A Framework for Component Selection in Collaborative Sensing Application Development

Jie Cao, Lingmei Ren, Weisong Shi
Department of Computer Science
Wayne State University
Detroit, MI, 48202
{jiecao, lingmei.ren, weisong}@wayne.edu

Zhifeng Yu
MobiHealth Technologies
Oakland Township, MI, 48363
zhifeng.f.yu@gmail.com

***Abstract*—Wireless sensor network-based technologies and applications have attracted a lot of attention in the past two decades because of their huge potential to change people's way of life. These applications usually need close collaboration among multiple sensors, gateways, services and end users. When developing these applications, system designers and practitioners usually face several performance requirements such as the accuracy, battery life and system reliability. Given the hard requirements in system performance, how to choose an optimal combination from various sensors, algorithms and collaborative systems to form the application is the most important problem that practitioners need to address. Ad hoc solutions were proposed in specific applications in the past; however, a general methodology that can be easily applied to future applications is lacking. In this paper, we take the challenge and propose a general framework aiming to address the component selection problem, illustrate how this framework can be applied to real life applications through a case study, and discuss challenging issues and two interesting finds from our implementation.**

Keywords : component selection, collaborative sensing

I. Introduction

The fast development and deployment of wireless communication technologies, and mobile devices, including sensors, robots, smart phones and tablets, have significantly changed the way we live [1], [2], [3], [4], [5], [6]. Wireless sensor network-based technologies and applications have been widely used in process management, health care monitoring, and environmental sensing, and so on. Trans-disciplinary collaboration is very common in these applications. For example, a wireless health application needs closely collaboration from health and medical research groups, mechanical engineers, computer scientists, doctors and nurses, and health insurance companies. System design is one of the most important tasks in the collaborative sensing application development. There are several challenges that need to be addressed in the design of collaborative sensing applications. One such challenge is meeting the performance requirements from service providers and end users. These requirements could cover quality of information, battery life, hardware size and weight, and system cost, and so on [7], [8], [9]. Moreover, in some applications, the performance requirements could not be fixed, but are adaptive. For example, a sensor could working in a high power mode to achieve good performance when it is powered by a plug, and it could also turn to a worse performance to conserve energy and extend battery life when powered by the battery.

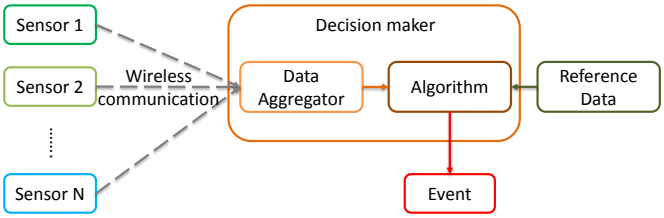

Fig. 1. System overview of a collaborative sensing application.

In designing collaborative sensing applications, practitioners normally face a great deal of choices in the components of the applications. Figure 1 shows a conceptual view of a typical collaborative monitoring application's system structure. When a target event happens, raw data is collected on the spot by wireless sensors, and then sent to data aggregator through wireless communication channels such as Wi-Fi, Zig-bee or Blue-tooth. The raw data is then processed in the decision maker, and then a decision is made based on the result of the detection algorithm with processed data as input. In the decision making process, the decision maker could also refer to the data reported by the outside systems. For example, a fire detection system makes decision mainly on the smoke detection sensor, however, it could also use a real-time camera to find the fire source and use this information in making the decision.

Combination of system components such as sensors, data transfer methods, detection algorithms and reference data sources obviously have significant impact on system's performance. Moreover, the number of combinations could be considerable large and practitioners could feels no place to start when facing such a large dataset. In this way, how to choose the appropriate combination of system components that can meet the performance requirement with less cost is the key issue in collaborative sensing application design. The practitioners could test all of the component combinations to find the best solution, if possible. However, it could be a tediously long process to go through the whole dataset. Best to the authors' knowledge, a much more efficient method

to optimize the component selection is still a missing part in the collaborative sensing area. In this paper, we take the challenge and propose a general framework aiming to address the component selection problem.

The remainder of this paper is organized as follows. In Section II, we survey the related research works in current literature. In Section III we propose performance vector, which is a metric can be used in evaluating the performance of multiple component combinations for different requirements. Section IV depicts a methodology for using performance vector in optimizing component combination selection. A case study using our methodology is introduced in Section V. We further discuss two interesting findings we have observed in the case study in Section VI. Finally, concluding remarks and future work are presented in Section VII.

II. Related work

Performance evaluation for collaborative sensing applications has been studied in extensive research work [10], [11], [12], [13], [14], [15]. In [16], Zeigel *et al.* proposed the notion of weighted diagnostic distortion (WDD), which is a novel measure for data quality in a health care related application. Based on this work, a general methodology for developing quality of information for body sensor design was proposed in [17]. In this work, the authors claim that the performance of the application should be evaluated based on their capability in finishing the desired job. However, the work above did not give a general framework that can be used in evaluate the performance of collaborative sensing applications from different requirement aspects.

In wireless sensor network area, various sensor selection schemes are developed to make trade off between power conservation and quality of service [18], [19], [20]. Moreover, several work mentioned that component selection is a important issue in system design [1], [2]. To the best of our knowledge, a general framework for component selection has not been proposed yet in the literature.

III. performance metrics vector

There are several performance requirements for a collaborative sensing application, including those from service providers, end users, and/or hardware constrains. In designing and developing collaborative sensing applications, practitioners need to understand the performance of the system very well in order to satisfy the requirements. With multiple requirements and a large number of component choices, it is very difficulty for the practitioners to evaluate and compare every choice. In this section, we propose performance vector, which is a metric can be used in evaluating the performance of different component combinations.

A. Performance requirements

Before designing the combination of components for an application, it is extremely significant for the practitioners to define the requirements of performance very clearly. Moreover, it is also very important that the proper metrics to describe the performance are selected. For instance, to demonstrate Quality of Information (QoI) [15], [21], [22], practitioners can choose root mean square error (RMSE), percentage RMS difference (PRD), or signal to noise ratios (SNR), and so on. Moreover, the requirements could cover various aspects such as accuracy, battery life, sensor size and weight, and so on. In this paper, we use R_x to indicate the performance requirements. For example, $R_{Accuracy}$ = 90% means the application requires accuracy to be at least 90%, and $R_{BatteyLife}$ = 24 hrs imply the system should work longer than one day before battery runs out.

