ActDetector: Detecting Daily Activities Using Smartwatches

Xiao Sun, Li Qiu, Yibo Wu and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA
{xxs118, lyq5023@psu.edu, yxw185@psu.edu, gcao}@cse.psu.edu

***Abstract*—Detecting daily activities is helpful for health care and clinical medicine. In this paper, we present ActDetector, a smartwatch based application which detects 8 common daily activities, including sitting, walking, running, going upstairs, going downstairs, eating, driving and sitting in a vehicle. By leveraging the built-in sensors on smartwatch, a multi-level classification system is proposed which considers both detection accuracy and energy efficiency. ActDetector is designed to work unobtrusively, no matter on which wrist the smartwatch is worn. We have implemented ActDetector on Sony Smartwatch 3 and evaluated its performance in real experiments involving 12 users. Experimental results show that ActDetector is energy efficient and can detect the daily activities with high accuracy.**

I. Introduction

Activities such as sitting, walking or eating, reflect routine movements in our daily lives. Monitoring these daily activities provides helpful information for health care and clinical medicine. For example, in [1], by analyzing the walking data of patients with stroke, researchers evaluated the efficacy of a task-oriented intervention in stroke rehabilitation, and in [2], eating activity was monitored to help people with obesity assess dietary intake and establish healthy eating habits. Even for people in good health, maintaining activity dairies for over an extended period of time plays an important role in improving their fitness. For example, by detecting and recording activities like sitting and walking, daily physical activities can be checked to help decide if more physical exercises are needed or not.

The rapid development of wearable technologies such as smartphones and smartwatches, has opened up opportunities for smart health [3]–[5], and it is possible to detect and record a user's health related daily activities unobtrusively by using wearable devices. Since smartwatch is integrated with motion sensors (e.g., accelerometer, gyroscope, etc.) and worn by a user on his/her wrist most of the time, we can collect motion related sensor data from the smartwatch to detect daily activities through analysis of wrist movement patterns. In this paper, we present a smartwatch based system called ActDetector, which leverages two widely equipped motion sensors, accelerometer and gyroscope, to detect 8 daily activities, including *sitting, walking, running, going upstairs, going downstairs, eating, driving,* and *sitting in a vehicle*. ActDetector is designed to record these activities in an unobtrusive way without interrupting users' normal smartwatch wearing and usage patterns.

As a popular research area with potential applications in health care, activity detection has been studied in previous works [6]–[12]. Radio based systems have been proposed in [6]–[10] to detect human activities by using radar or Channel State Information (CSI). However, these systems rely on the wireless devices installed indoor, which makes them unsuitable for detecting activities outdoor. In [11], nodes with motion sensors are placed at different parts of the body and data from all these sensor nodes were collected for activity detection. Although the system can work both indoor and outdoor, it is not practical for daily monitoring since users have to wear multiple sensor nodes on their wrists or ankles. Smartphone based approaches are proposed in [12], [13], where inertial sensors equipped on the smartphone are exploited for activity detection. However, in these approaches, to achieve more accurate detection, the smartphone has to be placed at a fixed position with respect to the body, which is inconvenient for users. Some simple activities such as walking or running can be detected by commercial wrist-worn devices like Fitbit or Jawbone. However, to use these devices, users have to set the preferred wrist (i.e., the wrist on which the device is worn) manually and these devices cannot detect complicated activities such as eating or driving.

Different from the aforementioned works, ActDetector is implemented on smartwatch to detect 8 common daily activities. Several practical issues are considered in developing ActDetector to meet the specific requirements of smartwatch, such as sensing capabilities and resource constraints of smartwatch, and different wearing habits of smartwatch users. The accelerometer and gyroscope data is collected and a multi-level classification system is designed for activity detection. First, a few time-domain features are extracted from the accelerometer data to detect the predominant activity sitting. Then, the remaining activities are classified into two categories, depending on whether they are activities with repeated patterns or not. Lastly, more time-domain and frequency-domain features are extracted and different classifiers are trained to identify each daily activity. ActDetector is designed to work unobtrusively and robustly, and it functions no matter on which wrist the smartwatch is worn. We have implemented ActDetector on Sony Smartwatch 3 and evaluated its performance in real experiments involving 12 users. Experimental results show that ActDetector is energy efficient and can detect daily activities with high accuracy.

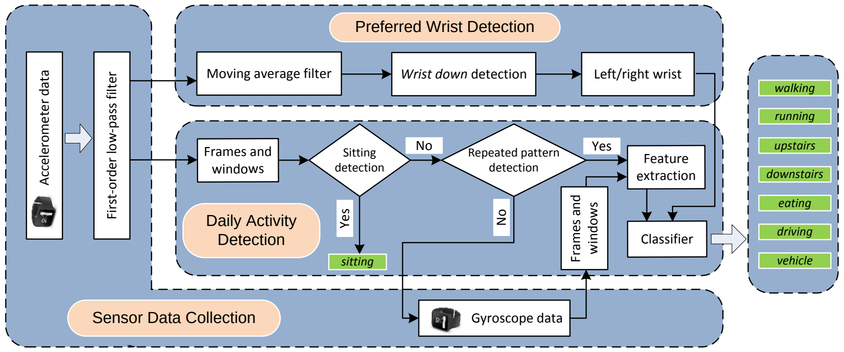


Fig. 1: The architecture of ActDetector.

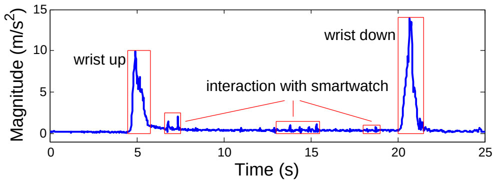


Fig. 2: The magnitude of the total acceleration when starting an app on smartwatch.

