Information Fusion 53 (2020) 55–65

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Adaptive sliding window based activity recognition for assisted livings

Congcong Ma[a,b,*], Wenfeng Li[a], Jingjing Cao[a], Juan Du[b], Qimeng Li[c], Raffaele Gravina[c]

[a] *School of Logistics Engineering, Wuhan University of Technology, Wuhan, 430063, China*
[b] *Nanyang Institute of Technology, Nanyang 473000, China*
[c] *Department of Informatics, Modelling, Electronics and Systems (DIMES), University of Calabria, Via P. Bucci, Rende, CS, 87036, Italy*

ARTICLE INFO

Keywords:
Adaptive sliding window
Activity recognition
Sitting posture
Smart cushion
Multivariate gauss distribution

ABSTRACT

The incessant diffusion of smart wearable devices and Body Area Networks (BANs) systems is pushing more and more researches on human activity recognition. Previous studies usually adopted fixed time window or chose an optimal window size depending on the characteristics of the activity. In this paper, we propose an adaptive time window-based algorithm of activity recognition, which addresses the problem of different types of activities having different time duration that usually causes poor recognition results. Multivariate Gaussian Distribution (MGD) method was used to detect the signal difference between each determined time window and the defined activity characteristics, and we more accurately define the time window size for each activity. To evaluate the proposed algorithm, we focused on the activities performed by wheelchair users in their daily life, so as to provide them with better healthcare services. We construct two different datasets, respectively considering static and dynamic wheelchair user activities on different ground surfaces. Then, we use time window expansion and contraction method to determine and dynamically adjust the window size for each activity. Different comparison criteria such as recognition precision and F-score were used to evaluate our algorithm. Experiment results revealed that, according to F-score, the proposed algorithm performs 15.3% better than traditional methods in static conditions. In dynamic scenarios we observed 6.4% and 24.5% improvements on flat and rough floors, respectively.

1. Introduction

In the context of BANs [1], multi sensor fusion [2] and wearable sensing [3], more and more user-centric applications such as mobile health monitoring [4], emotion-relevant activity recognition [5], and smart wheelchair systems [6] are being developed to facilitate our daily life.

More and more studies are being devoted to human activity recognition. Task-oriented [7], collaborative computing and multi-sensor data fusion based frameworks [8] have been proposed to support activity recognition in BANs. In most scenarios, activities should be monitored in real time, so power control [9] and energy harvesting [10] have to be addressed, too. Sodhro et al. [11] proposed a transmission power control based energy efficient algorithm for activity recognition in remote healthcare. Some studies focused on Quality of Service (QoS) on transmission [12] for activity recognition; to improve efficiency, an adaptive QoS computation algorithm Sodhro et al. [13] was proposed to balance transmission power, duty cycle and route selection. In terms of communication technology, 5G could better enhance and optimize the QoS for the communications of Internet-of-Things devices, including smart wearables used for activity recognition [14] especially in medical healthcare [15].

Compared with previous research, our paper mainly focuses on properties of wheelchair users activities. The main motivation of this work is to solve the problem of not optimal recognition of activities due to different activity time duration; being able to detect in the signal the start and end points of each activity would allow for more accurate dimensioning of the time window which would turns into improved recognition accuracy.

Therefore, in this paper, we propose an adaptive sliding window algorithm for more effective activity recognition based on a window expansion and contraction method to precisely set the time window of the activity. In order to evaluate the proposed algorithm, we also construct datasets of activities performed by wheelchair users in their daily life.

Specifically, the contributions of the paper are twofold:

- An architecture of adaptive sliding time window method is proposed to recognize different sitting activities for wheelchair users.
- A MGD-based dynamic time window adjustment algorithm is proposed to tune the window size by means of window expansion or contraction.

* Corresponding author.
E-mail addresses: macc@whut.edu.cn (C. Ma), liwf@whut.edu.cn (W. Li), bettycao@whut.edu.cn (J. Cao), dujuan-0218@163.com (J. Du), liqimeng@msn.com (Q. Li), r.gravina@dimes.unical.it (R. Gravina).

https://doi.org/10.1016/j.inffus.2019.06.013
Received 29 December 2018; Received in revised form 27 May 2019; Accepted 3 June 2019
Available online 3 June 2019

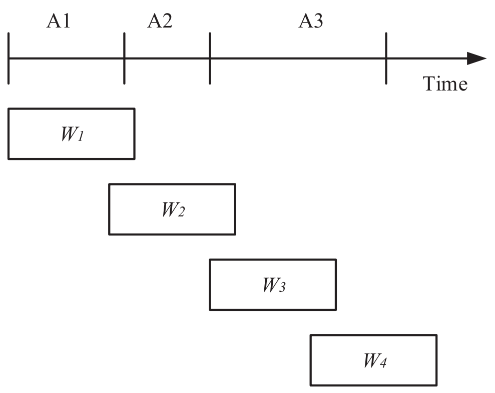


Fig. 1. Activity segmentation using fixed time window.

The remainder of the paper is organized as follows. Section 2 discusses the background and related works of window selection method in the context of activity recognition. Section 3 focuses on the methodology of this work. Section 4 presents the hardware design of a prototype system, data pre-processing, feature extraction and selection method. Section 5 describes the experiment protocol and data collection and presents performance evaluation results of our proposed method including comparison with other literature methods. Finally, in Section 6 we conclude the paper and outline future works.

2. Background and related work

Human motion is a complex sequence of simpler body movements which can be monitored using different approaches, notably including wearable inertial sensors. The collected data streams can be segmented in (consecutive or partially overlapping) time windows in which several features can be extracted and eventually used by recognition algorithms such as machine learning method [16] to classify the activities. However, if the time window is not properly selected, multiple activities could appear in the same window or, conversely, important portions of an activity could be missing because they “fall” in the following window.

Fig. 1 shows a schematic representation of a time series containing data of three activities, namely A1, A2 and A3, each one with a different time duration. If we use the naive method using fixed - partially overlapping - time windows of size W, in this example we have four time-windows, W_1, W_2, W_3, and W_4. In this situation, only A1 can be well recognized. W_2 contains data belonging to all the three activities, while A3 data is split in two windows.

In the following, we will introduce two types of time window selection methods: sliding window and event-based definition.

2.1. Sliding window method

Sliding window method is very popular in the context of activity recognition. However, determining the best value of the window size is a key, often non trivial, issue. The size of the sliding time window will affect the recognition results. If the time window is too large, it might contain information of multiple activities, and decreases the reactivity of the recognition system and increases computation load. On the contrary, if the time window is too small, some activity may be split into multiple consecutive windows, recognition task will be triggered too frequently, without, however, achieving high recognition results. Depending on specific application scenarios and requirements, the size of sliding time window is chosen from 0.08s [17] to 2s [18], and even up to 30s [19]. Given that the window length directly affects recognition performance, Wang et al. [20] compared different window sizes and found that 2.5s-3.5s can provide an optimal result for motion recognition and 0.5s for posture pattern recognition, but their method is still too specific.

As aforementioned, consecutive sliding windows could be non overlapping although many literature studies actually suggest to use 25% to 50% overlap [21]. If the window size is properly set, this simple approach allows excellent results when the activities being monitored have similar time duration.

A few previous researches focused on the adaptive sliding window. Noor et al. [22] proposed an adaptive window method that just expands the time window to detect the physical activity using single tri-axial accelerometer. Ni et al. [23] proposed a change point-based segmentation method to dynamically identify the window starting point and end point to get the information of the activity samples.

Despite of the sliding window based method, other researches investigated the use of event-based window definition to detect the time window.

2.2. Event-based window definition

With the sliding window-based approach, the window size is defined independently from the current activities being performed. The event-based window definition, instead, takes into consideration certain properties of the signal to determine the most appropriate window size.