B. Performance score

To demonstrate the performance of a component combination i, we introduce $P_{i,x}$. For example, $P_{i,Accuracy}$= 90% means the accuracy of component combination i is 90%. We also come up with a performance score $S_{i,x}$ to denote the relationship between the performance and requirement for a component combination.

$$S_{i,x} = \begin{cases} P_{i,x} & \text{if performance is better than} \\ & \text{or equal to } R_x, \\ Null & \text{if performance is worse than } R_x. \end{cases} \quad (1)$$

From Equation 1 we can see that the performance score $S_{i,x}$ is $Null$ if the performance of component combination i cannot satisfy the requirement R_x. We also set the value of performance score to be $P_{i,x}$, so that the practitioners can easily compare the performance of different component combinations.

C. Performance vector

With performance score $S_{i,x}$ one can describe the component combination's performance for one requirement very clearly. Nevertheless, there are normally multiple requirements for a collaborative sensing application. In this case, practitioners need a metric to demonstrate a component combination's performance from several requirement aspects. To overall evaluate the performance of a component combination, we define performance vector V as follows:

$$V_i =< S_{i,1}, S_{i,2}, ..., S_{i,n} > \quad (2)$$

In this way, practitioners can easily eliminate the combinations that do not meet all of the performance requirements if there is $Null$ in their performance vector. Moreover, they can also compare the performance over multiple requirement aspects between component combinations intuitively.

D. A toy example

In order to better demonstrate how the performance vector can help practitioners in component combination selection, we build a very simple toy example. In this toy example, the target application has two requirements in performance, $R_{Accuracy}$ and $R_{BatteryLife}$. For the practitioners, there are four available component combinations, denoted as CC_1, CC_2, CC_3, and CC_4. After measurement and calculation, we get the performance vectors of these four combinations

as $V_1 =< 0.3, Null >$, $V_2 =< 0.3, 3 >$, $V_3 =< Null, 4 >$, and $V_4 =< 0.5, 5 >$.

In this case, combination 1 can not be choose because $S_{1,BatteryLife} = Null$, which means the battery life of this component combination can not satisfy the requirement. Similarly, combination 3 should also be excluded since $S_{3,Accuracy}$ is $Null$. Combination 4 is the optimal choice because both $S_{4,Accuracy}$ and $S_{4,BatteryLife}$ are larger than that of combination 2, which means combination 4 has a better accuracy, while the battery life is longer than combination 2.

IV. Methodology for using performance vector

As shown in the toy example in last section, with performance vector, practitioners can make the optimal choice of component combinations for an application. However, There could be a large number of possible combinations in a complex application, and only a small portion of them can meet the requirements. In this paper, we name these component combinations as 'possible combinations'. In order to conserve the workload and improve the efficient in system designing, practitioners should calculate the performance vector only for the possible combinations rather than the whole sets. In this section, we propose a methodology that can be applied easily to eliminate the impossible combinations and find the optimal choice through calculating the performance vector of possible combinations.

1) Define performance requirement: Based on the performance requirements from service providers, end users, and/or hardware consratins, and so on, the practitioners need to define and quantize the requirements into a set of values $< R_1, R_2, R_3, ..., R_n >$.

2) List all the component combinations: In this step, the practitioners need firstly define the components in the system design, and then find all the available choices for each component. With the number of components m, and n_i for the number of available choices in component i, we can have

$$\text{Number of component combinations} = \prod_{i=1}^{m} n_i \qquad (3)$$

3) Sort the list for each requirement: For one component, it is usually not difficult to order the available choices based on the given requirement by using the empirical approach or a simple experiment. In this step, practitioners should sort the list in previous step for each component and each requirement separately.

4) find and test the best combination for each requirement: When the practitioners have the sorted list of component combinations for each requirement, it is very simple to find the best performance combination for each requirement. Practitioners should then test the best combinations to see if they can meet the requirements. If any of the performance vectors has $Null$ inside, which means for some requirement, even the best combination can not satisfy it. In this case, practitioners should either consider lowering the requirement, or trying to change the system design. For example, leverage other technologies to make new components.

5) Exclude impossible component combinations: To find the impossible component, practitioners can start with the best combinations in Step 4. By replacing only one component choice in the best combinations, practitioners can easily test if the replacement is possible or not for the application. For this test, practitioner should always start with the 'worst' choice in the sorted list in Step 3. This step has significant influence in reducing the workload for component combination selection. For example, in a dataset of 100 component combinations, if we can reduce the possible choices in one component from 5 to 4, which means we exclude only one choice for this component, the total dataset size will be reduced to $100 \times 4 \div 5 = 80$. In other word, we reduce 20 % of the dataset size by only eliminate one choice from one component.

6) Calculate performance vector for all the possible combinations: After exclude the impossible combinations, practitioners can calculate the performance vector for the remaining part. If any of the $S_{i,x}$ is $Null$, this combination i should be eliminated from the possible combinations, and there is no need to calculate other $S_{i,x}$ in V_i.

7) Find the optimal combination: Now practitioners should have a table of the performance vectors for all the possible combinations, and they can choose the optimal combination based on the highest score in the table. In some applications, there is no component combination that achieves best performance score for all the requirement. In this circumstance, the practitioners need to make a choice depend on the focus or key requirement.

V. Applying performance vector in a real application: Walking position detection

In this section, we declare how to use our methodology in a real collaborative sensing application. A wireless low-power fall detection application called Asgard [23] was used in this case study. An Asgard wireless sensor contains an accelerometer collecting acceleration, a flash memory that can store the data, and a CPU to process the data. Asgard sensor was fixed on the left ankle of each user, and each user was asked to walk 100 steps on flat ground and take stairs for another 100 steps. Practitioners wanted to identify the walking segment that is shown in the left part of Figure 2 and the stair segment that is depicted in the right part. In addition, they also wanted to count exactly how many steps were performed on flat ground and on stairs. The acceleration in the detection of gravity (Ag) was calculated by applying the Pythagorean Theorem.

$$Ag = \sqrt{x^2 + y^2 + z^2} \qquad (4)$$

In this case study, the decision-making algorithm put a threshold on Ag and used it to identify the walking segment and the stair segment.

