[19] Gineke A ten Holt, Marcel JT Reinders, and EA Hendriks. 2007. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, Vol. 300.

[20] Hsin-Ruey Tsai, Min-Chieh Hsiu, Jui-Chun Hsiao, Lee-Ting Huang, Mike Chen, and Yi-Ping Hung. 2016. TouchRing: Subtle and Always-available Input Using a Multi-touch Ring. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '16)*. ACM, New York, NY, USA, 891–898. https://doi.org/10.1145/2957265.2961860

[21] Sang Ho Yoon, Yunbo Zhang, Ke Huo, and Karthik Ramani. 2016. TRing: Instant and Customizable Interactions with Objects Using an Embedded Magnet and a Finger-Worn Device. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 169–181. https://doi.org/10.1145/2984511.2984529

[22] Cheng Zhang, AbdelKareem Bedri, Gabriel Reyes, Bailey Bercik, Omer T. Inan, Thad E. Starner, and Gregory D. Abowd. 2016. TapSkin: Recognizing On-Skin Input for Smartwatches. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces (ISS '16)*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/2992154.2992187

[23] Cheng Zhang, Xiaoxuan Wang, Anandghan Waghmare, Sumeet Jain, Thomas Ploetz, Omer Inan, Thad E. Starner, and Gregory D. Abowd. 2017. FingOrbits: Interaction with Wearables Using Synchronized Thumb Movements. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers (ISWC '17)*. ACM, New York, NY, USA.

[24] Cheng Zhang, Qiuyue Xue, Anandghan Waghmare, Sumeet Jain, Yiming Pu, Sinan Hersek, Kent Lyons, Kenneth A. Cunefare, Omer T. Inan, and Gregory D. Abowd. 2017. SoundTrak: Continuous 3D Tracking of a Finger Using Active Acoustics. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 30 (June 2017), 25 pages. https://doi.org/10.1145/3090095

[25] Cheng Zhang, Junrui Yang, Caleb Southern, Thad E. Starner, and Gregory D. Abowd. 2016. WatchOut: Extending Interactions on a Smartwatch with Inertial Sensing. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. ACM, New York, NY, USA, 136–143. https://doi.org/10.1145/2971763.2971775

[26] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. 2016. SkinTrack: Using the Body As an Electrical Waveguide for Continuous Finger Tracking on the Skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1491–1503. https://doi.org/10.1145/2858036.2858082

Received February 2017; accepted June 2017