In [24], the authors proposed an activity recognition method for woodworkers using wearable microphone and accelerometer. When they perform carpentry works, the voice volume recorded by a microphone attached on the wrist is a bit lower than the one recorded on the main arm. The authors exploited such property to set the size of the time window.

In [25], the authors use EMG sensor and a wearable accelerometer to collect data of body building activities for fatigue assessment. The authors analyze the accelerometer data during weight lifting to determine each time window. Then, they re-construct the EMG data using sliding windows, sized accordingly, to recognize fatigue conditions and provide the athlete with appropriate recommendations.

As the event-based method needs other kinds of sensors for a supplementary comparison, it will add complexity during implementation and computation load to the system.

Unlike previous related literature, our research better combines sliding window and event-based methods to support advanced activity recognition for wheelchair users.

3. Proposed method

In this section, we elaborate the proposed method. Firstly, we provide a definition of posture and activity. Secondly, we design the architecture of adaptive sliding window for activity recognition. Thirdly, we introduce the time window adjustment algorithm.

3.1. Posture and activity definition

In our former research [26], we used a smart cushion to detect the sitting postures and we listed some common postures usually performed while seated (i.e. Proper Sitting (PS), Lean Left (LL), Lean Right (LR), Lean Forward (LF) and Lean Backward (LB)). Regarding the posture recognition, we found that the time duration of 0.5s could well depict the information of sitting postures. Therefore, every half a second we determine a sitting posture condition.

As to the activities of the sedentary lifestyle users, we distinguish the activity categories from the aspect of posture representation as shown in Fig. 2.

So, we can get the activity category and the activity examples as shown in Table 1.

Examples of cumulative-posture activity are schematically depicted in Fig. 3. Fig. 3(a) is an example of proper sitting activity (PSA), and Fig. 3(b) is an example of left pressure relief activity (LPR). We recall that PS and LL respectively represent the proper sitting and leaning left postures.

Table 1
Activity categories and their examples.

Activity category	Activity example
Stable posture activity	Proper Sitting Activity (PSA), Left Sitting Activity (LSA), Right Sitting Activity (RSA)
Transitional posture activity	Left Pressure Relief (LPR), Right Pressure Relief (RPR)
Non-cumulative posture activity	Trunk Turn Left (TTL), Trunk Turn Right (TTR)

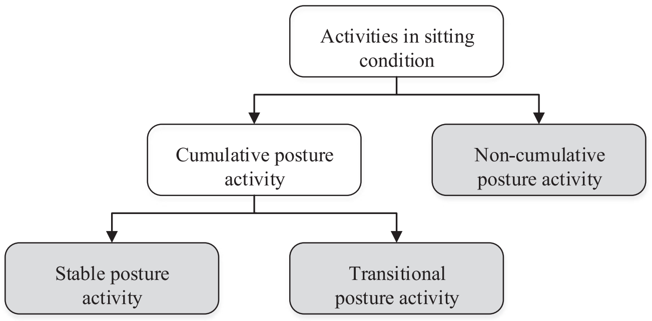

Activities in sitting condition
Cumulative posture activity
Non-cumulative posture activity
Stable posture activity
Transitional posture activity

Fig. 2. Activity categories sorted by posture representation.

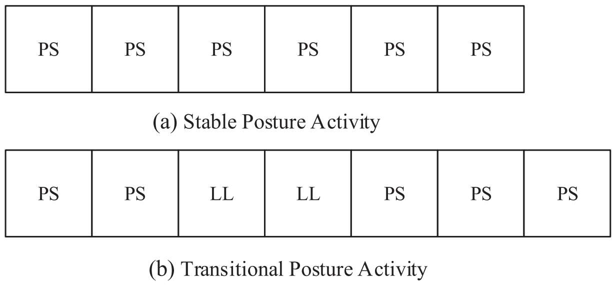

PS PS PS PS PS PS
(a) Stable Posture Activity
PS PS LL LL PS PS PS
(b) Transitional Posture Activity

Fig. 3. Examples of cumulative posture activity.

Stable posture activities refer to the activities kept for a given time period. In our paper, we selected the following three stable posture activities: Proper Sitting Activity (PSA), Left Sitting Activity (LSA) and Right Sitting Activity (RSA).

Transitional posture activities consists of a sequence of different individual sitting postures. A common example of such activities is pressure relief, which is a specific activity performed by wheelchair users to prevent pressure ulcers. In this paper, we focused on two kinds of transitional posture activities: Left Pressure Relief (LPR) and Right Pressure Relief (RPR).

Finally, non-cumulative posture activities, refers to sporadic activities. In this paper, we selected two non-cumulative posture activities: Trunk Turn Left (TTL) and Trunk Turn Right (TTR), as depicted in Fig. 4. Long-term sitting users often perform such activities for trunk exercise. In the initial position, the user is in proper sitting posture, then turns the trunk left/right 30° to 60°, then turns back to the initial proper sitting posture.

It has been shown [27] that trunk exercise can strength core stability and help preventing several pathologies including chronic stroke. Additionally, in the context of smart wheelchairs, monitoring these trunk activities while the wheelchair is in motion can support the assessment of dangerous driving conditions.

3.2. Adaptive sliding time window

We propose an adaptive sliding window algorithm whose flow chart is shown in Fig. 5. Firstly, we need to identify the activity category; then, according to the category, different time window adjustment methods are executed. Secondly, we need to calculate the Probability Density value using a MGD method; then, we have to determine whether the time window should be updated. Thirdly, the new time window is sized and the activity recognized; in addition, we obtain the start time point of the next activity.

In [28] the authors proposed a hierarchical signal segmentation approach exploiting the rich set of features in a large window to increase the general performance and a short window to detect transitions between activities. In contrast with this approach, our proposed method does not require to determine the features in the small window, with advantages in terms of computation workload.

As shown in Fig. 5, the proposed method involves two different classifiers. Classifier 1 is used to recognize the activity category, it processes the data of a fixed window size and distinguish if the activity belongs to Stable Posture Activity, Transitional Posture Activity or Non-cumulative Posture Activity. Classifier 2 is used to recognize the activity. When Stable Posture Activity is detected, Classifier 2 is directly used and the activity can be recognized. When Non-stable Posture Activity is detected, before evaluating time window expansion or contraction, we need to calculate the Probability Density Function (PDF) using MGD, then we can determine the most effective time window size. When the Transitional Posture Activity is detected, the algorithm will trigger the time window expansion; at each step of expansion, we calculate the PDF value of the features in a newly determined window compared with the features of the defined activity. When the PDF value continues to increase, the time window continues to expand. Similarly, we designed the time window contraction method. Using the continuously updated window size and the extracted features, Classifier 2 is used to determine the activity.

PDF value represents the likelihood that a signal belongs to a specific activity; a higher PDF value indicates that the data in window are nearer the real activity. Given a *d*-dimension feature set $X = \{x_1, x_2, x_d\}$ and an activity A_j, using the MGD method we can calculate the similarity of X compared with A_j, denoted by $p(X;\, \mu_j,\, \Sigma_j)$.

μ is the mean value of A_j, Σ is the covariance matrix corresponding to the feature sets X, N_j is the total number of observations belonging to activity A_j. μ_j and Σ_j are calculated as shown in Eqs. 1 and 2.

$$\mu_j = \frac{1}{N_j} \sum_{x \in A_j} x \tag{1}$$

$$\sum_j = \frac{1}{N_j} \sum_{x \in A_j} x x^{\mathrm{T}} - \mu_j \mu_j^{\mathrm{T}} \tag{2}$$

The calculation of PDF is shown as Eq. 3, where *n* is the dimension of activity *X*.

$$p(X|A_j) \propto p(X;\mu_j, \sum_j) = \frac{1}{(2\pi)^{n/2} |\sum_j|^{1/2}} e^{-\frac{1}{2}(X-\mu_j)^{\mathrm{T}} \Sigma^{-1} (X-\mu_j)} \tag{3}$$

The pseudo-code of the proposed adaptive sliding time window-based method is shown in Algorithm 1.

