2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)

Gesture-enabled Remote Control for Healthcare

Hongyang Zhao* Shuangquan Wang* Gang Zhou* Daqing Zhang†
*Computer Science Department, College of William and Mary
†Institut Mines-Télécom/Télécom SudParis
Email: {hyzhao, swang10, gzhou}@cs.wm.edu, daqing.zhang@telecom-sudparis.eu

***Abstract*—In recent years, wearable sensor-based gesture recognition is proliferating in the field of healthcare. It could be used to enable remote control of medical devices, contactless navigation of X-ray display and Magnetic Resonance Imaging (MRI), and largely enhance patients' daily living capabilities. However, even though a few commercial or prototype devices are available for wearable gesture recognition, none of them provides a combination of (1) fully open API for various healthcare application development, (2) appropriate form factor for comfortable daily wear, and (3) affordable cost for large scale adoption. In addition, the existing gesture recognition algorithms are mainly designed for discrete gestures. Accurate recognition of continuous gestures is still a significant challenge, which prevents the wide usage of existing wearable gesture recognition technology. In this paper, we present Gemote, a smart wristband-based hardware/software platform for gesture recognition and remote control. Due to its affordability, small size, and comfortable profile, Gemote is an attractive option for mass consumption. Gemote provides full open API access for third party research and application development. In addition, it employs a novel continuous gesture segmentation and recognition algorithm, which accurately and automatically separates hand movements into segments, and merges adjacent segments if needed, so that each gesture only exists in one segment. Experiments with human subjects show that the recognition accuracy is 99.4% when users perform gestures discretely, and 94.6% when users perform gestures continuously.**

I. Introduction

Healthcare is one important application scenario of gesture recognition technology. Lots of researchers and companies pay much attention to this area. According to the report published by MarketsandMarkets, the Healthcare application is expected to emerge as a significant market for gesture recognition technologies over the next five years [1]. In medicine, the ability of touch-free motion sensing input technology is particularly useful, where it can reduce the risk of contamination and is beneficial to both patients and their caregivers. For example, surgeons may benefit from touch-free gesture control, since it allows them to avoid interaction with non-sterile surfaces of the devices in use and hence to reduce the risk of infection. With the help of gesture control, the surgeons can manipulate the view of X-ray and MRI imagery, take notes of important information by writing in the air, and use hand gesture as commands to instruct robotic mechanism to perform complex surgical procedures. Wachs et al. [2] have developed a hand-gesture recognition system that enables doctors to manipulate digital images during medical procedures using hand gestures instead of touch screens or computer keyboards. In their system, a Canon VC-C4 camera and a Matrox Standard II video-capturing device are used for gesture tracking and recognition. The system has been tested during a neurosurgical brain biopsy at Washington Hospital Center.

Gesture recognition technology in healthcare can be mainly divided into two categories: computer-vision based gesture recognition and wearable sensor-based gesture recognition. The system developed by Wachs et al. [2] is an example of computer-vision based gesture recognition system. Though the system was tested in real-world scenarios, there still exists some disadvantages. It is expensive, needs color calibration before each use, and is highly influenced by lighting environment. Compared with computer-vision based recognition, wearable sensor-based gesture recognition technology is low cost, low power, requires only lightweight processing, no color calibration in advance, no violation of the privacy of the users, and is not interfered by lighting environment. Several wearable systems with gesture recognition technology have been proposed for healthcare application scenarios, e.g., upper limb gesture recognition for stroke patients [3] and for patients with chronic heart failure [4], glove-based sign language recognition for speech impaired patients, and for physical rehabilitation [5]. However, most wearable healthcare devices do not fit healthcare application scenarios well. There are mainly three problems in current wearable healthcare systems. (1) Not comfortable to wear. Many prototypes are too big which can not be used in reality. (2) No open Application Programming Interface (API). Most wearable healthcare prototypes do not open their API to public. Other developers cannot build applications based on their prototype. (3) Too expensive. Some wearable healthcare systems are quite expensive, e.g., a E4 healthcare monitoring wristband charges for $1690 with open API [6]. Additionally, most of gesture recognition prototypes can only recognize hand gestures one by one. Retrieving the meaningful gesture segments from continuous stream of sensor data is difficult for most gesture recognition prototypes [7].

To answer these problems, we address two research questions: (1) How does one design a hardware platform for gesture recognition and remote control, which is comfortable to wear, with open API, and at an affordable price? (2) How does one retrieve and recognize hand gestures from a continuous sequence of hand movements?

In this paper, we present Gemote, a wristband-based gesture recognition and remote control platform. The hardware platform integrates an accelerometer, gyroscope and compass sensor, providing powerful sensing capability for gesture recognition. We open our data sensing and gesture recognition

978-1-5090-4722-2/17 $31.00 © 2017 IEEE
DOI 10.1109/CHASE.2017.123

APIs to the public, so that developers and researchers can build their applications or carry out research based on our wristband platform. Because we only integrate hardware components that are necessary for gesture recognition, our wristband is small, comfortable, and affordable. We propose is a novel, lightweight, and high-precision continuous gesture segmentation and recognition algorithm. First, we separate data from a sequence of hand movements into meaningful segments. Next, nearby segments are merged based on gesture continuity and gesture completeness. The noise segments are then filtered out so that each segment contains one single gesture. Finally, we extract features from the acceleration and gyroscope data, and apply the Hidden Markov Model to recognize the gesture for each segment.

We summarize our contributions as follows:

1) We present Gemote, a wristband-based platform for gesture control. We design our platform with the consideration of user experience and practicality. It is comfortable to wear, with open API, and at an affordable price.
2) We present a continuous gesture segmentation and recognition framework. We propose a lightweight, and effective data segmentation mechanism to segment potential hand gestures from a sequence of hand movements. Then, we apply Hidden Markov Model recognize hand gestures.
3) Our experiment results show that our system can recognize hand gesture with 99.4% accuracy when users perform gestures discretely. When users perform gestures continuously, our system can segment hand gesture with 98.8% accuracy and recognize hand gesture with 94.6% accuracy.

