# Distributed Analytics and Edge Intelligence: Pervasive Health Monitoring at the Era of Fog Computing

Yu Cao, Peng Hou, Donald Brown, Jie Wang
Dept. of Computer Science, The University of Massachusetts Lowell
One University Avenue
Lowell, Massachusetts, 01854, USA
ycao@cs.uml.edu

Songqing Chen
Dept. of Computer Science, George Mason University
4400 University Drive
Fairfax, Virginia 22030, USA
sqchen@gmu.edu

## ABSTRACT

Biomedical research and clinical practice are entering a data-driven era. One of the major applications of biomedical big data research is to utilize inexpensive and unobtrusive mobile biomedical sensors and cloud computing for pervasive health monitoring. However, real-world user experiences with mobile cloud-based health monitoring were poor, due to the factors such as excessive networking latency and longer response time. On the other hand, fog computing, a newly proposed computing paradigm, utilizes a collaborative multitude of end-user clients or near-user edge devices to conduct a substantial amount of computing, storage, communication, and etc. This new computing paradigm, if successfully applied for pervasive health monitoring, has great potential to accelerate the discovery of early predictors and novel biomarkers to support smart care decision making in a connected health scenarios. In this paper, we employ a real-world pervasive health monitoring application (pervasive fall detection for stroke mitigation) to demonstrate the effectiveness and efficacy of fog computing paradigm in health monitoring. Fall is a major source of morbidity and mortality among stroke patients. Hence, detecting falls automatically and in a timely manner becomes crucial for stroke mitigation in daily life. In this paper, we set to (1) investigate and develop new fall detection algorithms and (2) design and employ a real-time fall detection system employing fog computing paradigm (e.g., distributed analytics and edge intelligence), which split the detection task between the edge devices (e.g., smartphones attached to the user) and the server (e.g., servers in the cloud). Experimental results show that distributed analytics and edge intelligence, supported by fog computing paradigm, are very promising solutions for pervasive health monitoring.


*Mobidata* '15, June 21, 2015, Hangzhou, China.

DOI: http://dx.doi.org/10.1145/2757384.2757398 .

## Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

## Keywords

Mobile Computing; Pervasive Health Monitoring; Distributed Analytics; Edge Intelligence; Fog Computing

## 1. INTRODUCTION

Biomedical research and clinical practice are entering a data-driven era [15]. The global size of Big Data in Biomedicine is roughly 200 Exabytes in 2012 [10], providing an unprecedented opportunity for making smart and optimized healthcare decision with improved outcomes, while reducing healthcare costs. One of the main application areas of biomedical big data research is pervasive health monitoring. During last few years, we have seen a worldwide deployment of mobile devices and wireless infrastructure that enabling the pervasive health monitoring applications. Most pervasive health monitoring applications employ inexpensive and unobtrusive sensors that can collect physiology data for chronic condition in natural living environment. Typically, large volume of sensor data will be collected and transmitted to the server in the cloud for further analysis.

However, real-world user experiences with mobile cloud-based health monitoring were poor, due to the factors such as networking latency and response time. On the other hand, fog computing [26, 6, 24, 7], a newly proposed architecture, utilizes one or a collaborative multitude of end-user clients or near-user edge devices to conduct a substantial amount of computing, storage, communication, and etc. This new computing paradigm opens an unprecedented opportunity to discover early predictors and novel biomarkers to support and enable smart care decision making in a connected health scenarios. In this paper, we employ a real-world pervasive health monitoring application (pervasive fall detection for stroke mitigation) to demonstrate the effectiveness and efficacy of fog computing application in health monitoring.

In the United States, stroke kills over 133,000 people per year [14]. It is projected that nearly 800,000 strokes will occur in 2013. Stroke is a leading cause of serious long term adult disability and is one of the leading causes of death [14]. There are more than seven million stroke survivors

in the U.S. whose age is over 20. While stroke has been a very serious problem and it is predicted to get worse, Recent research [16, 23] has indicated that: if we can help the potential victims reduce and/or mitigate the stroke-related risk factors, their chance of having stroke can be prevented by as much as one third; if the stroke patients could access the latest therapies in a timely manner, the disability from stroke can be greatly reduced.