The parameters used in Algorithm 1 are explained in Table 2. "ActivityType Classifier" is Classifier 1 and "Activity Classifier" is Classifier 2, shown in Fig. 5.

As shown in Algorithm 1, from line 6 to line 13, after the execution of time window expansion and contraction, we can obtain the stop time of the i_{th} activity and the start time of the $(i+1)_{th}$ activity, denoted as $S_{i+1,0}$.

For activity recognition, in [20] it was shown that 2.5s to 3s can provide an optimal result. We collected the data of the activities as shown in Table 1, according to our statistics, we set the initial time window *w*

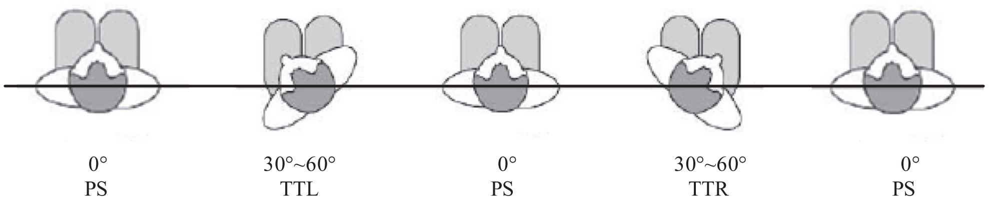


Fig. 4. Non-cumulative-posture activity examples (trunk rotation exercise).

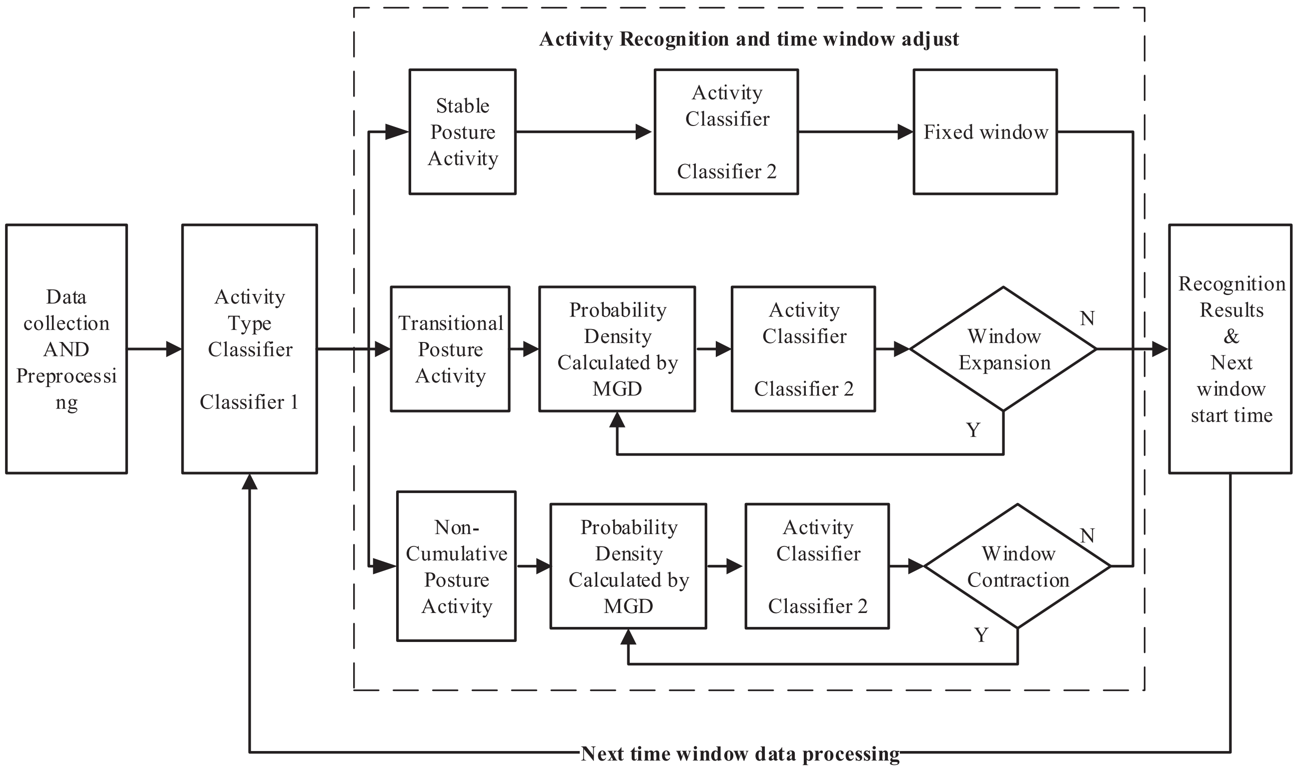

Activity Recognition and time window adjust
Stable Posture Activity
Activity Classifier
Classifier 2
Fixed window
Data collection AND Preprocessing
Activity Type Classifier
Classifier 1
Transitional Posture Activity
Probability Density Calculated by MGD
Activity Classifier
Classifier 2
Window Expansion
N
Y
Non-Cumulative Posture Activity
Probability Density Calculated by MGD
Activity Classifier
Classifier 2
Window Contraction
N
Y
Recognition Results & Next window start time
Next time window data processing

Fig. 5. Adaptive time window of activity recognition.

Algorithm 1 Adaptive sliding time window-based activity recognition algorithm.

Input: Raw activity data and the initial activity start time $S_{i,0}$
Output: Activity type of A and the next activity start time $S_{i+1,0}$

```
1:  BEGIN
2:    Using the base-window $w$, calculate the features in the data window
3:    execute ActivityType Classifier, calculate the Activity Type
4:    if ActivityType = StablePostureActivity
5:        Activity A = execute Activity Classifier
6:    elseif ActivityType = TransitionPostureActivity
7:        Activity A = execute Activity Classifier
8:        execute time window expansion method (as shown in Algorithm~ 2)
9:        after time window expands m times, we obtain the time window start time $S_{i+1,0} = S_{i,v}$, as $v = (N_{i,m} - 1) - round(f_o * w)$
10:   else ActivityType = NonCumulativePostureActivity
11:       Activity A = execute Activity Classifier
12:       execute time window contraction method (as shown in Algorithm~3)
13:       after time window expands n times, we obtain the time window start time $S_{i+1,0} = S_{i,v}$, as $v = (N_{i,m} - 1) - round(f_o * w)$
14:   endif
15: END
```

Table 2
Description of parameters in Algorithm 1.

Parameter	Description
w	Size of the initial time window
f_o	The overlap of each adjacent time window ($0 \le f_o \le 1$)
Round(i)	Rounding of value i to its nearest integer
$S_{i,0}$	The start time of i_{th} time window

to 3s by empirical evaluation. Also the expansion (f_e) and contraction (f_c) factors, time window overlap (f_o) will be used in Algorithms 2 and 3 as 17% (as the factor 17% was determined by time window duration of posture divided by w, it means that in every step of expansion or contraction, the time window expands or contracts by a time window of sitting posture).

3.3. Expansion method of adaptive sliding time window

As aforementioned, if the PDF value continues to increase, the time window continues to expand; when the computed PDF value stops increasing, the time window also stops to expand.

The time window expansion method is shown in Algorithm 2. Line 4 to line 8 represent the kernel code of the method that determine whether the expansion should be executed or not.

Algorithm 2 Adaptive sliding time window of expansion step.