The remainder of this paper is organized as follows. First, we discuss the related work in Section II. Then, we introduce the system framework in Section III. The design of hardware platform is introduced in Section IV. We present our continuous gesture segmentation and recognition algorithm in Section V. In Section VI, we evaluate the system performance. Finally, we draw our conclusion in Section VII.

II. Related Work

A. wristband-based platform

There have already been many wristband-based gesture recognition platforms. Some are developed by researchers in the university, while others are commercial products in the market. There are some common problems in current gesture recognition platforms. For example, most of current platforms do not open their API to the public [8][9][10][11], so they can not benefit other researchers and developers. Additionally, most platforms are too large to wear on wrist [10] [12] [9]. Some researchers just attach one smart phone on their wrist, which is inconvenient for daily wearing [13]. In the market, several wearable devices provide open API and are comfortable to wear, such as smart watch [14] and E4 healthcare monitoring wristband [6]. However, these products are either not targeted at healthcare, or too expensive. Table I shows the comparison of wristband-based gesture recognition platforms. From the table, we find that the platforms developed by researchers [8][9][10][11][12] usually do not provide open API, and are awkward to wear. Products on the market [6][14] are quite expensive, e.g., the price of Motorola Moto 360 (2nd Gen.) is over $300 and the price of E4 wristband is $1690. The motion sensors inside these platforms usually provide a high frequency sampling rate, e.g., Motorola Moto 360 (2nd Gen.) uses InvenSense MPU-6050 motion sensor which provides up to 1 kHz sampling rate for accelerometer and 8 kHz for gyroscope [15]. However, due to operating system and power requirement, the sampling rate for smart watch is limited to a lower frequency, e.g., 50 Hz [14]. This greatly limits the gesture recognition and motion sensing study in healthcare. Therefore, we are motivated to develop one wristband-based platform for gesture recognition and control for healthcare, which provides open API access, comfortable to wear, and at an affordable price.

TABLE I
Comparison of wristband-based gesture recognition platform

Platform	Open API	Wearable	Affordable price
Dong et al. [8]	×	×	×
Junker et al. [9]	×	×	√
eWatch [10]	×	×	√
RisQ [11]	×	√	√
E-Gesture [12]	√	×	√
E4 [6]	√	√	×
Moto 360 (2nd Gen.) [14]	√	√	×
Gemote	√	√	√

B. gesture recognition

Recently, smart wristband-based gesture recognition has been studied in mobile and pervasive computing. Various approaches dealing with the recognition of gestures or events have been presented. RisQ applies motion sensors on the wristband to recognize smoking gestures [11]. Xu et al. classify hand/finger gestures and written characters from smart watch motion sensor data [16]. Bite Counter utilizes a watch-like device with a gyroscope to detect and record when an individual takes a bite of food [17].

To recognize hand gestures, the first step is to extract potential gesture samples from a sequence of hand movements. A simple way to do this is to wear an external button on their fingers or hold it in their hand, and press this button to explicitly indicate the start and end of gestures [18][13]. In order to do this, users must wear an external button on their fingers or hold it in their hand. Unlike a wristband, wearing a button all day is burdensome and unnatural, limiting the usability of a hand gesture system. Another way to do this is to segment gestures automatically. The motion data are automatically partitioned into non-overlapping, meaningful segments, such that each segment contains one complete gesture.

Compared with button-enabled segmentation, automatic gesture segmentation provides a natural user experience. Park et al. apply a threshold-based method to detect short pauses

at the start and end of gestures [12]. They assume that a gesture is triggered when the gyroscope reading is higher than a threshold, and this gesture ends when the gyroscope reading is lower than this threshold. The authors define eight discrete gestures. When users perform these eight defined gestures, the gyroscope readings are always big. However, there are still many gestures that contain some small gyroscope readings, in which case their system mistakenly divides these gestures into multiple segments. Parate et al. assume that the gestures begin and end at some rest positions [11]. They segment gestures by computing the spatio-temporal trajectory of the wrist using quaternion data and tracking rest positions. However, all the segmentation and recognition procedures in their system are implemented in smartphone. They do not demonstrate their system's feasibility and performance in resource-limited wearable devices. Other researchers propose some complex segmentation methods, such as sequence analysis [9] and probability calculation [19]. However, these methods require huge computational effort, which can not be effectively and efficiently implemented in resource-limited wristbands.

III. System Overview

Our system, Gemote, is comprised of a wristband hardware platform and a continuous gesture segmentation and recognition framework.

The hardware platform contains a 9-axis motion sensor MPU-9250 for data sensing (including a accelerometer, a gyroscope, and a compass sensor), and a powerful Cortex-M4 processor for data processing and gesture recognition. We open our API to the public, so that other researchers and developers can build their own applications and systems upon our platform. The design philosophy of Gemote is the open API, the comfort of wear, and affordability in price.

The framework of the continuous hand gesture segmentation and recognition is shown in Fig. 1. It contains three modules: Sensing, Data Segmentation, and Hand Gesture Recognition. In the Sensing module, accelerometer, gyroscope and compass sensor readings are collected from 9-axis Inertial Measurement Unit (IMU) in the wristband. The sampling rate is set to be 20Hz with a balanced consideration of recognition accuracy, computational cost, and power restriction in the wristband.

The Data Segmentation module segments potential gestures from a sequence of hand movements. To segment gestures from hand movements, we first apply a threshold-based method to detect the start and end of the sequence of hand movements. As found in previous work [11] [20], while performing a hand gesture, people start from a static position, and then end in another static position. Therefore, the gestures tend to lie between these static positions. We develop a novel gesture segmentation algorithm to detect these static positions, and thus separate the sequence of hand movements into multiple segments, where one gesture may lie in one segment or several adjacent segments. To avoid splitting a single gesture's data into multiple segments, two metrics are proposed as post-processing to merge the adjacent segments so that each gesture only lies in one segment. Finally, the Noise Segments Removal module is applied to remove segments with noise gestures.