The rest of this paper is organized as follows. Section II discusses the design considerations. Section III presents the system architecture and illustrates the design details of each component. The implementation of ActDetector on Android Wear based smartwatch is presented in Section IV. We evaluate the performance of our system in Section V. Section VI reviews the related work and Section VII concludes the paper.

II. Design Considerations

ActDetector is designed to unobtrusively detect and record a user's daily activities using off-the-shelf smartwatches. To achieve this goal, the following design issues should be considered.

First, the system should use sensors which are commonly built in different types of smartwatches. A lot of smartwatch manufacturers (e.g., Sony, LG, Apple, Samsung, Huawei) have released their smartwatches recently. Depending on the brand and the device version, these smartwatches are equipped with different kinds of sensors. For example, the LG G Watch R is equipped with barometer, which can be used to measure the floors a user climbs, and the Samsung Gear S2 3G is equipped with GPS, which can be used to track a user's movement. However, these two sensors are not supported by many other smartwatches. To make ActDetector device independent, it should only use sensors which are commonly equipped on different types of smartwatches.

Second, the system must be lightweight. ActDetector is implemented on smartwatch to continuously detect and record the common activities occurred in user's daily life. However, compared to other mobile devices like smartphone, smartwatch usually has much less computing resources and power. To monitor user's daily activities for hours or even days, a large amount of sensor data needs to be collected and processed on smartwatch. Therefore, ActDetector must be lightweight.

Third, to make ActDetector work unobtrusively and robustly, users' wearing habits should be considered when designing the system. Although most users wear smartwatches on their wrists, different users have different preferences. Some may prefer wearing smartwatches on their left wrists, while others may prefer the right wrists. When a user performs a daily activity (e.g., walking, running or climbing stairs), the motion related sensor data collected from the left wrist and right wrist are different under most circumstances. Therefore, ActDetector should be able to detect the daily activities no matter on which wrist the smartwatch is worn.

III. System Design

In this section, we present the design of ActDetector. As shown in Figure 1, there are three main components in ActDetector. The sensor data is sampled from the accelerometer and gyroscope for processing. By using the accelerometer data, a light weight scheme is designed to detect on which wrist the user wears the smartwatch. Based on the detected result, the corresponding classifiers are chosen and a group of features are extracted from sensor data to identify each daily activity. The design details of each component are described as follows.

A. Sensor Data Collection

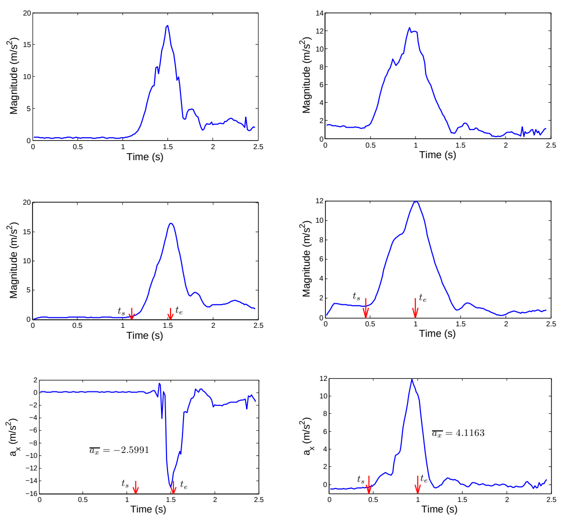
In ActDetector, accelerometer data and gyroscope data are collected and analyzed to detect daily activities. Once ActDetector is started, the accelerometer data is sampled with a sampling rate of 60 Hz. Since gyroscope consumes much more power than accelerometer, the angular velocity is sampled with a sampling rate of 100 Hz only when an activity with non-repeated pattern is detected (details in Section III-C). The accelerometer on the smartwatch measures all the accelerations that affect the device, which also include the gravity. However, in ActDetector, to detect a user's daily activities, we are only interested in the acceleration which is associated with the movement of the smartwatch. Therefore, a low-pass filter is applied to subtract the gravity components along X, Y and Z axis from the raw accelerometer data. For simplicity, in the remainder of this paper, we use acceleration to denote the filtered acceleration without the effect of gravity.

B. Preferred Wrist Detection

When a user performs a certain daily activity (e.g., walking, running, etc.), the movement of his/her wrist follows some patterns. By collecting accelerometer data from the wrist-worn smartwatch, ActDetector is able to capture the wrist movement patterns, and further detect which activity the user is performing. Although it is common for users to wear the smartwatches on the wrists, their preferred wrists may be different (i.e., some users prefer wearing smartwatches on their left wrists, while others prefer the right wrists). During a certain daily activity, the acceleration data obtained on the left wrist will be different from that obtained on the right wrist. Therefore, before using the accelerometer data to detect activities, ActDetector should detect whether the smartwatch is worn on the user's left or right wrist.

Figure 2 shows the magnitude of the total acceleration by combining the three accelerations along X, Y and Z axis (denoted as a_x, a_y, a_z respectively) when a smartwatch user starts an app. As can be seen, there are normally three wrist movement phases when a user starts an app: *wrist up*, *interaction with smartwatch* and *wrist down*. In the phase of *wrist up*, the user moves his/her preferred wrist (i.e., the one with the smartwatch worn on) towards his/her body and hold it for a while in front of his/her face to start the app. After opening the app with voice commands or touchscreen in *interaction with smartwatch*, the user moves his/her wrist down towards the other side of the body in the third phase *wrist down*. In both *wrist up* and *wrist down*, the magnitude of the total acceleration on the preferred wrist rises up and drops down very quickly in a short period. However, the moving directions of different wrists are just opposite (i.e., the left wrist moves from left to right in *wrist up* and from right to left in *wrist down*, while the right wrist moves the opposite way). Therefore, by analyzing the acceleration variation on the wrist when an app is started, we can detect *wrist up* and *wrist down*, and then infer on which wrist the smartwatch is worn. However, to detect *wrist up*, a daemon process which continuously collects accelerometer data is necessary because it is difficult to know when the user will start an app. Different from *wrist up*, the phase of *wrist down* starts after the app is opened, which has a deterministic starting point. Therefore, in ActDetector, we have the system started collecting accelerometer data once it is opened and detecting the user's preferred wrist by analyzing the phase of *wrist down*.

Figure 3 depicts the magnitude of the total acceleration and X axis acceleration during *wrist down* when the smartwatch is worn on the left and right wrist respectively. As shown in the bottom figures, since the X axis acceleration is obtained based on the smartwatch's local coordinate system and in this coordinate system the left and right wrist moves towards different directions along the X axis during *wrist down*, most of the X axis acceleration values are negative on the left wrist and positive on the right wrist. Therefore, to detect the preferred wrist, we first detect the period of *wrist down* and then check whether the X axis acceleration is positive or negative during this period. In practice, after the wrist is put down, it will shake slightly for a little while rather than becoming still immediately, which makes it difficult to detect the endpoint of *wrist down*. For simplicity, in ActDetector, we detect the first half of *wrist down* where the magnitude of the total acceleration increases. After ActDetector is started, the accelerations along the three axes a_x, a_y and a_z are collected and the total acceleration a is calculated. Then a *Moving Average Filter* [14] is used to filter out the noise in the total acceleration as shown in the middle figures of Figure 3. In the



(a) Left wrist (b) Right wrist

Fig. 3: The accelerometer data during *wrist down* when the smartwatch is worn on the left and right wrist respectively.

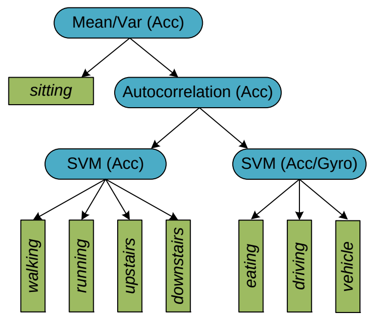


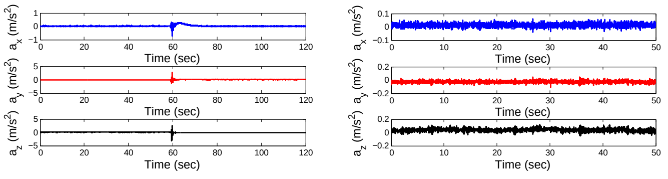
Fig. 4: Activity detection process.

filtered data set of the total acceleration, ActDetector detects a continuously increasing data sequence whose length is larger than a predefined threshold δ (δ is set to 10 in our system) and marks $[t_s, t_e]$ as the time period of the first half of *wrist down*, where t_s and t_e are the time stamps of the beginning and ending of the detected data sequence. Then, the X axis acceleration in $[t_s, t_e]$ is analyzed to detect the preferred wrist. To avoid the effect of the noise, we calculate the average a_x sampled in $[t_s, t_e]$ (denoted as $\overline{a_x}$), and as shown in the bottom figures of Figure 3, the preferred wrist is detected as left if $\overline{a_x} < 0$ and right otherwise.

In ActDetector, two sets of parameters are trained based on the training data collected from left wrist and right wrist respectively. Once the preferred wrist is detected, the particular set of parameters is used for activity detection. For the remainder of the paper, the system is illustrated based on the assumption that the sensor data is collected from the user's preferred wrist.

C. Daily Activity Detection

After the preferred wrist is detected, the sensor data is sampled and segmented to detect 8 common daily activi-



(a) (b)

Fig. 5: The accelerometer data along three axes during sitting.

ties, including sitting, walking, running, going upstairs, going downstairs, eating, driving and sitting in a vehicle (denoted as *sitting*, *walking*, *running*, *upstairs*, *downstairs*, *eating*, *driving* and *vehicle* respectively). As shown in Figure 4, a few time domain features are firstly extracted to detect *sitting* since it is a predominant activity in a user's daily life and can be easily detected with little computing resource. Then, autocorrelation is calculated to classify the activities as those with repeated patterns (i.e., *walking*, *running*, *upstairs* and *downstairs*) and with non-repeated patterns (i.e., *eating*, *driving* and *vehicle*). Lastly, Support Vector Machine (SVM) is applied to each category to classify all the activities.

1) Sensor Data Segmentation: In order to extract features for classification, a common practice is to segment the sensor data into frames. In our implementation, the sampled sensor data is segmented into 1-second frames. Although frame based features are able to capture some characteristics of the activities, they are insufficient to distinguish different activities accurately since the characteristics of some activities like *eating* or *driving* may be involved in multiple frames longer than 1 second. Therefore, after the samples are framed, we group every 8 continuous frames together as a window. Both frame based and window based features are extracted for activity detection.

2) Sitting Detection: Among all the predefined activities, *sitting* happens much more often than others. As observed, many people spend most of their daily time sitting at desks or in front of computers for study or work, especially for those who work at offices. To detect this predominant activity, a lightweight scheme is designed where only a few time domain features need to be extracted from the accelerometer data.

Figure 5a depicts 120-second long accelerometer data along the X, Y and Z axis when a user is sitting at the desk, and to show it more clearly, the first 50-second data is zoomed in and shown in Figure 5b. As shown in these figures, since a user's wrist remains still for most of the time in *sitting*, the corresponding accelerations along all three axes are close to 0 and vary slightly. These characteristics can be captured by two wildly used statistical measures: *mean* and *variance*. Therefore, for each frame f, we calculate its *mean* and *variance* along each axis ax (denoted as $mean_{ax}(f)$ and $var_{ax}(f)$ respectively) and set two thresholds γ_{ax}^{mean} and γ_{ax}^{var} to detect *sitting* frame. Any frame f with $|mean_{ax}(f)| < \gamma_{ax}^{mean}$ and $var_{ax}(f) < \gamma_{ax}^{var}$ will be detected as a *sitting* frame. In our implementation, the thresholds γ_x^{mean}, γ_x^{var}, γ_y^{mean}, γ_y^{var}, γ_z^{mean} and γ_z^{var} are empirically set to 0.15, 0.01, 0.15, 0.01, 0.15 and 0.01 respectively.

In practice, when a user is sitting, although his/her wrist will stay still for most of the time, it may move occasionally. As shown in Figure 5a, there is a wrist movement around the 60-th second caused by the change of sitting position, while the user is still in *sitting*. Since this kind of wrist movement lasts very shortly (less than 4 seconds as observed from our dataset), we classify the activity contained in a window as *sitting* as long as more than half of the frames in the window are *sitting* frames. If a *sitting* activity is detected, the sampling window is labeled and discarded; otherwise, it is preserved for further processing.

3) Repeated Pattern Detection: Sitting activities can be detected with little CPU and power consumption as described above. If an activity is not *sitting*, more features need to be extracted and more precise schemes are needed to classify it. However, it is difficult to design only one classifier to accurately classify all the rest of 7 activities. As shown in Figure 6, among the remaining activities, some are with repeated patterns, such as *walking*, *running*, *upstairs* and *downstairs*, and some are not, such as *eating*, *driving* and *vehicle* (denoted as *repeated* and *non-repeated* activities respectively). Therefore, before feeding the preserved sampling window into a classifier, we first detect whether it contains a *repeated* activity or a *non-repeated* activity.

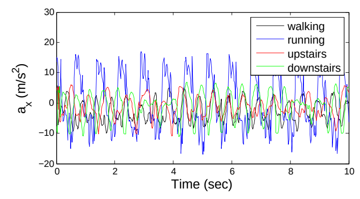
We use *autocorrelation* [15] for repeated pattern detection in our system. In signal processing, autocorrelation is the cross correlation between a signal and itself with different shifts. For a series of sensor samples $\{s_1, s_2, ..., s_i, ..., s_N\}$, the autocorrelation with shift k is defined as:

$$R(k) = \frac{\sum_{i=1}^{N-k} s_i s_{i+k}}{\sum_{i=1}^{N} s_i^2}$$

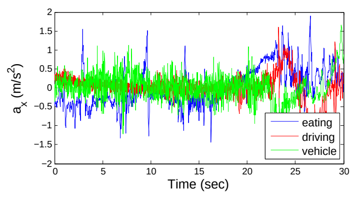
where the denominator $\sum_{i=1}^{N} s_i^2$ is used for normalization.

According to the definition, autocorrelation measures the similarity between a segment and a shifted version of itself. It can be proved that $R(k)$ is maximized when the sensor samples overlap itself (i.e., $k = 0$). As shown in Figure 7, if the sample sequence is periodic with period T, when the shift $k = mT$ ($m \in Z$), there is overlap between the sample sequence and its shifted version, and autocorrelation at these points will be large. Otherwise, if the sample sequence is non-periodic, autocorrelation will gradually decrease as k increases.

As observed in our data set, for *repeated* activities like *walking*, *running*, *upstairs* and *downstairs*, their periods are less than 1.5 seconds. Thus, in our implementation, we calculate the autocorrelation of the first 180 samples in a window (i.e., 3 seconds) with shift k increasing from 1 to 90 (i.e., 1.5 seconds) to detect the repeated pattern. After the autocorrelation is calculated, peak detection algorithms [16] can be used to detect if there are peaks in the autocorrelation or not. If peaks are detected, the window contains a *repeated* activity and the period is the corresponding k where the first peak occurs; otherwise, the window contains a *non-repeated* activity. Then,

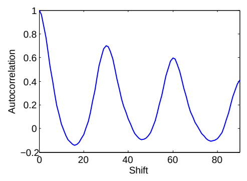


(a) *Repeated* activities

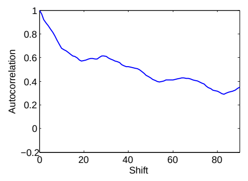


(b) *Non-repeated* activities

Fig. 6: The accelerometer data along X axis for *repeated* and *non-repeated* activities.



(a) Walking



(b) Eating

Fig. 7: Autocorrelation of 3-second accelerometer data along X axis when the user performs a *repeated* activity (*walking*) and a *non-repeated activity* (*eating*) respectively.

TABLE I: Features extracted in ActDetector.

Feature	Description	Category
mean_acc	the mean of the acceleration along each axis	*repeated* activities and *non-repeated* activities
var_acc	the variance of the acceleration along each axis	
cov_acc	the covariance between the acceleration along each pair of axes	
sc	the spectral centroid of the FFT spectrum for the acceleration along each axis	
sr(λ)	the spectral rolloff of the FFT spectrum for the acceleration along each axis	
period	the period detected in the acceleration along each axis	*repeated* activities
mean_gyro	the mean of the angular velocity along each axis	*non-repeated* activities
var_gyro	the variance of the angular velocity along each axis	
cov_gyro	the covariance between the angular velocity along each pair of axes	

based on the detection result, the corresponding classifier is applied to identify the activity contained in the window.

4) Feature Extraction and Classification: Extracting the proper features is important for classifying activities in each category. In our system, both time-domain and frequency-domain features are extracted. As listed in Table I, for time-domain features, we extract the *mean* and *variance* of the sensor data along each axis. In addition, the *covariance* between sensor data along every two different axes is extracted to describe the relationship between motions on these two axes during a certain activity. For *repeated* activities, the *period* detected in the previous step is also considered as a time-domain feature. The frequency-domain features *spectral centroid* and *spectral rolloff* are extracted based on the Fast Fourier Transform (FFT). Let p_i $(i = 1, 2, ..., n)$ denote the normalized magnitude of the i-th frequency bin obtained by using FFT, and spectral centroid [17] sc is calculated as:

$$sc = \frac{\sum_{i=1}^{n} i \cdot p_i^2}{\sum_{i=1}^{n} p_i^2}$$

Given a threshold λ $(0 < \lambda < 1)$, spectral rolloff $sr(\lambda)$ is calculated as:

$$sr(\lambda) = max(h|\frac{\sum_{i=1}^{h} p_i}{\sum_{i=1}^{n} p_i} < \lambda)$$

Spectral centroid measures the centroid of the spectral power distribution, and spectral rolloff measures the frequency bin below which λ of the spectral power is concentrated. In our system, $\lambda = 0.1, 0.5, 0.9$ and 3 spectral rolloff values are extracted.

Due to the periodicity, the characteristics of activities like *walking*, *running*, *upstairs* and *downstairs* can be captured from a sampling window of acceleration, and the above time-domain and frequency-domain features extracted from accelerometer data can be utilized to classify *repeated* activities accurately. However, for *non-repeated* activities, using only accelerometer data is insufficient to distinguish each activity accurately. To improve the classification accuracy for *non-repeated* activities, we also collect sensor data from the gyroscope, which measures angular velocities along three axes in the smartwatch's local coordinate system. As shown in Table I, only time-domain features are extracted from gyroscope data.

After features are extracted, SVM classifiers are trained to classify *repeated* and *non-repeated* activities. SVM is an effective supervised learning scheme in machine learning, and it has been widely used in different kinds of classification problems [18]. In our system, we use one-against-one strategy to build multi-class SVM classifiers to classify *repeated* and *non-repeated* activities.

Since most of the activities detected in our system are continuous, it is highly unlikely to have an activity interrupted by another activity which lasts very shortly. For example, it is not likely to have a sequence of windows containing activities like *sitting*, *sitting*, *sitting*, *driving*, *sitting*, *sitting*. Therefore, after windows sampled in 30 minutes are processed and labeled, the detection results are checked again and the isolated window whose predicted label is different from those of the prior and successive two windows will be relabled the same as its neighboring windows (e.g., in the previous example, *driving* will be relabled as *sitting*).

IV. Implementation

Sony Smartwatch 3 is chosen in our implementation, using Android Wear 1.3. When ActDetector is started, a sampling thread starts the accelerometer and begins reading data at 60 Hz. The accelerometer on Sony Smartwatch 3 can work at this sampling rate when it is registered in Android Wear OS with the parameter *SENSOR_DELAY_GAME*. Once the accelerometer data is sampled, a simple low-pass filter is applied to the raw data immediately to filter out gravity. This low-pass filter is always executed as long as accelerometer data is sampled. The filtered sensor data is then buffered and processed by a thread for preferred wrist detection. After the preferred wrist is detected, the buffered accelerometer data is dropped and the corresponding configurations for the particular wrist are chosen and used for the rest of time.

From then on, once 480 acceleration samples (i.e., 8 seconds) are collected by the sampling thread, they are fed to a processing thread as a window for activity detection. The processing thread segments the window into frames of 60 samples and calculates mean and variance for each frame. The predefined mean and variance thresholds illustrated in *Sitting Detection* are utilized to detect if a frame contains *sitting* or not. If more than 4 frames contain *sitting* activities, the window is labeled as *sitting* and the samples are discarded; otherwise, the window is preserved for further processing. For the preserved window, the autocorrelation of the first 180 samples (i.e., 3 seconds) with shift increasing from 1 to 90 is calculated to detect if the window contains activity with repeated pattern or not. Whether repeated pattern is detected or not, the window level time-domain and frequency-domain features are extracted and the SVM classifier is used to classify the activity contained in the window. However, if no repeated pattern is detected, the gyroscope will be started and features extracted from angular velocity will be used in the next sampling window. After a window is processed, the samples are discarded and the processing thread ends. The predicted labels are stored in a buffer. When 225 windows (i.e., 30 minutes) are processed, a recording thread checks the predicted labels in the buffer, and relabels the isolated window whose label is different from its prior and successive 2 windows. Then, the labels and the system times for the corresponding windows are written into a file for record.

In the processing thread, different SVM classifiers are used in different cases. To save the smartwatch's CPU and power, all the SVM classifiers in ActDetector are trained offline using Libsvm [19], and the support vectors and the coefficients are then provided to the processing thread for classification. Depending on the preferred wrist detection result, the classifiers built based on training data collected from the preferred wrist will be chosen. To classify a non-sitting activity, a classifier consisting of 6 binary SVMs is trained based on accelerometer data if the activity is *repeated*, and a classifier consisting of 3 binary SVMs is trained based on accelerometer and gyroscope data otherwise. Since gyroscope consumes more power than accelerometer, the sampling thread starts the gyroscope only when a *non-repeated* activity is detected. However, for the first window containing *non-repeated* activity, it only has accelerometer data. Since this happens rarely, in our implementation, this window is labeled the same as the following one. While the gyroscope is working, if *sitting* or a *repeated* activity is detected, the sampling thread stops the gyroscope to reduce power consumption.

Since activities like *sitting*, *walking*, *running*, *eating*, *driving* and *vehicle* will usually last for minutes or even hours, we can use duty cycle to save power and increase the smartwatch's battery life. If 5 continuous windows contain the same activity which may last for a while, the system enters a duty cycle state in which only one in every 3 windows is processed (i.e., 8 seconds out of every 24 seconds). If a window processed in the duty cycle state is detected as containing a different activity from the previous one, the system goes back to the normal state. The performance of using different duty cycles is evaluated in Section V.

V. Performance Evaluations

A. Experimental Setup

In order to collect sensor data and the ground-truth labels for the corresponding activities, two schemes are used in most of the existing works [7], [20], [21]. In [20], the sensor data is collected by smartphones automatically while users are performing some activities in their daily lives, and then users are asked to annotate and label their activities to provide ground truth. In [7], [21], to collect sensor data for a certain activity, users are asked to perform this activity under supervision in lab environment, and then smartphones are started for data collection. However, both of these schemes have disadvantages. For the first scheme, the labels for the activities may be incorrect due to users' carelessness or poor memories. For the second scheme, the sensor data may not reflect the activity accurately since a user may perform an activity under supervision in lab environment differently from that in his/her daily life. Therefore, we designed our own scheme to collect sensor data with ground-truth labels through a smartwatch based app *DataCollector*.

As shown in Figure 8, there are 10 buttons in *DataCollector*: 2 at the top and 8 at the bottom. For the 8 buttons at the bottom, each represents one of the 8 daily activities predefined in ActDetector, and the app will start sampling the sensors once the button is clicked. During the experiment, we ask the user to click the corresponding button when he/she is performing some activity, and then click the *REC* button at the top to write the collected data into a SD card with a file name indicating the corresponding label when this activity ends. In practice, users may forget to click *REC* until they find that they have already been in another activity. In this case, we ask the user to click *DEL* at the top to erase the sensor data which has been sampled most recently. In this way, the accuracy of the sensor data and the corresponding ground-truth labels can be guaranteed, and the users do not have to perform activities under supervision. In the design of ActDetector, to save power, gyroscope data is only collected if a *non-repeated* activity is detected. However, at the stage when we collect preliminary data for offline training and testing, we do not know which activity will be detected as *non-repeated*. Thus, in *DataCollector*, both accelerometer and gyroscope data is collected for each activity.

12 users (5 females and 7 males) were recruited to participate in our experiment, and each of them was given a Sony Smartwatch 3 with *DataCollector* installed. During the experiment, users were asked to wear their smartwatches every

TABLE II: The overview of the experimental data collected from 12 users.

Activity	Duration (left wrist)	Duration (right wrist)
sitting	1,096 min	1,235 min
walking	396 min	314 min
running	112 min	142 min
upstairs	26 min	27 min
downstairs	24 min	23 min
eating	168 min	348 min
driving	308 min	233 min
vehicle	387 min	462 min

TABLE III: The overall detection results.

Activity	TPR	PPV
sitting	0.962	0.943
walking	0.928	0.944
running	0.935	0.964
upstairs	0.870	0.857
downstairs	0.838	0.816
eating	0.758	0.830
driving	0.824	0.911
vehicle	0.884	0.851

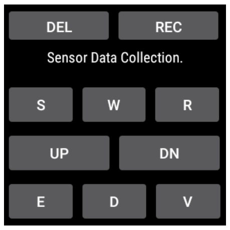


Fig. 8: The screenshot of *DataCollector*.

day and used *DataCollector* to record sensor data when they performed the predefined activities. We asked each user to wear the smartwatch on the left wrist for one week and then on the right wrist for another week. After each week's experiment, the user returned the smartwatch and we read sensor data from the smartwatch's SD card. The experiment has been approved by our IRB (Institutional Review Board), and the data collected in the experiment is shown in Table II.

B. Overall Detection Performance

After gathering the sensor data and the corresponding ground-truth labels from users, the detection accuracy of ActDetector is cross validated by using *Leave-one-user-out* strategy. At each time, data collected from 11 users is used as training data to train classifiers for *repeated* and *non-repeated* activities, and data from the remaining user is used for testing. This process is repeated 12 times so that each user's data is tested once. During training, data collected from left and right wrist is processed independently to train classifiers for different wrists. During testing, the preferred wrist is detected based on the accelerometer data collected at the beginning of each testing data file, and then the corresponding classifier is used for classification. The classification results are averaged to evaluate the overall detection accuracy.

The detection results are shown in Table III, using *True Positive Rate* (TPR) and *Positive Predictive Value* (PPV). For a certain type of activity, its TPR is defined as the ratio of the number of true positives (i.e., the activities which are correctly identified as such type) to the number of actual positives (i.e., such type of activities which are actually in the test set), and its PPV is defined as the ratio of the number of true positives to the sum of the number of true positives and false positives (i.e., the activities which are identified as such type but actually not). As can be seen, most of the activities can be detected with accuracy more than 82%. Due to their distinctive characteristics, more than 92% of *sitting*, *walking* and *running* are accurately detected. Generally speaking, *repeated* activities have larger TPR values than *non-repeated* activities because of their periodicity. Since the user's wrist may stay still some time during eating, many *eating* activities are misclassified as *sitting* and the TPR of *eating* (75.8%) is very low. Comparing with other *repeated* activities, *running* contains repeated pattern with shorter period and larger acceleration variance, and thus it is detected with higher accuracy (93.5%).

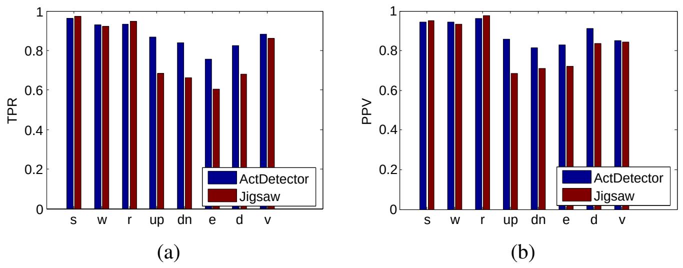


Fig. 9: TPR and PPV of each activity when using *ActDetector* and *Jigsaw* respectively.

Since there is no similar work which uses smartwatch to detect these 8 daily activities, we compare ActDetector with *Jigsaw*, a smartphone based activity detection approach proposed in [12], which detects *sitting*, *walking*, *running* and *vehicle*. For *upstairs*, *downstairs*, *eating* and *driving*, we segment the sensor data and extract the features as described in [12] for comparison. Figure 9 shows the overall detection results of *ActDetector* and *Jigsaw*, where the 8 activities are denoted as *s*, *w*, *r*, *up*, *dn*, *e*, *d* and *v* respectively. As can be seen, *ActDetector* has larger TPR and PPV than *Jigsaw* in general, especially for *upstairs*, *downstairs* and *non-repeated* activities. In *Jigsaw*, since many *upstairs* and *downstairs* activities are misclassified as *walking*, detection accuracies of these two activities are worse than many other activities. However, comparing with *Jigsaw*, these two activities are detected more accurately in *ActDetector* due to the usage of the feature *period*, which can effectively distinguish them from *walking*. For *non-repeated* activities, *ActDetector* can detect them more accurately than *Jigsaw* because unlike *Jigsaw*, where only accelerometer data is used, *ActDetector* also collects gyroscope data for detecting these activities.

C. Training Datasets

For a specific user, the classifiers used in ActDetector can be trained by different datasets: dataset collected from all the users and dataset collected only from this user. In order to evaluate ActDetector's detection accuracy when different training datasets are used, we collect two more weeks' data from a user in our experiment. ActDetector is trained based on the first two weeks' data of all the users and data of this particular user (denoted as *universal* and *individual* respectively), and

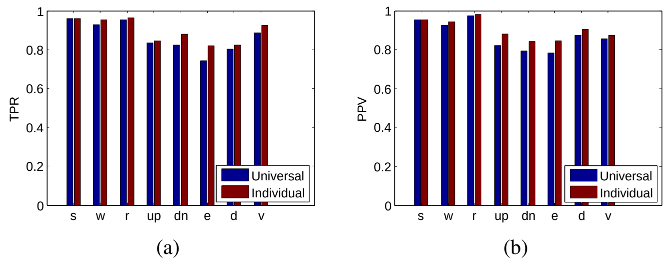


Fig. 10: TPR and PPV of each activity when using different training datasets.

TABLE IV: Power consumption of different activities when using *ActDetector* and *Jigsaw*.

Activity	Power consumption	
	ActDetector	*Jigsaw*
sitting	38.99 mW	77.18 mW
repeated activities	52.58 mW	80.55 mW
non-repeated activities	117.95 mW	82.73 mW

is tested by the new data collected from the user. Figure 10 shows the detection results of ActDetector when using different training datasets. As can be seen, most activities have larger TPR and PPV values in *individual* than in *universal*, indicating that more activities are accurately classified and less are misclassified when changing the training datsets from *universal* to *individual*. This is because the features extracted from a specific user's sensor data are more distinctive and stable than those extracted from all users' data, and thus they are better to describe and recognize the characteristics of this user's activities.

D. Power Consumption

Table IV shows the average power consumption when using *ActDetector* and *Jigsaw* to detect different types of activities (i.e., *sitting*, *repeated* activities and *non-repeated* activities). As can be seen, for *Jigsaw*, because it extracts the same features and uses the same classification algorithm for all activities, its power consumption for different kinds of activities does not vary too much. However, this is different for *ActDetector*. As shown in the table, for *ActDetector*, *sitting* detection consumes less power than the other activities because only two simple time-domain features need to be extracted for detecting *sitting*. Comparing with *sitting*, when detecting *repeated* activities, more time-domain and frequency-domain features need to be extracted, and thus more power is needed. Since gyroscope consumes more power than accelerometer, when using *ActDetector*, the power consumption for detecting *non-repeated* activities is larger than that of the other activities.

Comparing with *Jigsaw*, *ActDetector* consumes less power when detecting *sitting* and *non-repeated* activities. This is because in *ActDetector*, we carefully segment the sensor data and choose the extracted features to make it to be light weight. Although due to the usage of gyroscope, *ActDetector* consumes more power than *Jigsaw* when detecting *non-repeated* activities, it improves the detection accuracies for these activities (shown in Figure 9). Additionally, comparing with *non-repeated* activities, *sitting* and *repeated* activities (e.g., *walking*) are more predominant in a user's daily life. ActDetector can detect these predominant activities with litter power, and thus it is power efficient.

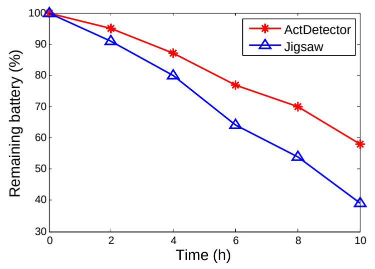
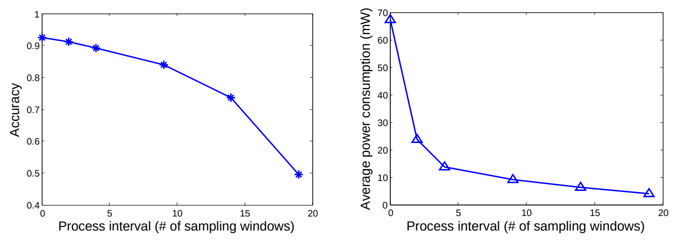


Fig. 11: Power consumption when using *ActDetector* and *Jigsaw* in daily lives.

To evaluate the system's overall power consumption, we asked a user to wear a smartwatch (Sony Smartwatch 3) with *ActDetector* and *Jigsaw* running respectively, and record the remaining battery every two hours. The smartwatch is fully charged and set to airplane mode during the experiment, and the results are shown in Figure 11. As can be seen, *ActDetector* consumes less power than *Jigsaw* when the smartwatch is used in daily lives. For a 10-hour experiment, it consumes only about 40% of the battery, while *Jigsaw* consumes about 60% of the battery. Our experiment shows that *ActDetector* can work more than one day on Sony Smartwatch 3.

E. Duty Cycles

Since most activities detected in ActDetector will usually last for minutes or even hours (e.g., *sitting*, *walking*), we use duty cycles to save power and increase the smartwatch's battery life. However, long duty cycle may have negative impact on the detection accuracy. In order to investigate the trade-off between detection accuracy and power consumption, we concatenate half of the collected data files as testing data and use different duty cycles to evaluate ActDetector, which is trained by the other half of the data. Figure 12 shows the overall detection accuracy and the average power consumption when ActDetector processes sampling windows in different intervals. In Figure 12a, the accuracy is calculated as the ratio of the number of activities that are correctly detected to the total number of activities. As can be seen, although the power consumption decreases when the process interval increases, the detection accuracy also drops dramatically. Only less than 50% of activities can be detected when the process interval is 19 (i.e., one out of every 20 sampling windows is processed). This is because when long duty cycle is used, one misclassified activity will cause many following windows that contain the same activity being labeled incorrectly. In addition, even the activity contained in a window can be correctly classified, many windows will still be mislabeled if a different activity starts just after this window is processed. As shown in Figure



(a) Detection accuracy (b) Power consumption

Fig. 12: Detection accuracy and power consumption with different duty cycles (each sampling window contains 8-second sensor data).

12, comparing with the scheme without duty cycle (i.e., the process interval is 0), when the process interval is 2, the detection accuracy is still high but the power consumption can be largely reduced. Therefore, in our implementation, the process interval was set to 2.

VI. Related Work

Radio based activity detection has been proposed in previous works. In [6], micro-doppler radars were used to detect activities by measuring the movement speeds of different parts of body when different activities were performed. Since human activities can affect Channel State Information (CSI) of wireless signals, CSI based approaches have been proposed in [7]–[10] to detect various activities. However, these systems rely on the wireless devices installed indoor, which makes them unsuitable for detecting activities outdoor.

Motion sensors have also been used to detect users' daily activities in previous works. In [11], Matthew *et al.* built an activity recognition system called Practical Body Networking, in which several activities were detected by using acceleration sampled from 5 Crossbow motes attached to a user's head, wrists and ankles. However, to make the system work, users have to wear multiple sensor nodes, which is inconvenient and difficult to implement in practice.

Smartphones are equipped with various sensors including accelerometer and gyroscope, and they have been used for activity detection in some research. In [13], Nicholas *et al.* proposed techniques for large-scale human activity inference by incorporating inter-person similarity measurements into the classifier training process. In [12], accelerometer data was collected from smartphone and different classification algorithms were implemented for activity detection. However, in these systems, the detection accuracy relies much on the position of the smartphone.

VII. Conclusion

In this paper, we proposed ActDetector, a smartwatch based application which detects 8 daily activities. By leveraging the built-in sensors on smartwatch, a multi-level classification system is designed which considers both detection accuracy and energy efficiency. A few time-domain features are extracted from the accelerometer data to detect the predominant activity *sitting*, and then more features are utilized to identify other activities. We have implemented ActDetector on Sony Smartwatch 3 and evaluated its performance in real experiments.

References

[1] N. Salbach, N. Mayo, S. Wood-Dauphinee, J. Hanley, C. Richards, and R. Cote, "A task-orientated intervention enhances walking distance and speed in the first year post stroke: a randomized controlled trial," *Clinical rehabilitation*, vol. 18, no. 5, pp. 509–519, 2004.

[2] Y. Dong, J. Scisco, M. Wilson, E. Muth, and A. Hoover, "Detecting periods of eating during free-living by tracking wrist motion," *Biomedical and Health Informatics, IEEE Journal of*, vol. 18, no. 4, pp. 1253–1260, 2014.

[3] X. Sun, Z. Lu, W. Hu, and G. Cao, "Symdetector: detecting sound-related respiratory symptoms using smartphones," in *ACM UbiComp*, 2015.

[4] X. Sun, Z. Lu, X. Zhang, M. Salathé, and G. Cao, "Infectious disease containment based on a wireless sensor system," *IEEE Access*, vol. 4, pp. 1548–1559, 2016.

[5] X. Sun, L. Qiu, Y. Wu, Y. Tang, and G. Cao, "Sleepmonitor: Monitoring respiratory rate and body position during sleep using smartwatch," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, 2017.

[6] M. A. A. H. Khan, R. Kukkapalli, P. Waradpande, S. Kulandaivel, N. Banerjee, N. Roy, and R. Robucci, "Ram: Radar-based activity monitor," in *IEEE INFOCOM*, 2016.

[7] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: device-free location-oriented activity identification using fine-grained wifi signatures," in *ACM MobiCom*, 2014.

[8] C. Han, K. Wu, Y. Wang, and L. M. Ni, "Wifall: Device-free fall detection by wireless networks," in *IEEE INFOCOM*, 2014.

[9] B. Wei, W. Hu, M. Yang, and C. T. Chou, "Radio-based device-free activity recognition with radio frequency interference," in *ACM IPSN*, 2015.

[10] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of wifi signal based human activity recognition," in *ACM MobiCom*, 2015.

[11] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles, "Pbn: towards practical activity recognition using smartphone-based body sensor networks," in *ACM SenSys*, 2011.

[12] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *ACM SenSys*, 2010.

[13] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao, "Enabling large-scale human activity inference on smartphones using community similarity networks (csn)," in *ACM UbiComp*, 2011.

[14] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*. Springer Science & Business Media, 2013.

[15] F. Kurth, "The shift-acf: Detecting multiply repeated signal components," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[16] F. Scholkmann, J. Boss, and M. Wolf, "An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals," *Algorithms*, vol. 5, no. 4, pp. 588–603, 2012.

[17] D. Li, I. K. Sethi, N. Dimitrova, and T. McGee, "Classification of general audio data for content-based retrieval," *Pattern recognition letters*, vol. 22, no. 5, pp. 533–544, 2001.

[18] G. Mountrakis, J. Im, and C. Ogole, "Support vector machines in remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 3, pp. 247–259, 2011.

[19] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[20] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[21] Y. E. Ustev, O. Durmaz Incel, and C. Ersoy, "User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal," in *ACM Ubicomp Adjunct*, 2013.

[22] K. Yatani and K. N. Truong, "Bodyscope: a wearable acoustic sensor for activity recognition," in *ACM UbiComp*, 2012.

[23] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen, "Iodetector: a generic service for indoor outdoor detection," in *ACM SenSys*, 2012.