Input: Activity A = TransitonPostureActivity
Output: the times of time window expansion step (m)

```
1:  BEGIN
2:    while $m < k_{\max}$ do
3:      Calculate the PDF value $P_{S_{i,j},m}(X|A)$
4:      if A is not changed or $P_{S_{i,j},m} > p_{\max}$
5:        $p_{\max} = P_{S_{i,j},m}$
6:        $N_{i,m} = w + round(f_e * w) * m$
7:        $m = m + 1$
8:        Activity A = execute Activity Classifier
9:      else
10:       stop time window expansion
11:     endif
12:   end while
13: END
```

Table 3
Description of parameters in Algorithm 2.

Parameter	Description
f_e	Expansion factor of the time window
m	Number of expansions
$k_{\max}$	The maximum number of time window expansion
$N_{i,m}$	Number of sample points after m times expansion of i_{th} window
$S_{i,j}$	i_{th} time window activity sample
X	i_{th} time window activity after expansion
$p_{\max}$	the maximum value of PDF value after k extensions

The time window stop criterion includes 3 conditions: (1) After each expansion step, the activity is changed; (2) after each expansion, the newly calculated PDF value is smaller than the former one; (3) the times of expansion step reaches the maximum number of expansions.

The parameters used in Algorithm 2 are shown in Table 3. The expansion factor is in the range of $0 \le f_e \le 1$, when f_e is 1, it means that the time window expands as the length of w, and when we choose the f_e as 17%, it means that it expands as the length of a posture duration as 0.5s every time. When m is equal to 0, then the window is not expanded. $P_{S_{i,j},m}$ is the PDF value of feature sets of X compared with activity A.

3.4. Contraction method of adaptive sliding time window

The adaptive sliding time window of contraction method is shown in Algorithm 3.

The parameters used in Algorithm 3 are shown in Table 4. The contraction factor is in the range of $0 \le f_c \le 1$; when f_c is 1, it indicates that the time window contract a initial time window w every time. When $m = 0$, it demonstrates that the window is not contracted. $Q_{S_{i,j},m}(X|A)$ is the PDF value of feature sets of X compared with activity A.

Algorithm 3 Adaptive sliding time window of contraction method.

Input: Activity A = NonCumulativePostureActivity
Output: the times of time window contraction step (n)

```
1:  BEGIN
2:    while $n < k_{\max}$ do
3:      Calculate the PDF value $Q_{S_{i,j},n}(X|A)$
4:      if A is not changed or $Q_{S_{i,j},n} > q_{\max}$
5:        $q_{\max} = Q_{S_{i,j},n}$
6:        $N_{i,n} = w - round(f_e * w) * n$
7:        $n = n + 1$
8:        Activity A = execute Activity Classifier
9:      else
10:       stop time window contraction
11:     endif
12:   end while
13: END
```

Table 4
Description of parameters in Algorithm 3.

Parameter	Explanation
f_c	Contraction factor of the time window
n	Number of contractions
$k_{\max}$	The maximum number of time window Contraction
$N_{i,n}$	Number of sample points after m times Contraction of i_{th} window
$S_{i,j}$	i_{th} time window activity sample
X	i_{th} time window activity after Contraction
$q_{\max}$	the maximum PDF value after k times Contraction

4. System implementation and data process

In a previous work [29], we designed a smart cushion to detect sitting postures and activities. In the following, we will describe the hardware design of the smart cushion, data pre-processing and also feature extraction.

4.1. Hardware design of the smart cushion

In this work, we use a smart cushion that was designed in our former research [29] to detect sitting postures [26] and activities [30]. The smart cushion is composed of two main parts: sensing hardware and signal processing module. Sensing hardware includes multiple pressure sensors deployed below the cushion foam to detect sitting postures and an inertial measurement unit (IMU) to detect movements of the human body. The signal processing module consists of an Arduino [31] board.

The sensors deployment on the smart cushion is shown in Fig. 6.

Force Sensing Resistor (FSR) [32] is used to measure the pressure. It is a light weighted, minimal sized and sensitive resistance-based sensor. As force is applied on the sensing areas, the resistance value of the pressure sensor will be correspondingly altered. Assuming a conventional size for the seat of 40 cm × 40 cm, we split it into 5 × 5 square zones. The six pressure sensors are individually placed in different zones as shown in Fig. 6(a). The central point is assumed as origin point O and the location of each pressure sensor is defined with relative distance to the origin point, as represented in Table 5.

In our smart cushion, the IMU is a MPU9250 [33] chip, a 9DoF (degree-of-freedom) unit placed on the back of the base board, as shown in Fig. 6(b). Using the I2C interface, it outputs three-axis acceleration, three-axis gyroscope, and three-axis digital compass data. In this work, we only monitor the 3-axis accelerometer and 3-axis gyroscope data.

Fig. 6(c) shows the top side of the board with its six pressure sensors and, on the left top side, the signal processing module (Arduino board), the Bluetooth module, vibration motor unit and power supply unit. Fig. 6(d) shows the bottom side of the board with the IMU on the center. The board is eventually embedded in the foam filling of the cushion for user comfort.

Table 5
The coordinates of each pressure sensor.

Sensor No.	Coordinate (x,y)
Fsr1	-1,2
Fsr2	1,2
Fsr3	-2,0
Fsr4	2,0
Fsr5	-1,-2
Fsr6	1,-2

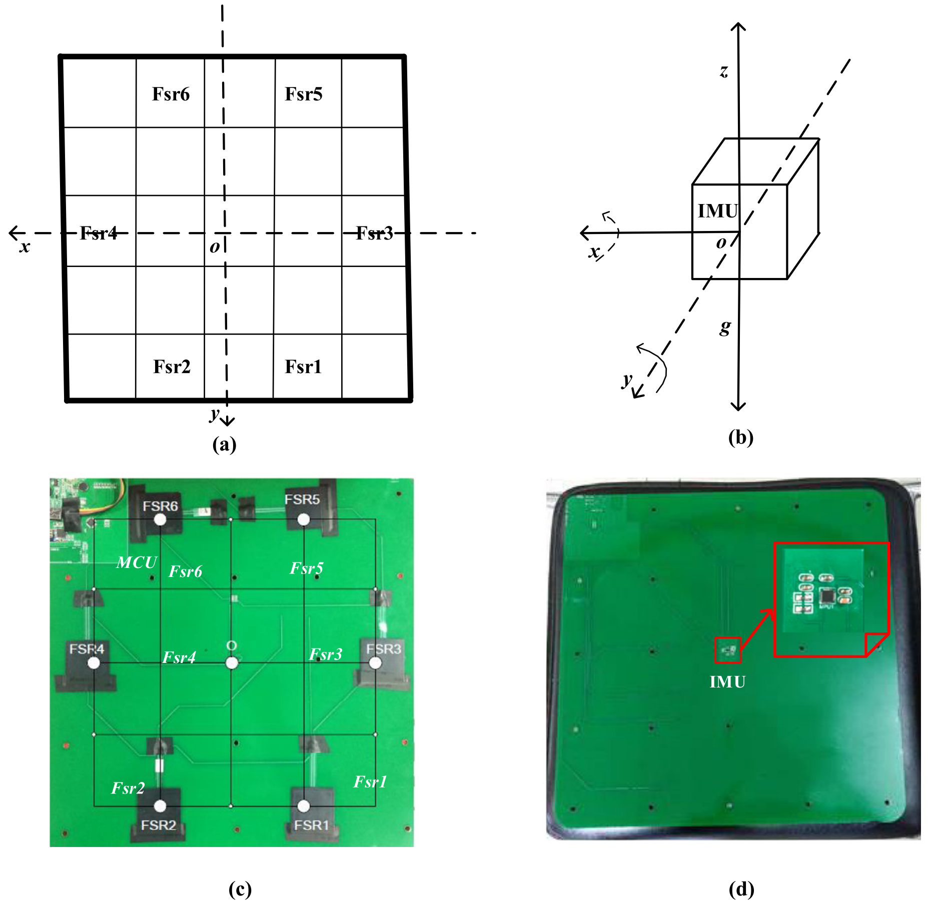


Fig. 6. Sensors deployment of the smart cushion and circuit board on a real chair: (a) schematic diagram of the sensors deployed on the base board; (b) IMU sensor and the three axis representation; (c) top side of the circuit board; (d) bottom side of the circuit board.