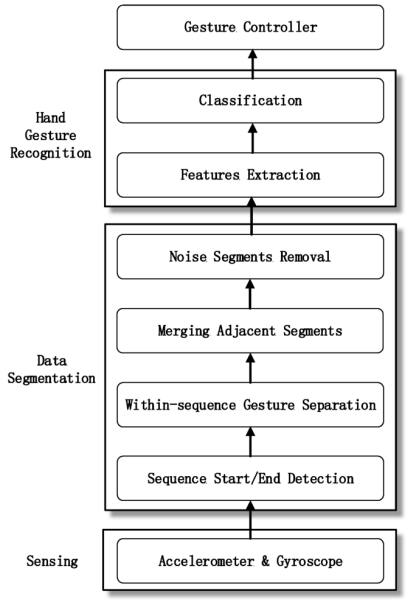

Fig. 1. Continuous Gesture Segmentation and Recognition Framework

The Recognition module receives segments from Data Segmentation module and classifies each segment as one predefined gesture or noise gesture. We apply the Hidden Markov Model to classify gestures because it has shown high recognition accuracy [9]. The recognized gesture can be utilized as a command to control the medical instruments or healthcare-related devices, such as a medical computer screen, X-ray or MRI navigation.

IV. Hardware Platform Design

Our smart wristband integrates a nRF52832 System on Chip (SoC) as the microcontroller unit. The nRF52832 SoC incorporates a powerful Cortex-M4 processor with 512kB flash and 64kB RAM, and a 2.4GHz transceiver that supports Bluetooth Low Energy (BLE) protocol. The Cortex-M4 processor provides strong computation capability for complex data processing and the gesture recognition algorithm. The BLE module enables our wristband to run for a long period of time, and communicate with other medical devices which also support BLE. In terms of motion sensing, a 9-axis motion sensor MPU-9250 is embedded in our smart wristband. The MPU-9250 is a System in Package that combines two chips: the MPU-6500, which contains a 3-axis gyroscope, a 3-axis accelerometer; and the AK8963, a 3-axis digital compass sensor. The sampling rates for the accelerometer, gyroscope, and compass sensor are up to 4 kHz, 1 kHz, and 8 Hz, which are configurable by the users. Compared to other smart wristbands in the market that integrate different types of sensors, we only focus on the motion sensors that are widely used by researchers. Our smart wristband is available online [21].

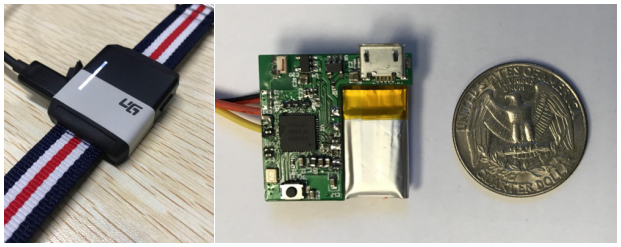

Fig. 2. Smart wristband

The capacity of the battery is a restriction for wearable devices. Our smart wristband is powered by a coin-size Li-Ion battery (3.7V, 90mAH). To account for the small capacity of the battery, we focus on the energy efficiency in our design. In the sensing and recognition state, our smart wristband consumes 10∼20mAH, depending on how many computing functionalities are used. In the sleep state, our smart wristband only consumes 1uAH, which greatly prolongs the battery lifespan. Compared with existing gesture recognition platforms, our wristband has three main advantages over other wristband prototypes:

Open API. We open our data sensing and gesture recognition APIs to smart phone developers. Developers can use our smart wristband as the data collector or gesture recognizer so that they can build their applications on top of our Gemote system. Table II shows the set of APIs currently supported by Gemote. It contains four classes. ConnectionManager is used to manage the Bluetooth connection to our smart wristband. DataSensingManager is used to collect the sensor data from our smart wristband, including acceleration, gyroscope, and compass data. GestureRecognitionManager is used to obtain the recognized gesture from our smart wristband. WidgetManager is used to get access to the widgets (battery level, button, LEDs) in our wristband.

Comfortable to wear. We carefully design our smart wristband to make it comfortable to wear. Fig. 2 shows the appearance and the printed circuit board (PCB) design of our smart wristband. The size of the PCB is very small with 26mm length and 25mm width, which is much smaller than previous wristband platforms [10] [12]. The size of the wristband shell is: 50mm length, 30mm width, and 11.7mm thickness, which is very easy to carry.

Affordable price. A practical, cheap, and open API platform is always strongly demanded by the researchers and developers in motion sensing. Some research groups use very large and expensive sensors ($2,000) [8], while others use generic smart watches (Motorola Moto 360 2nd Gen., over $300 [14]), or complex healthcare monitoring devices (E4 wristband: $1690 [6]). All these products are quite expensive for the motion sensing research and development. As our platform is designed for the motion sensing and gesture control, we only integrate necessary components into Gemote, e.g., one 9-axis motion sensor MPU9250 ($5), and one nRF52832 SoC ($5) that includes a Cortex-M4 processor and a BLE module. Therefore, our platform provides an affordable price for the customers to purchase and develop further.

V. Continuous Hand Gesture Recognition

The proposed continuous hand gesture recognition algorithm mainly contains three modules: Sensing, Data Segmentation, and Hand Gesture Recognition. The Sensing module collects the accelerometer and gyroscope sensor readings from IMU continuously, and outputs the sensor readings to the Data Segmentation module. The sampling rate of each sensor is set to be 20Hz with a balanced consideration of recognition accuracy, and computation and energy cost of the wearable device. The Data Segmentation module extracts individual gesture segments from a sequence of hand movements. The Hand Gesture Recognition module applies the HMM model to classify each individual gesture segment into one of the predefined gestures (Left, Right, Up, Down, Back&Forth, Clockwise, and Counterclockwise) or noise. The recognized gestures can be utilized to remotely control the medical instruments or healthcare related devices. In the following section, we first introduce the seven gestures defined in our system (Sec. V-A). Then, the data segmentation module (Sec. V-B) and the gesture recognition module (Sec. V-C) are presented in more detail.

