SmartJump: A Continuous Jump Detection Framework on Smartphones

Yantao Li
College of Computer Science
Chongqing University
Chongqing 400044, China
Email: yantaoli@cqu.edu.cn

Xiaoran Peng, Gang Zhou, Hongyang Zhao
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA

Abstract—**Performing jump exercise can maintain a healthy lymphatic system, which keeps human body in an optimal condition and is a critical component of human immune system. Accurate jump detection and count are crucial to patients with a dysfunctional lymph system. In this paper, we present a continuous jump detection framework on smartphones, SmartJump, for human jump detection and count, by leveraging the accelerometer and magnetometer ubiquitously built into smartphones. Specifically, SmartJump collects sensing data from the accelerometer and magnetometer, and processes these data through coordinate system translation and data smoothing filter. Then, jump features are extracted based on the smoothed z-axis acceleration data using the peak and valley detection algorithm and then are matched with the concluded three features from the analysis of physical jumps using a finite state machine for jump detection and count. We implement SmartJump on Samsung S6 Edge smartphones and recruit 6 subjects for data collection. We evaluate the accuracy of SmartJump in terms of five-fold cross-validation test, self-test, and leave-one-out cross-validation test, and the experimental results indicate that SmartJump achieves an average of** 96.4% **recall,** 97.2% **precision, and** 96.8% **F1 score in five different scenarios.**

1. Introduction

In the last three years, over one billion smartphones have been sold annually worldwide. The global market of smart wearable devices will reach 19 billion U.S. dollars in 2018, which is ten times more than that in 2013 [1]. Taking advantage of the large market and powerful smart devices, various healthcare and fitness applications emerge on the market providing users with health tracking and exercise planning functions. For instance, Apple Health [2] and Google Fit [3] can track steps, distance traveled, and time of continuous sitting. These applications typically rely on inertial measurement units (IMU) on smart devices to collect the motion data from users. By analyzing the motion data, these applications further provide users with personalized plans for their exercises. Thus, users can receive instant service with their existing smart devices, such as smartphones and smartwatches. However, most popular healthcare and fitness applications only track users' exercises in horizontal directions. For example, Apple Health can only provide users with measurements such as the steps and distance travelled, but it is not able to provide measurements on their vertical exercise, such as jump [2].

Jump is a common exercise in our daily life. Lymphologists suggest that performing jump exercise can maintain a healthy lymphatic system, which keeps human body in an optimal condition and is a critical component of human immune system [4]. It is important for people to be able to track the number of jumps performed. Although a few of jump count applications can provide users with the count and height, some application reviews show that they are not accurate [5]. Besides these amateur applications for jump count, some commercial solutions are designed for professional athletes [6], [7]. These commercial solutions require users to purchase additional peripherals that are expensive. For example, Vert requires users to purchase a measuring unit starting at $125 [6]. Other commercial solutions also cost users hundreds to thousands of dollars. Therefore, the existing commercial solutions are either inaccurate or too expensive.

Researchers also take advantage of IMU-powered devices to measure jump. Bojan et al. use their own hardware platform to measure the performance of counter-movement jump and plyometric jump, which mainly focus on the height of jumps [8]. Similarly, other jump-related research works also focus on the acceleration and speed during jumps [9], [10], [11]. Most of these methods are not able to detect the occurrence of jump. In addition to these research on jump acceleration, speed, and height, Jin et al. [10] and Kazuya et al. [12] manage to distinguish a variety of human activities including jump. However, these general human activity detection algorithms are validated with limited amount test cases, and the test case for each category is even smaller. In order to meet the three requirements in our jump detection solution, we address two research questions: 1) How to accurately detect and count jumps by only using smartphones? 2) How to extract jump features from data collected by smparphone sensors?