Table 6
The processed data of each sample data and description.

Pressure Sensor	IMU Sensor
CoP_x	a_x, a_y
CoP_y	a_{xz}, a_{yz}
IoP_x	a_{xyz}
IoP_y	ω_x, ω_y
RoP_x	ω_{xz}, ω_{yz}
RoP_y	ω_{xyz}

4.2. Data pre-processing

At the time t, we can obtain the raw data as shown in Eq. 4.

$$RawData_t = \{a_x, a_y, a_z, g_x, g_y, g_z, f_1, f_2, f_3, f_4, f_5, f_6\} \tag{4}$$

where f_1, f_2, f_3, f_4, f_5 and f_6 represent the pressure value of each pressure sensor.

All the data was collected and processed by the Arduino board at a sampling frequency of 50Hz (i.e. 50 samples per second for each signal). Former studies showed that human posture and activity is usually below 18Hz [34]. So, 50Hz is suitable enough to meet the requirements of posture recognition.

We extract a total of 16 different feature values from the pressure and inertial signals, as summarized in Table 6. In particular, the following 6 feature values are extracted from the pressure signals, while the remaining 10 feature values are extracted from the inertial signals and will be presented later.

- Center of Pressure (CoP), it is calculated as Eq. 5.

$$(CoP_x, CoP_y) = \left(\frac{\sum_{i=1}^{6} f_i * x_i}{\sum_{i=1}^{6} f_i}, \frac{\sum_{i=1}^{6} f_i * y_i}{\sum_{i=1}^{6} f_i}\right) \tag{5}$$

f_i represents the pressure value of each sensor, (x_i, y_i) are the coordinates of each sensor.

- Incline extent of Posture (IoP), it is calculated as Eq. 6.

$$(IoP_x, IoP_y) = \left(\frac{(f_2 + f_4 + f_6) - (f_1 + f_3 + f_5)}{\sum_{i=1}^{6} f_i}, \frac{(f_1 + f_2) - (f_5 + f_6)}{f_1 + f_2 + f_5 + f_6}\right) \tag{6}$$

- Ratio of Pressure (RoP): it is calculated as Eq. 7.

$$(RoP_x, RoP_y) = \left(\frac{f_2 + f_4 + f_6}{\sum_{i=1}^{6} f_i}, \frac{(f_1 + f_2) + 0.5 * (f_3 + f_4)}{\sum_{i=1}^{6} f_i}\right) \tag{7}$$

Regarding the sitting postural activities, we mainly focused on the analysis of the accelerometer and gyroscope data in the horizontal plane. From the raw acceleration and gyroscope data, we extract 10 feature values; the first four are a_x, a_y and ω_x, ω_y, while the latter ones, calculated by combining with z-axis, are defined in Eqs. 8 and 9.

$$\begin{cases} a_{xz} = \sqrt{a_x^2 + a_z^2} \\ a_{yz} = \sqrt{a_y^2 + a_z^2} \\ a_{xyz} = \sqrt{a_x^2 + a_y^2 + a_z^2} \end{cases} \tag{8}$$

$$\begin{cases} \omega_{xz} = \sqrt{\omega_x^2 + \omega_z^2} \\ \omega_{yz} = \sqrt{\omega_y^2 + \omega_z^2} \\ \omega_{xyz} = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \end{cases} \tag{9}$$

4.3. Feature extraction and selection

In this section, we describe seven different features that are extracted with our method, as shown in the following paragraph.

- *Mean* value represents the average value of a set of data; it is calculated as Eq. 10; N is the number of sample data, S_i is the i_{th} data.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} S_i \tag{10}$$

- *Standard Deviation* represents if the data sample are close to the mean value or not; it is calculated as Eq. 11.

$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S_i - \mu)^2} \tag{11}$$

- *Signal Energy* can distinguish different kinds of activity; it was calculated as Eq. 12.

$$EN = \frac{1}{N}\sum_{i=1}^{N} S_i^2 \tag{12}$$

- *Mean Trend* describes the trend of mean values over the window. The determined window size is N (number of samples); we further divide it into smaller windows (0.5s for each smaller window that contains 25 sample data) with no overlap. The number of samples in a smaller window is therefore N=25. The mean value of each of these 0.5s windows is calculated and subtracted from the mean of the following window. Then, the trend of the mean for every 0.5s is found for the entire time window. The sum of absolute values that are calculated between each adjacent smaller windows is calculated as Eq. 13, where μ_i represents the average value of each smaller window.

$$\mu T = \sum_{i=2}^{N/25} (|\mu_i - \mu_{i-1}|) \tag{13}$$

- *Window Mean difference*: the mean of each of these 0.5s windows is subtracted from the overall mean of the entire window sample data. The sum of absolute values of these distances called window mean difference is calculated as Eq. 14.

$$\mu D = \sum_{i=1}^{N/25} (|\mu - \mu_i|) \tag{14}$$

- *Variance Trend*: represents the variance trend of each small window and is computed similarly to the mean trend, as expected variance is calculated instead of mean for each of the windows. It is calculated as Eq. 15.

$$\mu T^2 = \sum_{i=2}^{N/25} (|\delta_i^2 - \delta_{i-1}^2|) \tag{15}$$

- *Window Variance difference*: it is calculated as Eq. 16, similarly to the window mean difference.

$$\mu D^2 = \sum_{i=1}^{N/25} (|\delta^2 - \delta_i^2|) \tag{16}$$

Using the processed data shown in Table 6, we can obtain 16 kinds of processed data from each time window; given that we extract 7 kinds of features, in conclusion we obtain a 112 (16 × 7)-dimension feature sets. Before the classification step, we need to normalize the features first, then we execute the feature selection step to reduce the complexity of the classification. Using the selected features, we can implement our adaptive time window method to recognize the defined activities.

5. Experiment and results

5.1. Experiment protocol and data collection

In order to validate our proposed method, we collected data of each activity from 12 different subjects using the smart wheelchair in static and dynamic conditions as shown in Table 7. With static condition, we refer to the subject sitting on a stopped wheelchair. Dynamic condition refers to the subject moving the wheelchair he/she is seated on (both on flat and rough floors). As shown in Table 7, NS(0) represents Nobody Sitting; the number in the bracket represents the execution order of the activity. Between each activity, there is an adjustment step for the subject to perform proper sitting activity, prior to execute the next one. For each round of data collection, the subject performs the series of activities 5 times.

5.2. Results comparison

In this section, we compare the recognition results in terms of Accuracy and F-score metrics. Accuracy is used to evaluate the entire performance of the algorithm; it is calculated as Eq. 17.

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{17}$$

where T_n (true negatives) is the correct classifications of negative examples, T_p (true positives) is the correct classifications of positive examples, F_n (false negatives) and F_p (false positives) are, respectively, the positive examples incorrectly classified into the negative classes and the negative examples incorrectly classified into the positive classes.

In addition, we calculated the F-score, according to Eq. 20, which represents the combination of precision and recall, defined, respectively, as follows:

$$precision = \frac{T_p}{T_p + F_p} \tag{18}$$

$$recall = \frac{T_p}{T_p + F_n} \tag{19}$$

$$F - score = \frac{2 \cdot recall \cdot precision}{recall + precision} \tag{20}$$

5.3. Results in static condition

In the static condition, we compare the results using both fixed- and adaptive-time sliding windows. Decision Tree and k-NN classifier algorithms are adopted for this evaluation. Our aim is to prove the effectiveness of the adaptive sliding window algorithm, so any machine learning classifier is suitable. Therefore, we chose Decision Tree as it is easy to construct and realize, and the kNN classifier because it does not require preliminary model construction.