A. Gesture Definition

There has been substantial research on gesture recognition. Some work defines gestures according to application scenarios, such as gestures in daily life [9], or repetitive motions in very specific activities [11], while others define gestures casually [12]. In this paper, we turn user's hand into a remote controller. We carefully design the hand gestures that best emulate a remote controller. Typically, a remote controller includes the following functions: left, right, up, down, select, play/pause, back. Therefore, we define the following seven gestures corresponding to these functions. At the beginning, the user extends his/her hand in front of his/her body. Then he/she moves towards a certain direction and moves back to the starting point again. We define the following gestures:

1) Left gesture: move left and then move back to the starting point
2) Right gesture: move right and then move back to the starting point
3) Up gesture: move up and then move back to the starting point
4) Down gesture: move down and then move back to the starting point
5) Back&Forth gesture: move to shoulder and then extend again to the starting point
6) Clockwise gesture: draw a clockwise circle
7) Counterclockwise gesture: draw an counterclockwise circle

These seven gestures are illustrated in Fig. 3. The defined hand gestures are very similar to the hand gestures defined by Wachs et al. [2]. Their gesture recognition system has been tested during a neurosurgical brain biopsy, which shows that

TABLE II
GEMOTE APIS

Class	Gemote APIs	Description
ConnectionManager	connect() disconnect()	connect to smart wristband by BLE disconnect from smart wristband
DataSensingManager	registerDataSensingListener(Sensors,CallbackListener, SamplingRate) startDataSensing() stopDataSensing() unregisterDataSensingListener()	register listener for given sensors with sampling rate start to collect sensor data from smart wristband stop sensor data collection unregister listener for sensors
GestureRecognitionManager	registerGestureRecognitionListener(CallbackListener) startGestureRecognition() stopGestureRecognition() unregisterGestureRecognitionListener()	register listener for gestures recognized by wristband start to recognize gesture stop gesture recognition unregister listener for gesture recognition
WidgetManager	getBatteryLevel() registerButtonListener(CallbackListener) setLED(ID, state) unregisterButtonListener()	get the battery level of wristband register listener for push-button event set LED with certain ID as certain state (on/off) unregister listener for push-button event

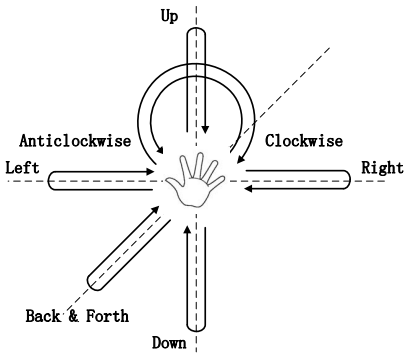

Fig. 3. Seven defined gestures for remote control

these gestures are suitable as a remote controller for healthcare applications. To be noticed, each defined gesture ends at the starting point. Therefore, each gesture is independent from the others. Users can continuously perform the same or different gestures, which enables continuous control.

B. Data Segmentation

A simple way to segment hand gestures from a sequence of hand movements is to use a hand-controlled button to clearly indicate the starting point and the end point of each individual gesture. However, in order to do so, the user must wear an external button on their fingers or hold it in their hands, which is obtrusive and burdensome. Another way is to segment gestures automatically. The motion data are automatically partitioned into non-overlapping, meaningful segments, such that each segment contains one complete gesture. Automatic segmenting a continuous sensor data stream faces a few challenges. First, the segmentation should extract exactly one entire hand gesture, neither more nor less than needed. Otherwise, the extracted segments contain non-gesture noises, or miss useful gesture information, which leads to inaccurate classification. In addition, when a user performs multiple continuous gestures, the segmentation should not split a single gesture into multiple segments, or put multiple gestures into a single segment. To deal with these challenges, a continuous gesture data segmentation method is proposed, which contains three main steps: sequence start and end points detection, within-sequence gesture separation, and merging adjacent segments.

1) Sequence start and end points detection: A lightweight threshold-based detection method is used to identify the start and end points of hand movements. To characterize a user's hand movement (HM), a detection metric is defined using the gyroscope sensor readings as

$$HM = \sqrt{Gyro_x^2 + Gyro_y^2 + Gyro_z^2}, \tag{1}$$

where $Gyro_x, Gyro_y, Gyro_z$ are the gyroscope readings of the X-axis, Y-axis, and Z-axis. When the user's hand is stationary, the HM is very close to zero. The faster a hand moves, the larger the HM is. When the HM is larger than a threshold, i.e. 50 degree/second, we regard it as the start point of hand movement. Once the HM is smaller than this threshold for a certain period of time, i.e. 400ms, we regard it as the end point of the hand movement. The time threshold is necessary as, in one single gesture, the HM may fall below this threshold occasionally, leading to unexpected splitting of this gesture [22][23]. Because the HM only keeps the magnitude of the vector sum of three axes and drops the direction information, this threshold-based detection method is independent of the device's orientation and therefore simplifies the gesture models.

Fig. 4 shows the gyroscope readings and the HM of one Left gesture and one Clockwise gesture. From Fig. 4(c), we see that the HM of the Left gesture falls below 50 degree/second at 1.6s. The Left gesture begins from moving left, then pauses, then moves right back to the original position. The low HM comes from the short pause in the Left gesture. The 400ms time frame prevents the Left gesture from being split into two separate hand movements.

Fig. 5 shows data processing for one continuous hand movement: raising hand horizontally→ performing Left gesture→ performing Back&Forth gesture→putting down hand. Raw gyroscope readings are shown in Fig. 5(a). The corresponding HM results for this hand movement sequence are shown in

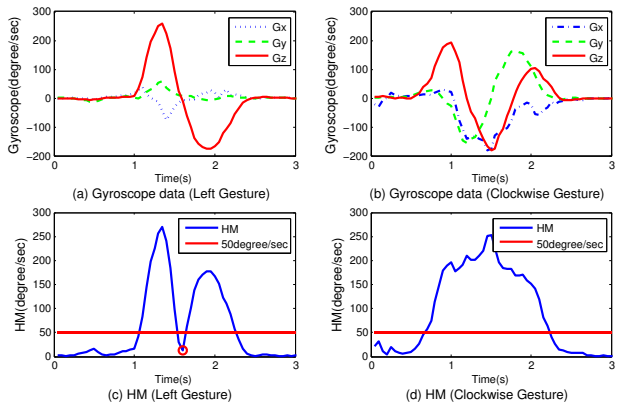

Fig. 4. HM based start and end points detection

Fig. 5(b).

2) Within-sequence gesture separation: After detecting the start and end points of one sequence of hand movements, we partition this sequence of hand movements into non-overlapping, meaningful segments so that one hand gesture lies in one or several consecutive segments.