In this paper, we present SmartJump, a continuous jump detection framework on smartphones for human jump detection and count, by leveraging the accelerometer and magnetometer ubiquitously built into smartphones. Specifically, we first analyze the physical process of jump activities and derive three key features of jump: 1) the peak of the taking-off phase, 2) the flat valley of the in-air phase, and 3) the peak of the landing phase. SmartJump collects sensing data from the accelerometer and magnetometer for acceleration and orientation, respectively, and processes these data through coordinate system translation and data smoothing filter. Then, jump features are extracted based on the smoothed z-axis acceleration data using the peak and valley detection algorithm and then are matched with the concluded three features using a finite state machine for jump detection and count. We implement SmartJump on Samsung S6 Edge smartphones and recruit 6 subjects for data collection in five different application scenarios including indoor and outdoor environments. We evaluate the accuracy of SmartJump in terms of five-fold cross-validation test, self-test, and leave-one-out cross-validation test, and the experimental results indicate that SmartJump achieves an average of 96.4% recall, 97.2% precision, and 96.8% F1 score in the five different scenarios. Compared with the existing solutions, we provide a smartphone sensor-based continuous jump detection and count framework that extracts jump features from smartphone sensors to match them with physical jump features for jump detection and count.

The contributions of this work are summarized as follows:

- We present SmartJump, a continuous jump detection and count framework by leveraging the accelerometer and magnetometer ubiquitously built into smartphones. SmartJump consists of three modules: jump sensing, data processing and jump detection.
- We extract three jump features from smoothed z-axis acceleration data by using the peak and valley detection algorithm and match them with the derived features from the analysis of physical jumps by using a finite state machine for jump detection and count.
- We implement SmartJump on Andorid smartphones and recruit 6 subjects for experiments and framework evaluation, and the experimental results demonstrate SmartJump achieves an average of 96.4% recall, 97.2% precision, and 96.8% F1 score in five different scenarios.

The rest of the paper is organized as follows: In Sec. 2, we present the SmartJump framework design in terms of framework overview, features of jump, data processing, and jump detection. We introduce the experiment setup and data collection, and evaluate SmartJump in accuracy in Sec. 3. We conclude this work in Sec. 4.

2. Framework Design

Jump is a general term for a variety of activities that people push themselves off the ground into the air by using their legs and feet and then return to the ground shortly. In this section, we present the design of the SmartJump framework. Specifically, we first overview the framework architecture of SmartJump. Then, we analyze the physical process of jump and derive three key features for jump detection. Finally, we elaborate the modules of data processing and jump detection for SmartJump.

2.1. Framework Overview

In this section, we overview the architecture of the continuous jump detection framework, SmartJump, as illustrated in Fig. 1. As shown in Fig. 1, SmartJump consists of three modules: 1) jump sensing, 2) data processing, and 3) jump detection.

SmartJump is designed for smart devices equipped with accelerometer and magnetometer. We select the Samsung S6 Edge smartphone as the smart device for implementation and experiments, which runs Android 5.0 operating system and equips with common sensors, such as accelerometer and magnetometer. Note that SmartJump can work on smart devices equipped with accelerometer and magnetometer and is a cross-Android operating system framework. To sense jumps, smartphone implemented with SmartJump can be put in users' pant pockets for experiments. The *jump sensing* module calls the Android APIs to collect the sensor data from accelerometer and magnetometer on smartphones with a data sampling rate of 50Hz. The accelerometer is used to detect the motion of users while the magnetometer is used to determine the orientation of the device according to the ambient geomagnetic field strength. The collected sensor data of accelerometer and magnetometer are then fed into the *data processing* module.

The *data processing* module processes the collected data of accelerometer and magnetometer through coordinate system translation and data smoothing filter. The coordinate system translation converts the collected acceleration data from the smartphone coordinate system into the Earth coordinate system. The acceleration perpendicular to Earth surface can detect jumps. In Android OS, the acceleration perpendicular to Earth surface is referred to as z-axis acceleration [13]. The date smoothing filter is used to smooth z-axis acceleration data since the translated z-axis data may contain hardware noise and other noises from the relative movements between users and phones.

The filtered acceleration data are fed into the *jump detection* module for jump detection and count. The *jump detection* module extracts three jump features from smoothed z-axis acceleration data by using the peak and valley detection algorithm and matches them with the derived features from the analysis of a physical jump activity by using a finite state machine. If the detected features match the three key features of jumps, it reports an occurrence of jump to the framework and the number will be counted.

Before we elaborate on the modules of the data processing and jump detection, we first analyze the physical process of jumps to derive jump features.

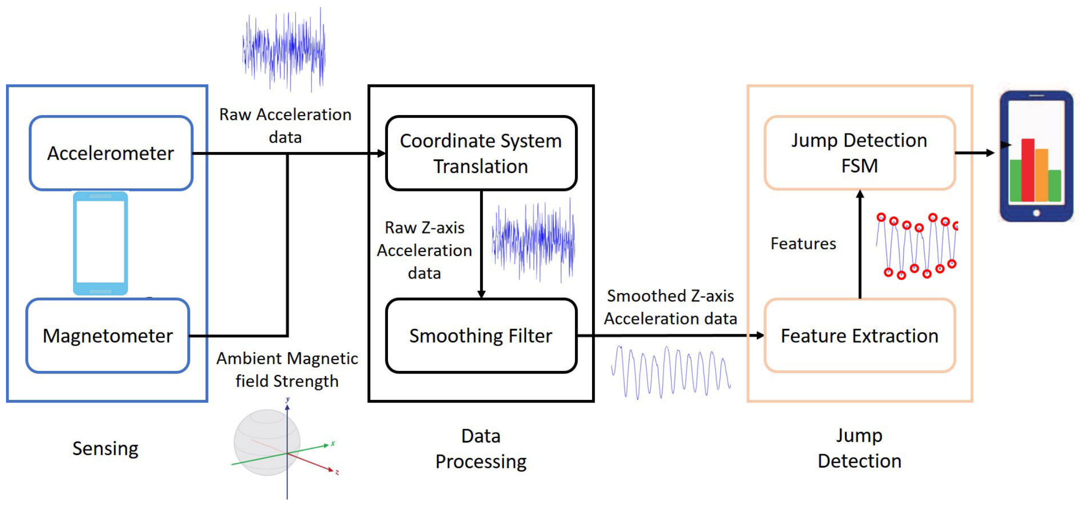

Figure 1: Framework Architecture of SmartJump

2.2. Jump Features

In order to distinguish jumps from various human activities, we need to find key features of jumps. A motion is usually characterized by acceleration, speed, and displacement. We characterize jump activity by acceleration, which can be captured by the accelerometer on smartphones. Since jump is a vertical movement, we consider the acceleration perpendicular to Earth surface in the analysis, which is the z-axis of the acceleration after the coordinate system translation in the *data processing* module. In this section, we analyze the physical process of a single jump and multiple consecutive jumps, respectively, to derive three key features of jumps for jump detection.

2.2.1. Physical process of a single jump. As described, jump is a process that a human subject pushes himself off the ground into the air by using his legs and feet, and then falls back on the ground shortly. From the description, we can segment a jump process into three phases: taking-off phase, in-air phase, and landing phase.

In the taking-off phase, the subject pushes himself off the ground by bending his legs and then straightening them. In this phase, the acceleration of the subject gradually increases to a maximum point when the legs start to bend, and then decreases to the gravitational acceleration until the subject leaves the ground. As soon as the subject leaves the ground, he reaches the in-air phase. In the air, gravity is the only force that influences the subject and thus the acceleration is equal to the gravitational acceleration. Then, the subject lands on the ground and gets into the landing phase. This phase is similar to the reverse of the taking-off phase. The subject's acceleration starts at gravitational acceleration, then rapidly climbs to a maximum point, and finally falls back to zero.

2.2.2. Physical process of multiple consecutive jumps. The physical process of multiple consecutive jumps is not a simple combination of single jumps. When two jump activities occur within a short time period (less than 0.5 second), the landing phase of the first jump is indistinguishable from the taking-off phase of the second one. This is because the two peaks are combined as one single peak. According to our experiment of a series of consecutive jumps, we observe that all three phases of the first jump, but there is no clear division between the landing phase of the first jump and the taking-off phase of the second one. However, if a subject pauses for a while between two jumps, the landing phase of the first jump can be distinguished from the taking-off phase of the second. Based on the experiment of jumps with pauses in between. In this experiment, the subject pauses for approximately half a second between two jumps. With the one-second pause, we can divide the landing phase of the first jump and the taking-off phase of the second.

2.2.3. Feature selection. Based on the analysis above, we derive three key features of the acceleration data to characterize jumps. The three features are: 1) the peak of the taking-off phase, 2) the flat valley of the in-air phase, and 3) the peak of the landing phase. We notice that the amplitude of the valley is always equal to that of gravitational acceleration, and thus we only accept a valley if its value is close to the value of gravitational acceleration. We do not set any limits to the amplitude of the peaks because the amplitude of the peaks can vary significantly from cases to cases. Therefore, we can only set a loose lower bound depending on the experiment results.

2.3. Data Processing

To extract the three key features, we process the collected data through coordinate system translation and data smoothing filter. We first convert the acceleration data from the smartphone coordinate system to the Earth coordinate system. On an Android smartphone, the coordinate system translation is achieved by `getRotationMatrix()` function and `remapCoordinateSystem()` function [14]. In these built-in functions, both accelerometer readings and the magnetometer readings are used for determining the orientation of the device.

After the coordinate system translation, we pick the acceleration perpendicular to the Earth surface for data smoothing. Since the key features of jumps lie in the vertical direction, we select a particular axis in the acceleration that is perpendicular to the Earth surface. In Android system, the particular axis in acceleration is referred to as z-axis acceleration [13]. Based on the translated z-axis acceleration data, we capture some peaks, flat valleys, and occasional spikes. These spikes may come from three sources: 1) error of the sensors, 2) relative movements between the device and the human subject, and 3) trembling of the subject. We apply Savitzky-Golay filter to the acceleration data to remove these spikes [15]. This is because Savitzky-Golay filter shows good smoothing effect on the acceleration data of jumps and meanwhile maintains the three key features of acceleration corresponding to jumps. In implementation, we set the data sampling rate as 50Hz through `SENSOR_DELAY_GAME` in `SensorManager`, according to Nyquist-Shannon sampling theorem, the frame length of the filter as 35, and order as 5. The Savitzky-Golay smoothing filter usually reduces the noise and small oscillation of the original data and it outlines the three key features. This amplification to the key features makes Savitzky-Golay smoothing filter the best for SmartJump. The filtered z-axis acceleration data are then fed into the *jump detection* module for final detection.

2.4. Jump Detection

The *jump detection* module consists of two parts: feature extraction and finite state machine based jump detection. Based on the above analysis, we extract features of peaks and valleys from the filtered z-axis acceleration data. We further use the peaks and valleys in acceleration data to match the three key features of jumps using a finite state machine. If the extracted features of the acceleration data appear in a peak-valley-peak pattern, a jump activity is detected and counted.

We find the peaks and valleys by using a peak and valley detection algorithm. Supposing that $a_i (i > 0)$ is the ith data point in the filtered z-axis acceleration, the jth point is a peak if and only if $a_j > a_{j-1}$ and $a_j > a_{j+1}$ and $a_j > 5$. We set a threshold of $5m/s^2$ to keep out the interference of other activities. The jth point is a valley if and only if $a_j < a_{j-1}$ and $a_j < a_{j+1}$ and $a_j < -10$. In the feature selection, we mentioned that the value of the valley should be equal to that of gravitational acceleration. However, the data processing module has altered the data and amplified the valley. The valley is no longer the ideal flat valley but becomes a steep valley, and the value of valley can reach $-15m/s^2$. We set a loose constraint $-10m/s^2$ as the threshold to accept the valley. The acceleration data are continuously streamed from the previous module. As soon as the peak and valley detection algorithm detects a peak or valley from the streamed data, it passes the peak or valley as an event to the jump detection part.

We use a finite state machine (FSM) to match the detected peak/valley events with "peak-valley-peak" patterns. The FSM has three states: Init, Peak, and Valley, where Init indicates the initial state. In the Init state, it can only transit into the Peak state when it receives a peak event. In the Peak state, it stays in Peak state if it receives another peak event and it transits to the Valley state when it receives a valley event. When the FSM receives a Peak event in the Valley state, it reports a jump activity and sets the state to Peak state. The Valley state stays at the Valley state if the FSM receives a valley event. Besides the normal state transition rule, the FSM also has a half-second reset rule. The FSM resets itself to Init state if no event is received within a half second.

3. Evaluation

In this section, we first introduce the experiment setup and data collection, and then evaluate SmartJump in terms of five-fold cross-validation test, self-test, and leave-one-out cross-validation test, respectively.

3.1. Experiment Setup and Data Collection

To evaluate SmartJump, we use the accelerometer and magnetometer on a Samsung S6 Edge smartphone to collect the acceleration and ambient magnetic field strength data. In our experiments, we recruited six subjects including three males (average weight and height: 70kg and 178cm) and three females (50kg and 165cm). The six subjects were required to perform five sets of activities, which represented five different application scenarios. The smartphones were put in their jeans' pockets during the data collection process. Note that jeans are tightly attached to human body which can precisely capture subjects' jump activities.

We test SmartJump in five application scenarios. The first application scenario is a simple indoor environment with enough space, where the subjects perform the following activities in order: 1) put phones in their jeans' pockets, 2) jump 25 times, 3) walk around in a small range, 4) and jump another 25 times. During the activities, the subjects can take a rest or stop for a while between two jumps, and there are no intense activities other than jump. We expect that SmartJump shows the best performance in this basic scenario. This scenario simulates some simple exercises performed at home. This scenario includes common activities in people's daily lives, and is the baseline for our evaluation. The second application scenario is a complex indoor environment. The subjects were required to perform the following activities in order: 1) put phones in their jeans' pockets, 2) jump 10 times, 3) go upstairs to the second floor, 4) jump 10 times, 5) go downstairs to the first floor, 6) jump 10 times, 7) run around, 8) jump 10 times, 9) sit on the sofa and stand up for 2-4 times, and 10) jump the last 10 times. This scenario contains more intense activities, such as climbing stairs which has obvious displacement on vertical direction. The complex indoor scenario simulates the intense exercises performed in a gym, where people have different intense exercises and move in a larger area than the simple indoor scenario. We expect that SmartJump can accurately distinguish the 50 jumps within a series of activities. The third application scenario is an outdoor environment. This scenario is similar to the complex indoor scenario, except the required activities that are performed in an open outdoor environment. The subjects were required to perform 50 jumps, walking, running, and riding bikes in an outdoor environment in any order. The only requirement is that they must travel at least 1km, which ensures that the subjects experience different terrains. The main difference is that the outdoor environment has more complicated terrains and unexpected events. The subjects can go up a hill or evade a coming car outside. The noise is much stronger than that in the indoor environments. This outdoor scenario simulates that the user performs some exercises in outdoor environment. With the various activities and complex terrains as noise, this scenario evaluates SmartJump's ability to distinguish jumps from noise. The fourth application scenario requires the subjects to freely perform their daily activities and exercises except for jumps. In this scenario, the subjects can freely move around in both indoor and outdoor environments. We only require them to stay in motion for at least 8 minutes. We design this application scenario to ensure that SmartJump does not detect other activities as jump. The fifth is rope jump scenario, in which the subjects were required to jump with the jump rope for 100 times. The subjects cannot rest during the 100 jumps. We expect SmartJump can accurately detect the total count of the jumps in this scenario.

3.2. Five-fold Cross-validation Test

In this section, we perform a five-fold cross-validation test on the six subjects' data. To construct the dataset, we randomly concatenate the smoothed z-axis acceleration of the six subjects' data together. Then, we divide the dataset into five subsets, where four of them are used for training and the rest one for testing. In particular, we tune the following parameters to achieve the best performance: frame length of the Savitzky-Golay filter, order of the Savitzky-Golay filter, and the thresholds of peaks and valley, in the training process. Each of the five subsets is used as the test sample once and we average the results of the tests.

Based on the five-fold cross-validation test, SmartJump achieves an average of 96.4% recall, 97.2% precision, and 96.8% F1 score for the whole dataset. The results demonstrate that SmartJump can effectively detect the jump and accurately count jumps, and meanwhile, it tolerates noises and other similar activities.

3.3. Self-test

In the self-test, we tune the same parameters of SmartJump for the best performance for each individual subject in different scenarios.

We conduct the self-test results for six subjects, and the three metrics for subjects 1 to 5 are above 98%. Only subject 6 has lower 95.2% recall, 96.0% precision, and 95.6% F1 score. We also conduct the self-test results for five scenarios. In the No-jump scenario, SmartJump only has one false detection for subject 1. In general, the three metrics of the rest four scenarios are all higher than 95%. The complex indoor scenario has the lowest values of 97.5% recall, 95.1% precision, and 96.3% F1 score. The outdoor scenario's performance is also lower than the average: 96.6% recall, 96.9%, and 96.7% F1 score. This is because the complex indoor scenario and outdoor scenario have strong noises from intense activities and complex environments.

3.4. Leave-one-out Cross-Validation Test

In the leave-one-out cross-validation, we leave the data of one subject or one scenario out, use the rest of the data to find the best parameters, and test them on the leave-out subject or scenario. For example, we use the data of subjects 1 to 5 to find the parameters of SmartJump, which can achieve the best performance for these five subjects, and then use the SmartJump to test subject 6. We repetitively do this process to each subject and scenario, respectively. The goal of the test is to ensure that SmartJump framework can be generalized to other users and scenarios.

We conduct leave-one-subject-out cross-validation, and the results are close to the results of overall performance test. The recall, precision, and F1 score of each test in leave-one-subject-out cross-validation are within the range of 95% to 99%. In the five-fold cross-validation test, we divide the dataset randomly. However, the results of randomly divided dataset are close to that of dataset divided by subjects. This result indicates that data grouped by subjects have no difference to randomly grouped data. We also conduct leave-one-scenario-out cross-validation. When we leave the no-jump scenario out, SmartJump still reports one false detection for subject 1 and two false detection for subject 6. We can observe that the three metrics of the simple indoor scenario and the jump rope scenario are still good. However, the performance of other scenarios are much worse. Compared with five-fold cross-validation test, the precision of complex indoor scenario drops to 85.8%, and the recall of outdoor scenario drops to 93.4%. These results show that complex indoor scenario and outdoor contain the most noises and activities similar to jump.

From the results of leave-one-out cross-validation, we conclude that collecting the data in different application scenarios is more important than that from different subjects. In this case, we consider scenarios from people daily scenarios to people exercise scenarios, which represent most of the daily lives of users.

4. Conclusion

In this paper, we present SmartJump, a continuous jump detection framework that detects human jump activity for exercise tracking. SmartJump consists of three modules: jump sensing, data processing and jump detection. We extract three jump features from smoothed z-axis acceleration data by using the peak and valley detection algorithm and match them with devised features from the analysis of physical jumps by using a finite state machine for jump detection and count. We implement SmartJump on Samsung S6 Edge smartphones and recruit 6 subjects for data collection in five different application scenarios. We evaluate the accuracy of SmartJump, and the experimental results indicate that SmartJump achieves an average of 96.4% recall, 97.2% precision, and 96.8% F1 score in the five scenarios.

Acknowledgment

We thank the subjects for collecting the experimental data and appreciate the anonymous reviewers for their valuable suggestions.

References

[1] Wearable device market value from 2010 to 2018. 2019. [Online]. Available: http://www.statista.com/statistics/259372/wearable-device-market-value/

[2] Apple Health. 2019. [Online]. Available: https://www.apple.com/ios/health/

[3] Google Fit. 2019. [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.fitness&hl=en

[4] D. Scrivens, "Rebounding: Good for the lymph system," *Well Being J.* vol. 17, no. 3, pp. 3-4, 2008.

[5] Jump Rope Counter + Calories. 2019. [Online]. Available: https://play.google.com/store/apps/details?id=pp.com.jumpix

[6] VERT. 2019. [Online]. Available: https://www.myvert.com/

[7] GyKo. 2019. [Online]. Available: http://gyko.it/

[8] B. Milosevic and E. Farella, "Wearable Inertial Sensor for Jump Performance Analysis," in *Proc. 2015 workshop Wearable Sys. Apps.*, Florence, Italy, 2015, pp. 15-20.

[9] M. R. Garcia, L.-J. M. Guzman, J.-S. B. Valencia and V. M. Henao, "Portable measurement system of vertical jump using an inertial measurement unit and pressure sensors," in *2016 XXI Symp. Signal Processing, Images and Artificial Vision* , Bucaramanga, Colombia, 2016, pp. 1-5.

[10] I. M. Pires, N. M. Garcia and M. C. C. Teixeira, "Calculation of Jump Flight Time using a Mobile Device," in *Proc. Int. Conf. Health Informatics*, Lisbon, Portugal, 2015, pp. 293-303.

[11] C. Balsalobre-Fernndez, M. Glaister and R. A. Lockey, The validity and reliability of an iPhone app for measuring vertical jump performance," *J. Sports Sci.*, vol. 33, no. 15, pp. 1574-1579, 2015.

[12] K. Murao and T. Terada, "A motion recognition method by constancy-decision," in *Proc. IEEE Int. Symp. Wearable Computers*, Seoul, South Korea, 2010, pp. 1-4.

[13] SensorManager. 2019. [Online]. Available: https://developer.android.com/reference/android/hardware/SensorManager.html

[14] Position sensors. 2019. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_position.html

[15] S. J. Orfanidis, *Introduction to signal processing*. Pearson Education, Prentice Hall, 2010.

Yantao Li is a Professor with the College of Computer Science, Chongqing University, Chongqing, China. He received the Ph.D. degree from the College of Computer Science, Chongqing University, China, in December 2012. His research area includes wireless communication and networking, sensor networks and ubiquitous computing, and information security. He was a recipient of the Outstanding Ph.D. Thesis Award in Chongqing in 2014, and the Outstanding Master's Thesis Award in Chongqing in 2011. Contact him at yantaoli@cqu.edu.cn.

Xiaoran Peng is a Software Engineer at Microsoft, Redmond, WA, USA. He received the M.S. degree from the Computer Science Department, William & Mary, Williamsburg, VA, USA, in 2018, and the B.S. degree from the College of Computer Science and Technology, Huazhong University of Science and Technology, Hubei, China. Contact him at xpeng02@email.wm.edu.

Gang Zhou is a Professor in the Computer Science Department at William & Mary. He was Graduate Program Director from 2015 to 2017. He received his Ph.D. degree from University of Virginia in 2007. He has published more than 100 papers in prestigious conferences and journals. His Google Scholar Citations are over 8000, with h-index 33 and i10-index 73. He received an NSF CAREER Award in 2013, the Best Paper Award of IEEE ICNP 2010, and Plumeri Award for Faculty Excellence in 2015. Dr. Zhou serves on the Journal Editorial Board of ACM Transactions on Computing for Healthcare (HEALTH), ACM Transactions on Sensor Networks (TOSN), Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), and Elsevier Smart Health. He also served on the Journal Editorial Board of IEEE Internet of Things and Elsevier Computer Networks. He serves as a Steering Committee member, General Chair, and TPC Chair of CHASE - ACM/IEEEs premier conference on Connected Health: Applications, Systems and Engineering Technologies. Contact him at gzhou@cs.wm.edu.

Hongyang Zhao received his Ph.D degree with the Computer Science Department, William & Mary, Williamsburg, VA, USA, in 2020. He received his M.S. degree from Zhejiang University, Zhejiang, in 2014 and his B.S. degree from Shanghai Jiao Tong University, Shanghai, in 2011, respectively. Contact him at hyzhao@cs.wm.edu.