5.3.1. Accuracy comparison analysis

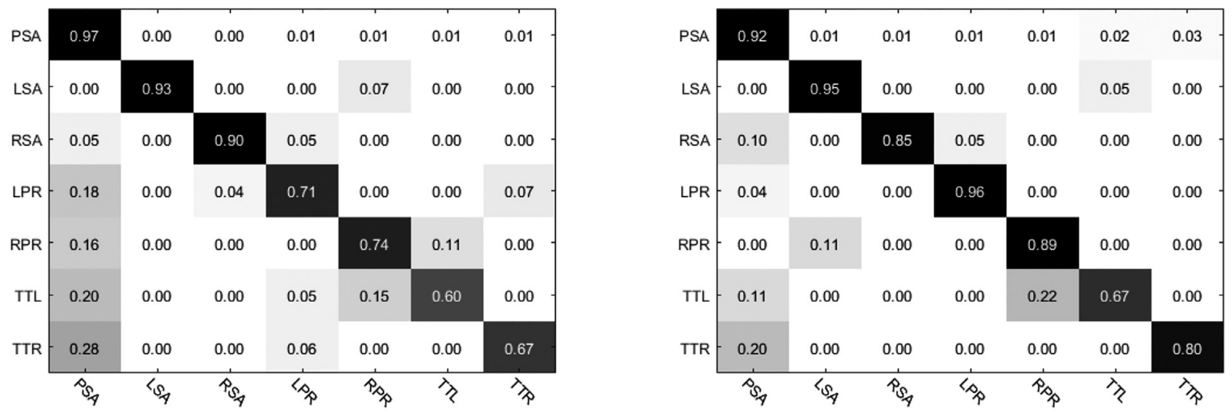
Fig. 7 shows the recognition results using Decision Tree and kNN. As we can see, for the stable posture activity, both the classifiers can achieve good results (above 90%). For the transition posture activity, the kNN classifier performs better than decision tree. As to the non-cumulative posture activity, none of them achieve satisfactory results, as the recognition performance never exceeds 80%.

To analyze the accuracy results, as shown in Table 8, we used 10-fold cross validation. DT stands for Decision Tree, FW is Fixed Window and AW means Adaptive Window.

We observe that in the condition of fixed time window, stable posture activities can be accurately recognized. On the contrary, transition posture and non-cumulative posture activities cannot be recognized with high accuracy. Instead, using the adaptive sliding window, both for the transition posture and non-cumulative activities are recognized better

Table 7
Data collection step in static condition and dynamic condition.

	Activity	Duration	Activity	Duration	Activity	Duration
Static condition	NS(0)	-	PSA(5)	30s	TTL(10)	30s
	PSA(1)	30s	LPR(6)	30s	PSA(11)	30s
	LSA(2)	30s	PSA(7)	30s	TTR(12)	30s
	PSA(3)	30s	RPR(8)	30s	PSA(13)	30s
	RSA(4)	30s	PSA(9)	30s	NS(14)	-
Dynamic condition	NS(0)	-	RSA(4)	30s	TTR(8)	30s
	PSA(1)	30s	PSA(5)	30s	PSA(9)	30s
	LSA(2)	30s	TTL(6)	30s	NS(10)	-
	PSA(3)	30s	PSA(7)	30s		



(a) Recognition results using Decision Tree. (b) Recognition results using kNN.

Fig. 7. Recognition results with two different classifiers.

Table 8
Accuracy comparison using the two time window selection methods in static condition (%).

	PSA	LSA	RSA	LPR	RPR	TTL	TTR
DT&FW	92.6	91.2	86.8	75.2	86.2	73.5	61.6
kNN&FW	93.9	94.4	91	81.3	88.4	76.9	85.4
DT&AW	93.5	92	92.4	87	93.7	88.4	86.6
kNN&AW	94	93	89.6	85.5	93.1	84.4	92.7

than using the fixed window size method. Especially for non-cumulative activities, our adaptive window method significantly outperformed traditional segmentation approaches.

5.3.2. F-Score comparison analysis

We compared the F-score using both fixed window and adaptive sliding window as shown in Table 9. Using the decision tree classifier, the F-score can improve by 15.9%, while with the kNN classifier, the F-score can improve by 15.3%.

Fig. 8 compares the results using the different window selection methods. As we can see in Fig. 8, for the stable posture activity recognition, similar recognition results can be achieved both with fixed time and adaptive sliding windows. For transition posture and non-cumulative activities, on the contrary, we observe better performance obtained with the use of our adaptive sliding window method.

Table 9
Results of F-score using the two time window selection methods in static environment.

	PSA	LSA	RSA	LPR	RPR	TTL	TTR
DT&FW	0.93	0.93	0.93	0.81	0.81	0.68	0.72
kNN&FW	0.93	0.91	0.91	0.70	0.84	0.67	0.76
DT&AW	0.94	0.95	0.94	0.92	0.86	0.87	0.85
kNN&AW	0.95	0.93	0.94	0.89	0.87	0.82	0.84

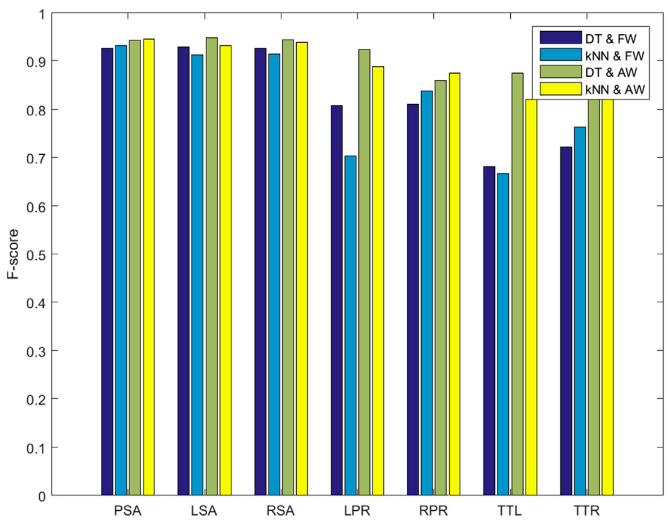
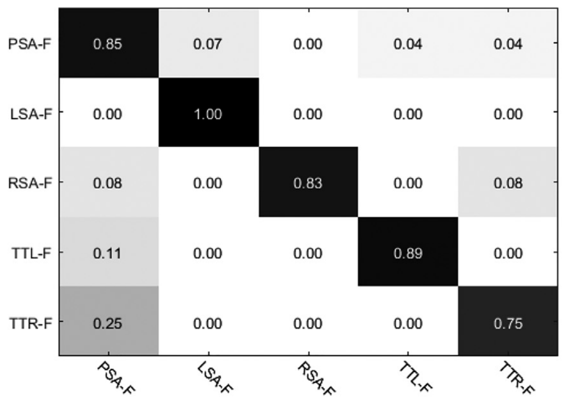


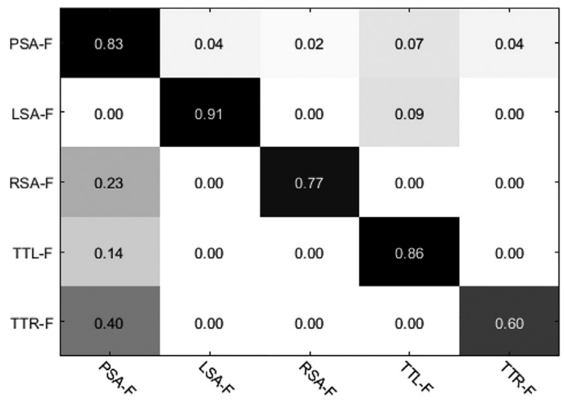
Fig. 8. The recognition results in static condition.