The hand gestures we defined start from and end in static positions that users feel comfortable with and choose according to their own free will. At static positions, the magnitude of hand rotation is relatively small. Therefore, the HM valley is a good indicator of the connecting point between two neighboring hand gestures. We employ a valley detection algorithm with a sliding window to detect valleys of the HM in the hand movement data. We utilize valleys' positions as the segment points to partition the hand movement data into multiple and non-overlapping segments. Specifically, the sample at time $t(i)$ is a valley if it is smaller than all samples in the time window of $[t\,(i) - t_w/2, t\,(i) + t_w/2]$. Since the duration of one hand gesture is normally longer than 0.6 second, the window size t_w is set to be 0.6s.

With the window size threshold, the proposed algorithm is able to identify the HM valleys. However, sometimes there are a few false valleys which are not real switches of hand gestures. The reason is that the valley recognition algorithm only compares the HM magnitude in the time window, but does not take the absolute magnitude of the HM into consideration. A false HM valley may have large value, which indicates obvious and drastic rotation or movement. We collected the gyroscope data of a set of the continuous hand gestures which was conducted under supervision and the magnitude of HM valleys was carefully checked. The results show that, in general, the magnitude of the real HM valleys is less than 100 degree/second. Therefore, another condition, i.e. HM is less than 100 degree/second at the valleys, is added into the valley detection algorithm to eliminate the false valleys.

Fig. 5(c) shows the segmentation result based on the proposed valley detection algorithm. In total, five HM valleys are detected and six segments are generated. In this way, the raw gyroscope readings can be partitioned into six segments, as shown in Fig. 5(d). Each segment is one complete gesture or part of one complete gesture.

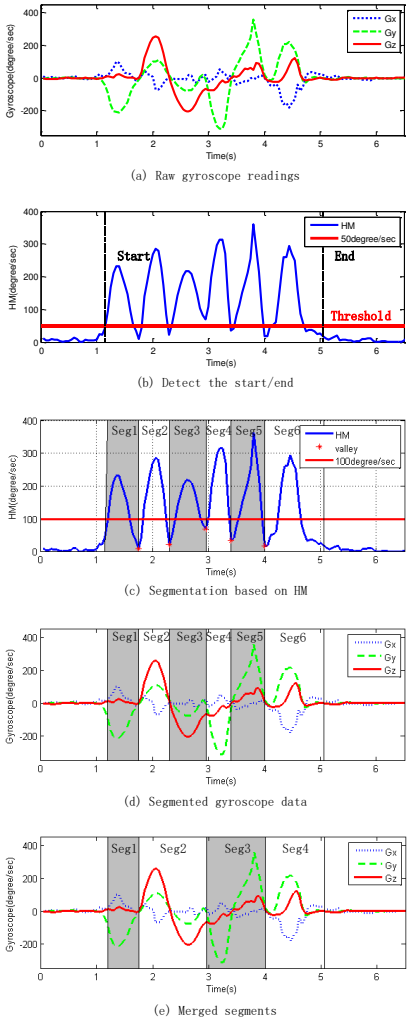

Fig. 5. Data processing for one continuous hand movement

One question here is why we use gyroscope readings in the proposed segmentation method, rather than the accelerometer readings. The accelerometer is mainly suitable for detection of speed change. Comparatively, gyroscope is more powerful for detection of orientation change. For hand movement during conducting hand gestures, the orientation change is more significant than the speed change. Thus, gyroscope-based segmentation method is more robust and accurate than accelerometer-based segmentation method, and can provide higher segmentation accuracy [12].

3) Merging adjacent segments: For one continuous gyroscope readings stream, after segmentation, we will get a series of partitioned segments. One gesture may lie in one segment or several continuous segments. In Fig. 5(d), segment 1 refers

to "raise hand horizontally" movement, segment 2 and 3 belong to Left gesture, segment 4 and 5 are from Back&Forth gesture, and segment 6 is "put down hand" movement. The Left gesture and the Back&Forth gesture are both partitioned into two segments. To merge the adjacent segments so that one gesture only lies in one segment, we propose two measurement metrics to decide whether two neighboring segments should be merged: Gesture Continuity metric and Gesture Completeness metric.

The **Gesture Continuity** metric measures the continuity of data in two neighboring segments. When two segments differ greatly in its signal shape at the connecting point, it is less likely that these two segments belong to the same single gesture. On the other hand, if two segments have similar slopes near the connecting point, these two segments may belong to one gesture. Based on this intuition, we compute the slopes near connecting points for each segments. If two slopes computed from two segments are similar, we say these two neighbor segments have similar shapes. Fig. 6 illustrates the computation of Gesture Continuity metric of a Right gesture:

For the sensor reading of each gyroscope axis, g_x, g_y and g_z (assume g_i), we do the following:

1) In g_i, we find the connecting point (t_1) between two segments $[t_0, t_1]$ and $[t_1, t_2]$, which is also a valley point in HM curve;
2) We extract the data points near connecting point (t_1) within one time window of 600ms, which is the same as the window size in valley detection algorithm. As the sampling rate is 20Hz, we pick 6 points before the connecting point (t_1) as t_a, t_b, t_c, t_d, t_e, t_f and 6 points after the connecting point (t_1) as t_g, t_h, t_i, t_j, t_k, t_l;
3) Twelve lines $\overline{t_a t_1}$, $\overline{t_b t_1}$,$\cdots$, $\overline{t_l t_1}$ are formed. For any 2 lines among the 12 lines, the angle between them is computed, and the maximum angle is defined as θ_{g_i};
4) We compute the weight w_{g_i} as the area size of the curve g_i in the time window $[t_0, t_2]$.