The ultimate goal of our research is to investigate and develop a smart device-based, real-time, low cost, and unobtrusive computer-aided fog computing system for use by stroke patients for abnormal event detection and predication. Because fall is a major source of morbidity and mortality among elderly and stroke patients often have multiple falls in a year [2], we use full detection as a case in point to demonstrate the efficacy and effectiveness of our proposed designs, algorithms, and systems.

While there are substantial research and development efforts on new algorithms and systems for early predictors and novel biomarkers for stroke patients using wearable devices [12, 25], real-world wearable computing tools and applications in stroke-related clinical practice with the capacity of real-time fall detection are rare, mainly because of the following two major barriers: (1) either based on simple thresholds that often produce many false alarms, (2) or based on sophisticated learning-based techniques that are too heavy to be executed on low-cost devices with limited computing resources. For the second case, the system relies heavily on the computing and storage resources in the cloud. This puts heavy pressure on the networking bandwidth and QoS. In many application scenarios where reliable networking bandwidth and QoS are not available, the system performance is very poor and the user experiences are inferior to the expectations.

In this study, we set to (1) investigate and develop new fall detection algorithms and (2) design and employ a real-time fall detection system, called *U-Fall*, using fog computing paradigm (e.g., distributed analytics and edge intelligence), which split the detection task between the edge devices (e.g., smartphones attached to the user) and the server (e.g., servers in the cloud). In U-Fall, the detection task is carefully split between the smartphone and the server. While the front-end mobile device quickly conducts light-weight computation for fall detection, the sensor data are also transmitted to the back-end server in real-time for more accurate detection. The front-end and the back-end run our newly devised detection algorithms. Such a two-stage detection can significantly improve the detection results while maintaining a low response time and consuming as little battery power as possible on the smartphone. Experimental results show that U-Fall achieves the high sensitivity (which also implies very low miss rate) close to that when using the simple threshold approach while it also achieves the high specificity (which also means low false alarm rate) close to that when using the learning techniques. At the same time, the response time and energy consumption of U-Fall are close to the minimum of the existing two approaches.

This rest of the paper is organized as follows. Section 2 introduces the background and the state-of-the-art of fall detection algorithms and systems. Section 3 presents the architecture, components, and algorithms for the proposed fall detection system. Section 4 describes the implementation detail of U-Fall and make concluding and Section 5 presents the evaluation results of U-Fall. We discuss U-Fall limitations and make concluding remarks in Section 6.

## 2. RELATED WORK

The first related area of research is fog computing [26, 6, 24, 7]. Under the fog computing paradigm, the cloud is shifted to the edge of the network and the routers may become the virtualisation infrastructure. By extending the cloud paradigm to the edge of the network, fog computing offers a unifying paradigm for cloud infrastructures and smart-items infrastructures. One of the main advantages of fog computing is that this paradigm could reduce the latency, offer location awareness, as well as support the users and/or sensor mobility. These benefits are particularly useful for pervasive health monitoring because

The second related area of research is using wearable devices for fall detection. It is gaining popularity recently and various detection algorithms have been designed. Most of the existing fall detection algorithms can be classified into two categories. The first and the major category often employs the threshold-based schemes [3, 8, 11, 17]. While threshold-based approaches can respond to fall events in almost real-time and many of them have very high sensitivity (low miss rate), a recent survey study [5] that validates thirteen published fall-detection algorithms using real-world fall data has indicated that most of them suffer from frequent false positives (high false alarm rate and low specificity).

To address the false alarm issue, some recent fall detection systems seek to employ sophisticated pattern-matching algorithms [9, 19, 20, 22], the second category of the existing fall detection algorithms. While introducing learning-based algorithms indeed reduces the false alarms, most of them are unfortunately too heavy to be trained in real-time and executed on the wearable devices (due to low computation capacity and limited storage space).

Some of the fall detection studies have considered to use mobile devices as the sensing devices. For example, in [19], authors use the accelerometers equipped in smartphones. However, real-time detection was not supported in this paper because both the training and testing were in an offline manner. In addition, their approach requires the user to fix the position and orientation of the phone. If the position and orientation of the phone were unknown, the accuracy of the fall detection is severely reduced.

Compared to existing detection algorithms and systems, our devised algorithms and developed U-Fall overcome these barriers by carefully splitting the detection task into two stages running on the front-end smartphone and the back-end resoureful and for most accurate and real-time detection results while maintaining a low battery consumption on the smartphone for prolonged lifetime.