5.4. Results in dynamic condition

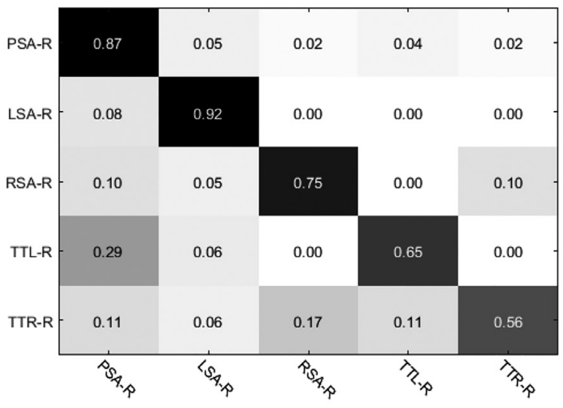
As to the activity recognition in dynamic condition, we perform the experiment on different ground surfaces; specifically, we collect data on flat floor indoor and rough floor outdoor (we use -F and -R postfix notation to represent the two kinds of conditions).



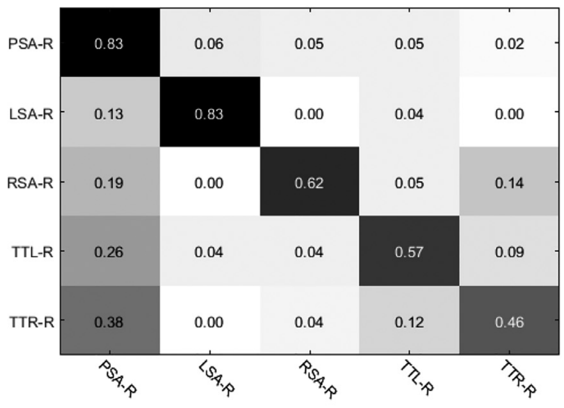
(a) Results using Decision Tree on flat floor.



(b) Results using kNN on flat floor.



(c) Results using Decision Tree on rough floor.



(d) Results using kNN on rough floor.

Fig. 9. Recognition results with the two kinds of classifier.

Table 10
Accuracy comparison using the two time window selection methods in dynamic condition (%).

		PSA	LSA	RSA	TTL	TTR
Flat floor	DT&FW	87.7	91.4	86	80.5	74.5
	kNN&FW	89.4	93.6	91.5	80.7	79.8
	DT&AW	93.5	92.2	91.7	79.3	83.0
	kNN&AW	91.7	92.6	93.1	81.8	81.0
Rough floor	DT&FW	85.2	89.7	86.4	60.3	68.4
	kNN&FW	87.2	87.3	82.4	58.8	56.2
	DT&AW	87.2	92.1	93.5	88	82
	kNN&AW	88.9	91	91.7	84.1	85.5

5.4.1. Accuracy comparison analysis

Fig. 9 shows the recognition results using decision tree and kNN classifier in the two kinds of environments. Compared with Fig. 7, in dynamic environment, only PSA is accurately recognized. When the wheelchair is in motion (dynamic environment), LSA and RSA are often misclassified as PSA, TTL or TTR. In addition, we observe that the non-cumulative posture activity results are lower in the rough floor than in flat floor.

We still use 10-fold cross validation to analyze the activity recognition results in the dynamic environment. As shown in Table 10, using the fixed time window, only with stable posture activity we can obtain good results. Using the adaptive sliding window, the recognition results of each activity increase. On flat floor, both the adaptive and fixed window method could recognize postural activities with about 90% accuracy. For non-cumulative activities, the recognition results drops instead around 80%, with the adaptive window method slightly outperforming the fixed time window approach. On rough floor, our proposed adaptive window method clearly shows improvement on recognizing non-cumulative activities.

5.4.2. F-Score comparison analysis

We compared the F-score using fixed window and adaptive sliding window, as shown in Table 11. Using the decision tree classifier, the F-score can improve by 12.3%, while with the kNN classifier, the F-score can improve by 6.4%. Especially in the rough floor, the F-score value improves by 34.4% and 24.5% using decision tree and kNN classifier, respectively.

Fig. 10 summarizes the comparison of different methods. For the stable posture activity recognition, recognition results are similar with both the fixed time window and adaptive sliding window methods. With transition posture and non-cumulative activities, however, it can be easily observed that the adaptive sliding window performs better.

Table 11
Results of F-score using two kinds of time window methods in dynamic environment.

	PSA	LSA	RSA	LPR	RPR	TTL	TTR
DT&FW	0.93	0.93	0.93	0.81	0.81	0.68	0.72
kNN&FW	0.93	0.91	0.91	0.70	0.84	0.67	0.76
DT&AW	0.94	0.95	0.94	0.92	0.86	0.87	0.85
kNN&AW	0.95	0.93	0.94	0.89	0.87	0.82	0.84

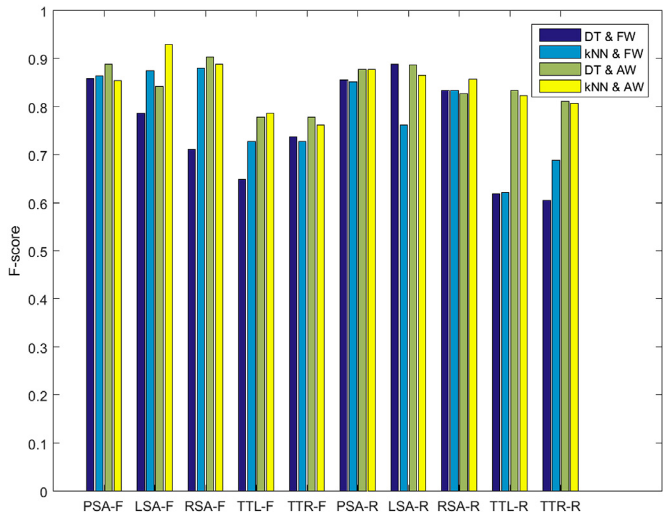


Fig. 10. The recognition results in dynamic condition.

6. Conclusion and future works

In this paper, we proposed an adaptive sliding window method for human activity recognition. This research contributes to the state-of-the-art in two distinct ways. Firstly, an architecture of adaptive sliding time window method was realized to recognize different sitting activities for wheelchair users. Secondly, a MGD-based dynamic time window adjustment algorithm was proposed to tune, by expansion and contraction, the window size of each individual activity before the classification step. To validate our algorithm, we constructed the test datasets collected from wheelchair users under static and dynamic conditions. The proposed method was compared against the traditional fixed time window size method in terms of accuracy and F-score. The results demonstrated that the proposed adaptive sliding window method can improve the activity recognition accuracy in both static and dynamic conditions.

In the following the limitations of the proposed adaptive sliding time window method are reported.

- The determination of the start time and end time of the activity increases the computation load, so the recognition results will be delayed.
- Although the proposed method is general-purpose and could be applied in many activity recognition contexts, currently our algorithm is validated only on wheelchair-related activities.

In the near future, we will devote efforts to employ the proposed method in a real time recognition prototype system. Computation load and time complexity of the algorithm will be investigated to optimize embedded implementations. Furthermore, we are planning to apply this method to recognize additional types of daily life activities for the wheelchair users. Finally, we will integrate our method in the SPINE framework [35] and the BodyCloud middleware [36] to provide reusable application modules to support the development of BAN-based systems for impaired individuals.