As there are three axes for gyroscope readings, we compute the three angles $(\theta_{g_i}, i \in \{x, y, z\})$ and three weights $(w_{g_i}, i \in \{x, y, z\})$ corresponding to the three axes. The Gesture Continuity (Con) at the connecting point t_1 is calculated as:

$$Con\,(t_1) = \frac{\sum\left(\mathrm{w}_{g_i} \cdot \theta_{g_i}\right)}{\sum \mathrm{w}_{g_i}} \tag{2}$$

The higher the angle θ_{g_i} is, the bigger difference the signal shape is, and the less likely for the two segments to belong to the same gesture. In addition, a larger gyroscope reading of one axis indicates greater hand rotation around this axis. Accordingly, we add w_{g_i} as weights to three axes. Con is the weighted version of the angle θ_{g_i}. It ranges from 0 degree to 180 degree. Small Con stands for similar signal shapes for two neighbor segments. We merge two segments if the Gesture Continuity metric Con is lower than a threshold.

In Fig. 6, the Right gesture is partitioned into two segments $[t_0, t_1]$ and $[t_1, t_2]$. From the figure, we see that angle θ_{g_z} is

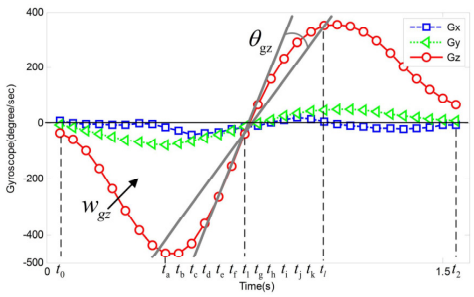

Fig. 6. Computation of Gesture Continuity and Gesture Completeness metric

quite small and weight w_{g_z} is very large. Therefore, the Con is small, and two segments $[t_0, t_1]$ and $[t_1, t_2]$ should be merged.

The **Gesture Completeness** metric measures the completeness of data in two neighboring segments if they belong to one complete gesture. To achieve continuous control, each gesture we chose to recognize starts from one user-chosen random position and ends with the same position. Even though the sensor readings vary during the procedure of a gesture, the sum of sensor readings should be close to zero for a complete gesture. Utilizing this gesture property, we calculate the Gesture Completeness metric as follows:

$$Com\,(t_1) = \frac{\left|\sum_{t_0}^{t_2} g_x\right| + \left|\sum_{t_0}^{t_2} g_y\right| + \left|\sum_{t_0}^{t_2} g_z\right|}{\sum_{t_0}^{t_2} |g_x| + \sum_{t_0}^{t_2} |g_y| + \sum_{t_0}^{t_2} |g_z|} \tag{3}$$

Here, g_x, g_y, g_z are sensor readings of each gyroscope axis, t_1 is the connecting point between two segments $[t_0, t_1]$ and $[t_1, t_2]$. Com ranges from 0 to 1. Small Com stands for that two neighboring segments belong to one gesture. We merge two segments if Com is lower than a threshold. In Fig. 6, we see that the sum of sensor readings for each axis is very close to zero. Therefore, Com is small and two segments $[t_0, t_1]$ and $[t_1, t_2]$ should be merged.

Fig. 7 shows the Con and Com values for 100 gestures performed by one user continuously. In these 100 continuous gestures, there are 177 connecting points. Of all these connecting points, 99 of them separate two gestures, which are marked as blue stars; the other 78 connecting points are inside gestures, which are marked as red circles. From Fig. 7 we find that almost all red circles are distributed in the left bottom of the figure, which indicates low gesture continuity and gesture completeness. As most red circles are within 40 degree in Con and 0.2 in Com, we set 40 degree as the threshold for Con and 0.2 as the threshold for Com. If Con and Com are smaller than these thresholds, we merge two segments into one.

From Fig. 5(d) to Fig. 5(e), we find that segment 2 and 3 in Fig. 5(d) are merged into segment 2 in Fig. 5(e), and segment 4 and 5 in Fig. 5(d) are merged into segment 3 in Fig. 5(e). Each segment in Fig. 5(e) contains exactly one complete gesture.

4) Noise Segments Removal: We extract the following three features from each segment to classify if it is a noise segment: (1) Duration of segment. Usually, the duration of one gesture is within a certain range. Among all the gesture data collected by us, no gesture lasts longer than 3 seconds, or shorter than

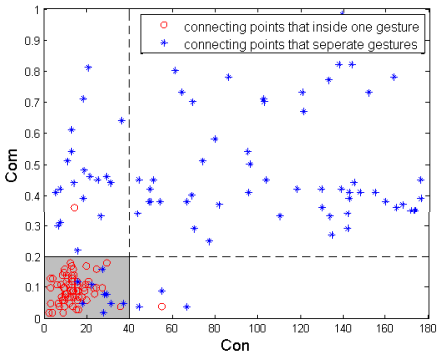

Fig. 7. Con VS Com

0.8 second. Therefore, if the duration of one segment is outside of these boundaries, this segment is filtered out as noise.
(2) HM of segment. The user is not supposed to perform the gesture too quickly. Therefore, HM, which measures the hand movement, is limited in a certain range. In our gesture dataset, we find that the max HM is 474 degree/second. Therefore, segments with the HM value above 474 degree/second are removed.
(3) Completeness of segment. The Gesture Completeness metric is used for segments merging as defined in Eq. (3). Here we use this metric again to remove noise segments. As each gesture defined by us starts from and ends in the same position, the Gesture Completeness metric (Com) for each gesture segment should be a small value. For all the gesture data collected, the Com values of more than 99% of gestures are smaller than 0.3. Therefore, if the Com of one segment is larger than 0.3, this segment is removed.

In Fig. 5(e), the Com value for Segments 1 to 4 are 0.98, 0.08, 0.04, 0.76, respectively. Therefore, Segment 1 and 4 are removed, Segment 2 and 3 are forwarded to the Hand Gesture Recognition module. Notably, Segment 1 and 4 are the "raise hand horizontally" movement and "put down hand" movement, which are not predefined gestures for our application.

C. Hand Gesture Recognition

According to the data segmentation results, we extract 6 representative features for model training and testing from the acceleration and gyroscope data of each segment: (1) raw acceleration data, (2) first-derivative of acceleration data, (3) the integral of the acceleration data, (4) raw gyroscope data, (5) first-derivative of gyroscope data, and (6) the integral of the gyroscope data. The first-derivative and the integral of the data are shown to be effective in improving recognition accuracy [12]. This is due to their ability to describe the main characteristics of the gesture: absolute trending (raw data), its relative change (first-derivative), and the cumulative effect (integral).