## 3. U-FALL DESIGN

### 3.1 Overview

Our proposed system utilizes two important principals from fog computing: distributed analytics and edge intelligence. Specifically, we distribute the data analytics between the edge device (smartphone) and server in the cloud. In another word, the fall detection in U-Fall is done in a collaborative manner between the edge devices (smartphone) and the resourceful cloud as shown in Figure 1. Please note,

our system also uses the smartphone for sensing and data capturing. As shown by Figure 1, the user is only aware of

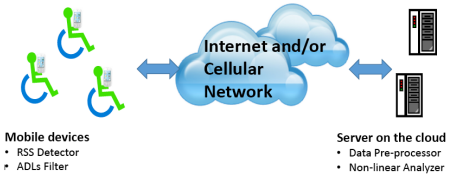


**Figure 1: Overall architecture of U-Fall**

the front-end (an App) running on the edge device (smartphone), while the back-end, connected via network, is totally transparent to the user. Note that both the front-end and the back-end can make independent detection results (e.g., when there is no connection to the cloud possible), but collaborative detection will be able to improve the accuracy and reduce the false alarms.

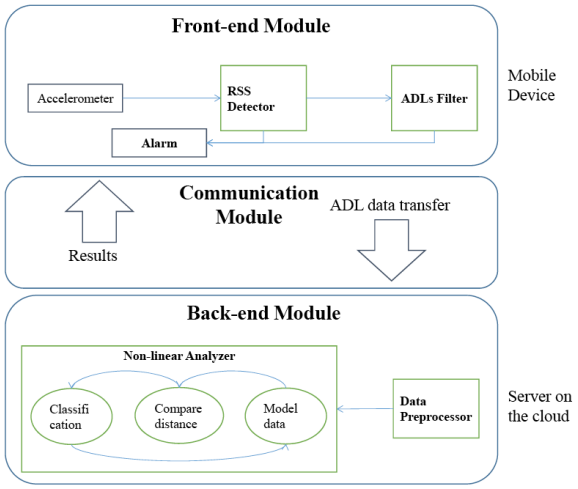


**Figure 2: U-Fall designs and components**

These design choices have lead to our system design as shown in Figure 2. As shown in this figure, U-Fall consists of the following three major modules:

**1. Front-end Module (FM)**: the FM runs on the edge device (smartphone). It further consists of three subcomponents, namely *RSS Detector*, *ADLs Filter*, and *Alarm*. RSS Detector runs the RSS threshold-based detection algorithm, while the ADLs Filter runs the fall-like ADLs filtering algorithm. We will discuss the alogroithm details in the later section. Alarm takes the input from either the server side or the previous two subcomponents directly (when there is no connection available with the server) and decides whether to contact the pre-stored emergency contact information or 911 if no emergency contact is set.

**2. Back-end Module (BM)**: the BM runs on the cloud. It consists of two sub-components, namely *Data Preprocessor*, and *Non-linear Analyzer*. Data Pre-processor is a pre-processing component to filter out the sensor data received from the front end. Non-linear Analyzer performs non-linear analysis for fall detection based on the pre-processed data.

**3. Communication Module (CM)**: CM provides two channels for communication between the FM and the BM. It transfers the sensor data from the FM to the BM via *Input Channel*, and it also passes the detection results from the BM to the FM via *Output Channel*.

## 3.2 Fall Detection Algorithms

As indicated by Figure 2, U-Fall is a two-stage fall detection system that demonstrates important principals of fog computing (e.g., distributed analytics and edge intelligence). The core of U-Fall is the fall detection algorithms running on the FM and the BM. In this subsection, we present these algorithms in detail.

### *3.2.1* Fall Detection Schemes on the Front-end

While the FM consists of three components, the detection related algorithms run in RSS Detector and ADLs Filter only. At a high level, the RSS Detector runs the threshold-based fall detection, while ADLs Filter runs the fall-like ADLs filtering algroithm. Details of these algorithms are as follows.

**1.1. RSS Detector: Threshold-based Fall Detection:** The threshold-based detection uses the value from root-sum-of-squares (RSS) of acceleration magnitude. The RSS is the root-sum-of-squares (or acceleration magnitude). By consulting with domain experts, we define that a fall begins with a period of free fall, followed by the impact with the ground, and then a period of inactivity, which can range from a matter of seconds to a few hours.

**1.2. ADLs Filter: Fall-like ADLs Filtering:** As we just discussed, while the RSS-based detection has great potential to deliver a system with a very low miss-rate, this approach may produce many false alarms [4]. One of the main false alarm sources is from the fall-like Activities of Daily Livings (ADLs) (i.e., jumping into a surface, sitting into a surface, and etc.). To address this issue, we apply Fall-like ADLs Filtering on the front-end, which is capable of filtering out certain fall-like ADLs.

### *3.2.2* Fall Detection Schemes on the Back-end

While the proposed algorithms in the front-end module can remove many false alarms, they are not able to remove all false alarms. There are several reasons for this issue. The first is that in our system, the smartphone is not required to be strapped on in certain locations. Rather, the smartphone can simply be placed in any pant pocket located near the waist. Additionally, there is no restriction as to how the smartphone must be oriented in the pocket. This is very convenient from the user's point of view. However, from the technical point of view, this is very challenging because we need to develop new fall detection algorithms that are robust to the changes of position, orientation, and location of the phone.

Motivated by the recent progress in sensor data filtering [21], we develop the orientation filtering technique. Due to the computation cost of the proposed algorithm, U-Fall runs it on the cloud.

**2.1. Data Pre-Processor: Orientation Filtering:** In this step, the Data Pre-Processor performs orientation filtering.

**2.2. Non-Linear Analyzer: Fall Detection based on Non-linear Time Series Analysis:**

While the proposed algorithms in both the FM and the

BM can filter out certain type of fall-like ADLs, such as jumping into a surface, sitting into a surface, they are unable to recognize more difficult cases such as bending-pick-up, squatting-down. To deal with these more complicated cases, we propose to use non-linear analysis and our proposed algorithm is rooted from time series mining. However, the direct application of existing time series mining algorithms, such as [18], will yield unsatisfied results, due to the unique characteristics of sensor data from smartphones. We propose a new learning-based time series mining algorithms based on nonlinear time series analysis. Traditionally, nonlinear time series analysis techniques have not been widely employed in the area of machine learning. However, recent advances in dynamical system models (e.g., parametric or nonparametric models, linear or nonlinear models) and related model matching algorithms have indicated that new computing approaches that model the human physiology data (such as data from wearable accelerometers) as a sequence of observations of a dynamical system could be a potential solution.

# 4. SYSTEM IMPLEMENTATION

We have implemented a prototype of U-Fall. The FM is implemented on Android 4.0.4 (Ice Cream Sandwich), and the BM is implemented using the Amazon AWS. The CM is implemented using Web services. In this section, we present implementation details.

## 4.1 Front-end Implementation

**RSS Detector Implementation:** The RSS Detector runs the threshold-based fall detection algorithm, the core of which is the choice of the threshold. We test three different UFTs: 1.6 g, 2 g, and 2.5 g. From our tests, we find that the 2 g threshold yields the best results. We thus use that as UFT for our algorithm.

**ADLs Filter Implementation:** The ADLs Filter is to filter out Fall-like ADLs. In this step, we mainly compute three values: the average acceleration magnitude variation (AAMV) index, the free fall interval (FFI), and the free fall average acceleration magnitude variation (FFAAMV). The AAMV is used to filter out two types of fall-like ADLs: "sitting quickly on soft surface such as sofa" and "sitting quickly on hard surface such as chair". The combination of FFI and FFAAMV can filter out the "jumping on the ground", which is another type of fall-like ADLs.

**Data Collector and Alarm Implementation:** To facilitate data collection efforts and support our experiments (next section), we also develop a mobile data collection App in the front-end module, together with the Alarm. The App has the following features: a user can choose to annotate the activities they perform, automatically collect the accelerometer data when the user performs the activities, and automatically transmit the data to the server in the cloud.

## 4.2 Back-end Implementation

The implementation of the back-end module is based on Amazon Web Services (AWS) [1]. We use a micro instance of 64-bit Windows Server 2008 R2 Base. MySQL database is installed in the server.

**Pre-Processor Implementation:** As aforementioned, due to the different sensor orientations, it is possible the same accelerometer measurements may generate different values. Our proposed orientation filtering method in Pre-Processor can correct the orientation by rotating the smartphone's frame of reference and aligning it to the earth frame of reference [21].

**Non-Linear Analyzer Implementation:** The Non-Linear Analyzer runs the nonlinear time series analysis Geometric Template Matching (GTM) [13]. GTM works by combining the conventional boosting machine learning algorithm with the feature extraction potentials of time delayed embedding and GTM. The weak classifier is the Naive Bayes. The features which are used for training are the similarity scores of the respective models.

# 5. PERFORMANCE EVALUATION

## 5.1 Experiment Setup

In all the following experiments, we use Samsung Galaxy S3 running Android 4.0.4 Ice Cream Sandwich as the front-end to install the FM of U-Fall. This smartphone uses Snapdragon S4 SoC (from Qualcomm) featuring a dual-core 1.5 GHz Krait CPU and an Adreno 225 GPU. It also has a 16 GB of internal storage. In the back-end, a micro instance of 64-bit Windows Server 2008 R2 Base provided by AWS EC2 service is used for experiments. The RAM of the AWS server is 4GB.

In order to evaluate the effectiveness and efficiency of U-Fall, we also implement two other systems running the state-of-the-art fall detection algorithms for comparisons. The first one, entitled as **T-System**, employs the threshold-based technique for impact detection (which is a typical indication of fall occurrence) and posture monitoring (because the last phase of fall event usually includes a lying posture). This has been used in the majorities of existing studies [3, 8, 11, 17].

As some other studies [9, 19, 20, 22] have also used pattern-matching for fall detection, we also implement into the second system, called **P-System**, for comparisons.

## 5.2 Data Set

After close discussion with our clinical collaborators (e.g., primary stroke care physicians), we have decided to seek volunteers for tests. For easy comparisons between different systems, we first collect a large volume, high quality, real-world (or near real-world) fall data set, which includes the following two parts: **1. evaluation data by volunteers:** 20 students volunteers were recruited to serve as subjects. After the training session offered by the doctor, each of them wears the smartphone for 2 weeks. During the 2-week period, they spent 2 hours per day to simulate the activity of daily living (ADL) of the stroke patients, including regular ADL (e.g., walking, jogging) and different types of fall events (different types of fall events are presented later). All the data were recorded and annotated. **2. evaluation data using a rescue dummy:** Since some of the falls and ADLs are difficult for volunteers, we also use a rescue dummy, called *Rescue Randy*, that we loaned from the local fire department. The *Rescue Randy* is a dummy whose height is 6 feet 1 inch and whose weight is 180 lbs. We choose *Rescue Rady* because it was developed for lifelike adult or juvenile victim handling, transportation, and extrication training. It is a manikin that can be safely used in situations too hazardous or uncomfortable for human volunteers. It also features with articulated joints, weight distribution according to human weight distribution chart. These features make them partic-

ular suitable for sibilating the real-world fall activities. For each fall activity, we attached the smartphone to *Rescue Randy* for a period of 2 hours and put Randy to conduct the fall activity under different scenarios. The sensor readings were recorded during each activity.

## 5.3 Evaluation Results

In the experiments, the front-end also uses the same Samsung Galaxy S3, the same as the one used in the data collection. To closely approximate the field tests, we use 4G LTE service by a major telecommunication company with 15 Mbps and 8 Mbps for download and upload, respectively, when communicating with the Amazon AWS server. For energy consumption, we use the PowerTutor [27].

**Table 1: Response Time (ms) of Three Systems Upon Fall Activities**

| System | T-System | P-System | U-Fall |
|---|---|---|---|
| Response Time | 35.38 | 47.85 | 35.67 |

Table 1 shows the comparison of the response time of the three systems in response to fall events. Each row in this table represents the one category of fall activity The first column of this table represents the name of the event (e.g., different types of falls and/or ADLs). The second column to the fourth column show the response time of each system being tested. The last row averages the comparison of response time from all the events. Table 1 shows that compared to the Pattern-matching system, the response time of U-Fall is reduced by 25% on average: as shown in the last column, the response time is reduced from 47.85 millisecond to 35.67 millisecond. Compared to the simple Threshold-based system, the response time of U-Fall is very close.

**Table 2: Response Time (ms) of Three Systems Upon ADL Activities**

| System | T-System | P-System | U-Fall |
|---|---|---|---|
| Response Time | 33.39 | 58.45 | 34.32 |

Table 2 shows the comparison of the response time of the three systems upon different types of ADL activities. While the data sets are different, both Table 1 and Table 2 show a similar pattern: the response time of our U-Fall system is very close to the response time of the Threshold-based system. At the same time, U-Fall can substantially reduce the response time when compared to the Pattern-matching system. On the other hand, in general, we can see that recognizing ADLs takes more time than recognizing fall activities in all three systems.

**Table 3: Energy Consumption (Joule) of Three Systems Upon Fall Activities**

| System | T-System | P-System | U-Fall |
|---|---|---|---|
| Energy Consumption | 1.33 | 2.26 | 1.63 |

Table 3 shows the corresponding energy consumption of the three systems upon different types of fall activities. Similar to table 1, the last row summarizes the average of energy consumption. This table shows that the energy consumption of U-Fall is very close to the energy consumption of the Threshold system. Compared to the Pattern-matching system, the energy consumption of U-Fall is reduced by 28%: as shown in the last column, the energy consumption is reduced from 2.26 Joule to 1.63 Joule. This is expected as most of the complicated computation in U-Fall is offloaded to the cloud.

**Table 4: Energy Consumption (Joule) of Three Systems Upon ADL Activities**

| System | T-System | P-System | U-Fall |
|---|---|---|---|
| Energy Consumption | 1.83 | 2.96 | 1.91 |

Table 4 summarizes of the corresponding energy consumption of the three systems upon different types of ADLs activities. From this table, we can still observe that U-Fall's energy consumption is close to that of the Threshold-based system, while it is much better than that of the Pattern-matching system. Not surprisingly, comparing Table 3 and Table 4, we find that recognizing ADLs takes more time and thus consumes more energy on the smartphone as well.

**Table 5: Fall Detection Accuracy of Three Systems**

| Event | T-System | P-System | U-Fall |
|---|---|---|---|
| Sensitivity | 90% | 78% | 88% |
| Specificity | 65.1% | 75.2% | 74.6% |

Finally, Table 5 lists the fall detection accuracy of three different systems. As shown in this table, while the sensitivity of U-Fall is slightly worse than Threshold-based system (88% VS 90%), U-Fall improves the accuracy by 13% (increase from 78% to 88%) when compared to the Pattern-matching System. This means that the ability of U-Fall to detect real falls is much higher. At the same time, compared to the Threshold-based system, U-Fall achieves much higher specificity (increase from 65.1% to 74.6%). Compared to the Pattern-matching system, the specificity was reduced very slightly. This means the ability of U-Fall to detect only real falls (in other words, its ability to filter out false positives) is also much higher.

From a medical standpoint, it is most important to have a high sensitivity, and the specificity is secondary to sensitivity. Clearly, it is essential to detect any real falls that occur; it is dangerous for a real fall to remain undetected. While specificity is secondary, it is still important. It is inconvenient and impractical for a fall detection system to have frequent false positives, which would make for an unreliable system, and any alarms would likely not be taken as seriously, making the system less effective. Table 5 clearly indicates that U-Fall achieves good results on both sensitivity and specificity.

# 6. DISCUSSIONS AND CONCLUSIONS

As discussed in the previous sections, U-Fall is able to achieve very high sensitivity (which means the chance of missing real fall events is low), as well as high specificity (which means the probability of making false alarms is also low). However, the specificity of U-Fall is around 75%. While this value is comparable to the state-of-the-art, we believe that more research should be devoted to further improving the specificity (in another word, to further reduce the false alarms). We plan to investigate some new pattern matching techniques and time delayed embedding and ensomble learning techniques.

The distributed analytics and edge intelligence proposed in our system utilized both smartphones and cloud services. The experiments show that U-Fall can achieve a low miss rate and a low false positive rate when compared against the state-of-the-art systems. Next, we plan to test U-Fall with larger scale data sets. We also plan to closely work with our clinical collaborators to deploy U-Fall in the clinical practice.

# 7. REFERENCES


[1] Amazon web services, inc. http://aws.amazon.com/.

[2] Stroke 101 fact sheet by national stroke association, centennial, co, u.s.a.

[3] S. Abbate, M. Avvenuti, G. Cola, P. Corsini, J. Light, and A. Vecchio. Recognition of false alarms in fall detection systems. In *CCNC 2011*, 2011.

[4] S. Abbate, M. Avvenuti, P. Corsini, J. Light, and A. Vecchio. *Monitoring of Human Movements for Fall Detection and Activities Recognition in Elderly Care Using Wireless Sensor Network: a Survey.* InTech, 2010.

[5] F. Bagalï£¡ï£¡, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk. Evaluation of accelerometer-based fall detection algorithms on real-world falls. *PloS one*, 7(5):e37062, 2012.

[6] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer, 2014.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[8] A. Bourke, P. Van de Ven, M. Gamble, R. O'Connor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. OLaighin, and J. Nelson. Evaluation of waist-mounted tri-axial accelerometer based fall-detection algorithms during scripted and continuous unscripted activities. *Journal of Biomechanics*, 43(15):3051–3057, 2010.

[9] J. C. Castillo, D. Carneiro, J. Serrano-Cuerda, P. Novais, A. Fernández-Caballero, and J. Neves. A multi-modal approach for activity classification and fall detection. *International Journal of Systems Science*, 45(4):810–824, 2014.

[10] C. P. Chen and C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.

[11] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. In *EMBC 2005*, pages 3551–3554. IEEE, 2006.

[12] J. Crosbie, S. Lennon, J. Basford, and S. McDonough. Virtual reality in stroke rehabilitation: still more virtual than real. *Disability and Rehabilitation*, 29(14):1139–1146, 2007.

[13] J. Frank, S. Mannor, J. Pineau, and D. Precup. Time series analysis using geometric template matching. *PAMI*, 35(3), 2013.

[14] A. S. Go, D. Mozaffarian, and et al. Heart disease and stroke 2013 statistical update. *Circulation Journal by American Heart Association*, 127:6–245, 2013.

[15] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. St Pierre, et al. Big data: The future of biocuration. *Nature*, 455(7209):47–50, 2008.

[16] R. Kahn, R. M. Robertson, R. Smith, and D. Eddy. The impact of prevention on reducing the burden of cardiovascular disease. *Circulation Journal by American Heart Association*, 108:576–585, 2008.

[17] Q. Li, J. Stankovic, M. Hanson, A. Barth, J. Lach, and G. Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Proc. of BSN 2009*, 2009.

[18] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proc. of the 8th SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003.

[19] J. Lockhart, G. Weiss, J. Xue, S. Gallagher, A. Grosner, and T. Pulickal. Design considerations for the wisdm smart phone-based sensor mining architecture. In *Proc. of the Fifth International Workshop on Knowledge Discovery from Sensor Data, Held in Conjunction with The 17th ACM SIGKDD*, volume 25-33, 2011.

[20] J. W. Lockhart, T. Pulickal, and G. M. Weiss. Applications of mobile activity recognition, 2012.

[21] S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 2010.

[22] V. Mirchevska, M. Lu?trek, and M. Gams. Combining domain knowledge and machine learning for robust fall detection. *Expert Systems*, 2013.

[23] H. PA, T. JG, K. OA, B. J, D. K, E. MD, F. EA, H. Y, J. SC, K. A, L.-J. DM, N. SA, N. G, O. D, W. PW, and W. YJ. Forecasting the future of cardiovascular disease in the united states: a policy statement from the american heart association. *Circulation Journal by American Heart Association*, 123:933–944, 2011.

[24] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm. Smart items, fog and cloud computing as enablers of servitization in healthcare. 2015.

[25] A. Tayal, M. Tian, K. Kelly, S. Jones, D. Wright, D. Singh, J. Jarouse, J. Brillman, S. Murali, and R. Gupta. Atrial fibrillation detected by mobile cardiac outpatient telemetry in cryptogenic tia or stroke. *Neurology*, 71(21):1696–1701, 2008.

[26] L. M. Vaquero and L. Rodero-Merino. Finding your way in the definition fog: Towards a comprehensive of fog computing. *ACM SIGCOMM COMPUTER COMMUNICATION REVIEW*, 44(5):27–32, 2014.

[27] Z. Yang. Powertutor-a power monitor for android-based mobile platforms. *EECS, University of Michigan, retrieved September*, 2, 2012.