A. Define performance requirement

In this application, there are three performance requirements that need to be satisfied:

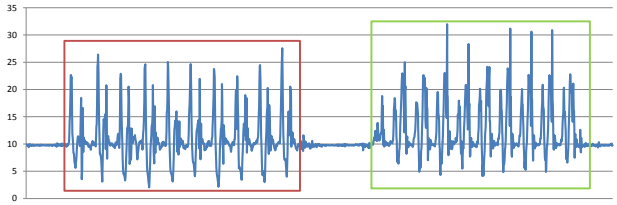

Fig. 2. The walking segment and the stair segment.

1) Accuracy: There is no doubt that the quality of data is the most notable feature in collaborative sensing applications. No matter if we consider service providers or receivers, the fidelity of the application is always the highest priority issue to be taken care of [24], [17]. In order to measure the accuracy of the detection, we introduce three measures in statistics, which are precision, recall and F_{score}. In this application, there are two events, walking on flat ground and taking stairs, so the algorithm will have the following four outputs.

True positive (TP): stair step correctly identified as stair step.

False positive (FP): flat ground step incorrectly identified as stair step.

True negative (TN): flat ground step correctly identified as flat ground step.

False negative (FN): stair step incorrectly identified as flat ground step.

Then we also list the definitions of precision and recall here.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

From the above equations we can deduce that both precision and recall scales are from 0 to 1, and in the ideal case, both precision and recall are equal to 1. In addition, we can understand that the larger precision and recall indicated the better performance our algorithm gives. In order to measure the accuracy of the system with precision and recall, we introduced F_{score}, which is the harmonic mean of precision and recall:

$$F_{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

We can also deduce that F_{score} scales from 0 to 1, and in the ideal case, $F_{score} = 1$. Similar to precision and recall, a larger F_{score} means our detection is more accurate. In this application, service providers came up with the requirement of F_{score} to be 0.9 for the fidelity of the data, so we set $R_{Accuracy} = 0.9$.

2) Battery Life: Because Asgard is powered by a battery, battery life is another key requirement to evaluate its performance [25], [26]. In this application, end user want that the Asgard sensor can work for more than one week without recharge the battery for convenience [27], which means the battery life should be longer than $24 \times 7 = 168$ hrs. In our expression, $R_{BatteryLife} = 168$.

3) Data quantity: The storage size of Asgard sensor is 512 MB, subjected to this hardware constrain, there should be a requirement in data quantity. In this case study, we set $R_{DataQuantity} = 512$ to restrict the amount of data collected by Asgard sensor.

As a conclusion, there are three performance requirements for this application, $R_{Accuracy} = 0.9$, $R_{BatteryLife} = 168$, and $R_{DataQuantity} = 512$.

B. Component combination analysis

1) Component with multiple choices: In this application, we found that there are three components that have multiple choices.

First is the number of axes of accelerometer we use in the application. Asgard sensor uses a 3-axes accelerometer, which means we can get the data from x, y, and z axes. In this case study, we have the following seven choices in axes: xyz, xy, xz, yz, x, y, and z. Using different number of axes does not affect battery life significantly since all the three axes are collecting data and we cannot shut down any axe while the sensor is working. However, it could affect the accuracy and data quantity a lot.

The second component that we can leverage is the sampling rate of the accelerometer. Asgard sensor allows us to configure the sampling rate to four distinctive values: 6 Hz, 15 Hz, 50 Hz, and 200 Hz. Using different sampling rate has influence on accuracy, battery life and data quantity.

The last component is decimal digits. In Asgard sensor, we can flexibly compress the data size to different decimal digits or even round it to integer. In this application, we tried four different decimal digits, which are a.bcd, a.bc, a.b, and integer part a only. Similar to the choice of axes, different decimal digits has influence on accuracy and data quantity, but can barely affect battery life because decimal digits is fixed to be a.bcd in raw data collection.

By applying equation 3, The total number of component combinations is $7 \times 4 \times 4 = 112$.

2) Sort the combinations: Common knowledge tells us that for the choice of axes, more axes could lead to a better accuracy, however, it could also use more space to store the data; and a higher sampling rate usually means a better accuracy, a larger data quantity and a shorter battery life, considering it could consume more power for data collection; similarly, more decimal digits stands for a better accuracy, but a larger data quantity.

Based on the above analysis, we sort all the component combinations for each of the requirements, as shown in the following table.

As we can see from this table, using more axes, more decimal digit, and higher sampling rate will take more space to store data and consumes more energy, however, accuracy could also be increased as a benefit.

3) The best combination: According to the above table, for accuracy, the best component combination is xyz + a.bcd + 200 Hz. We tested this combination and find out the accuracy

Accuracy

Data quantity

Axes	Decimal digit	Sampling Rate
xyz	a.bcd	200 Hz
xy/xz/yz	a.bc	50 Hz
x/y/z	a.b	15 Hz
	a	6 Hz

Battery life

$P_{Accuracy}$ is 1.0. Since $P_{Accuracy} > R_{Accuracy}$, it passes the test.

For battery life, the best combination is x + a + 6 Hz. The battery life is calculated by the following equation.

$$\text{battery life} = \frac{\text{battery capacity}}{\frac{\text{power dispassion}}{\text{working voltage}}} \tag{8}$$

Asgard sensor has the battery capacity of 2100 mAh with working voltage at 4V , and we measured the power consumption for 6 Hz sampling rate is 29 mW. By applying Equation 8, we have $P_{BatteryLife}$ = 290, which has a better performance than $R_{BatteryLife}$ = 168.

For data quantity, combination x + a + 6 Hz is still the best combination. For each data entry, we need to have at least two Bytes to store it, one is sign bit and another is the number. Each second, this combination will generate $6 \times 2 =$ 12 B data, and since $R_{BatteryLife}$ = 168, the total data size $P_{DataQuantity}$ is $168 \times 3600 \times 12 = 7.2576$ MB, which is smaller than $R_{DataQuantity}$ = 512.