We utilize the Hidden Markov Model (HMM) algorithm to train classifiers for online hand gesture recognition, as it has shown high accuracy in previous work [12][9][19]. For each gesture, an HMM model is trained based on a set of the same gesture data. We train the HMM models using the standard Baum-Welch re-estimation algorithm [24]. To filter out non-gestural movements or undefined gestures, a noise HMM model is trained using the ergodic topology [19]. At runtime, each data segment is classified as one of the predefined gestures or noise. We use the Viterbi algorithm [25] to efficiently calculate the likelihood of the input data segment for each gesture model. Then, the gesture model with the highest likelihood is selected as the classified gesture. If the noise model is classified, we reject the gesture.

VI. Performance Evaluation

The dataset for evaluating the hand gesture recognition algorithm is collected using Motorola Moto 360 (2nd Gen.), while our Gemote hardware platform was being completed. The motion sensor chip in Motorola Moto 360 (2nd Gen.) is InvenSense MPU-6050 [15], while the motion sensor chip in Gemote is InvenSense MPU-6500 [26]. The only difference between these two chips is that MPU-6500 has an on-board digitally programmable low-pass filter and a Serial Peripheral Interface, while MPU-6050 does not have those components.

In our experiment, the accelerometer and gyroscope data of seven hand gestures are collected from five male human subjects. The data collection experiment contains two independent steps: (1) each participant performs each gesture 10 times; (2) each participant randomly chooses 50 gestures and performs these gestures continuously. We evaluate the performances of the proposed algorithms on the non-continuous gestures and the continuous gestures separately. First, we evaluate the gesture recognition accuracy with five fold cross validation on each participant when performing gestures separately. We use gesture samples from four participants to train the HMM models, and then apply these HMM models to classify the gesture samples from the remaining participant. Second, we use all the gesture samples collected in the first step to train the HMM models, and evaluate the proposed algorithm on the continuous hand gesture samples collected in the second step. Based on the accelerometer and gyroscope data, 18 features are extracted to train the HMM models. Each HMM model is configured with 4 states and 2 Gaussian components.

Table III shows the confusion matrix for gesture classification when only using the acceleration features. The average accuracy is 95.4%. Table IV shows the confusion matrix for gesture classification when only using the gyroscope features. The average accuracy is 93.7%. Table V shows the confusion matrix for gesture classification when using both the acceleration features and gyroscope features. The average accuracy is 99.4%. From these tables, we find that the HMM models with the acceleration features as input perform a little better than the HMM models with the gyroscope features as input. We also find that HMM models with both the acceleration features and gyroscope features have the best performance.

Fig. 8 demonstrates the segmentation accuracy, recognition accuracy and overall accuracy of the continuous gesture recognition. The overall accuracy is the product of the segmentation

TABLE III
CONFUSION MATRIX FOR GESTURE RECOGNITION USING ACCELEROMETER FEATURES

	Left	Right	Up	Down	Back&Forth	Clockwise	Counterclockwise
Left	50	0	0	0	0	0	0
Right	0	48	0	0	0	0	2
Up	0	0	50	0	0	0	0
Down	0	0	0	40	0	6	4
Back&Forth	0	0	0	0	50	0	0
Clockwise	0	0	4	0	0	46	0
Counterclockwise	0	0	0	0	0	0	50

TABLE IV
CONFUSION MATRIX FOR GESTURE RECOGNITION USING GYROSCOPE FEATURES

	Left	Right	Up	Down	Back&Forth	Clockwise	Counterclockwise
Left	42	6	4	0	0	0	0
Right	0	40	0	0	10	0	0
Up	0	0	50	0	0	0	0
Down	0	0	0	48	2	0	0
Back&Forth	0	0	0	0	50	0	0
Clockwise	0	0	0	0	0	50	0
Counterclockwise	0	0	0	0	0	0	50

TABLE V
CONFUSION MATRIX FOR GESTURE RECOGNITION USING ACCELEROMETER AND GYROSCOPE FEATURES

	Left	Right	Up	Down	Back&Forth	Clockwise	Counterclockwise
Left	50	0	0	0	0	0	0
Right	0	50	0	0	0	0	0
Up	0	0	48	0	2	0	0
Down	0	0	0	50	0	0	0
Back&Forth	0	0	0	0	50	0	0
Clockwise	0	0	0	0	0	50	0
Counterclockwise	0	0	0	0	0	0	50

accuracy and recognition accuracy. The average segmentation accuracy is 98.8% (standard deviation: 1.83%), the average recognition accuracy is 95.7% (standard deviation: 4.08%), and the average overall accuracy is 94.6% (standard deviation: 3.99%). Good experimental results demonstrate that the proposed segmentation algorithm and recognition algorithm are very promising. Table VI shows the composite confusion matrices of the classification of the continuous hand gestures. From the table, we see that six Left gestures are mistakenly classified as Counterclockwise gestures, two Up gestures are mistakenly classified as Back&Forth gestures, and two Clockwise gestures are mistakenly classified as Left gestures. These misclassifications mainly come from the difference between training gesture samples and test gestures samples. We utilize gestures discretely performed by the users as the training set, and gestures continuously performed by the users as the test set. When users perform gestures continuously, sensor readings include lots of motion noises, which lead to the lower recognition accuracy.

Fig. 9 shows the classification accuracy of the continuous hand gestures with different HMM models: HMM models with the acceleration features, HMM models with the gyroscope features, and HMM models with both the acceleration and gyroscope features. The average accuracy for the HMM models with the acceleration features, the gyroscope features, and both the acceleration and gyroscope features are: 86.01% (standard deviation: 16.63%), 81.95% (standard deviation: 7.38%), and 94.6% (standard deviation: 3.99%), respectively. HMM models with the acceleration features achieve 100% recognition accuracy for Subject 1 and Subject 4. However, these models do not perform well for Subject 3 and Subject 5. Statistically, HMM models with both the acceleration and gyroscope features are more accurate (the highest average accuracy), and more stable (the lowest standard deviation). We plan to collect more data to further test the performance of our algorithm.