Acknowledgment

This research is financially supported by China-Italy Science & Technology Cooperation project, “Smart Personal Mobility Systems for Human Disabilities in Future Smart Cities” (China-side Project ID: 2015DFG12210, Italy-side Project ID: CN13MO7), and the National Natural Science Foundation of China (Grant No: 61571336, 61502360 and 61603280). This work has been also carried out under the framework of INTER-IoT, Research and Innovation action—Horizon 2020 European Project Grant Agreement #687283, financed by the European Union. The authors would like to thank the volunteers who participated in the experiments for their efforts and time.

References

[1] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, R. Jafari, Enabling effective programming and flexible management of efficient body sensor network applications, IEEE Trans. on Human-Mach. Syst. 43 (1) (2013) 115–133.
[2] R. Gravina, P. Alinia, H. Ghasemzadeh, G. Fortino, Multi-sensor fusion in body sensor networks: state-of-the-art and research challenges, Inform. Fusion 35 (2017) 68–80.
[3] M. Chen, Y. Ma, Y. Li, D. Wu, Y. Zhang, C.H. Youn, Wearable 2.0: enabling human–cloud integration in next generation healthcare systems, IEEE Commun. Mag. 55 (1) (2017) 54–61.
[4] R. Gravina, C. Ma, P. Pace, G. Aloi, W. Russo, W. Li, G. Fortino, Cloud-based activity-aaservice cyberphysical framework for human activity monitoring in mobility, Future Gener. Comput. Syst. 75 (2017) 158–171.
[5] R. Gravina, Q. Li, Emotion-relevant activity recognition based on smart cushion using multi-sensor fusion, Inform. Fusion 48 (2019) 1–10.
[6] C. Ma, W. Li, J. Cao, R. Gravina, G. Fortino, Cloud-based wheelchair assist system for mobility impaired individuals, in: International Conference on Internet and Distributed Computing Systems, 2016, pp. 107–118.
[7] S. Galzarano, R. Giannantonio, A. Liotta, G. Fortino, A task-oriented framework for networked wearable computing, IEEE Trans. Autom. Sci. Eng. 13 (2) (2014) 621–638.
[8] G. Fortino, S. Galzarano, R. Gravina, W. Li, A framework for collaborative computing and multi-sensor data fusion in body sensor networks, Inform. Fusion 22 (2015) 50–70.
[9] H. Ghasemzadeh, P. Panuccio, S. Trovato, G. Fortino, R. Jafari, Power-aware activity monitoring using distributed wearable sensors, IEEE Trans Hum Mach Syst 44 (4) (2014) 537–544.
[10] A.H. Sodhro, S. Pirbhulal, G.H. Sodhro, A. Gurtov, M. Muzammal, Z. Luo, A joint transmission power control and duty-cycle approach for smart healthcare system, IEEE Sens J (2019), doi:10.1109/JSEN.2018.2881611.
[11] A.H. Sodhro, P. Sandeep, Q. Marwa, L. Sonia, G.H. Sodhro, Z. Luo, Power control algorithms for media transmission in remote healthcare systems, IEEE Access 6 (2018) 42384–42393.
[12] A.H. Sodhro, S. Pirbhulal, A.K. Sangaiah, Convergence of iot and product lifecycle management in medical health care, Future Gener. Comput. Syst. 86 (2018) 380–391.
[13] A.H. Sodhro, A.S. Malokani, G.H. Sodhro, M. Muzammal, L. Zongwei, An adaptive qos computation for medical data processing in intelligent healthcare applications, Neural Comput. Appl. 30 (3) (2019) 1–12.
[14] A.H. Sodhro, S. Pirbhulal, A.K. Sangaiah, S. Lohano, G.H. Sodhro, Z. Luo, 5G-based transmission power control mechanism in fog computing for internet of things devices, Sustainability 10 (4) (2018).
[15] A.H. Sodhro, M.A. Shah, Role of 5g in medical health, in: International Conference on Innovations in Electrical Engineering & Computational Technologies, 2017, pp. 1–5.
[16] Z. Wang, D. Wu, R. Gravina, G. Fortino, Y. Jiang, K. Tang, Kernel fusion based extreme learning machine for cross-location activity recognition, Inform. Fusion 37 (C) (2017) 1–9.
[17] M. Berchtold, M. Budde, H.R. Schmidtke, M. Beigl, An extensible modular recognition concept that makes activity recognition practical, in: Advances in Artificial Intelligence, German Conference on Ai, Karlsruhe, Germany, September 21–24., 2010, pp. 400–409.
[18] W. Li, L. Wang, D. Yao, H. Yu, Fuzzy recognition of abnormal body behaviors based on motion feature analysis, J. Huazhong Univ. Sci. Technol. 42 (2014) 87–91.
[19] E.M. Tapia, S.S. Intille, W. Haskell, K. Larson, J. Wright, A. King, R. Friedman, Real–time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor, in: IEEE International Symposium on Wearable Computers, 2007, pp. 37–40.
[20] G. Wang, Q. Li, L. Wang, W. Wang, M. Wu, T. Liu, Impact of sliding window length in indoor human motion modes and pose pattern recognition based on smartphone sensors, Sensors 18 (6) (2018) 1965.
[21] Z. Chen, Q. Zhu, S.Y. Chai, L. Zhang, Robust human activity recognition using smartphone sensors via ct-pca and online svm, IEEE Trans. Ind. Inf. (99) (2017). 1–1
[22] M.H.M. Noor, Z. Salcic, I.K. Wang, Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer, Pervasive Mobile Comput. (2016) 102–107.
[23] Q. Ni, L. Zhang, L. Li, A heterogeneous ensemble approach for activity recognition with integration of change point-based data segmentation, Appl. Sci. 8 (9) (2018) 1695.
[24] J.A. Ward, P. Lukowicz, G. Troster, T.E. Starner, Activity recognition of assembly tasks using body-worn microphones and accelerometers., IEEE Trans. Pattern Anal. Mach Intell. 28 (10) (2006) 1553–1567.
[25] H. Dong, I. Ugalde, N. Figueroa, S.A. El, Towards whole body fatigue assessment of human movement: a fatigue-tracking system based on combined semg and accelerometer signals, Sensors 14 (2) (2014) 2052.
[26] C. Ma, W. Li, R. Gravina, G. Fortino, Posture detection based on smart cushion for wheelchair users, Sensors 17 (4) (2017) 719.
[27] S.H. An, D.S. Park, The effects of trunk exercise on mobility, balance and trunk control of stroke patients, J. Korean Soc. Phys. Med. 12 (1) (2017) 25–33.
[28] A. Akbari, J. Wu, R. Grimsley, R. Jafari, Hierarchical signal segmentation and classification for accurate activity recognition, in: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, 2018, pp. 1596–1605.

[29] C. Ma, W. Li, R. Gravina, J. Cao, Q. Li, G. Fortino, Activity level assessment using a smart cushion for people with a sedentary lifestyle, Sensors 17 (10) (2017) 2269.
[30] C. Ma, R. Gravina, W. Li, Y. Zhang, Q. Li, G. Fortino, Activity level assessment of wheelchair users using smart cushion, in: Eai International Conference on Body Area Networks, 2016, pp. 104–110.
[31] Arduino website (2019). https://www.arduino.cc/en/guide/arduinopromini, June 2019.
[32] Fsr website (2019). http://www.interlinkelectronics.com, June 2019.
[33] MPU9250 website (2019). https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/, June 2019.
[34] P. Gupta, T. Dallas, Feature selection and activity recognition system using a single triaxial accelerometer, IEEE Trans. Biomed. Eng. 61 (6) (2014) 1780–1786.
[35] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, Spine: a domain-specific framework for rapid prototyping of wbsn applications, Softw. Pract. Exper. 41 (3) (2015) 237–265.
[36] G. Fortino, G.D. Fatta, M. Pathan, A.V. Vasilakos, Cloud-assisted body area networks: state-of-the-art and future challenges, Wirel. Netw. 20 (7) (2014) 1925–1938.