Hindawi
Mobile Information Systems
Volume 2020, Article ID 8864627, 16 pages
https://doi.org/10.1155/2020/8864627

Research Article

Your Knock Is My Command: Binary Hand Gesture Recognition on Smartphone with Accelerometer

Huixiang Zhang,[1] **Wenteng Xu**,[1] **Chunlei Chen**,[2] **Liang Bai**,[3] **and Yonghui Zhang**[2]

[1]*School of Cyberspace Security, Northwestern Polytechnical University, Xi'an 710072, China*
[2]*School of Computer Engineering, Weifang University, Weifang 261061, China*
[3]*No. 203 Research Institute of China Ordnance, Xi'an 710065, China*

Correspondence should be addressed to Huixiang Zhang; zhanghuixiang@nwpu.edu.cn

Received 11 March 2020; Revised 21 May 2020; Accepted 1 July 2020; Published 26 July 2020

Academic Editor: Ali Kashif Bashir

Motion-based hand gesture is an important scheme to allow users to invoke commands on their smartphones in an eyes-free manner. However, the existing scheme is facing some problems. On the one hand, the expression ability of one single gesture is limited. As a result, a gesture set consisting of multiple gestures is typically adopted to represent different commands. Users must memorize all gestures in order to make interaction successfully. On the other hand, the design of gestures needs to be complicated to express diverse intensions. However, complex gestures are difficult to learn and remember. In addition, complex gestures set a high recognition barrier to smart APPs. This leads to an imbalance problem. Different gestures have different recognition accuracy levels, which may result in instability of recognition precision in practical applications. To address these problems, this paper proposes a novel scheme using binary motion gestures. Only two simple gestures are required to express bit "0" and "1," and rich information can be expressed through the permutation and combination of the two binary gestures. Firstly, four kinds of candidate binary gestures are evaluated for eyes-free interactions. Then, an online signal cutting and merging algorithm is designed to split accelerometer signals sequence into multiple separate gesture signal segments. Next, five algorithms, including Dynamic Time Warping (DTW), Naive Bayes, Decision Tree, Support Vector Machine (SVM), and Bidirectional Long Short-Term Memory (BLSTM) Network, are adopted to recognize these segments of knock gestures. The BLSTM achieves the top performance in terms of both recognition accuracy and recognition imbalance. Finally, an Android application is developed to illustrate the usability of the proposed binary gestures. As binary gestures are much simpler than traditional hand gestures, they are more efficient and user-friendly. Our scheme eliminates the imbalance problem and achieves high recognition accuracy.

1. Introduction

Eyes-free interaction is a method of controlling mobile devices without having to look at the device [1]. A variety of schemes have been developed to let users interact in an eyes-free manner. In [2], a digital calculator that operated with fingers on touch screens is developed. This method utilizes taps for digits input and uses swipes for other operations. Seventeen finger gestures are defined for arithmetic tasks. In [3], a nonvisual text entry method that uses the 6 bit Braille character encoding is presented. A signal is an input by touching the screen with several fingers where each finger represents one bit, either touching the screen or not. In addition to surface gestures, voice commands also provide a solution [4]. Siri is one of the most prominent examples of a mobile voice interface. Another important way is to use a motion-based hand gesture [5]. To command a smartphone to execute a task, a user needs to perform a hand gesture with that phone in hand. The type of gesture is recognized through analysing data samples captured by motion sensors, such as accelerometers, gyroscopes, and orientation sensors.

Motion-based hand gestures enjoy several advantages. Firstly, users do not need to pay visual attention to the touchscreen because the physical location of the smartphone can be perceived via proprioception [6]. Secondly, hand-motion-gesture interaction puts forwards a few restrictions on the surrounding environment. For example, voice commands are prone to error in noisy environments [7], but

motion gestures can be performed as long as the hands of users are free. Finally, motion-based hand gestures can be designed in three-dimensional space. Compared to surface gestures, there remains larger design space for a variety of interactive tasks [8–10].

However, the scheme using motion-based hand gesture to command smartphones is facing three problems.

(1) In order to represent different commands, a gesture set consisting of multiple gestures is required. For example, fourteen gestures are specified in the literature [5]; 11 gestures are proposed in the literature [11]. Users need to learn the set of hand gestures supported by a smartphone. They must memorize all gestures in order to make interaction successfully.

(2) In order to distinguish these different gestures, hand gestures are defined not only in terms of the movement shape but also based on the motion kinematics [12]. Users are required to learn the features of gestures, in terms of movement shape and kinematics. It could be a daunting barrier to grasp details of such features. In addition, gestures with complex features set up a barrier to achieving high recognition accuracy.

(3) The design of multiple gestures causes an uneven distribution of recognition accuracy levels among different gestures, which hinders the practical application of such design. For example, a deep feed-forward neural network is proposed to recognize 11 hand gestures in the literature [11]. They attained a minimum hit rate of 70.35% for Gesture 1 and a maximum hit rate of 100% for Gesture 10. As a result, the recognition accuracy levels of different gestures are dramatically different.

The root cause of the above problem is that multiple types of gestures are required to complete a specific interaction task with a phone. To address this problem, a novel interaction scheme using binary gestures is proposed in this paper. Only two kinds of hand gestures are needed to express binary bit "0" and "1." Through the permutation and combination of the two binary gestures, a bit sequence is constructed. The application installed on smartphones can identify the bit sequence by analysing sensors' signals. As the binary gestures are much simpler than traditional hand gestures, they are easy to learn and remember for users. High recognition accuracy can be achieved for both gestures. Thus, there will be no imbalance problem.

Taking the swiping movement gesture as an example, it is stipulated that users swipe of the smartphone horizontally to left and to right represent the bit "0" and "1," respectively. By combining binary gestures, complex meanings can be expressed. For instance, if the user swipes the phone to the left four times in succession, it means that the command is "0000." The permutation and combination of four binary gestures can represent up to 16 commands. We believe that it is easier for users to remember numbers than complex gestures.

It should be noted that we do not intend to design a set of gestures to meet the requirement of all kinds of interaction tasks. We just provide an alternative for the eyes-free interaction scenarios. Its typical application scenarios include visually disabled users [13], distracted interaction [14], and covert operation [15].

The main work and contribution of the paper are summarized as follows.

(1) A novel user-smartphone interaction scheme using binary gestures in an eyes-free manner is proposed.

(2) An online signal cutting and merging algorithm is designed to extract the independent gesture signal segment from the binary gesture sequence. This online algorithm achieves an accuracy rate comparable to the offline SVM algorithm.

(3) Five algorithms, including DTW, Naive Bayes, Decision Tree, SVM, and BLSTM, are adopted to recognize binary gestures, and BLSTM has reached a recognition accuracy of 98%.

(4) A prototype application that uses binary gestures to send SMS messages is implemented on the Android platform.

The rest of this paper is organized as follows. The definition of binary gestures is introduced in Section 2. Section 3 describes the segmentation process of binary gesture sequences in detail. In Section 4, five algorithms are exploited to recognize a segmented knock gesture. Section 5 introduces a prototype application that uses binary gesture interaction. Finally, the work of this paper is concluded.

2. Definition of Binary Gestures

We exploit four categories of binary gesture according to a standard 3-axis coordinate system. In the standard 3-axis coordinate system, the x-axis is horizontal and points to the right. As illustrated in Figure 1, the y-axis is vertical and points up, and the z-axis points toward the outside of the screen face [27].

The definition of the four binary gestures is shown in Table 1. In the definition, the phone is supposed to hold in its portrait orientation by users' two hands. The swipe, pitch, and flip gestures are performed along the z-axis, x-axis, and y-axis, respectively. For the knock gesture, the user holds the phone in one hand and taps on the screen with the index finger of the other hand.

A set of command encoded in binary is defined to represent the user's interactive intention. A specific command is transformed to a gesture sequence consisting of single-actions and double actions. In each gesture category, a single-action is defined to represent the meaning of "0," and a double-action is defined to represent the meaning of "1." A double-action gesture includes two consecutive single-action gestures. Multiple gestures constitute a binary gesture sequence for interaction. Take knock gesture as an example; if a user wants to issue a 4-bit command "0101" to a smartphone, he is required to perform 4 knock actions in sequence. In other words, the user needs to perform "single-knock, double-knock, single-knock, double-knock" on the smartphone within a specified time range.

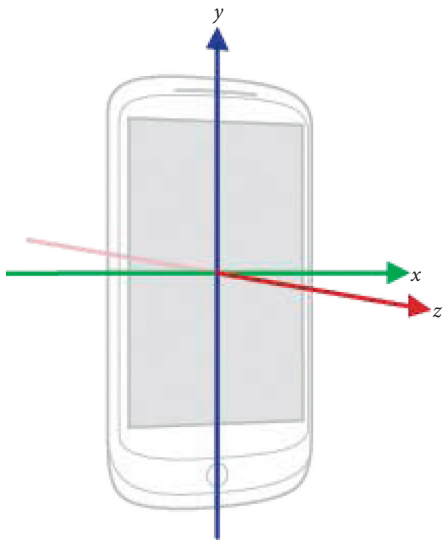

Figure 1: A standard 3-axis coordinate system [27].

Table 1: The definition of binary gestures.

Gesture category	Action	Meaning
Swipe (along the z-axis)	Single	0
	Double	1
Pitch (along the x-axis)	Single	0
	Double	1
Flip (along the y-axis)	Single	0
	Double	1
Knock (on the screen)	Single	0
	Double	1

An accelerometer is very common on smartphones. It is a vital sensor to monitoring device motion, such as tilt, shake, rotation, and swing. In addition, it uses about 10 times less power than other motion sensors [16]. For the aforementioned reasons, we consider collecting accelerometer data to identify user gestures. The application installed in the smartphone analyses the acceleration sensor data to identify the binary bit sequence.

Figure 2 illustrates the collected 3-axis accelerometer data while performing two binary gestures in succession under different categories. The two successive gestures represent a bit sequence of "01." The x, y, and z curves correspond to the 3-axis accelerometer data. It can be seen from Figure 2(a), there is a lot of noise in the acquired accelerometer signal of the swipe gestures. It is difficult to distinguish the two swiping action gestures. In contrast, the pitch, flip, and knock gestures are easier to distinguish. The single and double actions of these gestures are mainly distinguished according to the number of crests or troughs. From Figure 2(b), it can be clearly seen that the single-pitch gesture has a significant trough in the z-axis and a significant crest in the y-axis, while the double-pitch gesture has two troughs and crests in the corresponding axis. In Figure 2(c), the waveform of flip gestures is similar to that of pitch gestures, but the crests appear on the x-axis. For the knock gesture shown in Figure 2(d), the single-knock action has a significant crest, while double-knock action has two significant peaks. In summary, the pitch, flip, and knock gestures are considered in the following discussion.

In the next section, we will explain in detail how to identify the binary bit sequence passed by the user from the accelerometer signal.

3. Signal Segmentation

3.1. Overall Process. The overall processing flow is shown in Figure 3.

The 3-axis accelerometer signals are continuously acquired by an application installed on a smartphone. Before the start of each interaction, the phone is kept motionless for a period of time (more than 1 second). This motionless period is seen as a start signal of a gesture sequence. It is called the initial quiet period.

Firstly, the collected signals are preprocessed by synthesis and filtering. Then, the initial quiet period is detected. Once the start signal appears, an online bit cutting process is used to cut out independent gesture signal segments from a continuous signal stream. Next, the cut-out gesture signal segment is identified in its binary meaning. In an ideal state, a sequence composed of N binary gestures can be divided into N independent gestures signal segments. The final output is a N-bit binary sequence, which represents user's command message.

3.2. Signal Acquisition and Preprocessing

3.2.1. Sampling Frequency. In an Android smartphone, the sampling frequency of the various sensor is set in the system. There are four values that are available [17].

① SENSOR_DELAY_NORMAL, the sampling frequency is about 5 Hz.

② SENSOR_DELAY_UI, the sampling frequency is about 16 Hz.

③ SENSOR_DELAY_GAME, the sampling frequency is about 50 Hz.

④ SENSOR_DELAY_FASTEST, sample as fast as possible.

In the samples we collected, the duration of a single-knock gesture is about 0.2s-0.5s, which is equivalent to a gesture frequency of 2 Hz ~ 5 Hz. According to the Shannon sampling theorem, the sampling frequency of the signal should be no less than 10 Hz. If SENSOR_DELAY_FASTEST is used, the sampling frequency is much larger than 10 Hz, and too many samples are collected. This brings unnecessary overhead for subsequent calculations. The two frequencies of SENSOR_DELAY_UI and SENSOR_DELAY_GAME are more reasonable. Considering the accuracy of gesture recognition, we choose 50 Hz as the sampling frequency to obtain more sampling points.

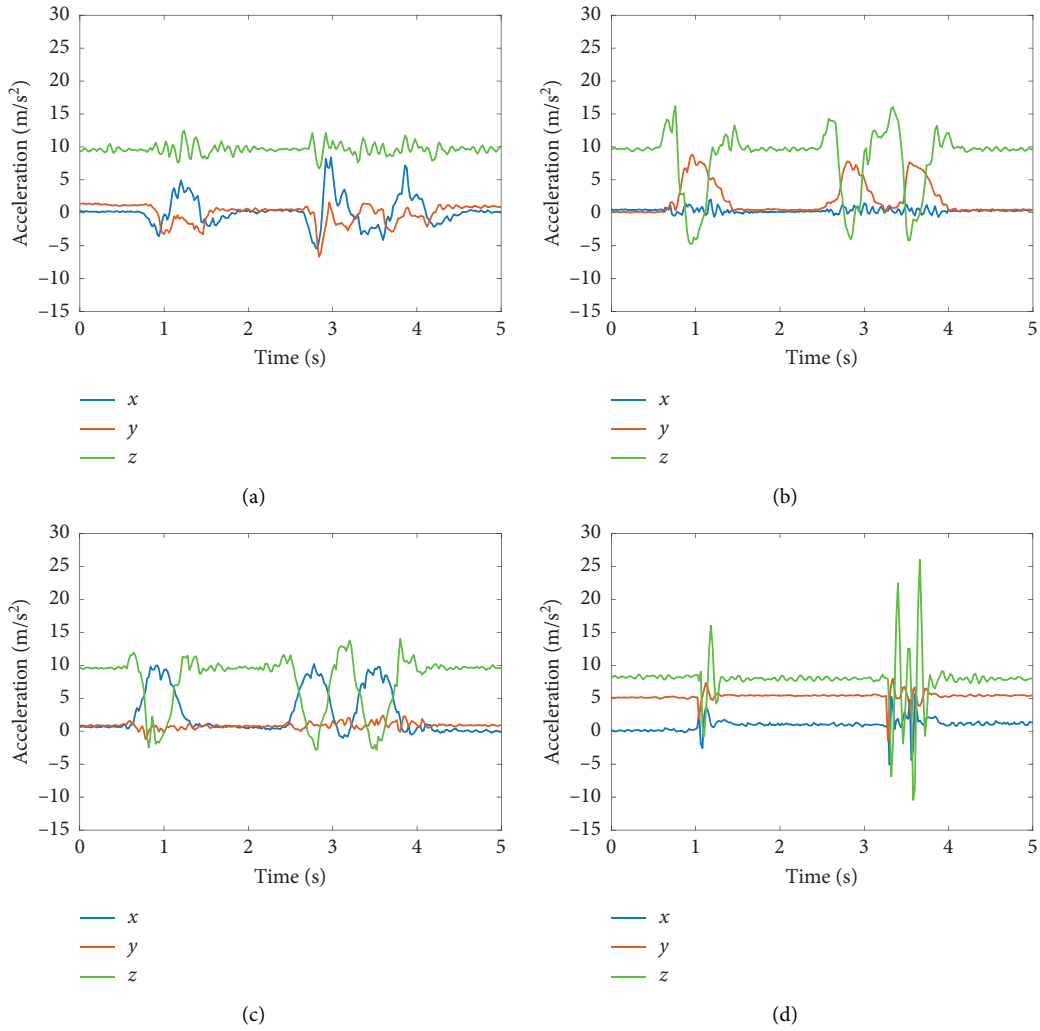

Figure 2: 3-axis accelerometer data for a bit sequence of "01" under different definitions. (a) Swipe. (b) Pitch. (c) Flip. (d) Knock.

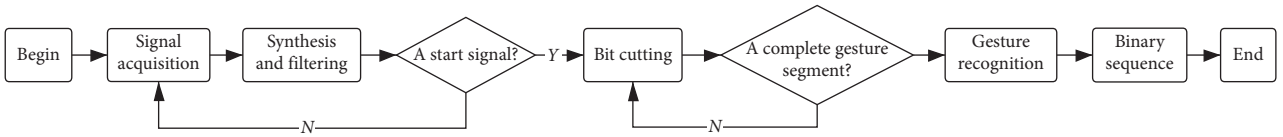

Figure 3: The overall processing flow.

3.2.2. Signal Synthesis and Filtering. To avoid the influence of the sensor's own drift and gravity, we have performed vector synthesis on the 3-axis data [18]:

$$A = \sqrt{A_x^2 + A_y^2 + A_z^2} - G, \tag{1}$$

where G represents the acceleration of gravity. A_x, A_y, and A_z represent the accelerometer sampling value of the X-axis, Y-axis, and Z-axis, respectively.

In order to filter out abnormal points and noise in the collected data, a low-pass filter is performed as follows:

$$\overline{A_i} = \alpha A_i + (1 - \alpha)\overline{A_{i-1}}. \tag{2}$$

Here, A_i represents the i^{th} synthesized accelerometer sample and $\overline{A_i}$ represents the value obtained after filtering. As the new sampling points are more significant for feature extraction and recognition, it is recommended to choose a large value of α to retain a large proportion of sampled values.

3.3. Bit Cutting Process. The bit cutting process attempts to separate independent gesture signal segments from the continuously collected accelerometer signal stream. The bit cutting process operates in an online mode. Instead of acquiring the complete binary gesture sequence signals,

cutting and analysing operations run simultaneously. Figure 4 shows the complete flowchart of the bit cutting process.

The classic Sliding Window (SW) and Sliding Window and Bottom-up (SWAB) algorithms [19] are used to perform online signal segmentation. These algorithms cannot cut out a single complete binary gesture signal at one time. By contrast, such algorithms obtain a large number of short signal segments. Therefore, a merge algorithm is designed to combine these short signal segments into a complete binary gesture signal segment. The pseudocode of a bit cutting process is illustrated in Algorithm 1.

3.3.1. Cutting Algorithm. SW and SWAB are two kinds of online signal cutting algorithms used to extract physical signal segments from time-series signals. The SW algorithm read sample into a sliding window continuously then uses linear regression to fit a line for the samples in the window. At some points, the cumulative error is greater than a user-specified threshold (denoted as $E_{\max}$), so the subsequence in the window is transformed into a segment. Then, the size of the sliding window is reduced to 0, and the process iterates until the entire time serial has been transformed into a piecewise linear approximation. The SWAB algorithm keeps a small buffer to gain a "semiglobal" view of the dataset for Bottom-Up. It scales linearly with the size of the dataset, requires only constant space, and produces high quality approximations of the data. That is beneficial to application in mobile devices.

The cumulative error E_{cum} of the linear approximation is calculated as follows:

$$E_{\text{cum}} = \sum_{i=1}^{n} \sqrt{\left(\overline{A_i} - \widehat{A}_i\right)^2}. \tag{3}$$

Here, $\widehat{A}_i$ is the fitted value of the i^{th} data sample after signal synthesis and filtering, and n is the current window size. Whenever the window size changes, the cumulative error is recalculated.

Figure 5 shows a preprocessed accelerometer signal sequence generated by two consecutive knock gestures, which are a single-knock and thereafter a double-knock. As illustrated in Figure 5, there is a relatively calm interval between two adjacent knock gestures, such as the interval of 2.5 s–4.5 s. This kind of interval is called the quiet period. In contrast, the signal period with relatively strong fluctuations is called the fluctuation period, such as the interval of 1.5 s~2.5 s and the interval of 4.5 s~6.0 s. Those are the signal segments corresponding to the user's knock gestures. Ideally, the quiet period and the fluctuation period alternate in the signal sequence of binary gestures.

After processing by the SW/SWAB, the signal sequence is cut into multiple short segments. As illustrated in Figure 5, these short segments are separated by blue vertical dashed lines. During the quiet period, there will be fitting errors due to small fluctuations. After a period of time, the cumulative error will eventually exceed the cutting threshold $E_{\max}$. Therefore, the signal in the quiet period will be cut into multiple sparse segments. During the fluctuation period, due to the relatively large fluctuation of the accelerometer signal, the cumulative error will exceed the cutting threshold $E_{\max}$ in a short time. Thus, the signal in the fluctuation period will be cut into multiple dense segments.

In order to extract a complete gesture, it is necessary to design a merge algorithm to combine multiple signal segments included in the fluctuation period. For the signal in Figure 5, two complete signal segments corresponding to the two knock gestures should be extracted after segments merging.

3.3.2. Merge Algorithm. For a segment, we can compute its average error E_{avg} as in the following equation:

$$E_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} \sqrt{\left(\overline{A_i} - \widehat{A}_i\right)^2}. \tag{4}$$

Here, $\overline{A_i}$ is the value of the i^{th} sample after signal synthesis and filtering, $\widehat{A}_i$ is the fitted value of the corresponding sample, and n is the number of samples in the segment. In particular, the average error of the initial quiet period is denoted as E_{avgb}.

Further, a characteristic p is defined to measure the fluctuation level of a segment. For the kth segment cut out by the SW/SWAB algorithm, its fluctuation characteristic $p(k)$ is set according to the following equation:

$$p(k) = \begin{cases} 0, & E_{\text{avg}}(k) < \beta * E_{\text{avgb}}, \\ 1, & E_{\text{avg}}(k) > \beta * E_{\text{avgb}}. \end{cases} \tag{5}$$

A fluctuation characteristic of 0 indicates that the segment's fluctuation is low and belongs to a quiet period. By contrast, a fluctuation characteristic of 1 indicates that the segment's fluctuation is high and belongs to a fluctuation period.

β is a coefficient used to balance $E_{\text{avg}}(k)$ and E_{avgb}. In general, the average error of the segments included in a quiet period is slightly larger than E_{avgb}. Thus, the value of β should be greater than 1. However, if β is set to a large value, segments that belong to a fluctuation period would be marked as segments belonging to a quiet period incorrectly.

After the above processing, we can get a binary numerical sequence of the fluctuation characteristic, that is, $P = [p(1), p(2), \cdots, p(k)]$. The merging algorithm processing flow is shown in Figure 6.

When the k^{th} segment is cut out ($k \geq 3$), the merge operation is performed according to the fluctuation characteristics of the last three segments, i.e., $[p(k-2), p(k-1), p(k)]$. There are three cases in which a merge operation can be performed:

(1) p_{k-1} equals p_{k-2}, the $(k-1)$th and the $(k-2)$th segment are merged into a new segment, and the fluctuation characteristic of the new segment remains unchanged.

(2) The sequence P matches [0, 1, 0], and the size of the $(k-1)$th segment is less than $C_{\min}$; it means that these three segments can be merged into a new segment with a fluctuation characteristic of 0.

(3) The sequence P matches [1, 0, 1], and the size of the $(k-1)$th segment is less than $C_{\max}$; it means that

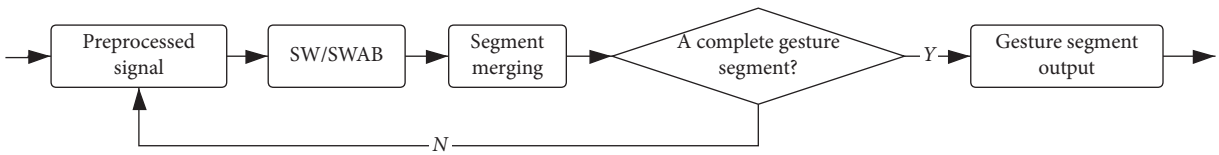

Figure 4: The flowchart of bit cutting process.

Input: α, the coefficients of the low-pass filter
β, the coefficients to adjust the fluctuation characteristic
E_{max}, the user-set maximum cumulative error threshold
E_{bavg}, the average error of the initial quiet period
Output: A complete gesture signal segment
/* initialization */;
$i \leftarrow 0$, $A \leftarrow []$;
$k \leftarrow 0$, *Segment* $\leftarrow []$, $P \leftarrow []$;
while *Get i*th *3-axis acceleration sample: Ax, Ay, Az* **do**
  /* signal synthesis and filtering */;
  $A_i \leftarrow \sqrt{A_x^2 + A_y^2 + A_z^2} - G$
  $\overline{A_i} \leftarrow (1 - \alpha)\overline{A_i} + \alpha A_i$;
  $A[i] \leftarrow \overline{A_i}$;
  $i \leftarrow i + 1$;
  /* SW or SWAB */;
  start $\leftarrow$ Segment$[k-1]$, end $\leftarrow i - 1$;
  $\widehat{A} \leftarrow$ linear regression to fit a line for A[*start*: *end*];
  for j = start $\longrightarrow$ end **do**
    $E_{cum} \leftarrow E_{cum} + \sqrt{(\overline{A}_j - \widehat{A}_j)^2}$
  end
  if $E_{cum} > E_{max}$ **then**
    /* a new segment is cut out by SW or SWAB */;
    Segment$[k] \leftarrow i - 1$;
    $E_{avg} \leftarrow 1/a_i E_{cum}$;
    /* set the fluctuation characteristic */;
    if Eavg $< \beta E_{bavg}$ **then**
      $P[k] \leftarrow 0$;
    else
      $P[k] \leftarrow 1$;
    end
    /* process by merge Algorithm */;
    Segment, P, k $\leftarrow$ *merge*(*Segment, P, k*);
    /* check if a complete segment is cut-out */;
    if [P [0], P [1], P [2]] = [1, 0, 1] **then**
      $TS \leftarrow A$[0: *Segment*[0]];
      output TS for recognition;
    end
    $k \leftarrow k + 1$;
  end
end

Algorithm 1: The pseudocode of bit cutting process.

these three segments can be merged into a new segment with a fluctuation characteristic of 1.

If none of the above cases are met, the current round of merging operation ends, then waiting for a new segment is cut out by the SW/SWAB.

There are two important parameters in the merge process, i.e., C_{max} and C_{min}. The size of the segment is actually the duration of the signal. In Case 3, C_{max} indicates the maximum duration of a quiet period allowed in a complete gesture signal. As a double-knock gesture is two consecutive single-knocks, there is usually a drop in the signal. The duration of the signal drop is about 100–300 ms in our experiments. Therefore, C_{max} is set to 15 under a sample frequency of 50 Hz.

In Case 2, C_{min} indicates the maximum duration of a fluctuation allowed in a quiet period. C_{min} is affected by many factors, such as the use scenario and sensor accuracy. Therefore, C_{min} is set to 3 conservatively in our experiments.

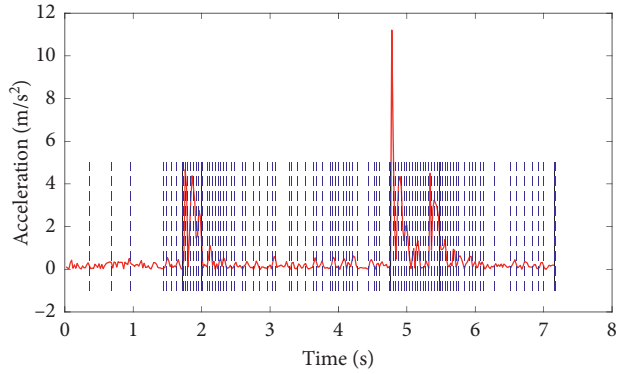

Figure 5: Cutting result of the SW/SWAB algorithm.

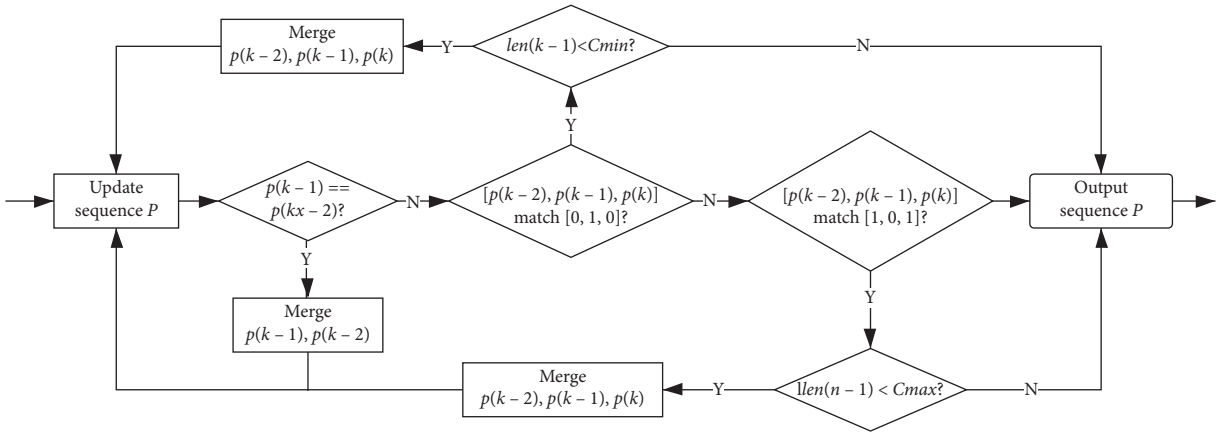

Figure 6: The merge algorithm processing flow.

Figure 7 illustrates the execution of the merge algorithm for an independent double-knock signal. In Figure 7(a), when the 3rd segment is cut out, the characteristic sequence P is [1,1,0] at this time. As $p_1 = p_2$, the first and the second segments are merged into a new segment with a fluctuation characteristic of 1. The characteristic sequence P is updated to [1,0]. Then, the 4th segment is cut out with a fluctuation characteristic of 0; thus, P is updated to [1,0,0] as in Figure 7(b). This does not fall into the three aforementioned merge cases. Next, the 5th segment is cut out with a fluctuation characteristic of 1. At now, the sequence P is changed to [1,0,0,1]. The fluctuation characteristics of the last three segments are checked. As $p_2 = p_3 = 0$, the two segments are merged into a new segment with a fluctuation characteristic of 0. After that, the sequence P is changed to [1,0,1], as in Figure 7(c). Suppose the size of the new segment is less than $C_{\max}$, that meets the merge Case 2. The three segments are merged into a big segment with a fluctuation characteristic of 1 as shown in Figure 7(d). Through continuous online cutting and merge processing, the complete segment of a knock gesture can be extracted. The pseudocode of the merge algorithm is shown in Algorithm 2.

3.3.3. Bit Cutting Experiments. Two experimental scenarios are designed. In Scenario 1, the smartphone is placed on the desktop; in Scenario 2, the smartphone is held on user's hand. A total of 8 volunteers participated in the experiments. Each volunteer is required to perform 4 knock gestures during an interaction. A round of experiments contains 16 interactions, corresponding to the bit sequences "0000"-"1111." Ten rounds of experiments were performed, and 2,560 gesture samples for each scene are obtained.

A metric of cut-out rate is used to evaluate the effect of the bit cutting process. The cut-out rate is defined as follows:

$$\text{cut of rate} = \frac{\text{number of bits actually cut out}}{\text{number of bits theoretically cut out}}. \quad (6)$$

The setting of parameters is shown in Table 2.

The experiments mainly analyse the cut-out rate of the binary gestures under different cumulative error thresholds $E_{\max}$. The threshold $E_{\max}$ is set as follows [19]:

$$E_{\max} = E * 2^m. \quad (7)$$

Here, E is 0.01, and m varies from 0 to 12. The experimental results are shown in Figure 7.

As illustrated in Figure 8, the cut-out rate decreases as $E_{\max}$ increases overall. When $E_{\max}$ is large, some gestures with low knock strength will be recognized as quiet periods incorrectly. That resulted in a situation of less cut-out, and the cut-out rate is less than 1.

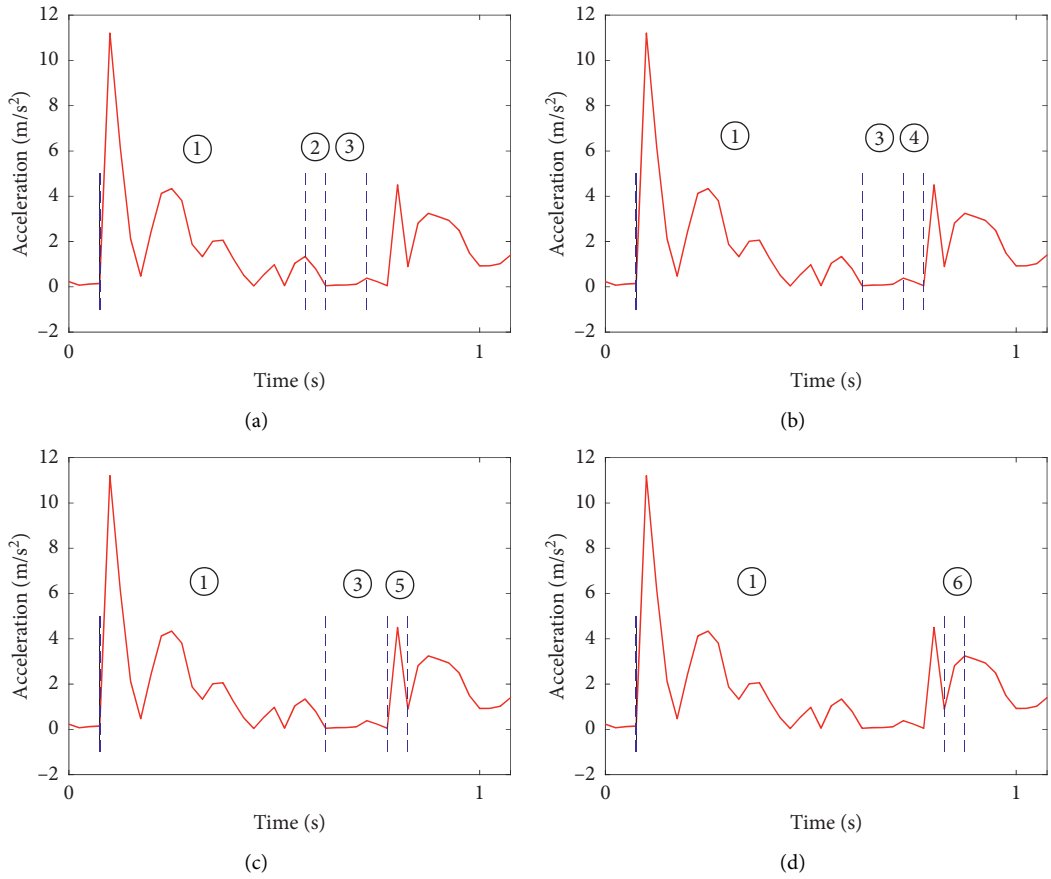

Figure 7: Schematic diagram of the merge algorithm. (a) $P = [1, 1, 0]$. (b) $P = [1, 0, 0]$. (c) $P = [1, 0, 1]$. (d) $P = [1, 1]$.

In the scenario of handheld, we can see the reasonable range of m is 0–7. However, the reasonable range of m is 0–4 in the scenario of the desktop. When a volunteer held the phone, a small shaking of the hand will cause continuous small fluctuations in the accelerometer signal. To discriminate between fluctuations caused by a handshake and those caused by knock gestures, $E_{\max}$ needs to be larger. In order to adapt to different scenarios, the setting of $E_{\max}$ is studied in the next subsection.

3.3.4. Setting of E_{max}. If an initial quiet period is detected, $E_{\max}$ is set as in the following equation:

$$E_{\max} = k \cdot N \cdot E_{\text{avgb}}. \tag{8}$$

Here, k is the linear adjustment coefficient, N is the current window size of SW/SWAB, and E_{avgb} is the average error of the initial quiet period. In this way, the setting of $E_{\max}$ can be dynamically adjusted according to E_{avgb} and the current window size. This achieves the purpose of scene adaptation.

The influence of k value on the bit cutting is analysed. k varies in [0.001, 0.01, 0.1, 0.5, 1, 3, 5, 10]. The experimental results are shown in Figure 9.

As shown in Figure 9, a reasonable range of k tends to be the same in both scenarios. Scene adaptation is achieved to a certain degree by adaptively adjusting $E_{\max}$. For different scenarios, only parameter k needs to be determined. When k is small, it has little effect on the cut-out rate. When k exceeds a certain threshold, the cut-out rate decreases rapidly. A smaller k means a small cumulative error threshold. This results in more segments being cut out, but a good bit cut-out rate can also be obtained by the merge algorithm. In contrast, a larger k means a large cumulative error threshold. This leads to less cut-out, and the cut-out rate is less than 1. From Figure 9, we can know that the cut-out rate is better when k is less than or equal to 1.

3.3.5. Effectiveness of Bit Cutting. In this section, we evaluated the effectiveness of the proposed bit cutting algorithm. The online bit cutting process is compared to an offline process using Support Vector Machine (SVM) [20]. The offline process is as follows.

A heuristic algorithm is used to cut the gesture signal sequence into multiple quiet and fluctuating segments. Then, the signal segments that are correctly cut out will be used to train an SVM model. Two features are extracted for each sampling point in a signal segment, namely, the 3-axis synthetic acceleration and the synthetic acceleration difference between the current and previous sampling point. The label of the sample point is the category of its segment, which is a quite segment or a fluctuating segment. After the

Input: *Segment*, the signal segments produced by SW or SWAB
P, the corresponding fluctuation characteristic of segments in Segment
k, the count of segment in Segment
Output: *Segment, P, k* after merging process
/∗ the merge algorithm runs only there are more than 3 segments. ∗/;
if $k > 3$ **then**
  if $P[k-2] == P[k-1]$ **then**
    /∗ [0, 0, 0] ⟶ [0, 0], [0, 0, 1] ⟶ [0, 1] ∗/;
    /∗ [1, 1, 0] ⟶ [1, 0], [1, 1, 1] ⟶ [1, 1] ∗/;
    $Segment[k-2] \leftarrow Segment[k-1]$;
    remove $(k-1)^{th}$ item in *Segment*;
    $k \leftarrow k-1$;
  else if $[P[k-2], P[k-1], P[k]] == [0, 1, 0]$ **then**
    /∗ [0, 1, 0] ⟶ [0] ∗/;
    if *count of* $(k-1)^{th}$ *segment in Segment* < Cmin **then**
      $Segment[k-2] \leftarrow Segment[k]$;
      remove $(k-1)^{th}$ item in *Segment* and *P*;
      $k \leftarrow k-2$;
    end
  else if $[P[k-2], P[k-1], P[k]] == [1, 0, 1]$ **then**
    /∗ [1, 0, 1] ⟶ [1] ∗/;
    if *count of* $(k-1)^{th}$ *segment in Segment* $> C_{max}$ **then**
      $Segment[k-2] \leftarrow Segment[k]$;
      remove $(k-1)^{th}$ item in *Segment* and *P*;
      $k \leftarrow k-2$;
    end
  else
    doing nothing;
  end
end
Return *Segment, P, k*

Algorithm 2: The pseudocode of the merge algorithm.

Table 2: Parameters setting.

Parameters	Value
Smartphone	Huawei honor 8x
α	0.8
β	5
C_{max}	15
C_{min}	3

above processing, we can get an SVM model to predict the category of each sampling point. Finally, the sample points are merged into segments according to their category labels. A similar merge process as shown in Figure 6 is utilized in the offline process.

The SVM algorithm has a global view, which simplifies the classification problem. All data samples are labelled and the 10-fold cross-validation is used to obtain the average bit cut-out rate of the SVM algorithm. For the bit cutting process, E_{max} is set based on equation (7), and k is 0.5. The experimental results are shown in Figure 10. The online bit cutting process designed in this paper achieves a cut-out rate comparable to the offline SVM algorithm. This shows that the proposed bit cutting process is suitable for online cutting of binary gesture signals.

3.3.6. Comparison of Different Gestures. In this section, the proposed bit cutting algorithm is applied to knock, pitch, and flip gestures sequence. The cut-out rate and the bit completion time are compared for these three gestures. Except for β, the parameters setting is the same as that in Table 2. As discussed in Section 3.3.2, the coefficient β should be greater than 1. Here, β *varies* from 1 to 10. As shown in Figure 11, the bit cutting algorithm is effective for all three gesture sequences. When β is set to 3, 4, and 5, the cut-out rate of the three gesture sequences is close to 1. That means all signal segment is cut-out correctly. As β increases, some segments in a fluctuation period were marked as segments belonging to a quiet period incorrectly. That causes the cut-out rate of the flip gesture sequence to increase to about 1.2.

Next, we counted the length of all correctly cut-out signal segments and obtained the average completion time to express bit "0" and "1." As illustrated in Figure 12, the bit completion time of pitch and flip gestures are longer than the knock gesture. The single-knock action takes about 0.3 seconds on average to express the bit "0," while the pitch and flip actions take more than 0.5 seconds. The double-knock action takes about 0.65 seconds on average to express the bit "1," while the pitch and flip actions take more than 1.0 seconds. To issue the same command to a phone, the time spent using the knock gesture is only about half of the pitch

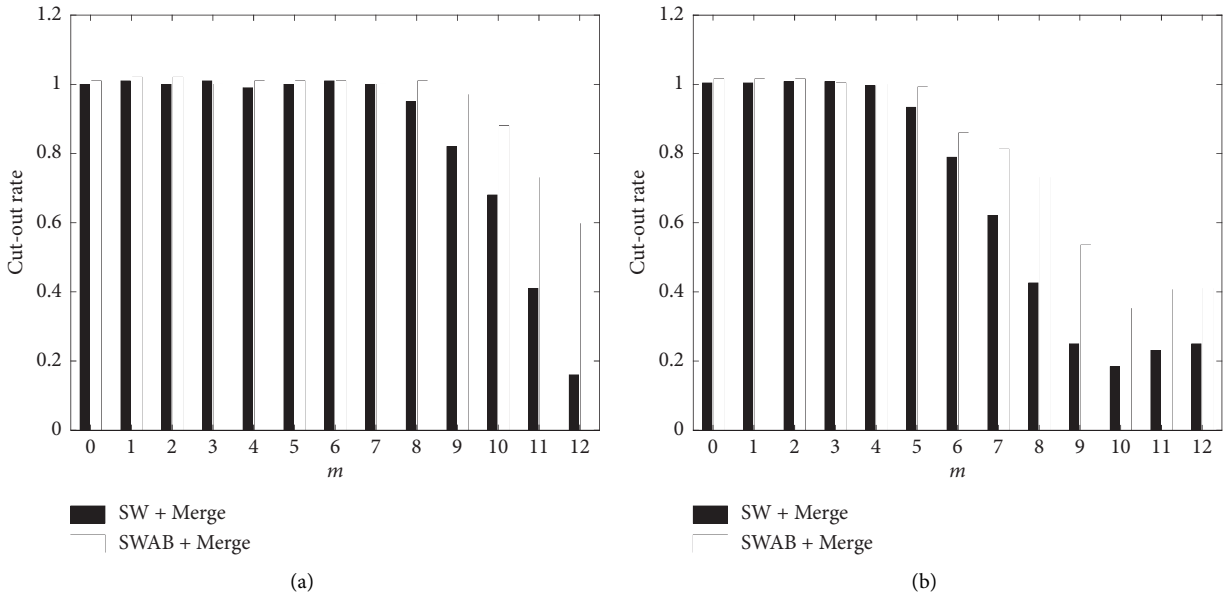

Figure 8: Comparison of cut-out rates in different scenarios. (a) Handheld. (b) Desktop.

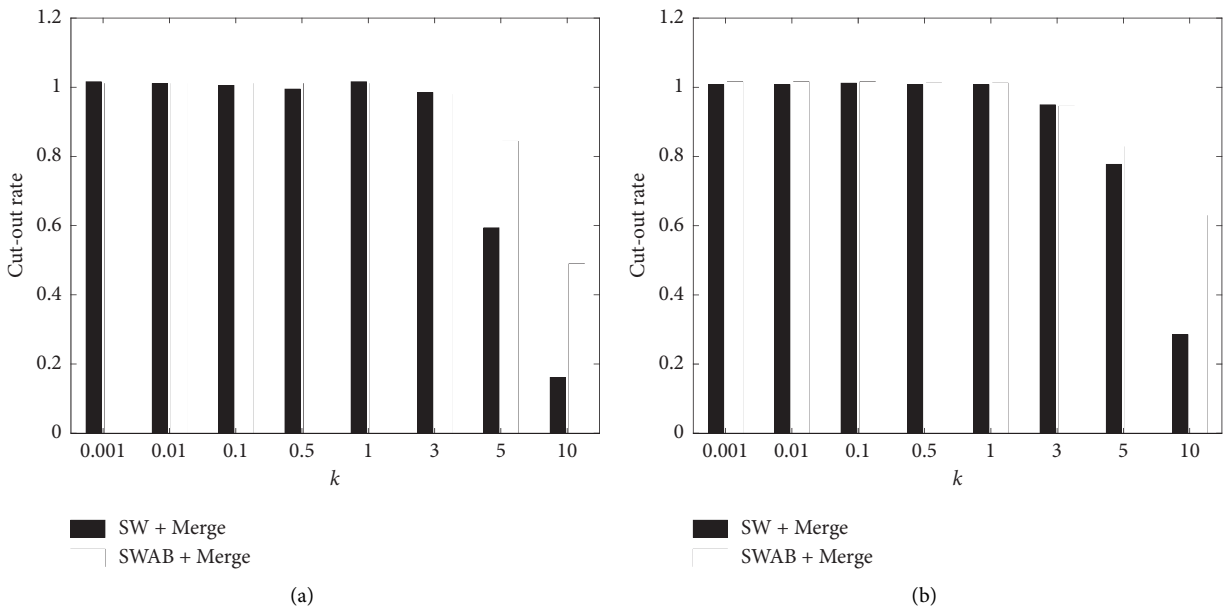

Figure 9: Comparison of cut-out rates for different k. (a) Handheld. (b) Desktop.

and flip gestures. Thus, the interaction efficiency of knock gesture outperforms the other two.

Since the proposed algorithm is better at cutting knock and pitch gesture sequences, how to recognize the cut-out signal segments of these two gestures to their binary meaning is studied in Section 4.

4. Binary Gesture Recognition

After bit cutting, a complete signal segment of a gesture sequence is obtained. To distinguish between single and double gesture action, the DTW, traditional machine learning, and BLSTM methods are exploited in this section.

4.1. DTW Method. Dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences, which may vary in length [21]. The temporal sequences of signature will be denoted as matrices like $S_{P\times Z}$, where P is the number of points in the cut-out signal segment and Z is the number of features extracted from each point. Here, the 3-axis raw acceleration data is used. As a result, the i^{th} point in the sequence is a 3-dimensional vector. In order to verify whether a sample ($S_{P\times Z}$) matches its corresponding template ($T_{Q\times Z}$), a dissimilarity score *dis* is computed between them based on the DTW algorithm. *dis* is a cumulative distance of the two gesture signal segments.

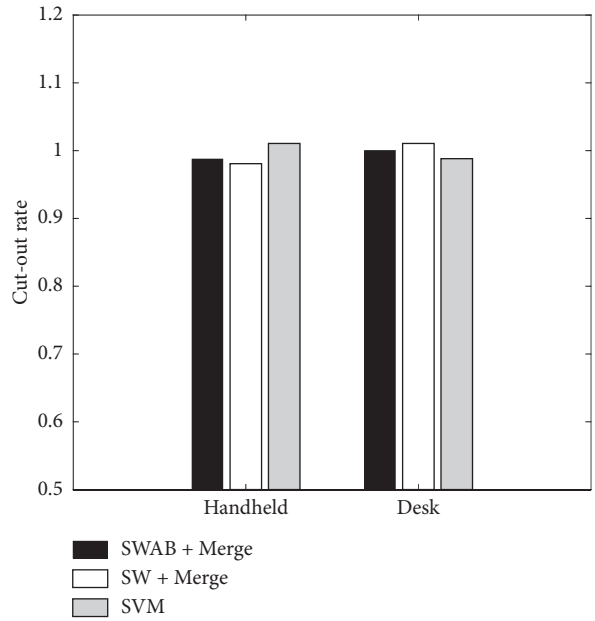

Figure 10: Comparison of cut-out rates of different algorithms.

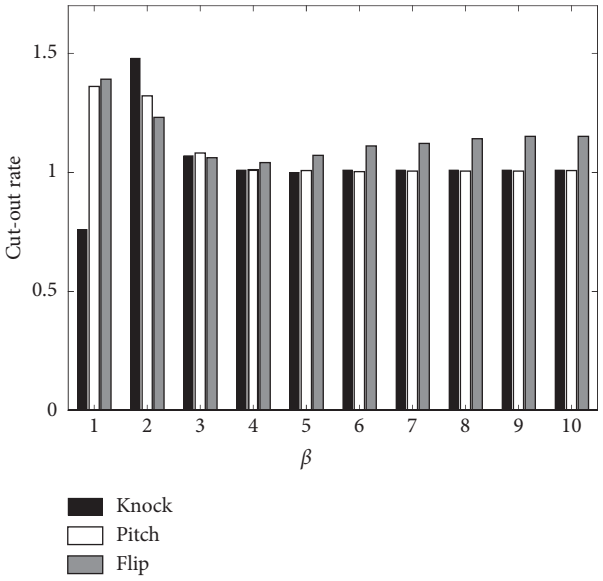

Figure 11: Cut-out rates for different β.

As seen in Table 1, a single-action and a double-action are defined in each gesture category. Therefore, a single-action signal segment and a double-action signal segment are manually selected for each volunteer as reference templates. When a signal segment is cut out, two dissimilarity scores are calculated between the segment and the two reference templates. The segment is classified as consistent with the template of a smaller dissimilarity score.

4.2. SVM Methods. Support Vector Machines (SVMs) are widely used for classification and regression tasks. Here, gesture recognition is treated as a binary classification problem. SVM constructs a hyperplane in a high-dimensional space to separate two class of gesture, single-action, and double-action gestures. We use LIBSVM as a classification algorithm and use the RBF kernel as a kernel function. Three features are extracted to construct a 3-dimensional feature vector for each gesture signal segment. They are the gesture size, gesture energy, and the first-order components of the signal after Discrete Cosine Transforms (DCT).

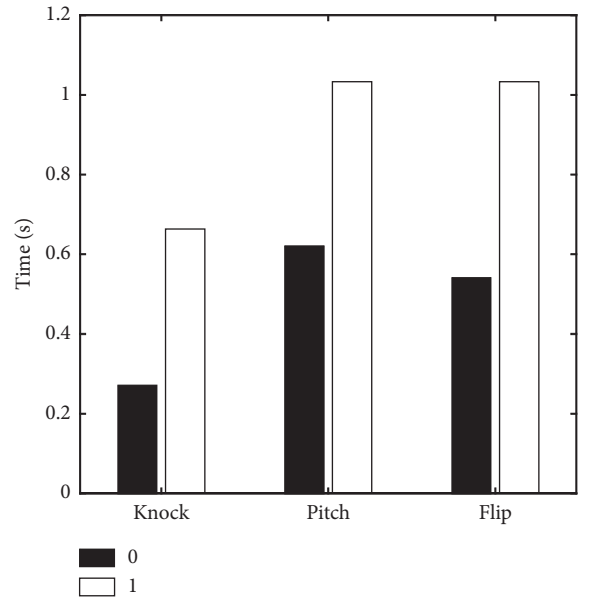

Figure 12: The average completion time for different binary gestures.

4.2.1. Gesture Size. The size of a gesture refers to the duration of the gesture. It is defined as the number of sampling points in a cut-out gesture segment. Obviously, a double-action gesture usually takes longer than a single-action gesture.

4.2.2. Gesture Energy. The energy consumption of an object's movement is closely related to its speed and acceleration. Bouten's research in recent years has proved that the absolute integral of the acceleration and angular velocity of an object's movement have a linear relationship with energy consumption [23]. This provides a theoretical basis for evaluating gestures' movements with an acceleration sensor. When the output signal is a digital signal, the following formula can be used to calculate the energy of a gesture:

$$E_n = \sum_{i=1}^{n} \left(|A_x| + |A_y| + |A_z| \right). \tag{9}$$

Among them, A_x, A_y, and A_z are the 3-axis values of the acceleration sensor. Since we have performed vector synthesis on the 3-axis data based on equation (1), the knock energy is defined as follows:

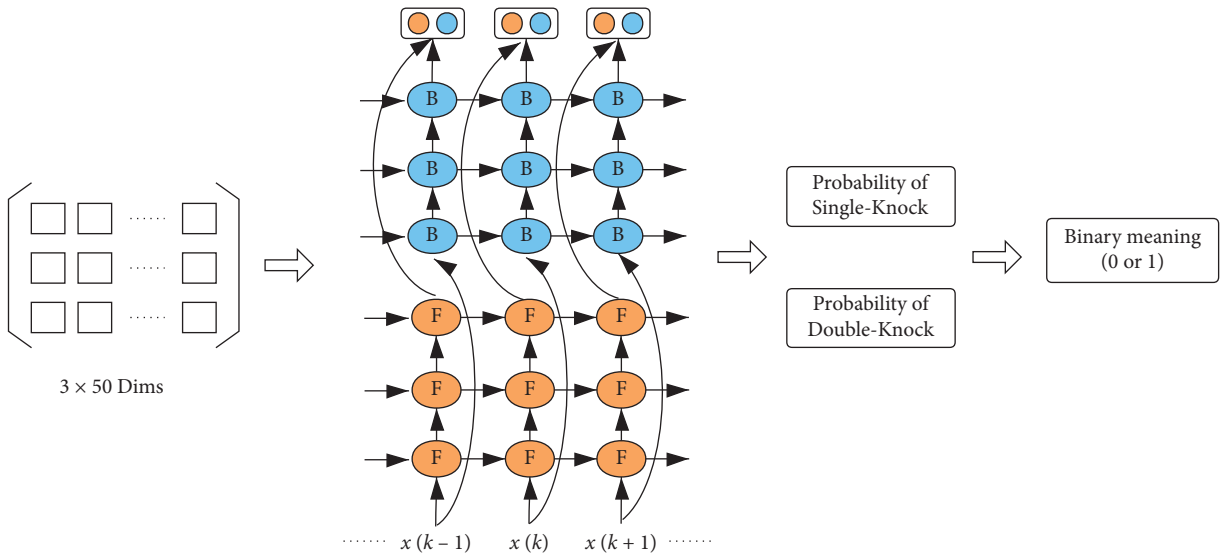

Figure 13: Knock gesture recognition using a 3-layer BLSTM model.

$$E_n = \sum_{i=1}^{n} \left(|A_i| \right). \tag{10}$$

4.2.3. DCT. A one-dimensional DCT is performed on a knock gesture signal segment. DCT converts the gesture signal into a set of frequencies. The first frequency in the set is the most meaningful. Therefore, the first-order components of the signal after DCT is selected as one of the features.

Two other machine learning methods, including Naive Bayes and Decision Tree, are also used to recognize binary gestures for comparison purposes [22]. These algorithms use the same feature vector as SVM for classification.

4.3. BLSTM Method. BLSTM is an extension of traditional LSTM that can improve model performance on sequence classification problems [24]. A 3-layer BLSTM architecture is used to model the gesture data in this paper. The process of knock gesture is illustrated in Figure 13.

Since the maximum duration of a knock gesture does not exceed 1 second, and the sample frequency is set to 50 Hz, up to 50 samples are captured for a cut-out gesture segment. Instead of using the synthesis and filtered values, the 3-axis raw acceleration data is used. Thus, a matrix of 3 by 50 is fed into the BLSTM model. The forward and the backward output are concentrated together to generate the probability for two knock gestures. The gesture of higher probability is selected as the predicted result, i.e., 0 for a single-knock and 1 for a double-knock.

The parameters of the BLSTM model are shown in Table 3. The same model is also applied to recognize the pitch gesture. Because the bit completion time of pitch gestures is longer than knock gestures, a matrix of 3 by 100 is used as input to the model.

Table 3: Parameters of the BLSTM model.

Parameters	Value
Learning rate	0.001
Batch size	32
Optimizer	Adam
Loss	Categorical cross entropy
Epochs	100
Dropout	0.5

4.4. Experimental Results. A metric defined in equation (11) is used to evaluate the recognition accuracy.

$$M_{\text{acc}} = \frac{(TP + TN)}{(P + N)}. \tag{11}$$

Here, P is the number of segments that belong to a single-action gesture. N is the number of segments that belong to a double-action gesture. TP is the number that is predicted to be a single-action gesture. And TN is the number that is predicted to be a double-action gesture.

A metric defined in the following equation is used to evaluate the imbalance of recognition for the two action gestures.

$$M_{\text{bal}} = \frac{M_{\text{acc}}(0)}{M_{\text{acc}}(1)}. \tag{12}$$

Here, $M_{\text{acc}}(0)$ represents the recognition accuracy of single-action gesture, and $M_{\text{acc}}(1)$ represents the recognition accuracy of double-action gesture. M_{bal} is expected to be around 1, which means the recognition accuracy for the two binary actions is similar. Moreover, the metrics of micro F1 and recall are also evaluated.

The experimental results are shown in Figure 14. All gesture recognition methods have achieved recognition accuracy of more than 90%. The BLSTM method outperformed the other algorithms and achieved the highest recognition accuracy of 98%. The metric of micro F1 also

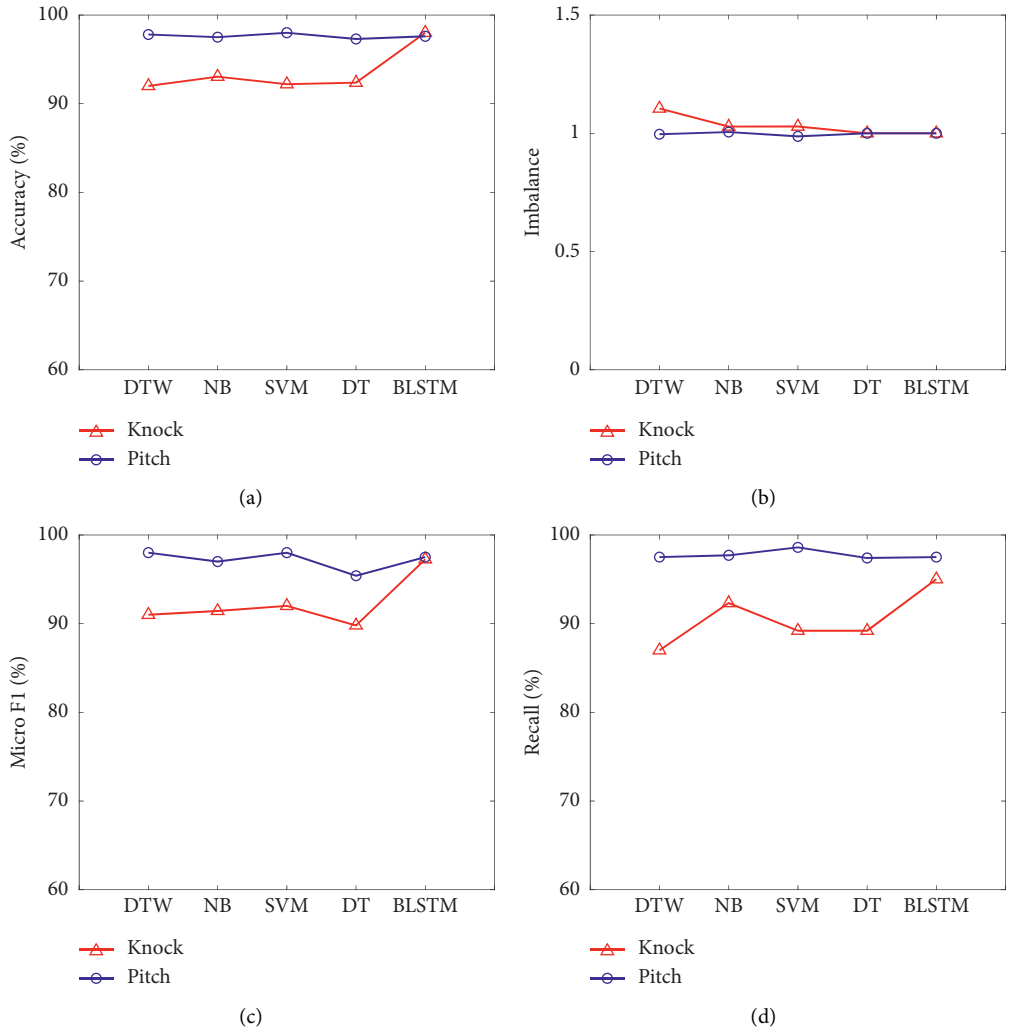

Figure 14: Gesture recognition results. (a) Recognition accuracy. (b) Recognition imbalance. (c) Micro F1. (d) Recall.

indicated that DTW, NB, SVM, and BLSTM methods can recognize the cut-out signal segments into its binary meaning in high accuracy.

From the perspective of recognition imbalance, the recognition accuracy of DTW for single-knock gesture is higher than a double-knock gesture, making its M_{bal} greater than 1. However, the recognition accuracy of DTW for single-pitch and double-pitch gesture are close. The imbalance of other recognition methods is good, and the experimental results are close to 1, among which the BLSTM method is the best.

As seen from Figure 14, these methods achieve superior performance in recognizing pitch gestures than knock gestures. An important reason is that the completion time of the pitch gesture is longer than that of the knock gesture. Compared with knock gestures, the difference in the feature of gesture duration between a single-pitch and a double-pitch is more significant; The same is true with regard to the energy difference between the two gesture types.

The experimental results show the effectiveness of using knock and pitch gestures for interaction. Only two simple gestures are required. High recognition accuracy can be achieved for both gestures and avoid imbalance problem at the same time.

5. Prototype Application and Discussion

In general, the knock gesture is simple and clear, which is convenient for users to operate. The bit completion time of knock gesture is shorter than that of pitch gesture. In addition, the knock gesture signal segment can be recognized in high recognition accuracy. Therefore, the knock gesture is selected to implement interaction between human and mobile applications.

In this prototype application, users utilize the single-knock and double-knock gestures to command the applications in an Android smartphone to send SMS messages. The prototype application is useful in some scenarios where private interaction is required; the user cannot speak or cannot light up the screen, which may attract others' attention. The binary knock gestures are inconspicuous and can be used to send text messages covertly.

As the BLSTM model performs best both in recognition accuracy and recognition imbalance, it is selected to

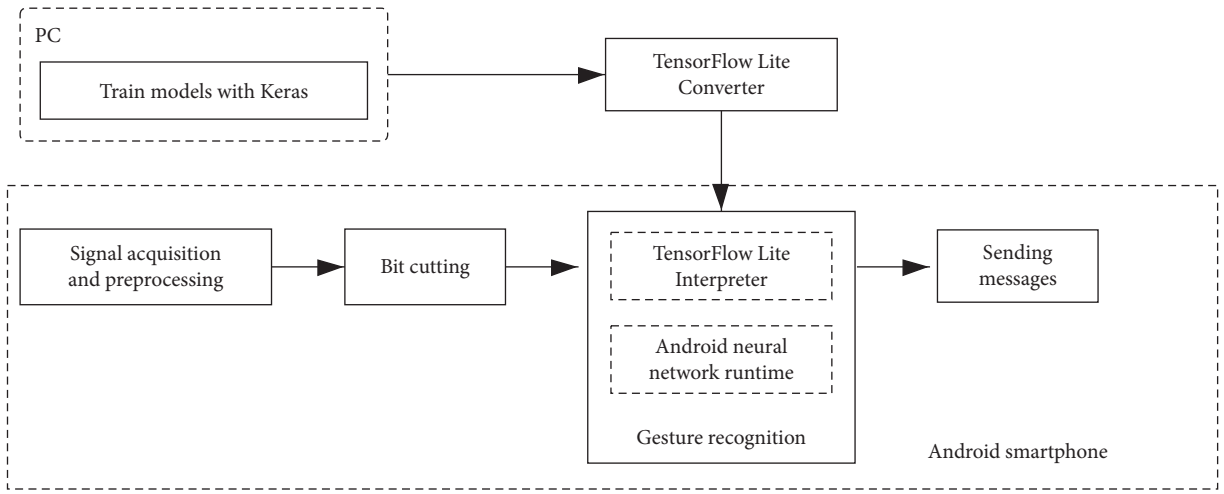

Figure 15: The development process of the prototype application.

implement in our prototype application. The framework of TensorFlow Lite [25] is used to integrate BLSTM into smartphones. The development process of the prototype application is shown in Figure 15.

A 3-layer BLSTM model is trained with Keras [26] in PC. Then it is converted into a TensorFlow Lite model using the TensorFlow Lite converter. The TensorFlow Lite interpreter executes the model on smartphones to make predictions based on input accelerometer data. If the predicted binary sequence is matched with the preset command, the application automatically sends a short message to the corresponding phone number.

The prototype application is tested with four different scenarios on how people interact with a smartphone. In the last three scenarios, users interacted in an eyes-free manner. (1) Normal: a person is sitting on a chair and holding a mobile phone on a desk. (2) Eyes-free: a person is sitting on a chair and holding a mobile phone beneath a desk. (3) Covert: a person is standing still with the phone in his pants pocket. (4) Walking: a person is walking at a constant speed with the phone in his pants pocket.

The metrics of the cut-out rate and accuracy are evaluated. Figure 16 illustrated the experimental results. In the scenarios of normal, eyes-free, and covert, they all achieved a cut-out rate close to 1. Most of the bit signal segments are split out from the gesture signal sequences successfully. Meanwhile, these bits are recognized with high accuracy. However, when people are moving, it greatly affects the cut-out effect. The proposed scheme is more suitable for interacting with smartphones when people are in a stationary state.

In addition to the above interaction cases, binary gestures can be used as a supplementary input modality for many scenarios. For example, it can be used as an interaction method of a blind assistive system. In [13], a blind person can establish a voice call to a predefined number using voice command. However, they got some error as a sound wave is affected much for noise and humidity. In such an environment, the blind person can use binary gestures instead of voice. In [28], a set of hand

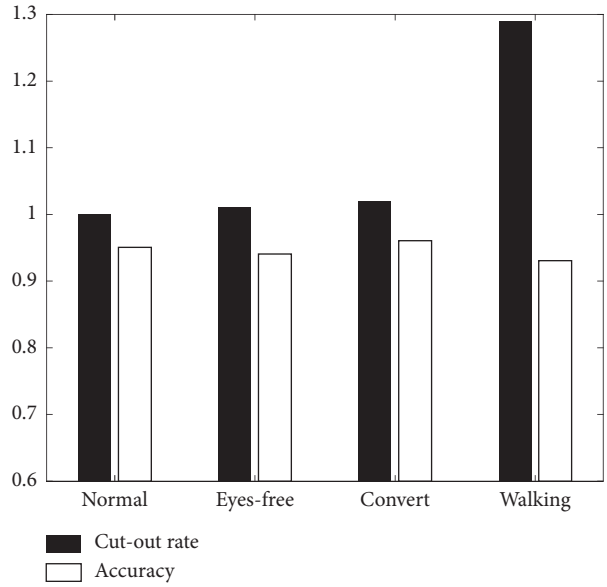

Figure 16: The development process of the prototype application.

gestures is proposed to control the smart lighting system. These vision-based hand gestures are more complex and difficult in terms of their recognition. Under such circumstances, smartphones can be adapted as a user interaction interface. By encoding these tasks into a binary command set, users can control the lighting system by binary motion-based gestures.

6. Conclusion

A novel user-smartphone interaction scheme using binary gestures is proposed in this paper. Firstly, four kinds of binary gestures are evaluated. The gestures of flip, pitch, and knock are selected as candidate interaction gestures. Then, the gesture extraction process is investigated in detail. The accelerometer signal is captured and preprocessed. An online signal cutting and merging

algorithm is designed to extract the independent gesture signal segment from the binary gesture sequence. Experiments show that the proposed method outperforms its counterparts in cutting knock and pitch gesture sequences. Next, five algorithms, including DTW, Naive Bayes, Decision Tree, Support Vector Machine, and BLSTM, are exploited to recognize the flip and knock gesture. Finally, an Android application is developed based on the binary command channel using knock gestures.

The proposed scheme only requires two meta gestures. And rich information can be expressed through the permutation and combination of the two gestures. As the binary gestures are much simpler than traditional gestures, our method achieves high recognition accuracy and avoids the imbalance problem.

The proposed scheme provides an alternative for eyes-free interaction scenarios. It is applicable to visually disabled user-smartphone interactions, distracted interaction, and covert operations.

As future work, we will enhance the ability to express more complex human-smartphone interaction commands.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the following foundations: the National Natural Science Foundation of China (Grant no. 31872847), the Natural Science Foundation of Shaanxi Province, China (Grant nos. 2019JM-244), the Industry-University Collaborative Education Program granted by the Ministry of Education of China (201902323022 and 201802217002), Weifang Science and Technology Development Plan (Grant no. 2017GX021), and Shandong University Scientific Research Development Plan (Grant no. J17KB183).

References

[1] K. Katsuragawa, A. Kamal, and Q. F. Liu, "Bi-Level thresholding: analyzing the effect of repeated errors in gesture input," *ACM Transactions on Interactive Intelligent Systems*, vol. 9, no. 2-3, pp. 1–30, 2019.

[2] B. Chaudhuri, L. Perlmutter, and J. Petelka, "GestureCalc: an eyes-free calculator for touch screens," in *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 112–123, Pittsburgh PA USA, October 2019.

[3] S. Azenkot, J. O. Wobbrock, S. Prasain, and R. E. Ladner, "Input finger detection for nonvisual touch screen text entry in Perkinput," in *Proceedings of Graphics Interface 2012*, pp. 121–129, Toronto, Canada, May 2012.

[4] A. Vtyurina, A. Fourney, and M. R. Morris, "Bridging screen readers and voice assistants for enhanced eyes-free web search," in *Proceedings of the World Wide Web Conference*, pp. 3590–3594, San Francisco, CA, USA, May 2019.

[5] J. Ruiz, Y. Li, and E. Lank, "User-defined motion gestures for mobile interaction," in *Proceedings of the International Conference on Human Factors in Computing Systems*, pp. 197–206, Vancouver, Canada, May 2011.

[6] M. Negulescu, J. Ruiz, Y. Li, and E. Lank, "Tap, swipe, or move: attentional demands for distracted smartphone input," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 173–180, Rome, Italy, May 2012.

[7] M. S. R. Tanveer, M. M. A. Hashem, and M. K. Hossain, "Android assistant EyeMate for blind and blind tracker," in *Proceedings of 2015 18th International Conference on Computer and Information Technology*, pp. 266–271, Istanbul, Turkey, September 2015.

[8] S. J. Castellucci, I. S. MacKenzie, M. Misra, L. Pandey, and A. S. Arif, "TiltWriter: design and evaluation of a no-touch tilt-based text entry method for handheld devices," in *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*, pp. 1–8, Pisa, Italy, Italy 2019.

[9] T. Vuletic, A. Duffy, L. Hay, C. McTeague, G. Campbell, and M. Grealy, "Systematic literature review of hand gestures used in human computer interaction interfaces," *International Journal of Human-Computer Studies*, vol. 129, no. 9, pp. 74–94, 2019.

[10] F. Hong, M. Wei, S. You, Y. Feng, and Z. Guo, "Waving authentication: your smartphone authenticate you on motion gesture," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 263–266, Seoul, South Korea, April 2015.

[11] Y. Jhang, Y. Chu, and T. Tai, "Sensor based dynamic hand gesture recognition by PairNet," in *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, Chengdu, China, pp. 994–1001, December 2019.

[12] A. Kamal, Y. Li, and E. Lank, "Teaching motion gestures via recognizer feedback," in *Proceedings of the 19th international conference on Intelligent User Interfaces*, pp. 73–82, Los Angeles, CA, USA, March 2014.

[13] J. Choi, K. Song, and S. Lee, "Enabling a gesture-based numeric input on mobile phones," in *Proceedings of 2011 IEEE International Conference on Consumer Electronics*, pp. 151-152, Las Vegas, NV USA, January 2011.

[14] S. S. A. Shimon, S. Morrison-Smith, N. John, G. Fahimi, and J. Ruiz, "Exploring user-defined back-of-device gestures for mobile devices," in *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 227–232, Copenhagen Denmark, August 2015.

[15] P. Mittal and N. Singh, "Speech based command and control system for mobile phones: issues and challenges," in *2016 Second International Conference on Computational Intelligence & Communication Technology*, pp. 729–732, Ghaziabad, India, February 2016.

[16] Motion sensors, https://developer.android.com/guide/topics/sensors/sensors_motion, 2020.

[17] Sensors overview, http://developer.android.com/guide/topics/sensors/sensors_overview.html, 2020.

[18] A. Hatori and H. Kobayashi, "A preliminary study of iot-device control using gestures recognition," in *Proceedings of*

56th Annual Conference of the Society of Instrument and Control Engineers of Japan, pp. 976–979, Kanazawa, Japan, September 2017.

[19] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, IEEE, San Jose, CA, USA, pp. 289–296, December 2001.

[20] R. Kumar and P. Singhal, "Review on offline Signature verification by SVM," *International Research Journal. Engineering and Technology*, vol. 4, no. 6, pp. 1771–1773, 2017.

[21] D. Kajiwara and K. Murao, "Gesture recognition method with acceleration data weighted by sEMG," in *Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and 2019 ACM International Symposium on Wearable Computers*, pp. 741–745, New York, NY, USA, September 2019.

[22] F. Pedregosa, G. Varoquaux, and A. Gramfort, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 10, pp. 2825–2830, 2011.

[23] C. V. C. Bouten, K. T. M. Koekkoek, M. Verduin, R. Kodde, and J. D. Janssen, "A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 3, pp. 136–147, 1997.

[24] M.-C. Lee and S.-B. Cho, "A recurrent neural network with non-gesture rejection model for recognizing gestures with smartphone sensors," *Lecture Notes in Computer Science*, vol. 8251, pp. 40–46, 2013.

[25] A. Campoverde and G. Barros, "Detection and classification of urban actors through TensorFlow with an android device," *Advances in Intelligent Systems and Computing*, vol. 1099, pp. 167–181, 2019.

[26] N. Ketkar, "Introduction to keras," *in Deep Learning with Python*, Apress, Berkeley, CA, USA, 2017.

[27] Motion and position sensors, https://google-developer-training.github.io/android-developer-advanced-course-concepts/unit-1-expand-the-user-experience/lesson-3-sensors/3-2-c-motion-and-position-sensors/3-2-c-motion-and-position-sensors.html, 2020.

[28] D. Park, Y. S. Lee, and S. Song, "User centered gesture development for smart lighting," in *Proceedings of HCI Korea*, pp. 146–150, Seoul, South Korea, December 2016.