4) Exclude impossible component combinations: In this case study, we first tested 200 Hz sampling rate. For 200 Hz sampling rate, the longest battery life it could achieve is using combination: x/y/z + 200 Hz + a. The $P_{BatteryLife}$ we got for this combination is 88, which is less than $R_{BatteryLife}$ = 168. So in this test, we exclude 200 Hz from the possible choice.

We then replace the number of axes in the best combinations. We tried to use only two axes, which gives us the combinations xy/xz/yz + 50 Hz + a.bcd for the best accuracy. The test result is show in Figure 3.

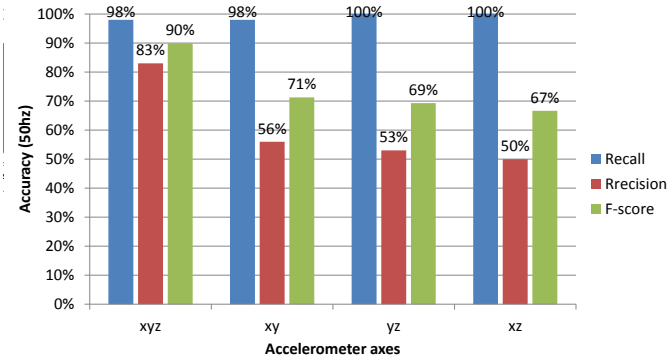

Fig. 3. Accuracy of different axes combination.

From Figure 3 we can see, only using all of the three axes can provide $P_{Accuracy} = 0.9 = R_{Accuracy}$, so in this test, we eliminate all the axes choices but xyz, since it is the only possible component choice.

We also tested 6 Hz sampling rate for the best accuracy requirement it can achieve. By testing combination xyz + 6 Hz + a.bcd, we got $P_{Accuracy} = 0.87$, which is smaller than $R_{Accuracy}$. So we exclude 6 Hz sampling rate from possible component choice list.

For decimal digits, we tested choice integer a. The best accuracy it can get is through combination xyz + 50 Hz + a, and the test result shows $P_{Accuracy} = 0.87$. This means only save the integer data is not accurate enough for this application.

5) Performance vector for all the possible combinations: After the above elimination, there are only 6 possible combinations left, we name them CC_1 to CC_6 and list all of them in the following table.

Name	Component combination
CC_1	xyz + 15 Hz +a.b
CC_2	xyz + 15 Hz +a.bc
CC_3	xyz + 15 Hz +a.bcd
CC_4	xyz + 50 Hz +a.b
CC_5	xyz + 50 Hz +a.bc
CC_6	xyz + 50 Hz +a.bcd

We then measure the performance of these 6 component combinations from accuracy, battery life and data quantity.

Accuracy:

In this case study, we calculate the F_{score} and show the result in Figure 4.

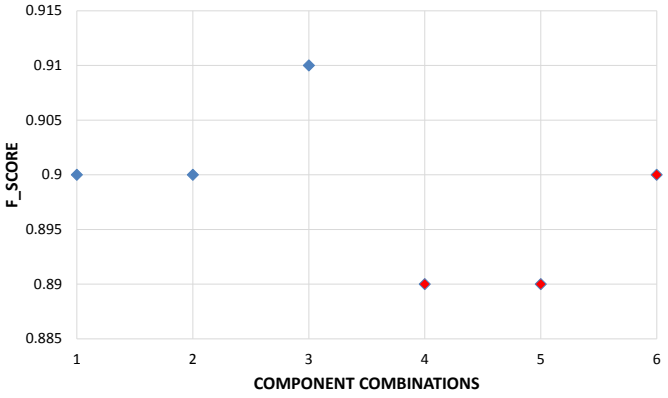

Fig. 4. F_{score} of component combinations.

By applying Equation 1, we get the $S_{i,Accuracy}$ for all the six component combinations and list them in the following table.

$S_{1,Accuracy}$	0.9
$S_{2,Accuracy}$	0.9
$S_{3,Accuracy}$	0.91
$S_{4,Accuracy}$	Null
$S_{5,Accuracy}$	Null
$S_{6,Accuracy}$	0.9

From the table above we can see that both $S_{4,Accuracy}$ and $S_{5,Accuracy}$ are $Null$, which means component combinations CC_4 and CC_5 cannot be used in the application, so in the following measurement, we only consider component combinations CC_1, CC_2, CC_3, and CC_6.

Data quantity:

In this case study, for each data entry, we need to have one Byte to store sign bit and one more for each number. In the table below we list the format of one data entry for component combinations CC_1, CC_2, CC_3, and CC_6.

CC_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	$\pm$	x	b_x	$\pm$	y	b_y	$\pm$	z	b_z						
2	$\pm$	x	b_x	c_x	$\pm$	y	b_y	c_y	$\pm$	z	b_z	c_z			
3	$\pm$	x	b_x	c_x	d_x	$\pm$	y	b_y	c_y	d_y	$\pm$	z	b_z	c_z	d_z
6	$\pm$	x	b_x	c_x	d_x	$\pm$	y	b_y	c_y	d_y	$\pm$	z	b_z	c_z	d_z

As we can see from the table, the length of one data entry for component combinations CC_1, CC_2, CC_3, and CC_6 respectively are 9 B, 12 B, 15 B, and 15 B. By applying the following equation:

$$P_{Dataquantity} = \text{Sampling rate} \times \text{Battery life} \times \text{Length of one data entry} \quad (9)$$

We can have the $S_{Dataquantity}$ for each component combination, as listed in this table:

$S_{1,DataQuantity}$	81.684
$S_{2,DataQuantity}$	108.864
$S_{3,DataQuantityy}$	136.08
$S_{6,DataQuantity}$	453.6

All of these four component combinations meets the requirement for data quantity, which is $R_{DataQuantity}$ = 512.

Battery Life:

As we have discussed before, only sampling rate has influence on battery life, and since we have collected $P_{BatteryLife}$ for 6 Hz and 200 Hz in the previous tests, here we present all of the results in one figure. Firstly, we measured the power dissipation for all of the four sampling rates, as shown in Figure 5.

By applying Equation 8 to the data in Figure 5, we get the $P_{BatteryLife}$ for all the four sampling rates, and the result is demonstrated in Figure 6. We continue calculating the $S_{BatteryLife}$ for all the four component combinations, and list the result in the table below.

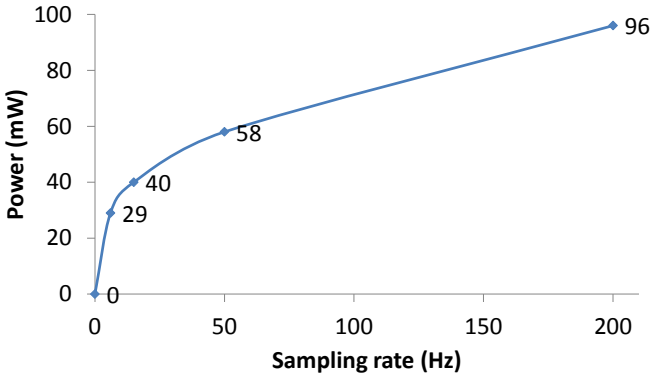

Fig. 5. Effect of sampling rate on power dissipation.

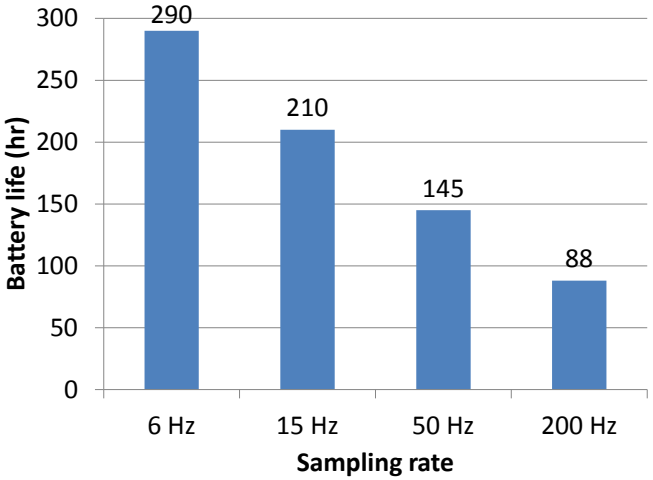

Fig. 6. Effect of sampling rate on $P_{BatteryLife}$.

$S_{1,BatteryLife}$	210
$S_{2,BatteryLife}$	210
$S_{3,BatteryLife}$	210
$S_{6,BatteryLife}$	Null

Since $S_{6,BatteryLife} = Null$, we need to exclude CC_6 from the possible combinations.

Now we have three component combinations left, and all of them meet the requirements from accuracy, battery life and data quantity. We summarize their performance and get their performance vector in this table:

Combination	$S_{i,Accuracy}$	$S_{i,BatteryLife}$	$S_{i,DataQuantity}$
CC_1	0.9	210	81.684
CC_2	0.9	210	108.864
CC_3	0.91	210	136.08

C. Find the optimal combination

From the table above we can see that all of the component combinations have the same performance in battery life, and

preform very similarly in accuracy. However, since CC_1 uses much less storage space compared with CC_2 and CC_3, in this application, we select CC_1 as the best combination and use it in the real application developent.

VI. Two Interesting Findings in the Case Study

When we are testing our methodology for using performance vector in our case study, we fully studied the relationship between data quality, battery life and data quantity for the application. We observed two interesting phenomenons during in case study. The first one is the effect of decision making algorithm on data quality, and the second one is different operation periods for battery.

A. *Effect of decision making algorithm on data quality*

In our case study, we first collected data using these four sampling rates and ran the classification algorithm on them, respectively. The threshold used in the algorithm was firstly set to 4.8 empirically, and the F_{score} was then calculated and shown in Figure 7. As expected, the F_{score} of the datasets has a positive correlation with the sampling rate, which suggests that with a larger quantity of data as input, decision makers can more easily make the correct diagnosis.

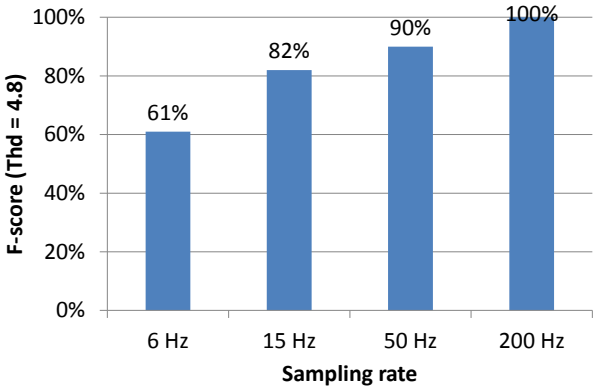

Fig. 7. Effect of sampling rate on F_{score}.

By analyzing the raw data collected at different sampling rates, we found that the information that is useful in diagnosis is presented in a more detailed fashion with a larger sampling rate. However, a problem was also introduced when increasing the sampling rate. The difference between the walking segment and the stair segment was reduced since both of them contain more details at a larger sampling rate, which could lead to a worse performance of classification even if the sampling rate is enlarged. Based on this observation, we applied a series of thresholds from 4.6 to 6.2 on the four datasets and tested their accuracy separately. Figure 8 and Figure 9 show the recall and precision, and the F_{score} is presented in Figure 10.

We can see from Figure 8 that no matter which threshold is used, the recall will not decrease when enlarging the sampling rate, or in other words, it is easier to identify the step movements from the non-step ones such as standing still. Nevertheless, Figure 9 tells us that in most cases, the precision

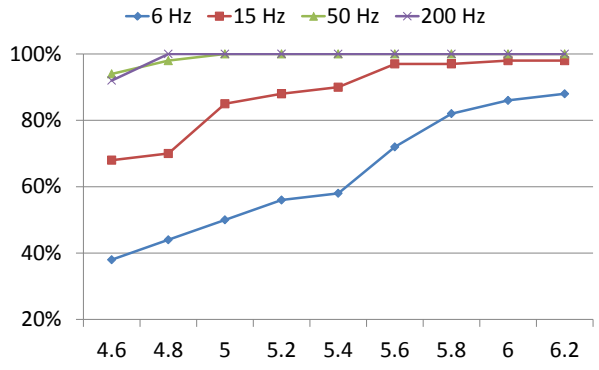

Fig. 8. Effect of threshold on Recall.

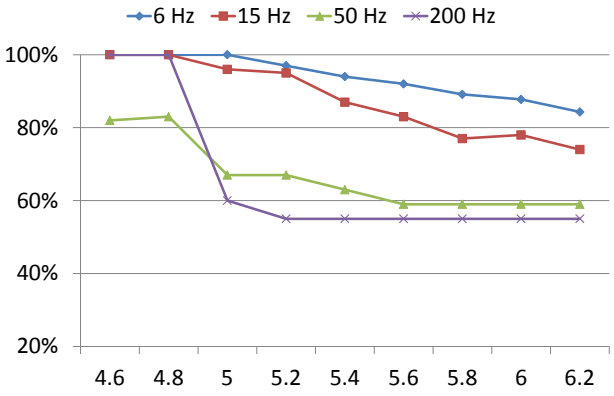

Fig. 9. Effect of threshold on Precision.

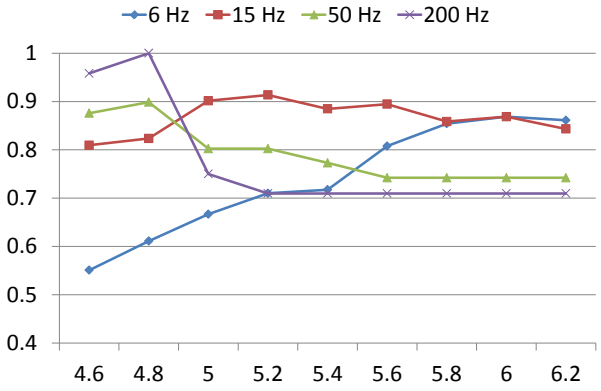

Fig. 10. Effect of threshold on F_{score}.

decreases while the sampling rate increases from 6 Hz to 200 Hz, which verifies our deduction that it is more difficult to identify the stair segments from the walking segments when the data is collected at a larger sampling rate. Then we took these two factors into consideration and got Figure 10, from which we can see that when the threshold is below 5, the performance increases with the sampling rate; however, if we take a look at the overall figure, the F_{score} of each sampling rate varies. For 50 Hz and 200 Hz, the accuracy of the

algorithm decreases when enlarging the threshold, and for 15 Hz the F_{score} does not vibrate too much; however, for 6 Hz the performance becomes better if we use a larger threshold. This means each sampling rate reaches its highest F_{score} at different thresholds and the algorithm should be adapted when the sampling rate is changed to get the best accuracy.

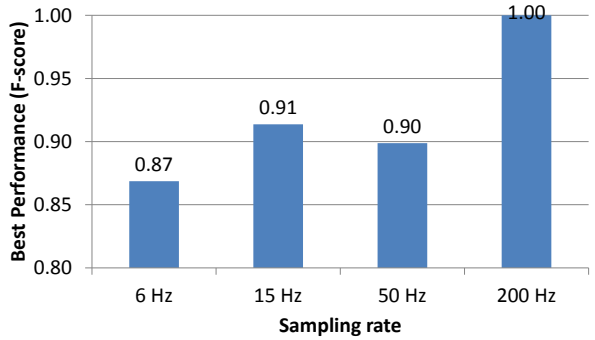

Fig. 11. Highest F_{score} each sampling rate can get.

Based on the above observation, we extracted the best performance that each sampling rate can get from adaptive thresholds and showed the results in Figure 11. Comparing it with Figure 7 we can make the following two conclusions. First is that enlarging the data quantity in this case study will not increase the data quality too much. As shown in Figure 11, both 6 Hz and 15 Hz can get the F_{score} around 0.9, which is the highest F_{score} that 50 Hz can get. Second, if the practitioners have to exchange some data quantity for a longer battery life, they can modify the algorithm to shrink the gap of data quality introduced by this trade off. In some cases, the practitioners can attempted to use a cheaper or more energy saving sensor but still achieve a good performance if the corresponding decision making algorithm is properly modified [28], [29], [30]. We believe the second finding here can be applied to other wireless health applications and benefit their practitioners in similar situations [31].

B. Three operation period for battery

In the case study, we have observed that when the battery is about to running out, Asgard sensor works abnormally by logging incorrect data. To better understand the reason of this abnormality, the operating characteristic of batteries was analyzed first to give us a better understanding of a battery's discharge process [32]. We first fully charged the battery of Asgard sensor, then configured the sensor to continually log acceleration data at 200 Hz sampling rate until the battery runs out. Considering 88 hours of battery life is too long for us to monitor, we changed to a 400 mAh battery. During the whole process we kept measuring the working voltage of the sensor to get the discharge curve of the battery, as shown in Figure 12.

Figure 12 shows that the Asgard sensor can continually work for 17 hours for one charge, and the working voltage of the sensor drops from $4.1V$ to $2.5V$ until the battery runs

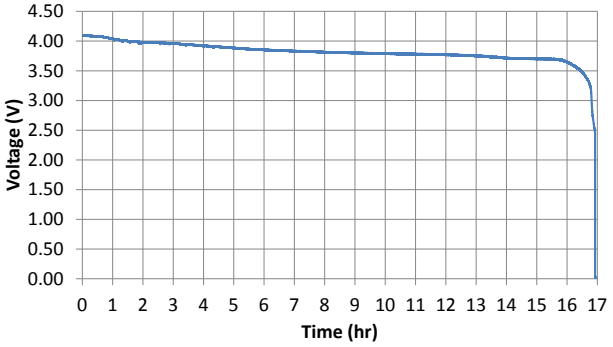

Fig. 12. Discharge curve of Asgard sensor battery.

out. Also, we can see from Figure 12 that during the first 16 hours of 17 hours working time, the working voltage is only changed subtly from $4.1V$ to $3.7V$. In the last hour, however, the working voltage decreases significantly until the sensor shuts down.

Working voltage has a significant impact on the performance stability of electronic devices. While the battery is running out, there is usually a certain stage in which the device can still be powered by the battery, yet the performance is not stable due to the low working voltage level. This phenomenon is more likely observed on a low-power device. For example, a flashlight could be dim or work intermittently when the battery runs out. Since wireless health applications generally use low-power sensors to guarantee a long battery life, such as in this case study the power dispassion of the Asgard sensor is lower than $100mW$, we suspect that this phenomenon could also appear in wireless health applications that involve low-power sensors. To verify this hypothesis, we performed an off-body analysis on an Asgard sensor. In the off-body analysis, one Asgard sensor powered by a battery was placed on a flat table as the treatment group, and we also set another Asgard sensor powered by a wire on the same table as the control group. We kept recording the acceleration reported by both sensors until the battery of the first sensor ran out. The results show that the wire powered sensor reports $9.8m/s^2$ throughout the process, and the data reported by the battery powered sensor is shown in Figure 13.

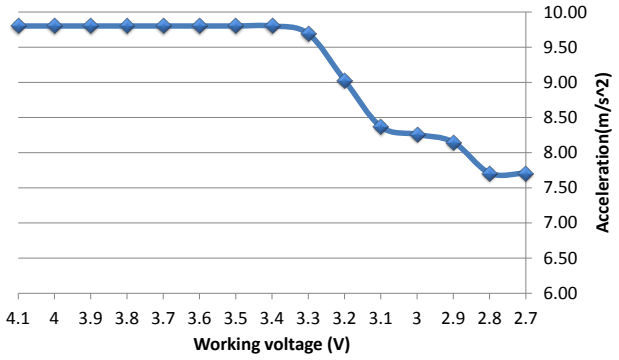

Fig. 13. Effect of working voltage on Asgard sensor.

Figure 13 shows that when the working voltage drops from $4.1V$ to $3.4V$, the sensor works normally and reports the correct acceleration. However, when the working voltage is below $3.4V$, the data reported by the sensor drops with the working voltage. Finally, when the working voltage is lower than $2.7V$, the sensor does not work. Based on this observation, we define the following three sensor operation periods:

Working Period (WP): when the battery capacity is sufficient so that the sensor can work normally.

Transfer Period (TP): when the working voltage is low and the sensor's performance is affected.

Non Working Period (NWP): when the working voltage is too low to support the sensor.

In this case study, the sensor first worked in the working period, which means the stage when the working voltage is between $4.1V$ and $3.4V$. Then the working voltage dropped from $3.4V$ to $2.7V$, which means the sensor worked in the transfer period. Last, the sensor went to the non working period when the working voltage was below $2.7V$. Obviously, The deviation between the values reported by these two sensors powered by the wire and battery affected the accuracy of the fall detection algorithm in the Asgard system. However, it is usually an arduous task to catch the transfer period during real usage of the sensor. In this case study, in order to evaluate the influence of the error introduced by the low working voltage on data quality, we used the normalized root-mean-square deviation (NRMSD) to show the difference between the values reported by the wire powered Asgard sensor and the values reported by the battery powered one, where lower values indicate less residual variance.

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(x_{b,i} - x_{w,i})^2}{n}} \tag{10}$$

$$NRMSD = \frac{RMSD}{x_{max} - x_{min}} \tag{11}$$

In Equation 10, x_b denotes the values collected by the battery powered sensor, and x_w denotes the data from wire powered sensor. Variable n was set to 1000, which means for every $0.1V$ from $2.7V$ to $3.4V$. We collected 1000 samples and used them to calculate the result. NRMSD of these two datasets is shown in Figure 14.

Figure 14 shows that data quality in the transfer period is not as good as when battery capacity is sufficient, and it becomes worse when working voltage keeps decreasing. In order to identify how long the transfer period lasts, we zoomed in on Figure 12 and located the points corresponding to $3.4V$ and $2.7V$. The result is presented in Figure 15. From Figure 15 we can see that the transfer period took about 15 minutes in the 17-hour working period. This may not be a large number, but if we extend the battery life to days or even months, the transfer period could also be prolonged to hours or days. Since the data collected in this stage is highly unreliable, the practitioners must be aware of this stage and take appropriate measures to avoid making incorrect decisions from it.

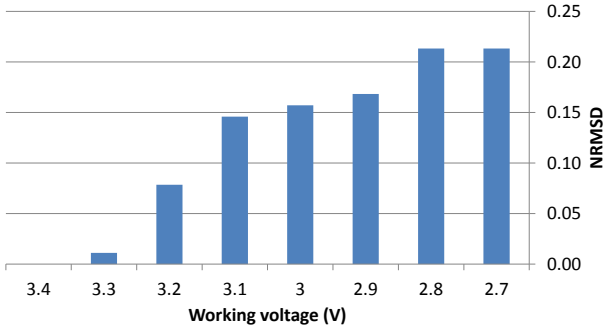

Fig. 14. Data quality recession in the transfer period.

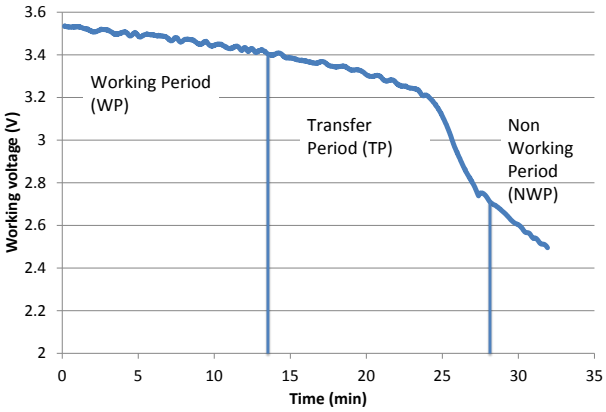

Fig. 15. Three operation periods of battery.

VII. Conclusions and Future Work

In the design and development of collaborative sensing applications, practitioners usually face several performance requirements from service providers, end users, and/or hardware constrains. Moreover, there could be a large number of available choices for different components in the application. To help the practitioners to select the optimal component combinations that can meet the performance requirements and reduce cost as much as possible at the same time, we proposed performance evaluation metrics and a methodology for using performance vector. We tested our methodology through a case study, and the result showed that our methodology is very efficient in finding the optimal component combination. Finally, we discussed two interesting findings we have observed in the case study, one is the effect of decision making algorithm on data quality, and the other is different operation periods for battery.

The case study used in this paper is rather simple, and the requirements are not difficult to measure and quantize. Finding an efficient way to deal with multiple multidimensional parameters is still a challenge. In the future, we will apply our methodology to more complicated applications with reference data from outside systems.

REFERENCES

[1] M. Hanson, H. Powell, A. Barth, K. Ringgenberg, B. Calhoun, J. Aylor, and J. Lach, "Body area sensor networks: Challenges and opportunities," *Computer*, vol. 42, no. 1, pp. 58–65, 2009.

[2] S. Kumar, W. Nilsen, M. Pavel, and M. Srivastava, "Mobile health: Revolutionizing healthcare through transdisciplinary research," *Computer*, vol. 46, no. 1, pp. 28–35, 2013.

[3] W. H. Wu, A. A. Bui, M. A. Batalin, L. K. Au, J. D. Binney, and W. J. Kaiser, "Medic: Medical embedded device for individualized care," *Artificial Intelligence in Medicine*, vol. 42, no. 2, pp. 137–152, 2008.

[4] B. Lo, S. Thiemjarus, R. King, and G.-Z. Yang, "Body sensor network-a wireless sensor platform for pervasive healthcare monitoring," in *The 3rd International Conference on Pervasive Computing*, vol. 13, 2005, pp. 77–80.

[5] G.-Z. Yang and M. Yacoub, "Body sensor networks," 2006.

[6] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *Journal of Mobile Multimedia*, vol. 1, no. 4, pp. 307–326, 2006.

[7] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," 2006.

[8] Y. Wang, J. P. Lynch, and K. H. Law, "A wireless structural health monitoring system with multithreaded sensing devices: design and validation," *Structure and Infrastructure Engineering*, vol. 3, no. 2, pp. 103–120, 2007.

[9] S. Chen, J. S. Brantley, T. Kim, and J. Lach, "Characterizing and minimizing synchronization and calibration errors in inertial body sensor networks," in *Proceedings of the Fifth International Conference on Body Area Networks*, ser. BodyNets '10. New York, NY, USA: ACM, 2010, pp. 138–144.

[10] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM, 2001, pp. 139–150.

[11] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," *Computers, IEEE Transactions on*, vol. 52, no. 6, pp. 753–763, 2003.

[12] R. Iyer and L. Kleinrock, "Qos control for sensor networks," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 517–521.

[13] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, 2002.

[14] M. A. Yigitel, O. D. Incel, and C. Ersoy, "Qos-aware mac protocols for wireless sensor networks: A survey," *Computer Networks*, vol. 55, no. 8, pp. 1982–2004, 2011.

[15] N. Sarma and S. Nandi, "Qos support in mobile ad hoc networks," in *Wireless and Optical Communications Networks, 2006 IFIP International Conference on*, 2006.

[16] Y. Zigel, A. Cohen, and A. Katz, "The weighted diagnostic distortion (wdd) measure for ecg signal compression," *Biomedical Engineering, IEEE Transactions on*, vol. 47, no. 11, pp. 1422–1430, 2000.

[17] I. Armenti, P. Asare, J. Su, and J. Lach, "A methodology for developing quality of information metrics for body sensor design," *Wireless Health 2012*, 2012.

[18] H. Rowaihy, S. Eswaran, M. Johnson, D. Verma, A. Bar-Noy, T. Brown, and T. La Porta, "A survey of sensor selection schemes in wireless sensor networks," in *Defense and Security Symposium*. International Society for Optics and Photonics, 2007, pp. 65 621A–65 621A.

[19] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM, 2004, pp. 129–143.

[20] J. S. Brantley, A. T. Barth, and J. Lach, "Optimizing battery lifetime-fidelity tradeoffs in bsns using personal activity profiles," in *Proceedings of the 7th International Conference on Body Area Networks*, ser. BodyNets '12.

[21] Z.-Y. Demetrios, "A glance at quality of services in mobile ad-hoc networks," in *Seminar in Mobile Ad Hoc Networks*, 2001.

[22] S. Chakrabarti and A. Mishra, "Qos issues in ad hoc wireless networks," *Communications Magazine, IEEE*, vol. 39, no. 2, pp. 142–148, 2001.

[23] L. Ren, Q. Zhang, and W. Shi, "Low-power fall detection in home-based environments," in *Proceedings of the 2nd ACM international workshop on Pervasive Wireless Healthcare*. ACM, 2012, pp. 39–44.

[24] K. Sha and W. Shi, "Consistency-driven data quality management of networked sensor systems," *Journal of Parallel and Distributed Computing*, vol. 68, no. 9, pp. 1207 – 1221, 2008.

[25] R. Istepanian, E. Jovanov, and Y. Zhang, "Guest editorial introduction to the special section on m-health: Beyond seamless mobility and global wireless health-care connectivity," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 8, no. 4, pp. 405–414, 2004.

[26] A. Milenkovic, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Computer Communications*, vol. 29, no. 13-14, pp. 2521 – 2533, 2006.

[27] Q. Zhang, L. Ren, and W. Shi, "Honey: A multimodality fall detection and telecare system," *Telemedicine and e-Health*, 2013.

[28] J. P. Lynch, K. H. Law, A. S. Kiremidjian, T. W. Kenny, E. Carryer, and A. Partridge, "The design of a wireless sensing unit for structural health monitoring," in *Proceedings of the 3rd International Workshop on Structural Health Monitoring*, 2001, pp. 12–14.

[29] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, and J. Stankovic, "An advanced wireless sensor network for health monitoring," in *Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2)*, 2006, pp. 2–4.

[30] M. A. Hanson, H. C. Powell, Jr., A. T. Barth, and J. Lach, "Application-focused energy-fidelity scalability for wireless motion-based health assessment," *ACM Trans. Embed. Comput. Syst.*, vol. 11, no. S2, pp. 50:1–50:21, Aug. 2012.

[31] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, "Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach," in *Wearable Computers (ISWC), 2012 16th International Symposium on*. IEEE, 2012, pp. 17–24.

[32] D. Raychev, Y. Li, and W. Shi, "The seventh cell of a six-cell battery," in *Proceedings of the third Workshop on Energy-Efficient Design (WEED 2011)*, 2011.