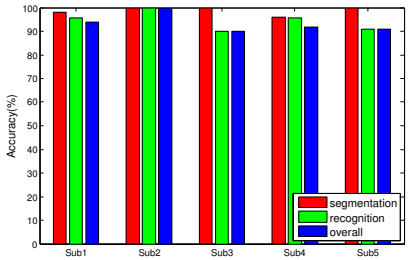

Fig. 8. Segmentation and recognition accuracy of continuous hand gestures. Sub1 means human subject one.

VII. CONCLUSION

In this paper, we present the wristband platform Gemote for gesture recognition for future remote control use in healthcare

TABLE VI
CONFUSION MATRIX FOR CONTINUOUS GESTURE RECOGNITION

	Left	Right	Up	Down	Back&Forth	Clockwise	Counterclockwise
Left	28	0	0	0	0	0	6
Right	0	40	0	0	0	0	0
Up	0	0	36	0	2	0	0
Down	0	0	0	29	0	0	0
Back&Forth	0	0	0	0	44	0	0
Clockwise	2	0	0	0	0	23	0
Counterclockwise	0	0	0	0	0	0	23

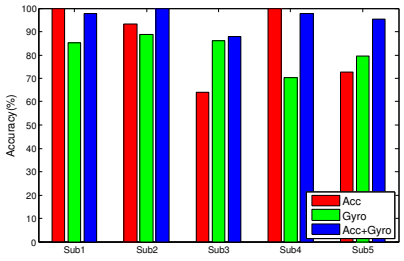

Fig. 9. Classification accuracy of continuous hand gestures with different HMM models. Sub1 means human subject one.

settings. It is comfortable to wear, with open API, and at an affordable price. Gemote employs a novel continuous gesture segmentation and recognition algorithm. For a sequence of hand movement, Gemote separates data into meaningful segments, merges segments based on gesture continuity and gesture completeness metrics, removes noise segments, and finally recognizes hand gestures by HMM classification. Evaluation results show that Gemote can achieve 94.6% recognition accuracy when users perform gestures continuously.

ACKNOWLEDGMENTS

Special thanks to our participants in our user studies and all the anonymous reviewers for their great reviews and suggestions to improve the quality of this paper. This work was supported by NSF CNS-1253506 (CAREER) and NSF CNS-1618300.

REFERENCES

[1] marketsandmarkets.com. (2015). [Online]. Available: http://www.marketsandmarkets.com/Market-Reports/touchless-sensing-gesturing-market-369.html

[2] J. P. Wachs, H. I. Stern, Y. Edan, M. Gillam, J. Handler, C. Feied, and M. Smith, "A gesture-based tool for sterile browsing of radiology images," *Journal of the American Medical Informatics Association*, vol. 15, no. 3, pp. 321–323, 2008.

[3] A. Tognetti, F. Lorussi, R. Bartalesi, S. Quaglini, M. Tesconi, G. Zupone, and D. De Rossi, "Wearable kinesthetic system for capturing and classifying upper limb gesture in post-stroke rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, no. 1, p. 8, 2005.

[4] S. W. Davies, S. L. Jordan, and D. P. Lipkin, "Use of limb movement sensors as indicators of the level of everyday physical activity in chronic congestive heart failure," *The American journal of cardiology*, vol. 69, no. 19, pp. 1581–1586, 1992.

[5] M. Milenkovic, E. Jovanov, J. Chapman, D. Raskovic, and J. Price, "An accelerometer-based physical rehabilitation system," in *Proceedings of IEEE SSST*. IEEE, 2002, pp. 57–60.

[6] E4 wristband. [Online]. Available: https://www.empatica.com/e4-wristband

[7] N. C. Krishnan, C. Juillard, D. Colbry, and S. Panchanathan, "Recognition of hand movements using wearable accelerometers," *Journal of Ambient Intelligence and Smart Environments*, vol. 1, no. 2, pp. 143–155, 2009.

[8] Y. Dong, A. Hoover, and E. Muth, "A device for detecting and counting bites of food taken by a person during eating," in *Proceedings of IEEE BIBM*. IEEE, 2009, pp. 265–268.

[9] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, 2008.

[10] U. Maurer, A. Rowe, A. Smailagic, and D. P. Siewiorek, "ewatch: a wearable sensor and notification platform," in *Proceedings of IEEE BSN*. IEEE, 2006, pp. 4–pp.

[11] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis, "Risq: Recognizing smoking gestures with inertial sensors on a wristband," in *Proceedings of ACM MobiSys*. ACM, 2014, pp. 149–161.

[12] T. Park, J. Lee, I. Hwang, C. Yoo, L. Nachman, and J. Song, "E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices," in *Proceedings of ACM SenSys*. ACM, 2011, pp. 260–273.

[13] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[14] Moto 360 2nd gen. [Online]. Available: https://www.motorola.com/us/products/moto-360

[15] Invensense mpu6050 datasheet. [Online]. Available: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[16] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch," in *Proceedings of ACM HotMobile*. ACM, 2015, pp. 9–14.

[17] Y. Dong, A. Hoover, J. Scisco, and E. Muth, "A new method for measuring meal intake in humans via automated wrist motion tracking," *Applied psychophysiology and biofeedback*, vol. 37, no. 3, pp. 205–215, 2012.

[18] Wii controller. [Online]. Available: http://wii.com/

[19] H.-K. Lee and J.-H. Kim, "An hmm-based threshold model approach for gesture recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 10, pp. 961–973, 1999.

[20] C. Lee and Y. Xu, "Online, interactive learning of gestures for human/robot interfaces," in *Proceedings of IEEE ICRA*, vol. 4. IEEE, 1996, pp. 2982–2987.

[21] UG smart wristband. [Online]. Available: http://www.ultigesture.com/

[22] W.-C. Bang, W. Chang, K.-H. Kang, E.-S. Choi, A. Potanin, and D.-Y. Kim, "Self-contained spatial input device for wearable computers," in *Proceedings of IEEE ISWC*. IEEE Computer Society, 2003, p. 26.

[23] A. Y. Benbasat and J. A. Paradiso, "An inertial measurement framework for gesture recognition and applications," in *International Gesture Workshop*. Springer, 2001, pp. 9–20.

[24] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[25] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[26] Invensense mpu9250 datasheet. [Online]. Available: https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf