ArmIn: Explore the Feasibility of Designing a Text-entry Application Using EMG Signals

Qiang Yang, Yongpan Zou*, Meng Zhao, Jiawei Lin, Kaishun Wu
yangqiang2016@email.szu.edu.cn,{yongpan,wu}@szu.edu.cn
{zhaomeng.szu.edu,lin364884292}@gmail.com
College of Computer Science and Software engineering, Shenzhen University

ABSTRACT

EMG is becoming an emerging interface for human-computer interface and has been applied to gesture recognition in previous work. However, those existing EMG-based interfaces can only recognize gestures at a coarse-grained level such as hand and arm gestures, which constraints their usage in applications involving fine-grained activities such as text entry via keystrokes. As a result, in this paper, we attempt to push the limit of existing EMG-based interfaces and propose the first wearable text-entry system, named ArmIn, with EMG signals. ArmIn is designed to recognize keystroke gestures with the help of a finger on printed and physical keyboards. We implement ArmIn using commodity EMG sensors and custom hardware board, and conduct experiments to evaluate its performance. By carefully designing the data processing scheme, ArmIn can recognize keystrokes on both kinds of keyboard, with 89.5% and 87.5% accuracy respectively, when it is worn on a user's left arm.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**; *Interaction techniques*; *Ubiquitous and mobile devices*;

KEYWORDS

EMG, HCI, Text-entry

ACM Reference Format:
Qiang Yang, Yongpan Zou*, Meng Zhao, Jiawei Lin, Kaishun Wu. 2018. ArmIn: Explore the Feasibility of Designing a Text-entry Application Using EMG Signals. In *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '18), November 5–7, 2018, New York, NY, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3286978.3287030

1 INTRODUCTION

EMG, short for electromyography, is a kind of electrodiagnostic technique for evaluating and recording electric activities produced by skeletal muscles. By analyzing myoelectric signals induced by muscle activities, one can rely on EMG to assess the states of muscles, analyze gait and postures, prevent risks, assist sports training and so on. However, EMG-embedded devices were previously regarded as specialized equipment and were only adopted in certain areas such as physiology and ergonomics for academic purposes. In the most recent years, with the rapid development of hardware technologies, EMG sensors have been gradually designed to be fashionable devices [19] and become the potential candidates to be embedded in wearable devices such as smartwatches and smartbands. This emerging trend is motivated by the vision to provide real-time physiological states monitoring service and to construct personalized health-care system for users. Hence, we anticipate the pervasiveness of EMG sensors in an expanded manner in the near future.

Besides supporting conventional features, EMG has actually attracted tremendous attention in the case of developing some sophisticated novel applications. Among them, EMG-based human-computer interaction (HCI) is an interesting topic which explores the feasibility of designing interfaces with EMG sensors [7, 16, 20, 23, 25]. Previous works in this scope mainly focused on gesture recognition to convey simple commands and interact with other devices. However, these works can only recognize several predefined hand gestures at a coarse-grained level, which consequently cannot be applied to those applications that require recognizing fine-grained gestures. For example, text entry via keystrokes requires dealing with the challenges related to fine-grained finger gestures recognition. For such kind of HCI applications, existing EMG-based methods are obviously inapplicable. Motivated by this, we have a question: *is it possible that utilizing EMG sensors to recognize fine-grained gestures like keystrokes?*

In this paper, we make an attempt to respond to this question by proposing a text-entry interface, namely ArmIn, in the form of armbands using low-cost EMG sensors. The key insight of developing such an EMG-based text-entry system is that EMG sensors are sensitive to corresponding muscle activities induced by keystrokes. Although tapping a keyboard only involves a finger, it produces obvious responses for the corresponding muscles. At a high level, ArmIn is designed to act as an independent middleware between a user and any device to interact with. We claim that this is a new interface for extending text-entry interactions of mobile devices especially for wearable devices, as it is usually difficult to enter texts on wearable devices due to their size restriction.

However, it is non-trivial to realize ArmIn due to the following challenges. First, there are several kinds of noises polluting EMG signals, including power line interference (PLI), baseline wandering (BW) and Gaussian white noise (WGN). This means that it is hard to extract pure signals caused by keystrokes. Second, electrodes are

*Yongpan Zou is the corresponding author.

MobiQuitous '18, November 5–7, 2018, New York, NY, USA

ACM ISBN 978-1-4503-6093-7/18/11.
https://doi.org/10.1145/3286978.3287030

attached at different positions of muscles, EMG signals cannot be captured simultaneously in multi channels. This asynchrony infers that traditional event detection methods are failed in ArmIn as we cannot segment signals of five channels at the same time simply according to a single one. Finally, previous works in this scope did not reveal which learning model is effective for EMG signals, so it is a challenge to choose suitable features, models and parameters to achieve a good performance of ArmIn.

To deal with challenges of EMG-based keystroke recognition, we carefully design a data processing pipeline first to remove background noise and interference. Different filters are utilized to eliminate these noises according to their characteristics. For the second challenge, we define a special feature to enhance the valid signals and integrate five channels. Following that, an adaptive threshold is utilized to encode signals and an onset and offset detection algorithm can be performed effectively. As for the last challenge, we make an effort to conduct a feature engineering, train many learning models and tune their parameters carefully in order to select the optimal feature set, learning model and parameters.

We implement a prototype of ArmIn using commercial electrodes and custom hardware board, and conduct experiments to evaluate its performance on physical and printed keyboards. The experimental results show that the accuracy of keystroke recognition can be up to 89.5% and 89.4% for the printed and physical keyboard, respectively. In the remaining parts of this paper, we shall demonstrate the detailed description of design, evaluation methodology of ArmIn.

2 RELATED WORKS

2.1 EMG-based applications

As a medical clinical method, EMG-based method is usually utilized to investigate the fatigue patterns and the diseases of muscle [11]. In recent years, researchers have developed EMG-based applications in Human-Computer Interaction (HCI) paradigms. Saponas et al. [16] designed an EMG-based system to classify four-finger gestures including pinching, holding a travel mug, and carrying a weighted bag. Zhang et al. [25] built a framework accompanied with 3-axis accelerometer and multi-channel EMG sensors in order to classify 72 Chinese sign language (CSL) words using decision tree and hidden Markov models (HMMs), and created a prototype of an interactive system with gesture-based control unit. However, acceleration signals play a much more significant role in this method than EMG. Christoph et al. [4] utilized high-density EMG sensor array with 192 electrodes to detect and recognize subtle finger gestures. In total, 27 gestures were defined and then recognized using Bayes classifier with 90% accuracy. Meanwhile, the authors in this work investigated the effect and benefit of various electrodes and presented two methods to solve the error, caused by position shift. But this technique requires plenty of electrodes. Consequently, Donny et al. [7] proposed a forearm-based EMG system that can recognize fine-grained thumb gestures, including left swipes, right swipes, taps, long presses and user-defined complex motions with high accuracy.

2.2 Techniques for text input

It is known that limited input space of mobile devices limits user experience. Although touch screen is available nowadays, typing errors occur frequently when user types on a small keyboard. Some commercial and common device and sensors are used for this purpose, which are introduced as follows:

Based on acoustic sensors: The acoustic hardware, speakers and microphones, are used as transmitter and receiver in the studies. Liu et al. [10] explore the application of keystroke snooping. With time distance of arrival (TDoA) measurement, they can identify some features keystrokes and can recover 94% keystrokes. On the other hand, LLAP [22] performs device-free tracking of a hand or a finger. It adopts the phase shift concept to obtain fine-grained movement direction, and tracks the gestures based on measured distance. It can be used to recognize characters and short words with 92.3% and 91.2% accuracy, respectively.

Based on IMU: PhonePoint Pen [1] utilize the IMU sensors (accelerometer, gyroscope) embedded in a smartphone to write in the air for text entry. The acceleration due to hand gestures can be translated into geometric strokes, and recognized as characters with 83% accuracy. Similarly, Airwriting [3] is an input system that enables complex hands-free interaction ability through recognizing 3D handwriting. Users can write texts in the air as if they use an imaginary blackboard.

Based on WiFi: According to Channel State Information (CSI) corresponding to a certain keystroke, WiKey [2] creates a unique pattern to detect and classify keystrokes, that can recognize keystrokes in a continuously typed sentence with 93.5% accuracy. WiDraw [18] harnesses the Angle-of-Arrival values at the mobile device to calibrate and then utilizes it to track hand trajectory. It can track with a median error lower than 5 cm and achieves 91% mean word recognition accuracy.

Based on camera: CamK [24] uses front-facing camera of mobile phone to build an input space with a papery keyboard, to capture the motion of keystrokes and to localize the keystroke with fingertips tracking. The accuracy of this method can reach above 95%.

Others: Some studies explore the other new methods for text input with other sensors or hybrid sensors, e.g., TypingRing [12]. TypingRing is a wearable ring platform that enables a user to type as if there is an invisible standard keyboard underneath his hand. By using the embedded sensors, including accelerometer, proximity sensors and displace sensor, TypingRing discriminate what key is pressed and then sends the key event to a remote computer.

Compared with existing works, we argue that ArmIn is a new EMG-based application and opens opportunities to add novel features to EMG-embedded armbands.

3 SYSTEM DESIGN

ArmIn is composed of a hardware platform and a data processing pipeline. The architecture of ArmIn is shown as Fig.1. Once EMG signals are recorded by sensors, they are fed into a data processing procedure. The first functional block in the pipeline is for the signal denoising. In this step, background noise and random interference contained in raw EMG signals are filtered out according to their characteristics. Next, onset and offset of a keystroke activity should

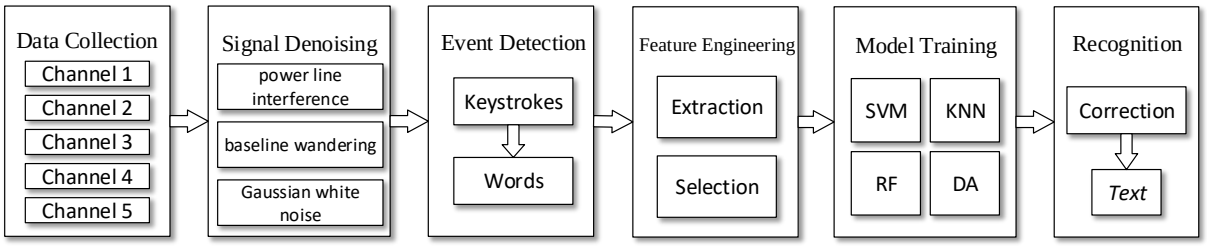


Figure 1: System architecture of ArmIn.

be detected. We define a special feature, use it to encode the signal series, and then the signal segment corresponding to each keystroke can be extracted. Using the natural short pause between the word-typing, we can further detect the start point and end point of a word. Following that, in order to recognize different keystrokes, we perform feature extraction and selection on each signal segment. By collecting certain number of samples, we carefully train learning models including support vector machine (SVM), random forest (RF), K-nearest neighbor (KNN) and discriminate analysis (DA). By comparing their performances, we select an optimal model for the implementation of ArmIn. Finally, a Bayesian algorithm based word correction method is performed to correct inputting typos or wrong output keystrokes so as to improve the text-entry performance. In the following, we shall conduct a detailed description of the aforementioned steps.

3.1 Positions of electrodes

Our system works on armbands and detects the activities of the muscles of left and right forearms. Without loss of generality, in this paper, we only focus on the muscles of the left one. Fig. 2 shows the structure of left forearm and indicates relevant muscle, including extensors and flexors of the fingers, which are sensitive to the typing actions. We attach 5 pairs of electrodes for 5 channels on these 5 positions to capture the fluctuation of the EMG signal when a user types. To avoid the crosstalk, the distance between two electrodes is set to at least 2 cm. Moreover, one more pair of electrodes is added as the reference to eliminate common-mode signal introduced by environment, and it is attached on the elbow (not shown in this figure).

3.2 Signal preprocessing

Raw signals are collected with a 500 Hz sampling rate so that the valid range of frequency is in the $[0, 250]$ Hz because of Nyquist Law. During transmission, the interference including power line interference (PLI), baseline wandering (BW) and Gaussian white noise (WGN) pollute the EMG signal as shown in Fig.3. To eliminate these noises, ensemble empirical mode decomposition (EEMD)[26] is utilized to filter signals first. Zhang et al.[7] verify that EEMD-based methods perform better than conventional filters and result in a high signal-to-noise ratio (SNR). With EEMD, the intrinsic mode functions (IMFs) is decomposed as follows:

$$S(t) = \sum_{k}^{K} IMF_k + \overline{r(t)} \tag{1}$$

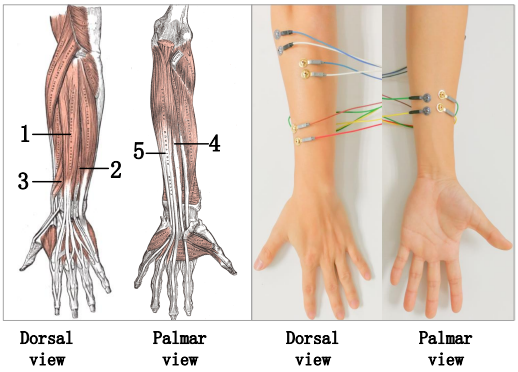


Figure 2: Positions of the relevant muscles: drosal view, the left part, shows the extensor muscles, that includes extensor digitorum (1), extensor digiti minimi (2) and abductor pollicis longus (3); and palmar view, the right one, shows the flexor muscles, that includes parts of flexor digitorum profundus (4/5).

where $S(t)$ denotes raw EMG sigals, and IMF_k represents kth IMF. The sum of noises is defined as $\overline{r(t)}$. The noise is eliminated in every IMF, and then we integrate IMFs together to recover the clean EMG signal $\overline{S(t)}$:

$$\overline{S(t)} = \sum_{k}^{K} \overline{IMF_k} \tag{2}$$

where $\overline{IMF_k}$ refers to the filtered IMF.

Power line interference: The PLI is produced by alternating current (AC) in an electric power grid and its frequency, shown in Fig. 3, is 50 Hz. Moreover, PLI produces harmonic components in different frequency domains, including 100 Hz, 150 Hz and 200 Hz. All these interferences in the received signal are eliminated with an elliptic filter-based 3-order notch filter with the stop band set to $[49, 51]$ Hz/$[99, 101]$ Hz/$[149, 151]$ Hz/$[199, 201]$ Hz, which can remove the noise efficiently as well as have a slight influence on the signal in the pass band. Fig.4 shows its frequency response.

Baseline wandering: For the movement of electrodes, skin and cable, the signal is distorted with low frequency but high amplitude signal. According to Stegeman and Hermens et al. [15–17], the frequency components of muscle activities usually fall into $[15, 250]$ Hz range. In this consequence, a 3-order bandpass Butterworth filter is utilized to remove the noise efficiently.

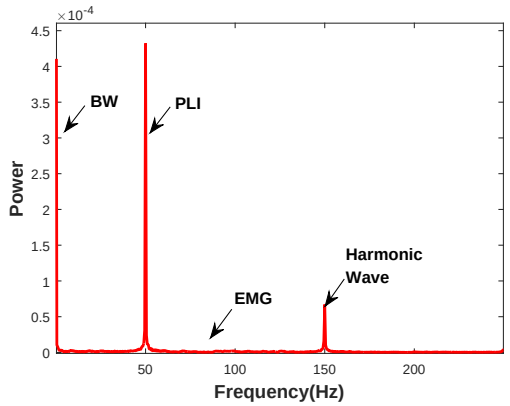


Figure 3: Spectrum of raw EMG signals.

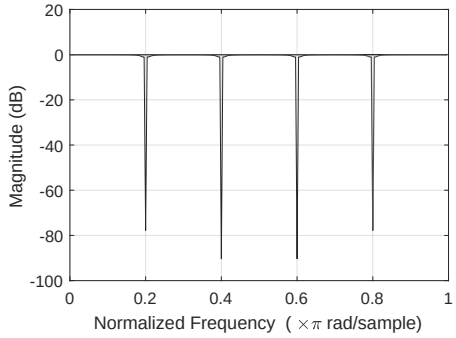


Figure 4: Frequency response of Notch filter. The maximal frequency is 250Hz.

Gaussian White Noise: A similar approach to Zhang et al. [26], the soft threshold wavelet-based denoising method[6] can be used to eliminate the Gaussian White Noise.

3.3 Onset and offset detection of keystrokes

Next, segmentation is required to detect the onset and offset of a valid stroke motion in the signal sequence. A straight-forward solution for signal segmentation is using a threshold to detect every channel whether the amplitude is higher than it or not. However, due to the asynchrony of five channels mentioned in 1, EMG signals do not start and end simultaneously. Moreover, EMG signals are very sensitive to typing strength, so a fixed threshold can not be used to check the onset and offset directly. As a result, we must find a new method to deal with this problem.

We introduce two features first, named Root Mean Square (RMS) and Sample Entropy (SE). RMS represents the energy in a signal window, which can be calculated as:

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} S^2(i)}{n}} \tag{3}$$

where $S(i), i = 1, 2, .., n$ denotes a window of EMG signal, where n is the number of samples. SE is a feature that indicates the complexity or instability of a signal. Readers are suggested to refer [14] for more details. Fig.5 shows SE and RMS of EMG signal of the first channel. Obviously, real EMG signal owns more power so that can be described by RMS. Because of the randomness of noises, SE can be regarded as an indicator of it. Considering both features, for a same window, SE can be used as a weight to balance EMG signal and noise. Based on this observation, we introduce a variable $C(w)$, which can be defined as:

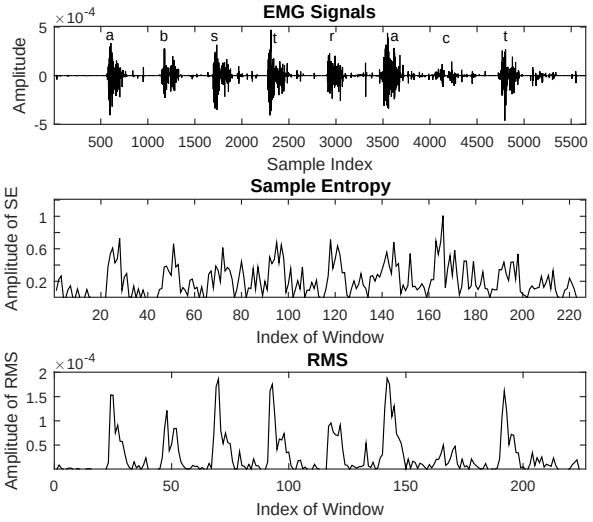


Figure 5: EMG signals, SE and RMS of the first channel.

$$C(w_i) = SE_i \cdot RMS_i^T \tag{4}$$

$$= \sum_{j=1}^{5} (SE_i^j \cdot RMS_i^j) \tag{5}$$

$$SE_i = (SE_i^1, SE_i^2, SE_i^3, SE_i^4, SE_i^5)$$
$$RMS_i = (RMS_i^1, RMS_i^2, RMS_i^3, RMS_i^4, RMS_i^5)$$

where w_i denotes the ith window, and SE_i^j means the SE of ith window in jth channel. Similarly, the RMS of ith window in jth channel is defined as RMS_i^j. Fig. 6 shows that $C(w)$ changes with a sliding window, the length of which is 30 samples. After integrating five channels together, Powerful EMG signal is enhanced and has a higher amplitude than before. At the same time, random noise and fluctuations are further suppressed.

After that, an adaptive threshold-based approach is used to check and detect the onset and offset of keystrokes. We setup following two steps to segment the signal precisely.

3.3.1 Initializing threshold. Everyone has different typing habit with various strength, so the threshold should be initialized by some calibration keystrokes of each user. Guided by the initial program, users are required to type keys A, S, D, and F respectively, and every keystroke has to be completed in 1.2 second. These keys correspond to initial positions of left-hand fingers. Then the threshold T can

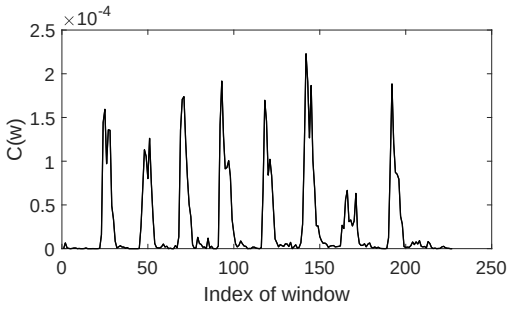


Figure 6: C(w) of EMG signals.

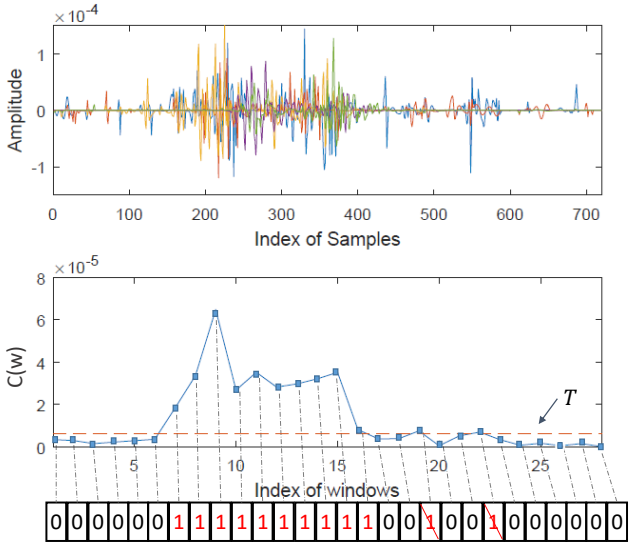


Figure 7: Signal encoding using threshold T. The above subfigure denotes EMG signals of five channels, and the under one shows $C(w)$ to be encoded.

be calculated by:

$$T = \theta \cdot \frac{1}{4} \sum_{k=1}^{4} \frac{1}{n_k} \sum_{i=1}^{n_k} C_k(w_i) \tag{6}$$

where n_k denotes the number of windows of kth keystroke, and θ is set to 0.5 according to experiments for suppressing invalid noises as far as possible.

3.3.2 keystroke segmentation. After obtaining threshold T, we can utilize it to segment keystroke signals. When a user conducts a new keystroke, the $C(w)$ can be calculated using a slide window with 30 samples, and be encoded by threshold T simultaneously:

$$Code(W_i) = \begin{cases} 1 & if\ C(W_i) > T \\ 0 & else \end{cases} \tag{7}$$

where $Code(W_i)$ represents the code of ith window. Fig.7 illustrates the encoding result of a keystroke signal. We can see that efficient signal segment is detected but some wrong results exist due to some bursting noise. As a consequence, two observations are performed to revise the encoded sequence:

- A special case may happen which is an inactive window (0) that appears between two active windows (1). Corresponding to raw EMG signal, it means a very short pause or shift. Therefore, if sequence $(1-0-1)$ appears, we change inactive (0) value to the active (1) one.
- Bursting noises were easily recognized as valid keystroke signals as their high energy. However, they always live last a very short time, which is much shorter than normal keystroke signals. Generally, a keystroke last 0.3s at least (i.e., 5 windows), so we can remove the continuous '1' sequence within 5 windows to eliminate this kind of interference.

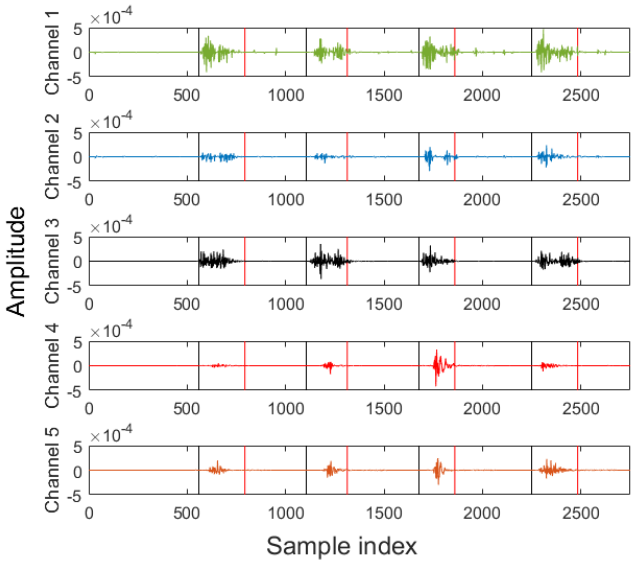


Figure 8: An example of onset and offset detection using adaptive threshold method in five channels.

By doing this, we can finally detect the onset and offset of a valid keystroke signal. Sequence(0-0-1) and (1-0-0) are identified as onset and offset respectively. Fig.8 exhibits the segmentation results of five EMG channels. Although valid signals do not start or end simultaneously as well as hold different signal amplitudes, they are still captured preciously by our adaptive threshold detection algorithm.

3.4 Endpoint detection of words

After detecting each keystroke, the start point and endpoint of a word should further be determined. According to the input habit, when users type two continuous words, they always stop for a short time or stroke 'space' key between two words. In our work, we set the pause, which lasts over 0.8s (around 400 samples), or recognizing a 'space' key, as the gap of two consecutive words. Consequently, when the endpoint event occurs, the nearest character (i.e., keystroke) is the end one of a word. Fig. 9 shows a sample of words segmentation.

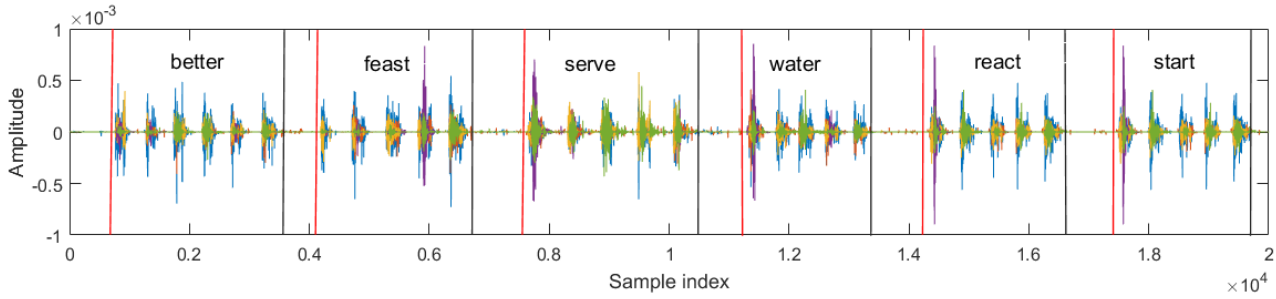


Figure 9: Words endpoint detection.

Table 1: Feature Subsets

SubsetID	Time domain	Frequency domain
1	MAV2,SamEn,MAV	MDF,VCF,MNP SM3,PSR
2	SamEn,WL	MDF,SM2,VCF,MNP
3	MAV2,SamEn,MAV	MDF,SM3,TTP
4	MAV,RMS,SamEn VAR,WL	MDF,MNF,SM1,VCF
5	ACC,MAV1,MAV SamEn,VAR	MDF,SM1,SM3,VCF
6	MAV2,SamEn,WL	MNF,MDF,SM1,VCF
7	MAV2,SamEn,MAV	SM3,TTP,PSR
8	IEMG,MAV,MAV1 SamEn,WL	MNF,MDF,SM1 SM3,VCF
9	ACC,MAV1,MAV SamEn,VAR,WL	SM3,VCF
10	ACC,MAV,MAV WL,SampEn,SSI	MNF,VCF,SM1

3.5 Feature selection

After detecting each stroke, we extract features in both frequency and time domains from corresponding signal segment, which were widely used in previous EMG research. We have extracted about 24 EMG common features such as root mean square (RMS), mean absolute value (MAV), sample entropy (SamEn), spectral moments (SM) and the like which can be referred to [13].

However, it is not possible to apply all these features in the learning stage. Instead, it is better to select an optimal feature set for the sake of achieving satisfying performance and reducing computational overhead. The feature selection in ArmIn is implemented by a wrapper-like method[21], which randomly selects feature subsets and testing the corresponding recognition performance of one user with the SVM model embedded. Then the candidate feature subsets are generated and used for all the users to evaluate with average accuracy using 10-fold cross validation and sort out the best subset. As a result, Table 1 shows the corresponding features with their abbreviation, referring to [13], and Fig. 10 shows the top ten performance of different subsets. Therefore, taking the accuracy and overhead into consideration, we select a best feature set 3 that consisting of 3 time-domain features and 3 frequency-domain features as shown in Table 2 to be used in the learning model. Due to the limited space, readers are suggested to refer to [13] for specific definition of these features.

Table 2: Selected Features in ArmIn

Feature domain	Feature	Description
Time domain	MAV2	Modified mean absolute value type 2
	MAV	Mean absolute value
	SampEn	The sample entropy of signals
Frequency domain	MDF	The Median frequency
	SM3	The 3rd Spectral moments
	TTP	The total power

3.6 Model training

The learning models that we have tested in the design of ArmIn include support vector machine (SVM), K-nearest neighbor (KNN), random forests (RF) and Discriminant Analysis (DA). In order to achieve a high recognition accuracy, we carefully tune key parameters of each model by comparing the average performance with a 10-fold cross validation method.

Considering that these learning models are not new techniques, we only introduce some parameters tuning processes and show corresponding performances briefly here.

- For SVM model, we have tested different kernel functions and finally select the radial basis function (RBF), and results are shown in Fig.11. The critical parameters for RBF kernel are penalty coefficient C and kernel function coefficient γ, which are tuned to 2 and 0.05 respectively, as shown in Table.3, where the P denotes performance of a corresponding pair of parameters.

Table 3: Performance of different SVM parameters

C	1	1	2	2	2	4	4	4	8	8
γ	0.05	0.125	0.05	0.025	0.125	0.025	0.125	0.25	0.05	0.125
P	87%	85.7%	89.2%	87.5%	87%	87.5%	88.6%	85.7%	86.4%	87.6%

- For KNN model, the value of K has significant effect on the recognition accuracy as a small value of K results in large variance and the larger one brings huge biasing phenomenon. We test the performances under different values of K in the range $[1, 50]$. Fig.12 show that this model can obtain the

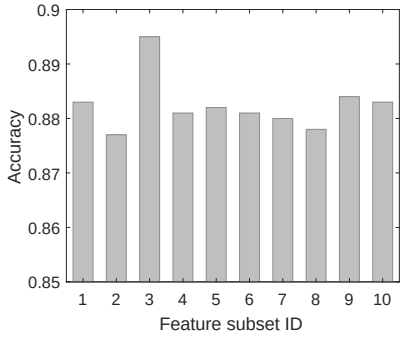


Figure 10: Performance of top 10 feature subsets.

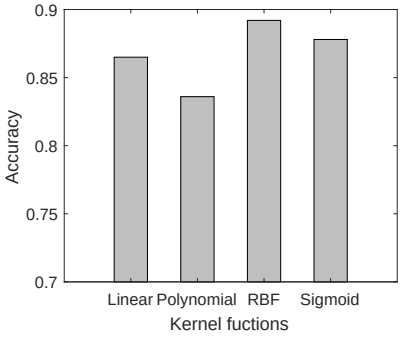


Figure 11: Performance of different kernel functions.

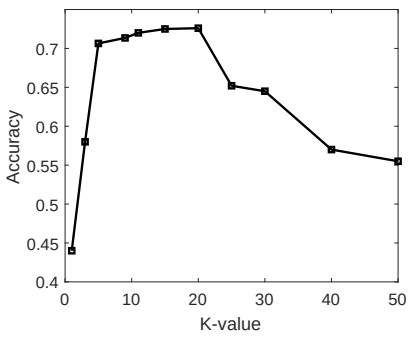


Figure 12: Performance of different K-values.

best performance when K=20, but the accuracy has become almost stable at K=5. Considering that the model which k value is 20 cost too much time to be trained. As a result, we pick the optimal value as 5 in order to achieve a balance between training time and performance.

- For RF model, we evaluate the performances of ArmIn with different parameter settings which mainly include the number of trees and the number of branches in each node represented as 'trees' and 'Dim' in Table.4. The corresponding optimal values are 180 and 4, respectively.

Table 4: Performance of different RF parameters

Set	1	2	3	4	5	6	7	8	9	10
trees	180	140	200	160	120	120	100	100	140	140
Dim	4	4	8	12	4	4	8	4	8	4
P	85.7%	84.6%	84.7%	84.5%	84.2%	84%	83.9%	83.8%	83.6%	83.6%

- For DA model, we test many methods including Liner Discriminant Analysis (LDA, with an 84.43% accuracy), Diaglinear Discriminant Analysis (with 82.57%)and quadratic discriminant analysis (QDA, with 83.25%), and choose LDA as it excels other two methods slightly.

3.7 Word correction

Actually, because of the wrong input or the wrong output, the system may not produce an absolute right word when the user type one word, e.g., the 'bear' is typed as 'vear' or recognized as 'tear'.

We assume the output string is a typo, then use Damerau-Levenshtein [5] algorithm to calculate distances between the typo word and similar words and generate a candidate list, which contains the words whose distance is equal at most 2. For each candidate word, we estimate the probability of this word given the typo word. Words in the candidate list are sorted based on the probability, and that produces a suggestion list. A Bayesian-based correction algorithm[9] can be used:

$$P(W|T) = \frac{P(T|W) \times P(W)}{P(T)}. \tag{8}$$

Since the probability of typing any string (i.e., $P(T)$) is equal, our objective is

$$\max_W P(W|T) \approx \max_W P(T|W) \times P(W), \tag{9}$$

where $P(T|W)$ means the probability of typo word T when the target word is W and $P(W)$ means the probability of the emergence of W, which can be found in the corpus[8].

We regard the string as typo ($T = t_1 t_2 ... t_n ...$), which has the same length as W has. Note that the letter t_i in T is influenced by w_i in W rather than other letters. So, the probability of T given W can be computed as

$$P(T|W) = P(t_1 t_2 ... t_n ... | w_1 w_2 ... w_n ...). \tag{10}$$

Since the letters in T are independent, the resultant probability can be written as

$$P(T|W) = \prod_i P(t_i|w_i). \tag{11}$$

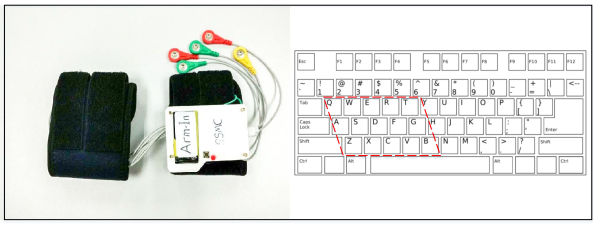
Actually, after training with the recognition model, we can capture all statistics in a confusion matrix (CM), which each cell (pth row and qth column) reveals the possibility that the letter w_p (i.e., w_i) is recognized as the letter w_q (i.e., t_i). According to the confusion matrix, we can obtain the probability of letter t_i given w_i. Hence, we obtain

$$P(t_i|w_i) = CM(w_i, t_i). \tag{12}$$

While taking consideration of the above equation, we can figure out the formula of $P(W|T)$ as

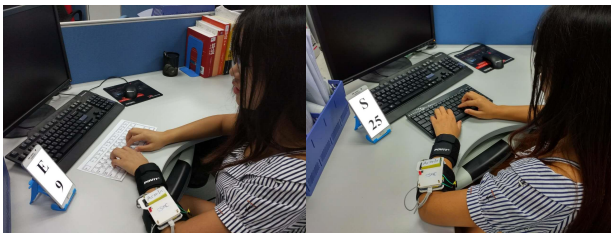
$$\begin{aligned} \max_W P(W|T) &\approx \max_W P(T|W) * P(W) \\ &\approx \max_W \prod_i P(t_i|w_i) * P(W) \\ &\approx \max_W \prod_i CM(w_i, t_i) * P(W). \end{aligned}$$

The word list is sorted by the possibility $P(W|T)$, and we can obtain the target word with the highest rank from the list. With this method, ArmIn can correct wrong words automatically and improvement the inputting efficiency.



(a) Hardware components of ArmIn (b) Keyboard layout and testing keystrokes

Figure 13: The hardware components of ArmIn.



(a) Experiments on printed keyboard (b) Experiments on physical keyboard

Figure 14: Experimental setup for evaluating ArmIn.

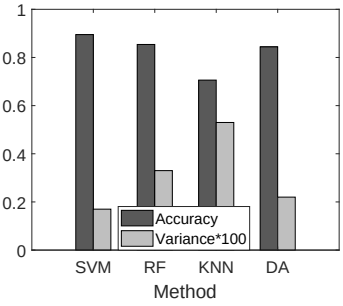


Figure 15: Accuracy of different methods with different variance.

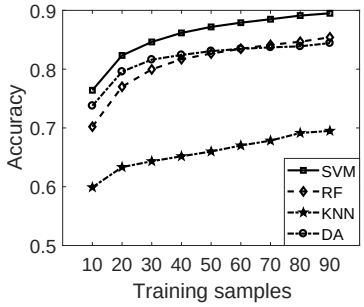


Figure 16: Training overhead for different learning models.

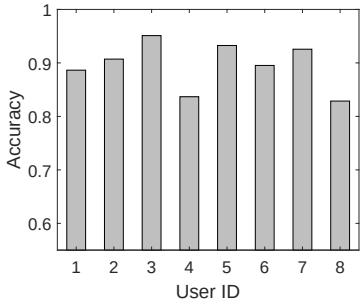


Figure 17: The performance of different users.

4 EXPERIMENTAL SETUP

We design ArmIn in the form of armbands worn by a user on the forearm, which is built on a custom EMG board including an ADS1298 analog front-end device, pins for connecting EMG electrodes, a microSD memory card, and a Bluetooth module for communicating with other devices. The EMG sensing module consists of three parts, which includes electrodes, special leads and ADS1298 development kit. The ADS1298 development kit consists of an ADS1298 processing chip, preprocessing modules and an Arduino Micro motherboard, which are responsible for data collection, preprocessing and transmission. In the experiments, we attach metal EMG electrodes to the skin to sense muscle activities and collect EMG signals with a 500 Hz sampling rate. When EMG signals are collected and preprocessed by the kit, it sends the processed signals to the connected mobile device via Bluetooth. In the mobile end, we implement data processing and keystrokes recognition algorithms. However, it is noted that as an interactive medium, ArmIn can either run the whole data processing pipeline independently or just conduct partial tasks as in the present version.

In order to evaluate the performance of ArmIn, we recruit 8 participants (3 females and 5 males) from our university who do not have any special training background and relevant knowledge of EMG. We evaluate the performance of ArmIn on two different kinds of keyboards, which are a real physical keyboard and a paper-printed one with ArmIn worn on the user's left arm. In the former case, participants are requested to perform keystrokes on physical keyboards as normal. In the case of printed keyboard, we print a keyboard on a paper, place it on a table and request participants to conduct keystrokes similar to a normal keyboard. For each kind of keyboard, we collect data of 16 letters, namely, 'A~G', 'Q~T', 'Z~B', and the space key, corresponding to one's left hand with 130 repetitions for each letter. In order to evaluate word recognition performance, we choose 15 words consisting of these letters covered by the left hand. Each word is typed 30 times per participant.

5 EVALUATION

5.1 Performance of different methods

We test the performance of ArmIn with different learning methods, namely, SVM, Random Forests, KNN and Discriminate Analysis, in order to select the optimal learning method. Specifically, we test the recognition accuracy of each user using a 10-fold cross validation scheme with the aforementioned methods. Shown as Fig. 15, SVM achieves the best performance in the sense of average accuracy (89.5%) over all participants with the lowest variance (0.17%). Following SVM, random forests achieve a favorable recognition accuracy of 85.4% with a variance of 0.33%. The KNN method performs the worst among these methods. To summarize, SVM perform better and is more stable than other methods.

Another perspective to compare different methods is training overhead. We also evaluate the performance of different methods under different number of training samples. In order to do this, we vary the number of training samples from 10 to 90 for each letter and calculate the average recognition accuracy over different participants. The results are displayed in Fig. 16. As seen, the performances of four models increases with the number of training samples. When the quantity of training samples exceeds a certain threshold, the accuracy increases slowly. However, the slope of

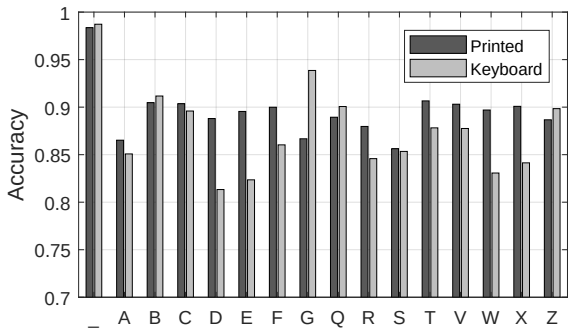


Figure 18: The overall performance of different letters with printed and physical keyboards.

Figure 19: The average confusion matrix over all participants of recognizing 16 letters with the SVM model.

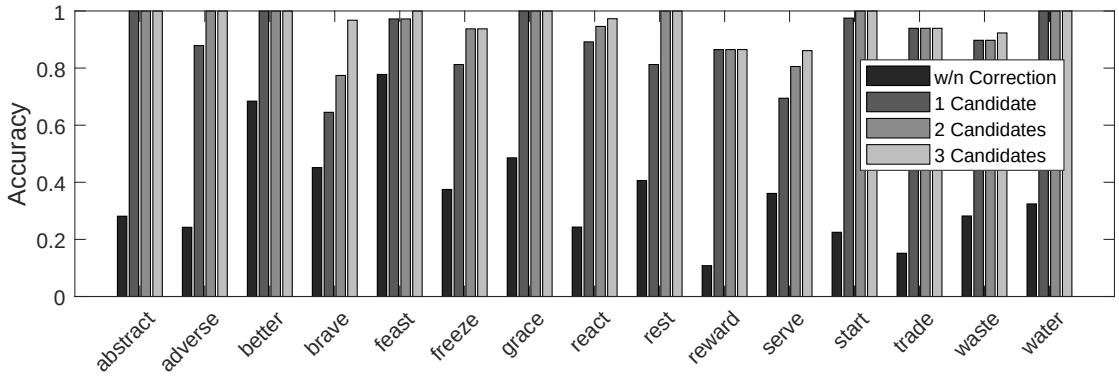


Figure 20: The performance of words correction technique over 15 words.

the accuracy increment varies among different models. Obviously, when the number of training samples exceeds 40 per letter, performances of RF and DA models stay nearly stable. Nevertheless, for SVM and KNN, the performance still goes up until this number reaches 70 per letter. However, it is noted that although SVM has a higher training overhead threshold, it still achieves the highest performance when the number reaches 40. As a result, in this perspective, we can also regard SVM model as the optimal one.

5.2 Performance of different conditions

Different users: We investigate the performance of ArmIn when it works for different users. According the result mention above, we calculate the average recognition accuracy of the total 16 keys with SVM and 80 training letters. Fig. 17 shows that the best performance is 95.1% and the worst is 82.9%.It illustrates that ArmIn is available for different users but difference exists, because although they conduct similar typing actions, their habits and typing strength are diverse.

Different material of keyboards: We also evaluate the performance of ArmIn when it works on two different kinds of keyboards, namely, printed keyboard and physical keyboard. Consequently, we evaluate its performance on them. The result is shown in Fig. 18. For printed and physical keyboards, the average recognition accuracy can achieve about 89.5% and 87.5%, respectively. The lowest recognition accuracy of these two keyboards are 85.6% and 81.3%, respectively. This result shows that performance on printed keyboard is slightly higher than physical one, and the reason behind it is that people type keys with different strength due to their own habits, which can be avoided when typing a printed one without the mechanical feedback.

5.3 Performance of different letters

All the above evaluation only involves overall performance of recognizing letters. In this part, we give the detailed performance of recognizing each letter only when users stroke the printed keyboard in order to obtain deeper insight of ArmIn. Since SVM is selected as the optimal model for ArmIn, we display the average confusion matrix outputted by SVM model, shown in Fig. 19. This confusion matrix is obtained by averaging the results of all participants with 40 training samples for each letter. It is obvious that the recognition error distribution among these letters presents a certain pattern. That is, errors are distributed symmetrically on either

side of the diagonal line. The reason behind this phenomenon is that keystrokes belonging to the same group, are conducted by the same finger, are more easily mis-classified with each other. This is because these keystrokes can cause much similar muscle activities and consequently induce EMG signals with highly similar features. Inspired by this observation, the possible way to further improve the recognition performance is that providing candidate letters near the recognized keystroke. We will verify this solution in our future work. The other result is shown in Fig. 18. The performance of all letters that the lowest accuracy is 85.6%. It means that ArmIn holds a stable recognition accuracy among different letters.

5.4 Performance of words correction

We combine the characters obtain from the recognition model in a consecutive manner to create a word. However, any wrong typing or failed recognition of letters will bring about the error word. So it is much possible that the constructed word is not the expected one. In our correction method, American national corpus [8] is utilized for providing the $P(W)$ mentioned in section 3.7, and the correction algorithm produces a suggestion word list to the user. Fig. 20 shows the results under different conditions, that includes 1) without correction, 2) one candidate word after correction, 3) two candidate words, 4) three candidate words. If the target word is equal to the expected word or the one in the suggestion word list, the result is regraded as a correct one. Without correction, the accuracy of word recognition is not good, which is only 36%. With one candidate word, the accuracy rises and it is 43.6%. When two candidate words are displayed, the system can achieve 92.5% accuracy. The performance can be enhanced further by considering more candidate words, e.g., 93% accuracy for three candidate words. As depicted in Fig. 20, two candidate words in the suggestion list is enough for achieving a reasonable level of accuracy.

6 CONCLUSION

Motivated by the emerging EMG-based human-computer interface, we step further to explore the feasibility of designing a text-entry system, termed ArmIn, using EMG which can recognize more fine-grained keystrokes. We implement ArmIn with commercial EMG electrodes and custom signal processing board, and conduct experiments to evaluate its performance of keystroke recognition on physical and printed keyboards. The experimental results show that ArmIn can recognize keystrokes on both kinds of keyboards with a similar accuracy of 89.5%. Providing two candidates, the word recognition can achieve a 92.5% accuracy. The results verify the feasibility of recognizing fine-grained keystrokes with EMG sensors and open up a new vision of HCI applications using EMG techniques.

ACKNOWLEDGMENTS

This research was supported in part by the China NSFC Grant 61802264, 61872248, 61472259, Joint Key Project of the National Natural Science Foundation of China (Grant No. U1736207), Guangdong Natural Science Foundation 2017A030312008, Shenzhen Science and Technology Foundation (No. JCYJ20170302140946299, JCYJ20170412110753954), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (Grant No.161064), Guangdong Talent Project 2015TX01X111 and GDUPS (2015), and Tencent "Rhinoceros Birds" - Scientific Research Foundation for Young Teachers of Shenzhen University.

REFERENCES

[1] Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves, and Frank DeRuyter. 2011. Using mobile phones to write in air. In *ACM MobiSys*.
[2] Kamran Ali, Alex X Liu, Wei Wang, and Muhammad Shahzad. 2015. Keystroke recognition using wifi signals. In *ACM MobiCom*. 90–102.
[3] Christoph Amma, Marcus Georgi, and Tanja Schultz. 2012. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3D-space handwriting with inertial sensors. In *IEEE ISWC*. 52–59.
[4] Christoph Amma, Thomas Krings, Jonas Böer, and Tanja Schultz. 2015. Advancing muscle-computer interfaces with high-density electromyography. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 929–938.
[5] Gregory V Bard. 2007. Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric. In *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*. Australian Computer Society, Inc., 117–124.
[6] R. R. Coifman and D. L. Donoho. 1995. *Translation-Invariant De-Noising*. Springer New York. 125–150 pages.
[7] Donny Huang, Xiaoyi Zhang, T Scott Saponas, James Fogarty, and Shyamnath Gollakota. 2015. Leveraging Dual-Observable Input for Fine-Grained Thumb Interaction Using Forearm EMG. In *ACM UIST*.
[8] Ide, Nancy and Suderman, Keith. 2015. American National Corpus Project. http://www.anc.org/. Accessed on 2017-05-10.
[9] Mark D Kernighan, Kenneth W Church, and William A Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics*. 205–210.
[10] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. 2015. Snooping keystrokes with mm-level audio ranging on a single phone. In *In ACM MobiCom*. 142–154.
[11] Joseph KF Ng, Carolyn A Richardson, and Gwendolen A Jull. 1997. Electromyographic amplitude and frequency changes in the iliocostalis lumborum and multifidus muscles during a trunk holding test. *Physical therapy* 77, 9 (1997), 954.
[12] Shahriar Nirjon, Jeremy Gummeson, Dan Gelb, and Kyu-Han Kim. 2015. Typingring: A wearable ring platform for text input. In *ACM MobiSys*.
[13] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. 2012. Feature reduction and selection for EMG signal classification. *Expert Systems with Applications* 39, 8 (2012), 7420–7431.
[14] Joshua S. Richman and J. Randall Moorman. 2000. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology Heart and Circulatory Physiology* 278, 6 (2000), H2039.
[15] T Scott Saponas, Desney S Tan, Dan Morris, and Ravin Balakrishnan. 2008. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *ACM CHI*. 515–524.
[16] 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 *ACM UIST*.
[17] DF Stegeman and H Hermens. 2007. Standards for surface electromyography: The European project Surface EMG for non-invasive assessment of muscles (SENIAM). *Proceedings of the Third General SENIAM Workshop on Surface Electromyography* (2007), 108–112.
[18] Li Sun, Souvik Sen, Dimitrios Koutsonikolas, and Kyu-Han Kim. 2015. Widraw: Enabling hands-free drawing in the air on commodity wifi devices. In *In ACM MobiCom*. 77–89.
[19] Thalmic Labs Inc. 2013. Myo Gesture Control Armband. https://www.thalmic.com/en/myo/. Accessed on 2016-08-16.
[20] Tomáš Tisančín, Adam J Sporka, and Ondřej Poláček. 2014. EMG Sensors as Virtual Input Devices. In *ACM International Conference on Multimedia, Interaction, Design and Innovation*.
[21] Minh Phuong Tu, Z Lin, and R. B Altman. 2005. Choosing SNPs using feature selection. In *Computational Systems Bioinformatics Conference, 2005. Proceedings*. 301–309.
[22] Wei Wang, Alex X Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *ACM MobiCom*. 82–94.
[23] Kevin R Wheeler, Mindy H Chang, and Kevin H Knuth. 2006. Gesture-based control and EMG decomposition. *IEEE Transactions on Systems, Man, and Cybernetics* 36, 4 (2006), 503–514.
[24] Yafeng Yin, Qun Li, Lei Xie, Shanhe Yi, Edmund Novak, and Sanglu Lu. 2016. CamK: A camera-based keyboard for small mobile devices. In *IEEE INFOCOM*. 1–9.

[25] Xu Zhang, Xiang Chen, Yun Li, Vuokko Lantz, Kongqiao Wang, and Jihai Yang. 2011. A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Transactions on Systems, Man, and Cybernetics* 41, 6 (2011), 1064–1076.

[26] Xu Zhang and Ping Zhou. 2013. Filtering of surface EMG using ensemble empirical mode decomposition. *Medical engineering & physics* 35, 4 (2013), 537–542.