Tracking Human Motions in Photographing: A Context-Aware Energy-Saving Scheme for Smart Phones

YAFENG YIN, LEI XIE, YUANYUAN FAN, and SANGLU LU, State Key Laboratory for Novel Software Technology, Nanjing University

Due to the portability of smart phones, more and more people tend to take photos with smart phones. However, energy-saving continues to be a thorny problem, since photographing is a rather power hungry function. To extend the battery life of phones while taking photos, we propose a context-aware energy-saving scheme called "SenSave." SenSave senses the user's activities during photographing and adopts suitable energy-saving strategies accordingly. SenSave works based on the observation that a lot of energy during photographing is wasted in preparations before shooting. By leveraging the low power-consuming embedded sensors, such as accelerometer and gyroscope, we can recognize the user's activities and reduce unnecessary energy consumption. Besides, by maintaining an activity state machine, SenSave can determine the user's activity progressively and improve the recognition accuracy. Experiment results show that SenSave can recognize the user's activities with an average accuracy of 95.5% and reduce the energy consumption during photographing by 30.0%, when compared to the approach by frequently turning ON/OFF the camera or screen. Additionally, we enhance "SenSave" by introducing an extended Markov chain to predict the next activity state and adopt the energy-saving strategy in advance. Then, we can reduce the energy consumption during photographing by 36.1%.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing design and evaluation methods**; *Empirical studies in ubiquitous and mobile computing*;

Additional Key Words and Phrases: Activity sensing, energy saving, smart phones, photographing

ACM Reference format:
Yafeng Yin, Lei Xie, Yuanyuan Fan, and Sanglu Lu. 2017. Tracking Human Motions in Photographing: A Context-Aware Energy-Saving Scheme for Smart Phones. *ACM Trans. Sen. Netw.* 13, 4, Article 29 (September 2017), 37 pages.
https://doi.org/10.1145/3085578

A preliminary version of this work was presented in *Proceedings of the 12th IEEE International Conference on Mobile Ad Hoc and Sensor Systems* (IEEE MASS 2015) (Fan et al. 2015).
This work is supported in part by National Natural Science Foundation of China under Grant Nos. 61472185, 61373129, 61321491, 61502224; JiangSu Natural Science Foundation under Grant No. BK20151390; Fundamental Research Funds for the Central Universities under Grant No. 020214380035; 2016 Program A for outstanding PhD candidate of Nanjing University under Grant No. 201601A008. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.
Authors' addresses: Y. Yin, L. Xie (corresponding author), Y. Fan, and S. Lu, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China; emails: yyf@dislab.nju.edu.cn, lxie@nju.edu.cn, fyymonica@dislab.nju.edu.cn, sanglu@nju.edu.cn.

1 INTRODUCTION

1.1 Motivation

Nowadays, smart phones are widely used in our daily lives. Due to the portability of smart phones, more and more people tend to take photos with their smart phones, for example, taking photos at a tourist attraction. However, energy-saving continues to be an upsetting problem for smart camera phones, since photographing is a very power-hungry function. For example, according to KS Mobile's (KS Mobile Inc. 2014) report in 2014, the application *Camera360 Ultimate* is listed in first place of the top 10 battery-draining applications for Android. Therefore, the huge energy consumption becomes a non-negligible pain point for the users of smart camera phones. Consequently, it is essential to reduce the unnecessary energy consumption during photographing to extend the battery life of smart camera phones.

1.2 Limitations of Prior Art

Prior work on energy saving of smart phones can be classified into the following three parts: energy consumption of hardware (Fan et al. 2007; Bellosa et al. 2003; Rajan et al. 2006; Balasubramanian et al. 2009), power consumption models, and energy-saving schemes for specific applications. *For hardware*, Chen et al. (2013a) analyze the power consumption of AMOLED displays in multimedia applications and reveal that camera recording incurs high power cost. LiKamWa et al. (2013) report the experimental and analytical characterization of CMOS image sensors and reveal two energy-proportional mechanisms for energy saving. *For models*, Dong and Zhong (2011) propose Sesame, with which a mobile system constructs an energy model of itself without any external assistance. Xu et al. (2013) propose a new way called V-edge to generate power models based on battery voltage dynamics. *For specific applications*, Han et al. (2013) study the energy cost made by human-screen interaction, such as scrolling on the screen. Dietrich and Chakraborty (2013) detect the game's current state and decrease the processor's voltage and frequency whenever possible to save energy. Hu et al. (2013) propose a Mobility-Assisted User Contact detection algorithm (MAUC), which utilizes the accelerometer of the phone to detect user movements for energy-saving. The Bluetooth scans only when user movements have a high possibility to cause contact changes. LiKamWa et al. (2013) improve the energy efficiency of image sensors based on hardware modifications. There are fewer energy-saving schemes for photographing.

Being different from these prior work, we aim to propose an energy-saving scheme for photographing. We aim to recognize the user's activity and reduce unnecessary energy cost when the user is not taking photos. The scheme does not need hardware modifications and user interaction, to guarantee a good user experience.

1.3 Proposed Approach

A straight solution to reduce energy cost is to turn off the camera or screen while not taking photos. However, frequently turning ON/OFF the camera or screen is very annoying and leads to a bad user experience. Besides, frequently turning on the camera or screen will incur high energy consumption. Take the Samsung Galaxy Nexus smart phone as an example, the energy consumption of the pair of operations, that is, turning off the camera and screen and then turning on the screen and camera, can keep the camera working on preview mode for about 7s.

To propose an efficient energy-saving scheme, we conduct extensive observations. We find that during photographing, a fairly large proportion of energy is wasted in preparations before shooting. For example, the user usually first turns on the camera. Then, he/she will probably adjust the phone time and again, to find a good camera view. Finally, when the camera focuses on the target, the user will press the button to shoot. Between two consecutive shots, the camera works with the same settings (e.g., the same preview size), which result in comparable energy consumption to that of shooting photos. If we can recognize the user's activity and detect the duration between two consecutive shots, then we can decrease the screen brightness, preview size, or the preview frame rate to reduce energy cost.

In this article, by leveraging activity sensing, we propose a context-aware energy-saving scheme for smart camera phones. Our idea works based on the observation that most smart phones are equipped with low power-consuming sensors, such as the accelerometer and gyroscope. We can leverage these tiny sensors to recognize the user's activities, such that the corresponding energy-saving strategies (e.g., decreasing the screen brightness, decreasing the frame rate, etc.) can be applied. To reduce the error of activity recognition, we maintain an activity state machine to determine the activity state progressively. In addition, we also introduce an extended Markov chain to predict the next activity state, to adopt a suitable energy-saving strategy in advance to further reduce energy cost. Without user interaction, we can reduce the energy consumption during photographing while guaranteeing a good user experience.

1.4 Technique Challenges and Solutions

There are some challenges in activity sensing and designing the energy-saving scheme for taking photos with smart phones.

- **Activity sensing**: The first challenge is how to use the sensor data for activity recognition. To address this challenge, we propose a three-level architecture, which classifies the activities into three levels: body level, arm level, and wrist level. For the sensor data of a potential activity, we first utilize the variance and periodicity of the sensor data to classify the activity into one of the three levels. For activities in the same level, we combine data from different sensors to distinguish one from another based on the features of activities. To reduce the error of activity recognition, we maintain an activity state machine and determine the user's activity state progressively.
- **Energy-saving scheme design**: The second challenge is how to design an appropriate energy-saving scheme with the recognized activities during photographing. To address this challenge, we propose a context-aware energy-saving scheme SenSave, which adopts suitable energy-saving strategies based on the user's activities. In body level, SenSave focuses on turning ON/OFF sensors, camera, and screen. In arm level, SenSave focuses on adjusting the screen brightness, starting or stopping the camera preview. In wrist level, SenSave focuses on adjusting the preview size, the preview frame rate of the camera. In each level, we will adjust the parameters in an energy-saving strategy for the specific activity.
- **Trade-off between activity sensing and energy saving**: The third challenge is how to make an appropriate trade-off between the accuracy of activity sensing and energy consumption. Obviously, more types of sensor data and larger sampling rates contribute to higher accuracy of activity sensing, while resulting in more energy consumption. To address this challenge, we only leverage the low power-consuming sensors like accelerometer and gyroscope for activity recognition. When guaranteeing the recognition accuracy, we choose the sampling rates of sensors as small as possible. For further energy saving, we introduce an extended Markov chain to predict the next activity state and adopt the suitable energy-saving strategy in advance.

1.5 Key Contributions

We make the following contributions in this article (a preliminary version of this work appeared in Fan et al. (2015)).

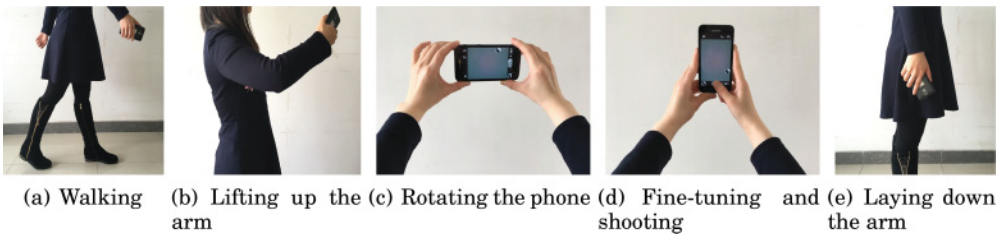


Fig. 1. Human activities during photographing.

— First, we propose a context-aware energy-saving scheme for smart camera phones, by leveraging the built-in sensors for activity sensing. Based on the activity recognition results, we can adopt corresponding energy-saving strategies.
— Second, we build a three-level architecture for activity sensing, including body level, arm level, and wrist level. We use the low power-consuming sensors like accelerometer and gyroscope to extract representative features to distinguish one activity from another. By maintaining an activity state machine, we can determine the user's activity progressively and reduce the error of activity recognition.
— Third, we design an efficient energy-saving scheme, which can adaptively choose a suitable energy-saving strategy without user interaction, according to the activity state. Besides, we also introduce an extended Markov chain to predict the next activity state, to adopt a suitable energy-saving strategy in advance for further energy saving.
— Fourth, we have implemented a system prototype in android-powered smart camera phones. The experiment results show that our solution is able to recognize the user's activities with an average accuracy of 95.5%. Besides, we can reduce the energy consumption during photographing by 30.0%, when compared to the approach by frequently turning ON/OFF the camera or screen. By introducing the extended Markov chain, we can reduce the energy consumption during photographing by 36.1%.

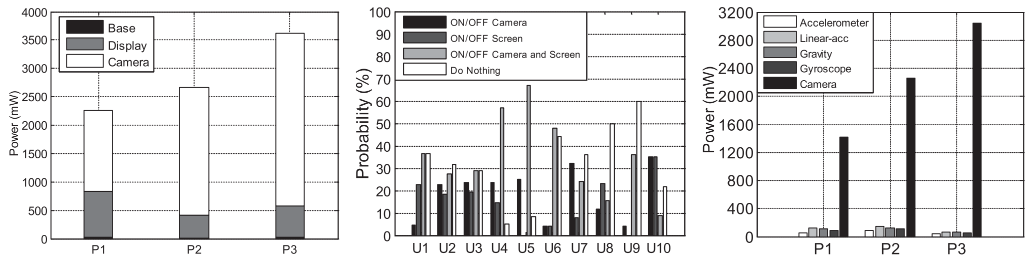
2 OBSERVATIONS ON PHOTOGRAPHING

2.1 Human Activities Related to Photographing

During photographing, the users tend to have similar activities, as shown in Figure 1. Before or after the user takes photos, he/she may stay motionless or keep moving, for example, walking, jogging, and so on. While taking photos, the user usually lifts up the arm, rotates the phone, makes fine-tuning, shoots a picture, then lays down the arm. We categorize all the activities into three levels. (1) *Body level:* motionlessness, body movement. (2) *Arm level:* lifting up the arm, laying down the arm. (3) *Wrist level:* rotating the phone, making fine-tuning, shooting a picture. If the user wants to take multiple photos, then he/she may keep the camera working on the preview state, to take the next photo conveniently. However, it is rather energy consuming to keep the camera working. Therefore, many users tend to turn off the camera between two consecutive shots, unless he/she needs to take multiple photos in a short time.

2.2 Energy Consumption Related to Photographing

According to Figure 1, before shooting a photo, there will be a *preparation time*, during which the user needs to move his/her locations, adjusts the position of the phone or makes fine-tuning, to find a good camera view. Obviously, if the user keeps the camera working with large preview size, it will incur much energy consumption, because the preparation time cannot be ignored. Besides,



(a) Energy consumption of three components (b) Probability of turning ON/OFF camera or screen of 10 users (c) Energy consumption of built-in sensors

Fig. 2. In (a) and (c): P1, Samsung Galaxy Nexus; P2, Samsung Galaxy S5; P3, Samsung Galaxy Note4.

frequently turning on/off the camera can also incur extra energy consumption. To optimize the energy-consuming parts and propose an efficient energy-saving scheme, we first use Monsoon power monitor (Monsoon Solutions Inc. 2015) to measure the energy cost in photographing.

2.2.1 Energy Consumption in Preparation for Photographing. Power consumption in preparation time of photographing is large. We observe the power consumption in the following three android-based phones, that is, Samsung Galaxy Nexus, Samsung Galaxy S5, and Samsung Galaxy Note4. In Figure 2(a), we show the power of the phone for three components, that is, "Base," "Display," and "Camera." Here, "Base" means the power when the screen is turned off and the phone stays in the idle state, that is, no app runs except for the system program. "Display" represents the added power (i.e., compared with "Base") by keeping the screen on. "Camera" represents the added power (i.e., compared with "Display" and "Base") by keeping the camera working in preview mode with default settings. During the experiment, we measure the energy consumption in an office with the same light conditions, while each phone adjusts its screen brightness in an automatic way. According to Figure 2(a), the power of keeping the screen on is dozens of times larger than that of "Base," while the power of keeping the camera working is several times of that of "Display." Therefore, when keeping the camera working in preview mode, much energy will be wasted in preparation for photographing. However, if we can recognize the user's activities and detect the preparation time accurately, we can decrease the screen brightness, preview size, preview frame rate, and so on, to reduce the unnecessary energy cost.

2.2.2 Energy Consumption of Turning ON/OFF the Camera. Frequently turning ON/OFF the camera and screen is annoying and wasting energy. Usually, when the user needs to take multiple photos in a period of time, he/she tends to turn ON/OFF the camera or screen frequently for energy saving, instead of keeping the camera working all the time. To know how users adopt the common energy-saving schemes, that is, turning ON/OFF the camera or screen, we invite 10 users to take photos freely in our campus for 20min. For each user, when he/she takes photos, there will be an observer, who will record the user's behavior, that is, the number of times for taking photos, turning ON/OFF camera, turning ON/OFF screen, turning ON/OFF camera and screen. Here, "turning ON/OFF" means a pair of operations, that is, turn off and turn on. For example, turning off the camera and turning it on again means one time for turning ON/OFF camera.

In regard to the three energy-saving schemes, "Turn ON/OFF Camera" means the user turns off the camera after photographing and turning on the camera again for retaking photos. In the process, the user does not turn on/off the screen. "Turn ON/OFF Screen" means the user turns off the screen after photographing and turning on the screen again for retaking photos. Considering the privacy issues, users usually need to unlock the screen (e.g., enter password), when they turn

Table 1. Energy Consumption for Different Operations

Phone \ Operation	Turn on camera (uAh)	Turn off camera (uAh)	Turn on screen (uAh)	Turn off screen (uAh)	Preview (uAh/s)
Nexus	324.61	314.39	301.36	275.89	169.18
S5	456.57	380.40	311.86	267.83	198.42
Note4	763.02	714.58	439.64	255.22	272.01

on the screen. Each user participating in our experiments unlocks the screen after he/she turns on the screen. Thus, unlocking the screen is included in the "Turn ON/OFF Screen." In regard to "Turn ON/OFF Camera and Screen," it means the user turns off both camera and screen after photographing, and then turns on screen and camera for retaking photos.

In Figure 2(b), we show the probability that the user "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen" during photographing. On average, the users adopt one of the above energy-saving schemes with 65.8% probability, instead of doing nothing. Here, "Do Nothing" means that the user keeps the camera working in preview mode during photographing. It usually occurs in the case when taking multiple photos in a short time. "Do Nothing" leads to large energy consumption, as described in Section 2.2.1. Among the three energy-saving schemes, the users prefer to turn off both camera and screen for further energy consumption. On average, the probability of "Turn ON/OFF Camera and Screen" achieves 32.9%. However, frequently user interaction (e.g., touch the screen, press the button) degrades the user experience and it can also increase the energy consumption (Dietrich and Chakraborty 2013).

In Table 1, by using Samsung Galaxy Nexus, Samsung Galaxy S5, and Samsung Galaxy Note4, we show the energy consumption of turning ON/OFF camera or screen one time. In regard to the measurement, take "ON/OFF Screen" as an example, we first keep the screen on and then turn off the screen. The "turning off" operation incurs the power saltation, then we record the energy consumption $E1$ in the following 20s from the moment of saltation. After that, we record the energy consumption $E2$ while keeping the screen off for 20s. Then, we use $E1 - E2$ to represent the energy consumption of "Turning OFF Screen." Similarly, we can measure the energy consumption of other operations. We invite 10 volunteers to participate in the experiment. Each volunteer repeats the operations 50 times, and we average the experiment results. After that, we compare the energy consumption of turning ON/OFF camera or screen with that of keeping camera on. Take Samsung Galaxy Nexus for an example, Equation (1) shows that "Turn ON/OFF Camera," "Turn ON/OFF Screen,"and "Turn ON/OFF Camera and Screen" one time is equivalent to keeping camera in preview mode for 3.78s, 3.41s, and 7.19s, respectively. Similarly, "Turn ON/OFF Camera and Screen" one time in Samsung S5 and Samsung Note4 is equivalent to keeping camera in preview mode for 7.14s and 7.99s, respectively.

$$\begin{aligned}
&\text{ON/OFF Camera: } \frac{324.61\text{uAh} + 314.39\text{uAh}}{169.18\text{uAh/s}} = 3.78\text{s} \\
&\text{ON/OFF Screen: } \frac{301.36\text{uAh} + 275.89\text{uAh}}{169.18\text{uAh/s}} = 3.41\text{s} \\
&\text{ON/OFF Camera and Screen: } 3.78\text{s} + 3.41\text{s} = 7.19\text{s}
\end{aligned} \tag{1}$$

It indicates that frequently turning ON/OFF the camera or screen manually is indeed inconvenient and rather energy consuming. If we can decrease the energy consumption while not requiring the

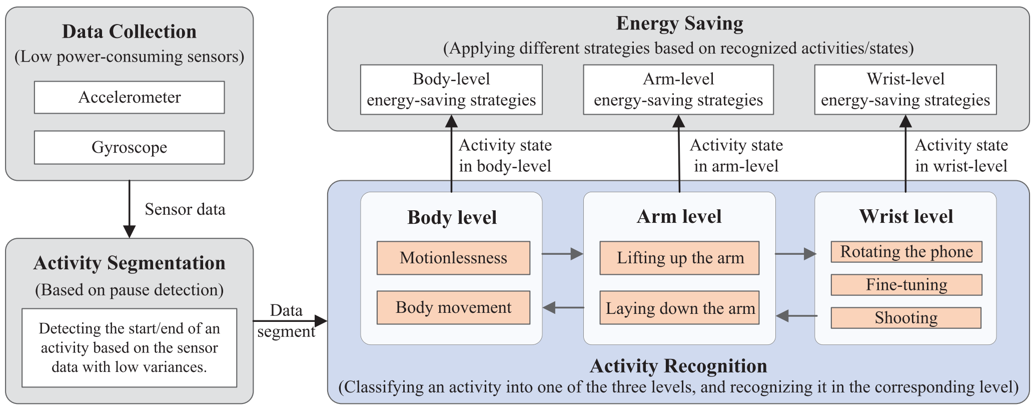

Data Collection
(Low power-consuming sensors)
Accelerometer
Gyroscope
Sensor data
Activity Segmentation
(Based on pause detection)
Detecting the start/end of an activity based on the sensor data with low variances.
Data segment
Energy Saving
(Applying different strategies based on recognized activities/states)
Body-level energy-saving strategies
Arm-level energy-saving strategies
Wrist-level energy-saving strategies
Activity state in body-level
Activity state in arm-level
Activity state in wrist-level
Body level
Motionlessness
Body movement
Arm level
Lifting up the arm
Laying down the arm
Wrist level
Rotating the phone
Fine-tuning
Shooting
Activity Recognition
(Classifying an activity into one of the three levels, and recognizing it in the corresponding level)

Fig. 3. System architecture.

user to frequently turn ON/OFF the camera or screen, then we can extend the battery life and provide a better user experience.

2.2.3 *Energy Consumption of Using Built-in Sensors. It is possible to use low power-consuming built-in sensors for activity sensing and reduce the energy consumption during photographing.* Figure 2(c) shows the power consumption of running the built-in sensor in Samsung Nexus, Samsung S5, and Samsung Note4. All these sensors work with the same sampling rate, that is, 100Hz, which is the maximal sampling rate of sensors in Samsung Nexus. When we measure the power consumption of a sensor, we turn off other sensors. We first record the power consumption without running any sensors as P_{w1}. Then, we record the power consumption by running an sensor as P_{w2}. For P_{w1} and P_{w2}, the phone has the same screen display. Besides, we average the power consumption in 20s when the phone is in a steady state, that is, eliminating the power saltation of starting the sensor. After that, we use $P_{w2} - P_{w1}$ to obtain the power consumption of the sensor.

In Figure 2(c), the sensors include accelerometer, linear accelerometer, gravity sensor, gyroscope, and camera. Based on the definition of *SensorEvent* in Android APIs (Google Inc. 2016a, 2016c), linear acceleration (linear-acc for short) and gravity data are generated from accelerometer, that is, we use a low-pass filter to isolate the force of gravity, then we use the acceleration to minus the gravity data in each axis to obtain the linear acceleration. Therefore, the energy consumption of linear accelerometer is a little larger than that of gravity sensor. Both of them are larger than that of accelerometer. In regard to gyroscope, it usually has larger energy consumption than accelerometer (Park et al. 2011). However, when we turn on the camera and keep the camera working in preview mode, the increase of power consumption is much larger than that of other sensors. Therefore, we can utilize the low power-consuming built-in sensors (i.e., accelerometer and gyroscope) of the phone to detect the user's activities and reduce the energy consumption during photographing. A simple example could be decreasing the screen brightness or turning off the screen, when we find that the user is not taking photos.

3 SYSTEM OVERVIEW

Based on the above observations, we consider using the built-in sensors of smart phones to detect the user's activities and reduce the unnecessary energy cost in photographing. The architecture of our system is shown in Figure 3. First, we collect the data from low power-consuming built-in sensors, that is, accelerometer and gyroscope, as shown in the *Data Collection* module. Second, we extract the activity segment from the sensor data, as shown in the *Activity Segmentation* module. Third, we classify an activity into one of the three levels: body level, arm level, wrist level. Then, we recognize the activity in the corresponding level, as shown in the *Activity Recognition* module. Fourth, we adopt an appropriate energy-saving strategy based on the recognized activity, as shown in the *Energy Saving* module. In the following paragraphs, we will describe how to do activity sensing and adopt energy-saving strategies.

3.1 Activity Sensing

Based on Section 2.1, the user's activities are categorized into three levels: body level, arm level, wrist level. In each level, there may be more than one activity. Besides, the user can transfer from one activity to another. To reduce the error of activity recognition, we maintain an activity state machine to describe the transfer relationship of the activities and determine the user's activity progressively. When we get the sensor data of an activity, we first utilize the variance and periodicity of sensor data to classify the activity into one of the three levels. Then, we combine the data from different sensors to recognize the specific activity in each level.

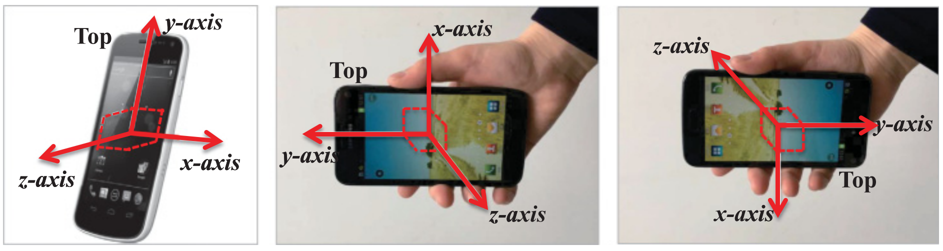
- **Body level:** It includes motionlessness and body movement. Motionlessness can be recognized with its low variance of linear acceleration (linear-acc for short) and gyroscope data. In regard to body movement, which can be walking, jogging, and so on, we utilize the periodicity of an activity to infer whether the current activity belongs to body movement. That is to say, we do not distinguish walking, jogging, and so on, we aim to recognize an activity as body movement or not.
- **Arm level:** It includes lifting up the arm and laying down the arm. We utilize the relationship between the gravity data and linear acceleration to distinguish the two activities. Besides, we use a voting mechanism to improve the recognition accuracy.
- **Wrist level:** It includes rotating the phone, making fine-tuning, and shooting a photo. We make use of a linear SVM model to distinguish them with the following features: variance, mean, maximum value and minimum value of linear acceleration and gyroscope data in each axis.

3.2 Energy Saving

Considering the specific feature of each activity, we propose an adaptive energy-saving scheme called "SenSave" for photographing. Our solution SenSave does not need user interaction (e.g., turn ON/OFF the camera or screen manually) for energy-saving, it utilizes the low power-consuming sensors to detect the user's activity and adopts a corresponding energy-saving strategy adaptively. For example, when you walk, jog or stay motionless, we can keep the screen off. When you lift your arm up, it is better to turn on the screen and adjust the screen brightness based on the ambient light. When you make fine-tuning to observe the camera view before shooting a photo, it is better to make the camera work in preview mode. In this way, we can make context-aware energy-saving strategies for camera phones.

4 SYSTEM DESIGN

In this section, we will show how to use the built-in sensors (i.e., accelerometer and gyroscope) for activity sensing and design an efficient energy-saving scheme for photographing in smart camera phones.



(a) Coordinates of sensors (b) Hold horizontally naturally (c) Hold horizontally backward

Fig. 4. Coordinate system of the phone and the attitude of a phone being held.

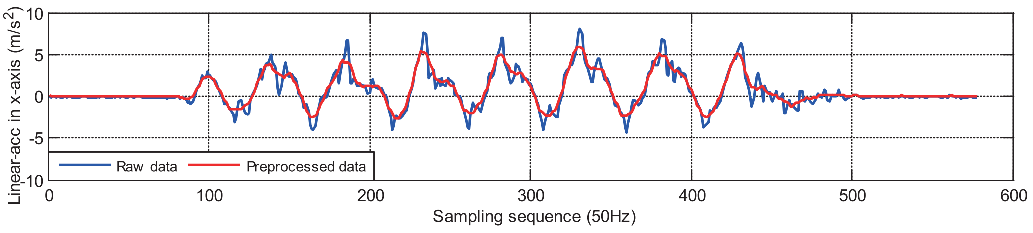


Fig. 5. Data preprocessing.

4.1 Activity Sensing

4.1.1 Raw Data Collection. We collect sequential data from accelerometer and gyroscope. From the acceleration, we use a low-pass filter to isolate the gravity data. Then, we use the acceleration to minus the gravity data in each axis to obtain the linear acceleration (Google Inc. 2016a, 2016c). In the phone, the sensors use a standard 3-axis coordinate system, as shown in Figure 4(a), which may be different from the earth coordinate system. For example, when we hold the phone as shown in Figure 4(b), the gravity data in x-axis x_v almost equals to g (i.e., 9.8m/s^2). When we hold the phone as shown in Figure 4(c), x_v equals to $-g$ (i.e., -9.8m/s^2). Here, g represents the gravity acceleration, which always points towards the ground. It is noteworthy that the direction of the gravity sensor data is opposite to that of the gravity acceleration, according to the definition of *SensorEvent* in Android APIs (Google Inc. 2016a). In this article, we utilize gravity data for activity recognition, while not transforming gravity data to the gravity acceleration. From sensors, we obtain the raw data of linear acceleration (linear-acc for short), gravity data, and gyroscope data, which will be used for following data processing and activity recognition.

4.1.2 Data Preprocessing. The raw sensor data collected from the built-in sensors is usually noisy. To use the sensor data for activity recognition, the data offset and noises should be removed first. For data offset, it means when we keep the device static, the linear-acc and gyroscope data are not equal to zero. Thus, we remove the data offset in each axis. For noises, we use a smooth function to mitigate the effect of noise. As shown in Figure 5, when the sampling rate of the sensor is 50Hz, the blue line shows the raw sensor data of linear acceleration in x-axis. After removing the data offset, we smooth the sensor data in blue line with a 9-point moving average. The smooth function is like a low-pass filter, which removes the high-frequency noise. The red line in Figure 5 represents the smoothed sensor data. The effect of outliers and noises has been mitigated. Unless otherwise specified, we will use the preprocessed sensor data to do activity segmentation for activity detection and recognition, as described in the following sections.

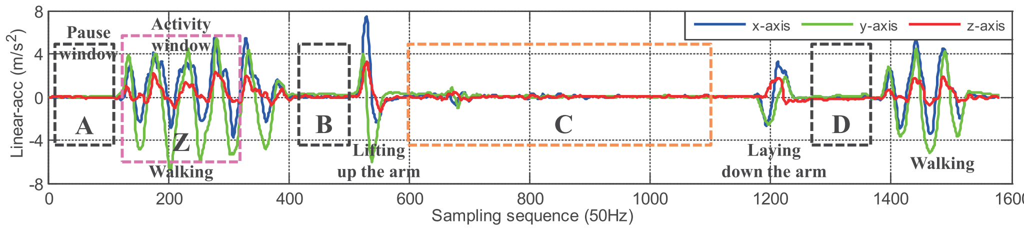


Fig. 6. Linear acceleration.

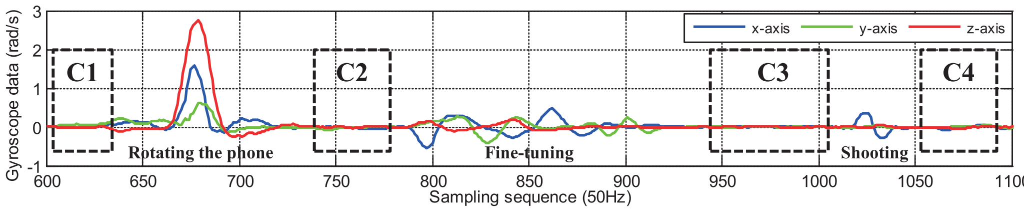


Fig. 7. Gyroscope data in rectangle C.

4.1.3 Activity Segmentation. To detect the user's activity, it is necessary to segment the data of an activity from the sensor data. In Figure 6, we show the sensor data of linear accelerometer during photographing. For the activities in photographing, there is often a short pause between two different activities. In Figure 6, we mark each short pause with a black rectangle, that is, rectangle A, B, and D. Therefore, the linear-acc can be used to separate the following activities: motionlessness, walking (body movement), lifting up the arm, and laying down the arm.

However, there still exist some micro activities like rotating the phone, fine-tuning and shooting, which are too gentle to be distinguished by the linear acceleration, as the rectangle C shown in Figure 6. In this case, we use gyroscope data to assist for activity segmentation. This is because gyroscope is more sensitive to micro activities like rotating, fine-tuning and shooting, when compared to the linear accelerometer. In Figure 7, we show the gyroscope data corresponding to the activities in rectangle C of Figure 6, and mark the detected short pauses with black rectangles C1, C2, C3, and C4. Therefore, we can use gyroscope data to detect the short pause between different micro activities and separate rotating, fine-tuning and shooting. For shooting, we can also use the screen touch operation to assist for activity recognition.

According to the above observations, we utilize the linear acceleration and gyroscope data to do activity segmentation, as shown in the following steps.

— **Step 1:** We introduce a sliding *pause window*, and then calculate the variances of linear-acc during the pause window. Suppose there are n sampling data of linear-acc located in the sliding pause window, we represent the data in x-axis, y-axis, and z-axis as $\{x_{a1}, x_{a2}, \ldots, x_{an}\}$, $\{y_{a1}, y_{a2}, \ldots, y_{an}\}$, and $\{z_{a1}, z_{a2}, \ldots, z_{an}\}$, respectively. The corresponding resultant linear-acc is $a_i, i \in [1, n]$. Here, $a_i = \sqrt{(x_{ai}^2 + y_{ai}^2 + z_{ai}^2)}$. The variances of linear-acc during the pause window is s_a^2, which can be calculated based on Equation (2):

$$s_a^2 = \frac{1}{n-1} \sum_{i=1}^{n} (a_i - \overline{a})^2. \tag{2}$$

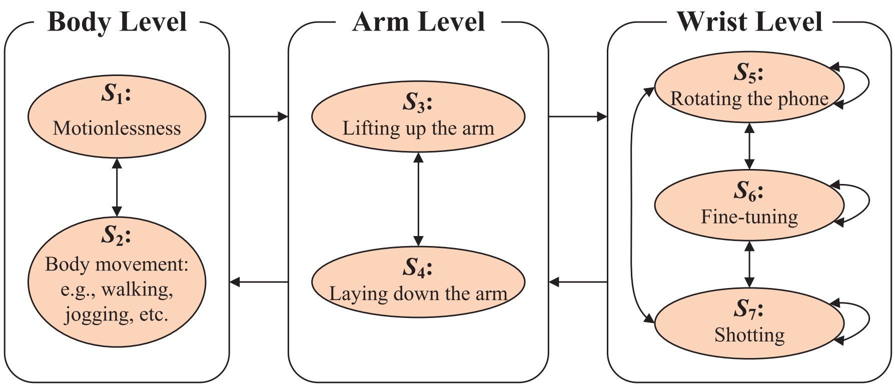

Body Level
Arm Level
Wrist Level
S_1: Motionlessness
S_2: Body movement: e.g., walking, jogging, etc.
S_3: Lifting up the arm
S_4: Laying down the arm
S_5: Rotating the phone
S_6: Fine-tuning
S_7: Shotting

Fig. 8. The activity state machine.

Here, $\overline{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$. If the variance s_a^2 is below the corresponding threshold, that is, satisfying Equation (3), then the window will be regarded as the start/end segment of an activity, as the pause window A, B, D shown in Figure 6. The left side of the window is regarded as the end point of the last activity, while the right side of the window is regarded as the start point of the next activity. Based on extensive observations, we set ϵ_a to 1, and set the size the sliding pause window w_p to 0.5s,

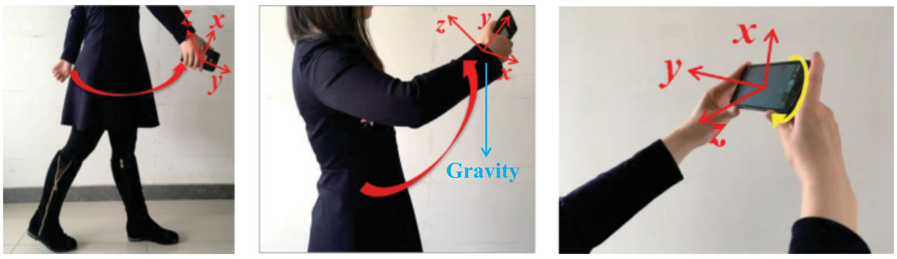
$$s_a^2 < \epsilon_a. \tag{3}$$

—**Step 2:** We move the sliding pause window with the step length $l = \frac{w_p}{2}$, w_p represents 0.5s. If two sliding windows have overlaps and are both regarded as start/end segments, then it may indicate that micro activities (or motionlessness) occur. At this time, we will use the corresponding gyroscope data in the pause window to calculate the variances s_g^2, as described in Step 1. Similarly, if the variances satisfy Equation (4), the window will be regarded as the stat/end segment of an micro activity (or continuous pauses). Otherwise, the sliding window will keep moving forward until it finds the pause between micro activities, as shown in Figure 7. Based on extensive observations, we set ϵ_g to 0.012:

$$s_g^2 < \epsilon_g. \tag{4}$$

—**Step 3:** For a detected pause, it indicates the end of last activity and the start of next activity. Therefore, when we detect a pause, we start a new *activity window* to obtain the sensor data of the following activity. As shown in Figure 6, we mark the activity window with a rectangle Z. Obviously, an activity window represents a potential activity. We recognize the user's activity in the activity window and adopt the corresponding energy-saving strategy until the next pause occurs. Based on extensive observations, we set the size w_a of the activity window to 2s, because 2s is usually enough to obtain the features of an activity, as shown in Figure 6. If the next pause without overlaps occurs in 2s, then we adaptively change w_a to the duration between two consecutive pauses.

4.1.4 Recognition among Different Levels. When we get sensor data of an activity, we should first know which level the activity belongs to, then we can recognize the activity in the corresponding level. To describe the relationship between the levels and activities, we introduce an *activity state machine*, as shown in Figure 8. In the state machine, each activity is represented as a state, and the states are classified into three levels accordingly. Considering the behavior features during photographing, the user cannot transfer from one state to any other state arbitrarily. For



(a) Walking (b) Lifting up the phone (c) Rotating the phone

Fig. 9. Analysis of activity features.

example, if the user has laid down the arm, he/her will probably not lay down the arm again. For body level, the user keeps still or moving (e.g., walking, jogging), it means he/she stays in the same state. Therefore, there is no self-cycle activities in body level. For arm level and wrist level, the short pause between two activities is used for segmenting two consecutive activities, as described in Section 4.1.3. The transfer relationship of states is shown in Figure 8. By maintaining the activity state machine, we can determine the activity state progressively and reduce the error of activity recognition. For example, we may wrongly recognize the activity "Rotating the phone" as "Fine-tuning". Nevertheless, the two states have similar energy-saving strategies, there will not be a sudden decrease of user experience. Usually, we will not recognize "Rotating the phone" as "Walking," which turns off the screen for energy saving and leads to a bad user experience. Therefore, the activity state machine can control the recognition error in a tolerable range and guarantee a good user experience.

For different activities, the amplitude and direction of linear-acc changes differently, as shown in Figure 9. For body level, the linear-acc changes periodically, as shown in Figure 9(a). For arm level, the user usually lifts his/her arm one time, as shown in Figure 9(b). While considering different holding gestures of the phone, the direction variation of linear-acc cannot be mapped to the activities of lifting up or laying down the arm directly. Thus, we introduce the gravity data, which reflects the direction of gravity to assist for activity recognition. In regard to the wrist level, the user tends to rotate the phone to adjust the camera view, as shown in Figure 9(c). Therefore, we combine the linear-acc and gyroscope data to recognize the activities.

With the activity segment from Section 4.1.3, we show how to classify an activity into one of the three levels in Figure 10. We utilize the variance of linear-acc, gyroscope data and the periodicity of sensor data to distinguish body level, arm level and wrist level. For *body level*, if the user stays motionless, the value of linear-acc and gyroscope data is close to zero. The rectangle B in Figure 6 and the rectangle C1 in Figure 7 represent the linear-acc and gyroscope data of pause (or motionlessness), respectively. If the sensor data in a relative long time (e.g., larger than 15s) satisfies Equations (3) and (4), then the activity will be recognized as motionlessness, that is, be classified into body level. In regard to body movement, it does not satisfy Equations (3) and (4). However, we can utilize the periodicity to distinguish body movement from other activities, as the sensor data of "Walking" shown in Figure 6. Here, we simply use the period t_p of the activity for body movement detection. We use $t_{p_1}, t_{p_2}, \ldots, t_{p_{k-1}}, t_{p_k}$ to represent the period of last k activities in body movement, respectively. If t_p satisfy Equation (5), then the activity will be classified into body movement, that is, body level. Based on extensive experiments, we set ϵ_p to 0.2:

$$\left| \frac{t_p - \frac{1}{k}\sum_{j=1}^{k} t_{p_k}}{\frac{1}{k}\sum_{j=1}^{k} t_{p_k}} \right| < \epsilon_p. \tag{5}$$

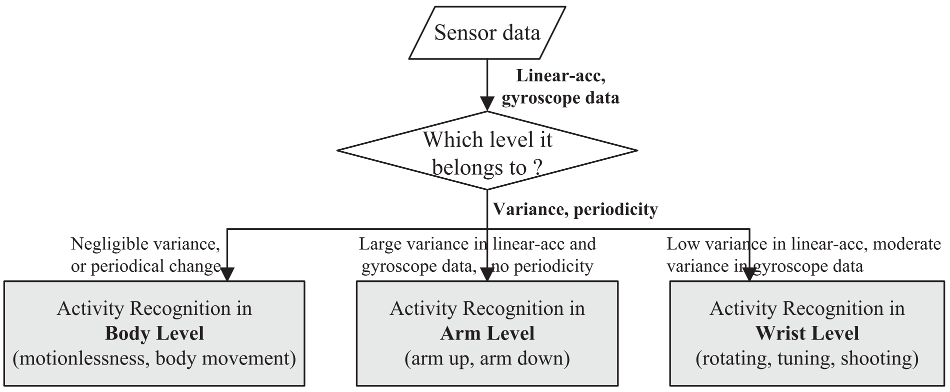

Sensor data
Linear-acc, gyroscope data
Which level it belongs to ?
Variance, periodicity
Negligible variance, or periodical change
Large variance in linear-acc and gyroscope data, no periodicity
Low variance in linear-acc, moderate variance in gyroscope data
Activity Recognition in Body Level (motionlessness, body movement)
Activity Recognition in Arm Level (arm up, arm down)
Activity Recognition in Wrist Level (rotating, tuning, shooting)

Fig. 10. Activity recognition among different levels.

For *arm level*, the user lifts up the arm or lays down the arm. There are no periodicity of consecutive activities, for example, the user cannot continuously lift up the arm. Therefore, if the activity does not satisfy Equations (3), (4), and (5), it will be classified into arm level. For *wrist level*, the user rotates the phone, makes fine-tuning, shoots a photo. The micro movements of the phone satisfy Equation (3) while not satisfying Equation (4). Without loss of generality, we assume the initial activity of the user belongs to body level. That is to say, before the phone runs our scheme "SenSave," the user is motionless or keeping moving (e.g., walking, jogging). The process of activity recognition among three levels is shown in Algorithm 1. When we know which level the current activity belongs to, then we can do activity recognition in the corresponding level, as described in Section 4.1.5.

ALGORITHM 1: Recognition among Three Levels

```
Input: Activity segment $A_i$.
Output: Classified level $L_i$.
Calculate the variance $s_a^2$ of linear-acc of $A_i$ with Equation (2).
Calculate the variance $s_g^2$ of gyroscope data of $A_i$ like Equation (2).
if $s_a^2$ satisfies Equation (3) and $s_g^2$ satisfies Equation (4) then
    Classify $A_i$ into Body Level and recognize the activity as motionlessness.
if $s_a^2$ dissatisfies Equation (3) and $s_g^2$ dissatisfies Equation (4) then
    if Period $t_p$ of $A_i$ satisfies Equation (5) then
        Classify $A_i$ into Body level and recognize it in body movement.
    else
        Classify $A_i$ into Arm level.
if $s_a^2$ satisfies Equation (3) and $s_g^2$ dissatisfies Equation (4) then
    Classify $A_i$ into Wrist level.
Ruturn the classified level $L_i$.
```

4.1.5 Activity Recognition in Each Level. According to Section 4.1.4 and Figure 9, the activities in different levels usually have different features in sensor data. For body-level activities, they usually have large variation in linear-acc and gyroscope data. For arm level activities, we need to combine linear-acc and gravity data to distinguish the lifting-up activity and laying-down activity. For wrist level activities, we combine linear-acc and gyroscope data to distinguish the three micro

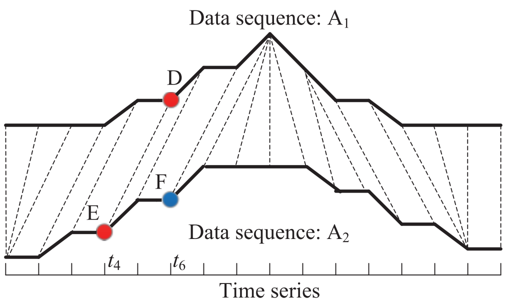


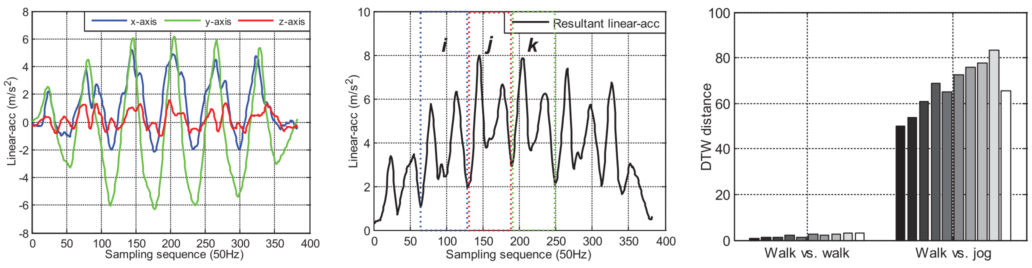
Fig. 11. Working principle of DTW.

activities. In the following paragraphs, we will show how to utilize the sensor data to distinguish one activity from another in each level.

Body Level: As shown in Section 3.1 and Figure 3, the body level includes motionlessness and body movement (e.g., walking, jogging). According to Section 4.1.4, we can utilize the variance of linear-acc and gyroscope data, that is, Equations (3) and (4), to distinguish motionlessness and body movement easily. For body movement, we need to verify that the activity really belongs to body movement instead of disturbances or noises, thus we use Dynamic Time Warping (DTW) (Sempena et al. 2011) for further activity recognition in body level, as follows.

Dynamic time warping (DTW) is used for measuring the similarity between two temporal sequences, which can vary in speed. For example, the user can walk slow or fast, thus the sensor data of an activity segment can be different. Fortunately, by using DTW, we can calculate an optimal match between two activity segments. Thus, we can still recognize the activity, although the variation exists in sensor data. In Figure 11, we show the working principle of DTW. The data sequences A_1 and A_2 are warped non-linearly in the time dimension, to obtain the optimal match between them. For example, at time t_6, the data in A_1 and A_2 are represented as D and F, respectively. However, DTW matches D with E (data at time t_4 in A_2) instead of F. In body movement, we use DTW to measure the similarities between consecutive body movements, as shown in the following steps.

- **Step 1:** *Calculating the resultant linear-acc.* Considering that people may hold the phone in different ways, the coordinate systems can be different, as shown in Figures 4(b) and 4(c). Thus, we will not compare the sensor data of body movements in each axis. Instead, we use the resultant linear-acc for similarity comparison. For an activity segment, the sampling data of linear-acc in x-axis, y-axis, and z-axis are represented as x_{ai}, y_{ai}, and z_{ai}, respectively; $i \in [0, N-1]$. Then, we use $\sqrt{x_{ai}^2 + y_{ai}^2 + z_{ai}^2}$ to obtain the resultant linear-acc. Figure 12(a) shows the linear-acc in each axis, while Figure 12(b) shows the resultant linear-acc of Figure 12(a).
- **Step 2:** *Calculating the DTW distances.* As shown Figure 12(b), we can obtain the resultant linear-acc of each activity segment, such as segment i, j, k. Although some difference exists in the segments, we can use DTW to obtain the best match between them. For any two activity segments (e.g., i and j), we use DTW (Sempena et al. 2011) to calculate the distance between them for similarity comparison. Smaller distance means higher similarity. In Figure 12(c), take the segment of "walking" as an example, we calculate the DTW distance between the segment with 10 "walking" activities, 10 "jogging" activities, respectively. It indicates that the DTW distance between the same type of activities is very small, while



(a) Sensor data of walking (b) Resultant linear-acc (c) DTW distance

Fig. 12. DTW distance calculation.

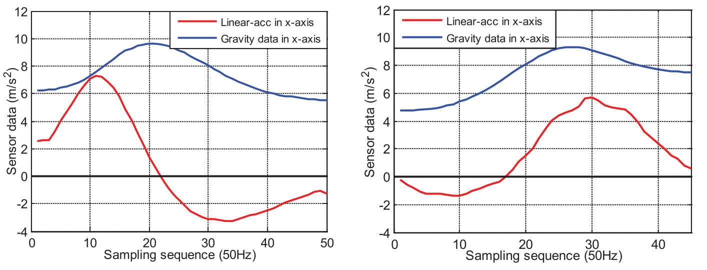
that between different types of activities is rather large. Therefore, DTW distance can be used for detecting the periodical change of body movement, that is, do the same activity. Here, we do not recognize what the specific activity is. Instead, we only verify that the body keeps moving periodically. The user can change from one activity (e.g., walking) to another activity (e.g., running) in body movement, when he/she keeps doing the same activity (e.g., walking), we can recognize it as body movement.

—**Step 3:** *Comparing with previous body movements.* For an activity segment A_i, we calculate the DTW distance between A_i and the last n_t recognized segments in body movement. Therefore, we obtain n_t DTW distances for A_i. Then, we use a voting mechanism for body movement detection. For a DTW distance, if it is less than D_w, then A_i gets a vote from body movement. Finally, A_i gets n_v votes from body movement. If $\frac{n_v}{n_t} \geq \epsilon_b$, then the activity is recognized as body movement. Otherwise, it is treated as a disturbance. We set $n_t = 8$, $D_w = 25$, and $\epsilon_b = 75\%$ by default.

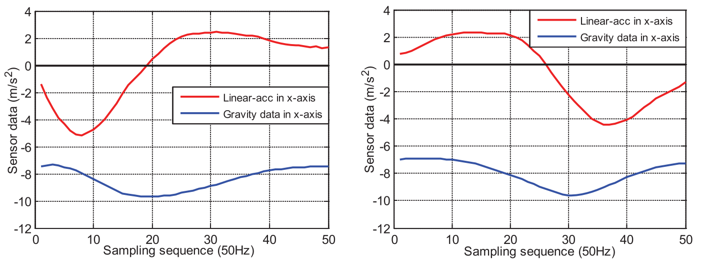
Arm Level: Arm level contains two activities, which are lifting up the arm and laying down the arm. According to Figures 9(b) and 6, lifting up or laying down the arm will cause a large variation of linear-acc in x-axis. When we hold the phone as shown in Figure 4(b), the linear-acc of lifting up the arm and laying down the arm is shown in Figures 13(a) and 13(b), respectively. Lifting up and laying down the arm incur different direction changes of linear-acc in x-axis. Thus, we can utilize the difference of direction changes in linear-acc to distinguish the two activities.

However, considering that the phone can be held in different gestures, the direction changes of linear-acc may be different. When we hold the phone like Figure 4(c), the linear-acc in x-axis of lifting up/laying down the arm is shown in Figures 13(c) and 13(d). For the same activity (e.g., lifting up the arm), if we hold the phone in different gestures, the direction changes of linear-acc are different, as shown in Figures 13(a) and 13(c). Thus, we cannot map the direction changes (from positive to negative/from negative to positive) of linear-acc to activities (lifting up/laying down the arm) directly. Therefore, we introduce the data of gravity sensor to assist for activity recognition in arm level. In Figure 13(a), the user lifts up the arm, the gravity data in x-axis is positive, while the linear-acc in x-axis changes from positive to negative. It indicates that the signs (i.e., positive or negative) of the two sensor's data change from same to different, when the user lifts up the arm. Based on Figure 13(b), when the user lays down the arm, the signs of the two sensor's data change from different to same. When the user holds the phone like Figure 4(c), the signs of the two sensor's data still have the above change rule, that is, changing from same to different for lifting up the arm, changing from different to same for laying down the arm.

In fact, in addition to the above two gestures shown in Figures 4(b) and 4(c), the phone can also be held in other attitudes. When we lift up the arm and rotate the phone at the same time, the

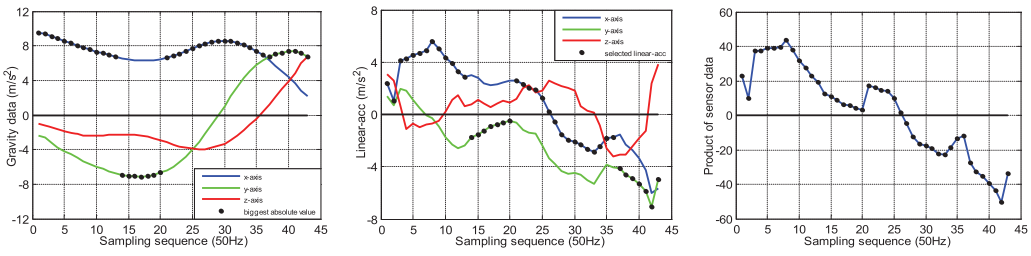


(a) Lift up the arm (horizontally, normal) (b) Lay down the arm (horizontally, normal)



(c) Lift up the arm (horizontally, backward) (d) Lay down the arm (horizontally, backward)

Fig. 13. Sensor data in arm level when the phone is held horizontally in normal/backward direction.



(a) Gravity data (b) Linear-acc (c) Product of selected data

Fig. 14. Lift up the phone while rotating the phone.

corresponding gravity data and linear acceleration are shown in Figures 14(a) and 14(b). At this time, we cannot immediately figure out the relationship between the two sensor's data, because the phone rotation will incur the change of sensor data in all axes. Therefore, we describe the biggest direction change of the activity in one of the three axes. We select the direction (i.e., x-axis, y-axis, or z-axis) of gravity data that has the biggest absolute value, as the dark dots shown in Figure 14(a). Then, we select the linear-acc in the corresponding axis selected by the gravity sensor, as shown in Figure 14(b). We represent the selected gravity data as $d_{vi}, i \in [1, N]$, while the corresponding linear-acc is $d_{ai}, i \in [1, N]$. Then, we can use Equation (6) to get the product of the selected sensor data, as shown in Figure 14(c):

$$P_i = d_{vi} \cdot d_{ai}, \quad i \in [1, N]. \tag{6}$$

In Figure 14(c), the signs of the product have the same change rule with that in Figures 13(a) and 13(c). Therefore, no matter how the phone is held by the hand, when the user lifts up the arm, the signs sign(P_i) of the product $P_i, i \in [1, N]$ change from positive to negative. In regard to laying down the arm, the signs sign(P_i) of $P_i, i \in [1, N]$ change from negative to positive. Here, the sign function is shown in Equation (7):

$$\text{sign}(P_i) = \begin{cases} 1, & P_i > 0 \\ 0, & P_i = 0 \\ -1, & P_i < 0 \end{cases}. \tag{7}$$

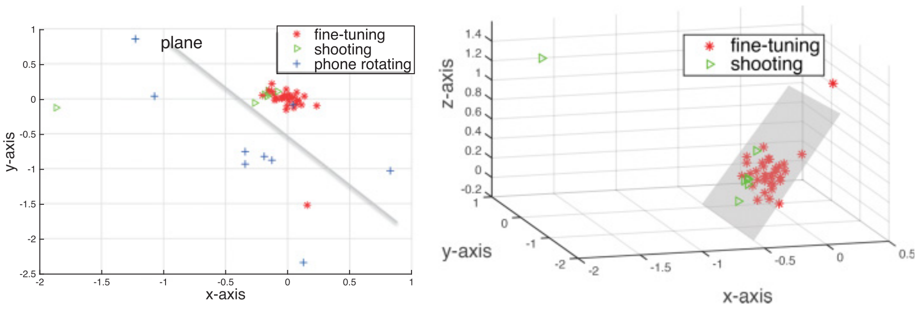
Due to the effect like unconscious shaking of the hand, the signs sign(P_i) of the product $P_i, i \in [1, N]$ may not change smoothly. Therefore, we introduce a *voting mechanism* to mitigate the effect of noise. For signs sign(P_i), $i \in [1, N]$, we obtain the sign $Sign1$ of the first half signs (i.e., sign(P_i), $i \in [1, \frac{N}{2}]$) by voting, if more than $\frac{N}{4}$ sign(P_i) are 1, then $Sign1 = 1$. Otherwise $Sign1 = -1$. In a similar way, we can get $Sign2$ of the other half signs (i.e., sign(P_i), $i \in (\frac{N}{2}, N]$). If $Sign1$ is positive and $Sign2$ is negative, then the activity is lifting up the arm. Otherwise, the activity is laying down the arm, as shown in Algorithm 2.

ALGORITHM 2: Recognition in Arm Level

```
Input: Gravity data {$x_{gk}, y_{gk}, z_{gk}, k \in [1, N]$} and linear-acc {$x_{ak}, y_{ak}, z_{ak}, k \in [1, N]$} in each axis of the
       activity segment $A_i$.
Output: Recognized activity in arm level.
Calculate the absolute value $|x_{gk}|$, $|y_{gk}|$, $|z_{gk}|$ of gravity data in each axis.
Select the gravity data $G_k$ in the axis $M_j$ (i.e., x-axis, y-axis or z-axis), which has the biggest value among
$|x_{gk}|$, $|y_{gk}|$ and $|z_{gk}|$.
Select the linear-acc $LA_k$ in the corresponding axis $M_j$.
Get the product $P_k$ of $G_k$ and $LA_k$, and store the signs of $P_k$ in set $\{sign(P_k), k \in [1, N]\}$.
Get $Sign1$ by voting in $\{sign(P_k), k \in [1, \frac{N}{2}]\}$, and get $Sign2$ by voting in $\{sign(P_k), k \in (\frac{N}{2}, N]\}$.
if $Sign1$ is positive and $Sign2$ is negative then
    Return "Lifting up the arm".
else
    Return "Laying down the arm".
```

Wrist Level: Wrist level contains three activities, which are rotating the phone, fine-tuning and shooting. Considering that the user may rotate the phone (see Figure 9(c)) to make a slight adjustment, we introduce the gyroscope data to assist for activity recognition. However, the micro activities may not incur obvious change of sensor data. Therefore, we use a classifier called Support Vector Machine (SVM) (Qian et al. 2010) to classify the three activities. For the sensor data of linear accelerometer and gyroscope, we extract the variance, mean, maximum, minimum value of the data in each axis (x-axis, y-axis, z-axis) as features, that is, 24 features for classification. Then, we use a linear kernel to train a SVM model, aiming to construct a hyperplane in high-dimensional space, to classify the training data into different classes. When we obtain the sensor data of an activity, we use the SVM classifier to get the class that the activity belongs to, that is, recognizing the activity.

To describe the principle of the SVM classifier, we show a simplified illustration with Figures 15(a) and 15(b). Figure 15(a) shows the linear-acc of the three activities in X-Y plane. Then, we can use a hyperplane to classify the activities into two classes: rotating the phone, the other activities (i.e., fine-tuning and shooting). After that, by considering more features, for example, the data in X-Y-Z space, we get another hyperplane to separate fine-tuning and shooting, as shown in Figure 15(b). Therefore, using more features and applying the binary classification process multiple times, we can recognize the activities in wrist level.



(a) Three activities shown in X-Y plane (b) Fine-tuning and shooting shown in X-Y-Z space

Fig. 15. Linear-acc of three activities in wrist level.

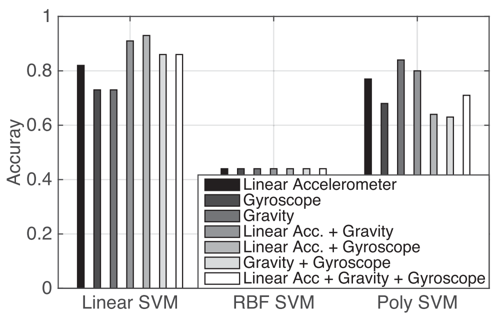


Fig. 16. Accuracies of different SVM models.

In this level, we select the linear-acc and gyroscope data for classification based on extensive experiments. As shown in Figure 16, we have tried to train the SVM model with seven combinations of sensors and three different kernels. When we combine the linear-acc with gyroscope data and use the linear kernel to construct the SVM classifier, we can get the highest accuracy for activity recognition in wrist level.

4.2 Energy-Saving Scheme

When we obtain the user's state based on activity recognition, we will use the following energy-saving scheme to extend the battery life, while guaranteeing a good user experience.

4.2.1 Energy Saving Points. Based on the analysis in Section 2.2, we know that using the camera module and keeping the screen on lead to large energy consumption. *For the camera*, the adjustable parameters are window size of camera (i.e., surface view size), picture size, preview size, and preview frame rate (Google Inc. 2016b). Among the parameters, changing the surface view size will affect the user experience, because users are used to using the camera in full screen. Small surface view will degrade user experience. In regard to picture size, it means the size of the photo stored in the phone. Therefore, we discard the adjustment of surface view size and picture size during photographing. Instead, we observe how the preview size and the preview frame rate affect energy consumption with Samsung S5, as shown in Figures 17(a) and 17(b). Because the screen size of Samsung S5 is $1,920 * 1,080$ pixels, we set the surface view size as $1,920 * 1,080$ pixels, to make the camera work in full screen for a good user experience. Unless otherwise specified, the preview

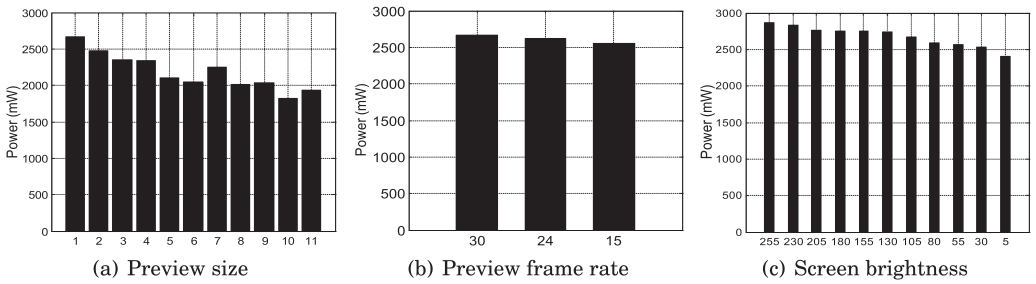


Fig. 17. Energy consumption in Samsung S5 vs. preview size, preview frame rate, screen brightness. (a) 1–11, respectively, represent the following preview size (in pixels): 1,920*1,080; 1,440*1,080; 1,280*720; 1,056*864; 960*720; 800*480; 720*480; 640*480; 352*288; 320*240; 176*144.

size is $1,920 * 1,080$ pixels, the preview frame rate is 30Hz, the screen brightness is 100 and the phone works in auto focus mode.

According to Figure 17(a), small preview size usually leads to small energy consumption. However, when the aspect ratio of preview size changes, small preview size may not save energy. In Figure 17(a), the 6th preview size is 800*480 pixels (i.e., ratio is 5:3), while the 7th preview size is 720*480 pixels (i.e., ratio is 3:2). Although $720 * 480 < 800 * 480$, the 7th preview size has larger power consumption than the 6th one. Nevertheless, if the aspect ratio of preview size keeps unchanged, small preview size leads to small power consumption, as the power under 2nd, 5th, 8th, and 10th preview size in Figure 17(a). Besides, to ensure that there is no deformation of the photo in surface view, the preview size should be supported by the phone. We are expected to decrease the preview size for energy consumption, while not distorting the photo in camera view window. In regard to the effect of preview frame rate, Figure 17(b) indicates that using a smaller preview frame rate is a possible way to reduce energy cost. However, due to the limitation of hardware configurations in Samsung S5, we cannot arbitrarily assign a frame rate for the camera. Even though we assign a preview frame rate, the camera may not work in the fixed frame rate. This leads to the limited energy reduction by decreasing the preview frame rate.

For the screen, Figure 17(c) shows the energy consumption, when we vary the screen brightness. The brightness range is [0, 255], where 0 is corresponding to the darkest screen and 255 represents the brightest screen. Therefore, by reasonably decreasing the screen brightness, we can reduce the energy consumption during photographing.

4.2.2 *Energy Saving Scheme.*

By combining the observations in Section 2.2 and the analysis in Section 4.2.1, for each recognized activity/state, we apply a corresponding energy-saving strategy, as shown in Figure 18.

If the recognized activity is in *body level*, then the screen and the camera will be turned off, because the user doesn't need to look at the screen for photographing. Furthermore, if the user stays in body level for a long time (e.g., 5min), the sensors will be turned off and our system SenSave will stop running until the user needs to take photos and starts SenSave again.

If the recognized activity is in *arm level*, then we will keep the screen on and adjust the brightness of the screen based on the ambient light. As shown in Table 2, we classify the brightness [0, 255] into five categories, according to different environments. When the user lifts his/her arm, he/she may want to take photos. Thus, we will adjust the screen brightness based on Table 2, and start the camera view with the lowest available configurations. On the contrary, if the user lays down the arm, it usually indicates that he/she finishes taking photos and it is unnecessary to look at the

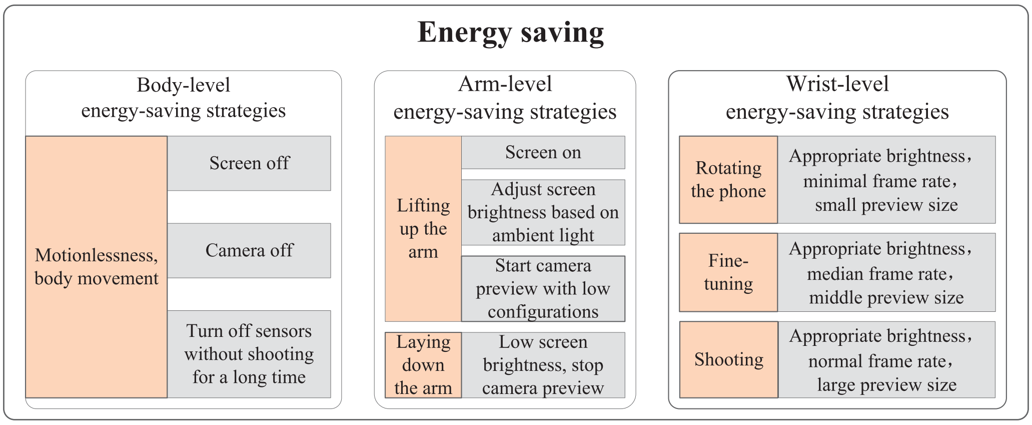

Energy saving
Body-level energy-saving strategies
Motionlessness, body movement
Screen off
Camera off
Turn off sensors without shooting for a long time
Arm-level energy-saving strategies
Lifting up the arm
Screen on
Adjust screen brightness based on ambient light
Start camera preview with low configurations
Laying down the arm
Low screen brightness, stop camera preview
Wrist-level energy-saving strategies
Rotating the phone
Appropriate brightness, minimal frame rate, small preview size
Fine-tuning
Appropriate brightness, median frame rate, middle preview size
Shooting
Appropriate brightness, normal frame rate, large preview size

Fig. 18. Energy saving strategies for different activities/states.

Table 2. The Screen Brightness in Different Environments

Number	Environment	Ambient Light (SI lux)	Brightness of Screen
1	Daytime, outdoor, sunny	>6,000	180
2	Daytime, outdoor, cloudy	500~6,000	130
3	Daytime, indoor, no lamp	100~300	80
4	Night, outdoor, street lamp	<100	55
5	Night, indoor, lamp	300~500	105

screen. Then, we will decrease the screen brightness to the lowest possible brightness of the smart phone and stop the camera view.

If the recognized activity is in *wrist level*, then the screen stays on and the camera stays in preview mode, because the user is probably going to take photos. When the user rotates the phone, the camera is set to work with the smallest frame rate supported by the phone (e.g., 15fps in Samsung S5) and a small preview size. Take Samsung S5 as an example, to see the photo clearly in camera view window, the small preview size is set to $640 * 480$ pixels, which is supported by the phone, as shown in Figure 17(a). For other phones, we set small preview size in the same way and ensure that the preview size is no less than $640 * 480$ pixels. For the fine-tuning activity, the camera works with the median frame rate and middle preview size, for example, 24fps and $960 * 540$ pixels in Samsung S5. For the shooting activity, the camera works with the default/maximum frame rate and large preview size, for example, 30fps and $1920 * 1080$ pixels in Samsung S5, to capture the target timely. It is worth mentioning that the size $960 * 540$ pixels does not occur in Figure 17(a). However, because the middle preview size has the same aspect ratio with the surface view, the camera can work with it well. In addition, we also allow the user to adjust the screen brightness, select the preview frame rate, choose the preview size according to his/her preference, to provide a good user experience.

5 ACTIVITY RECOGNITION WITH PREDICTION BASED ON AN EXTENDED MARKOV CHAIN

In the above sections, we recognize an activity after we obtain the whole or partial data of the activity, then we adopt a corresponding energy-saving strategy. In fact, when we have recognized

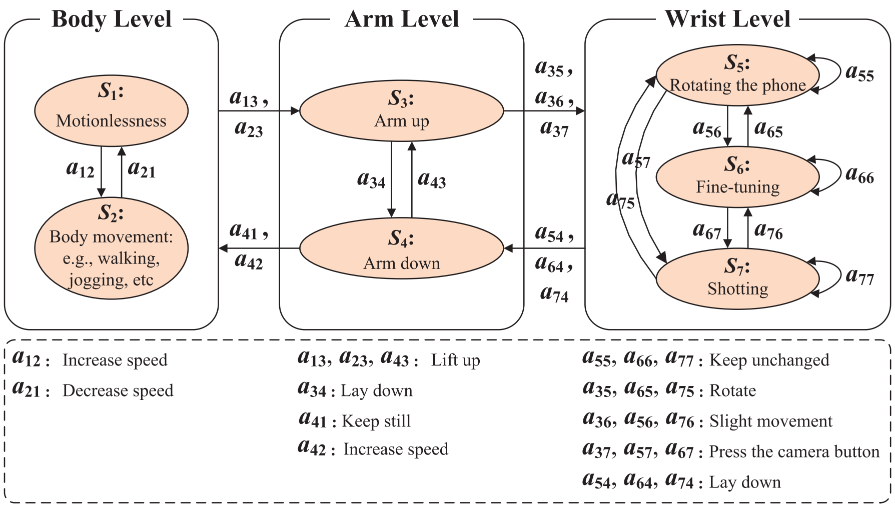


Fig. 19. The extended Markov chain.

the current state of the user, we may have missed the best chance to adopt the most efficient energy-saving strategy.

For example, the user has finished taking a photo and he/she is laying down the arm. During this process, we will wait for the sensor data of laying down the arm for activity recognition, and we still use the energy-saving strategy in wrist level, whose energy-saving strategy is less efficient than that of other levels. If we can predict the next activity state in advance, then we can adopt a more efficient energy-saving strategy. Therefore, we introduce an *extended Markov chain* for activity state prediction. Then, we enhance the previous scheme "SenSave" as "SenSave-MC."

In the extended Markov chain, there are seven activity states for taking photos, as mentioned in Figure 8. Besides, we also introduce the actions that lead to the transition of states in the extended Markov chain. As shown in Figure 19, the states in the extended Markov chain represent the user's activities during photographing, while the actions a_{ij} represent the events causing state transitions. As described in Section 4.1.4, the user cannot transfer from one state to another state arbitrarily during photographing. The transfer relationship of states and the actions that cause state transition are shown in Figure 19. The action a_{ij} causes the transition from state i to state j. We use $\{S_1, S_2, \ldots, S_7\}$ to represent the seven states. We use p_{ij} to describe the transition probability from state i to state j, $i, j \in [1, 7]$ and $p_{ij} \in [0, 1]$. To get the value of p_{ij}, we need to use the history data to calculate transition probability. Suppose that there are m_i history samples for state S_i, then the total number of history samples is $N_s = \sum_{i=1}^{7} m_i$. The state of the uth history sample is represented as s_u, $u \in [1, N_s]$. We can calculate the transition probability p_{ij} by Equation (8):

$$p_{ij} = \frac{\sum_{u=1}^{N_s-1} T_{ij}(u, u+1)}{m_i}, \tag{8}$$

where $T_{ij}(u, u+1)$ is the indicator function, which is shown in Equation (9):

$$T_{ij}(u, u+1) = \begin{cases} 1, & \text{if } s_u = S_i \wedge s_{u+1} = S_j \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

In Table 3, we show an example of the state transition matrix of a user. The transition matrix is calculated by more than 500 state samples collected in our campus, where the user takes photos for

Table 3. State Transition Probability during Photographing

$p_{ij}(\%)$	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	—	9.96	90.04	0	0	0	0
S_2	95.86	—	4.14	0	0	0	0
S_3	0	0	—	0.33	22.81	73.75	3.11
S_4	1.30	94.67	4.03	—	0	0	0
S_5	0	0	0	1.74	3.25	94.12	0.89
S_6	0	0	0	0.03	7.71	0.11	92.15
S_7	0	0	0	71.19	7.28	18.67	2.86

about 2h. When SenSave detects a potential action that indicates a potential state transition, it will use Table 3 to predict the next state. In regard to the action/event detection, we also use the sliding *pause window* (see Section 4.1.3) to detect the variance/peak value/valley value of linear-acc and gyroscope data. If the sensor data in the window has the similar feature with that of the current state, then SenSave will stay in the current state. For example, suppose the maximum resultant linear-acc in the current state is a_m, while that in the pause window is a'_m. If $|\frac{a'_m - a_m}{a_m}| \leq 20\%$, then it indicates the resultant linear-acc of them has the similar feature. Based on Section 4.1.3, the size of the sliding pause window is 0.5s, thus the window usually can catch a peak/valley value or both of them of an activity segment. If two consecutive sliding windows have similar features in all the following aspects, that is, peak/valley value, then variances of the sensor data, it indicates the user keeps in the same activity state. Otherwise, if we detect a pause, it indicates the user will prepare to transfer to another state. During the pause, "SenSave-MC" still use the previous energy-saving strategy. If "SenSave-MC" detects the variation of sensor data/activity features, then it indicates the user is probably transferring to a new state. At this time, we predict the next state based on Table 3.

Suppose that the current state is S_3 (i.e., arm up) and we detect an action. By observing the third row of Table 3, S_3 has the highest probability of transferring to S_6. Then, we can predict that the next activity is probably S_6, that is, fine-tuning the phone. At this time, we can adopt the suitable energy-saving strategy immediately. For example, we can use the energy-saving strategy in wrist level in advance to provide a better user experience. Without the extended Markov chain, we will need to wait for the sensor data of next activity for recognition. In that way, we may not select the suitable preview size, frame rate and screen brightness timely, when the user slightly adjusts the phone for taking photos, leading to a worse user experience.

In Figure 20, we show the process of using extended Markov chain for state prediction and utilizing the state recognition result to update the state transition matrix. We use D_u to represent the sensor data of the uth action and s_u to represent the final recognized state corresponding to the uth action, $s_u \in \{S_i\}, i \in [1, 7]$. Suppose that we have got u states, we will utilize the transition matrix (see Table 3) and the last state s_u to predict the next state as s_{u+1}. That is to say, if s_u represents state S_i, then we find the state S_j that has the largest probability p_{ij} in the ith row of Table 3, and then predict the next state as $s_{u+1} = S_j$. After that, we can immediately adopt a corresponding energy-saving strategy for state S_j. However, in fact, a state i can transfer to another state k. When we finally recognize the activity containing the action, we may find the next state is different from the predict one. For example, the current user state is S_1. When SenSave-MC detects a potential action in a short time, the predicted state is S_3. In fact, the user increases the speed and enters into the body movement state. When SenSave-MC gets enough sensor data containing the action (e.g., 2s), it will recognize the corresponding activity as body movement. At

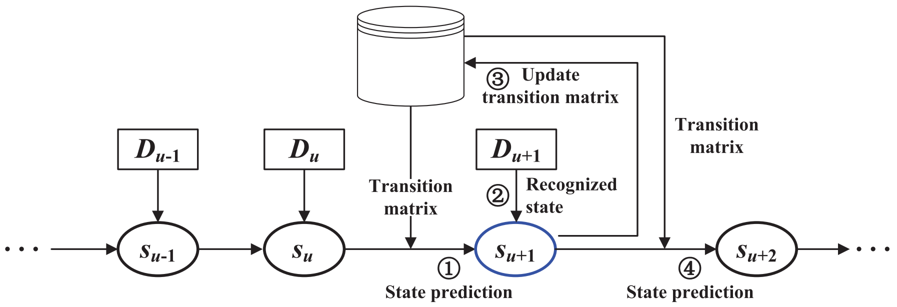
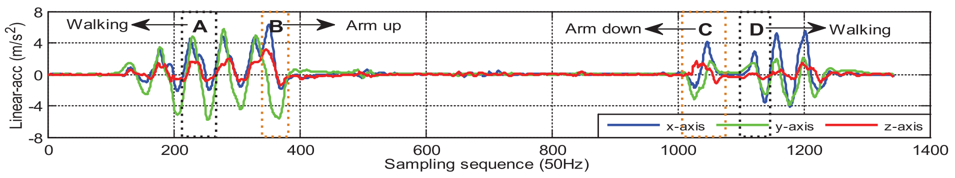
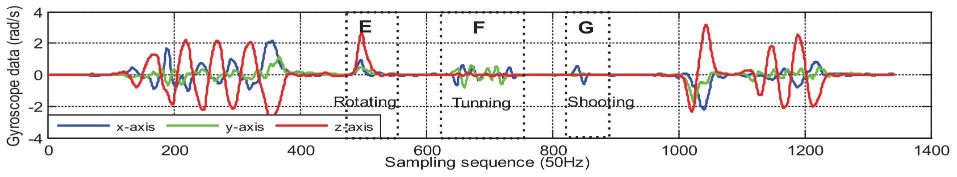


Fig. 20. Working principle of SenSave-MC.



(a) Linear-acc



(b) Gyroscope data

Fig. 21. Taking photos while walking.

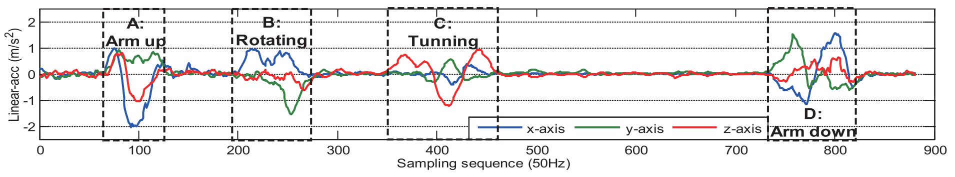
this time, SenSave-MC will calibrate the prediction state as S_2. That is to say, if the predicted state s_{u+1} is different from the actual state S_j, we need to calibrate the state s_{u+1} and the corresponding energy-saving strategy according to S_j. Besides, from s_u to s_{u+1}, we get a new transition sample of states, that is, $s_u \rightarrow s_{u+1}$, then we use this new sample to update the transition matrix based on Equations (8) and (9). Similarly, we will use the recognized state s_{u+1} and the updated transition matrix to predict the next state and repeat the above process.

6 DISCUSSION ABOUT ACTIVITY SENSING AND ENERGY SAVING IN COMPLEX ENVIRONMENTS

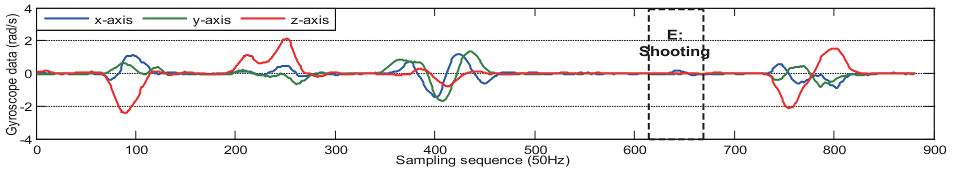
In Section 4, we utilize the pause between activities for segmentation. However, in real life, we can also take photos in a moving state. During photographing, the disturbances can also occur. In this section, we will show how to make SenSave work in complex environments.

6.1 Activity Sensing in Moving States

As mentioned above, the user can take photos in a moving state. The typical examples could be taking photos while walking, and taking photos in a bus. In Figure 21, we show the sensor data of taking photos while walking. By comparing Figures 6 and 21(a), taking photos while walking affects activity recognition in arm level. This is because the arm-up activity occurs with walking. We may wrongly segment the arm-up activity into body movement, thus recognizing the arm-up activity as body movement. In regard to the following activities, most of the users (i.e., 17 of 20 in



(a) Linear-acc



(b) Gyroscope data

Fig. 22. Taking photos in a bus.

our experiment) will stop and take photos. That is to say, taking photos while walking may lead to the confusion of body movement and arm movement, we can still recognize activities in wrist level based on Section 4.1.5. As shown in Figure 21(b), we recognize activities in wrist level by combining linear-acc and gyroscope data. In regard to the activity "Shooting," it also can be recognized by the screen touch operation. Besides, according to Figure 18, even if we cannot recognize activities in arm level accurately, we can adjust the screen brightness, preview size and preview frame rate in wrist level immediately, to provide a good user experience.

In Figure 22, we show the sensor data of taking photos in the bus. Due to the movement of the bus, the sensor data contains more noises. By smoothing the sensor data with a moving average, we obtain the linear-acc in Figure 22(a) and the gyroscope data in Figure 22(b). Considering the noises, we increase the parameter value in Equations (3) and (4), then we can still segment the activities for recognition. This is because the activity data is not buried by noises. However, if the bus does not move steadily, for example,taking a sudden turn, slamming the brakes on, it becomes very hard to detect the activities of photographing. For the segmented activity, we can recognize it based on Section 4.1 and adopt the energy-saving strategy in Figure 18. Therefore, our SenSave can work well in a bus moving steadily.

6.2 Changing the Positions of Phones and Targets

According to Figure 10, we use the variance and periodicity of sensor data to classify an activity into a level, and then recognize it in that level. The activity recognition is mainly related to the features of sensor data instead of the transfer relationship between activities. Therefore, changing the positions of phones or targets will not affect activity recognition.

In regard to the energy-saving strategy, it is related to the user's activity. Usually, before the user takes a photo, he/she lifts up the arm. However, if the phone is put in the coat pocket or the user aims to take a photo below the chest, the user may lay down the arm before shooting a photo. According to Figure 18, when the user lays down the arm, SenSave will decrease screen brightness and stop the camera view. This energy-saving strategy may degrade the user experience during the following activities in wrist level. To solve this problem, we also introduce the adjustment of screen brightness and camera parameters in wrist level, as shown in Figure 18. That is to say, even if SenSave does not choose an appropriate energy-saving strategy in arm level, we can correct the error immediately in wrist level.

6.3 Avoiding Disturbances from Other APPs

Figure 18 shows the energy-saving strategies in SenSave. In our article, we adjust the preview size, frame rate, screen brightness of SenSave, instead of the system. SenSave does not change the system setting, while the disturbances from other APPs will not change the settings of SenSave.

When we turn off SenSave, the camera and the system of the phone will recover to the default settings. If we want to do activity sensing and reduce energy consumption during photographing, then we first need to turn on SenSave. Without turning on SenSave, we will not do activity sensing and adopt energy-saving strategies. If SenSave has been turned on, then SenSave will detect whether the user has transferred to another APP, that is, what the current active window is, from time to time. If the user uses another APP with the photographing function (e.g., video call), then SenSave will be turned off until the user turns on SenSave again. In this way, SenSave and other APPs have no effect on each other, that is, we do activity recognition and energy saving in SenSave instead of other APPs.

7 PERFORMANCE EVALUATION

To verify the efficiency of our scheme "SenSave" and the enhanced scheme "SenSave-MC," we implement the system prototype on Samsung Galaxy S5 smart phone running on Google's Android platform. The version of the Android system is 5.0. Unless otherwise specified, the phone is in airplane mode, the WiFi is turned off, the phone stays in the idle state, that is, no APP runs except for system program. We repeat each experiment 50 times and average the experiment results. In the experiments, we first observe what factors affect the accuracy of activity recognition and select suitable parameters of sensors for activity recognition. Then, we invite multiple users and use different smart phones to measure the activity recognition accuracy. After that, we use Monsoon power monitor (Monsoon Solutions Inc. 2015) to measure the power consumption of the phone, and compare our solutions with the existing methods in terms of energy consumption during photographing.

7.1 Selection of Suitable Parameters for Sensors

7.1.1 Activity Recognition Accuracies Under Different Sampling Rates. In this experiment, we select six different sampling rates, that is, 2, 5, 10, 20, 50, and 100Hz, to observe how the sampling rates of sensors affect the recognition accuracy with Samsung S5 smartphone. Figures 23(a), 23(b), and 23(c), respectively, show the activity recognition accuracy in body level, arm level, wrist level. Usually, using larger sampling rates will increase the recognition accuracy. When the sampling rate is larger than 20Hz, the recognition accuracy is close to or larger than 90%. To guarantee a good activity recognition accuracy, the sampling rate of the sensor should be equal to 20Hz at least.

7.1.2 Energy Consumption Under Different Sampling Rates. To observe how the sampling rate affects the energy consumption, we utilize Samsung S5 to measure the power consumption of linear acclerometer and gyroscope from 5 to 100Hz. As mentioned in Section 2.2.3, linear-acc and gravity data are generated from accelerometer. When we obtain linear-acc, it means we also obtain gravity data. Therefore, we only observe the power consumption of linear accelerometer and gyroscope. We measure the power consumption of a sensor in the same way described in Section 2.2.3. As shown in Figure 23(d), as the sampling rate increases, both the power consumption of linear accelerometer and gyroscope increases. The power at 100Hz is much larger than that at 20Hz. If we aim to save energy, then we should select the sampling rate as small as possible.

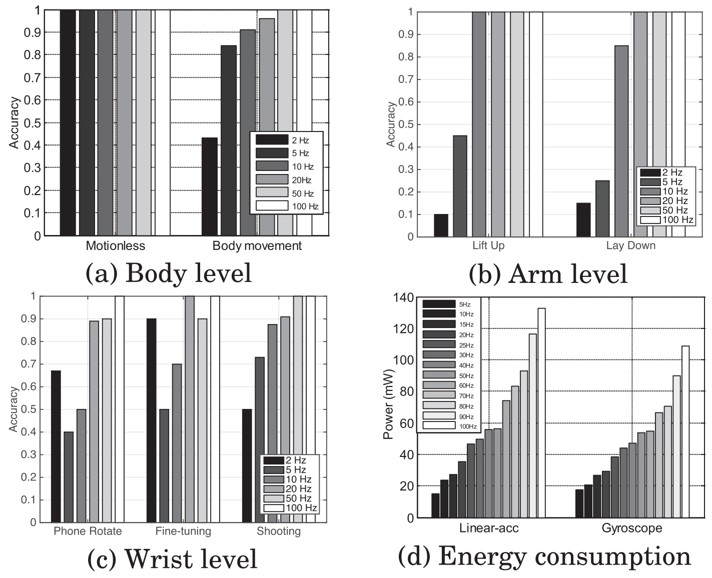
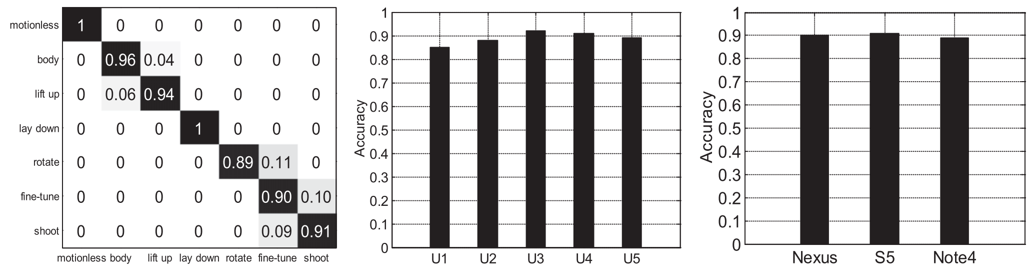


Fig. 23. Parameter selection: (a) Sampling rate vs. recognition accuracy in body level. (b) Sampling rate vs. recognition accuracy in arm level. (c) Sampling rate vs. recognition accuracy in wrist level. (d) Sampling rate vs. energy consumption.

7.1.3 Energy Consumption of Processing Sensor Data. Energy consumption of processing the sensor data is mainly related to the calculated data size. In a certain time period, as the sampling rate of sensors increases, the energy consumption also increases. Take Samsung S5 as an example, when the sampling rate is 20Hz, the power consumption of processing the sensor data for body movement (body level), activities in arm level, activities in wrist level are 1430, 1140, and 1,295mW, respectively. For body movement recognition, due to the calculation of DTW distances, the power consumption is a little high. However, the time for calculating DTW distances is usually less than 20ms, while the processing time for arm-level activities and wrist-level activities is smaller. The energy consumption in DTW distance calculation is about 2.1uAh, which is much smaller than the energy consumption (about 100uAh) of shooting a photo (i.e., the operation of pressing the shutter), turning ON/OFF the camera (about 837uAh), and so on. Besides, we do not perform activity recognition all the time. Instead, we only do activity recognition when we detect a possible activity, thus the energy consumption in data processing is insignificant. Consequently, processing the sensor data with a reasonable sampling rate (e.g., 20Hz) does not incur much extra energy consumption. Reducing the energy consumption during photographing mainly relies on adjusting the settings of camera and screen.

7.1.4 Trade off Between Recognition Accuracy and Energy Consumption. According to Figures 23(a)–23(c), the high sampling rate means the large recognition accuracy. However, based on Figure 23(d), large sampling rate leads to high energy consumption. Therefore, we need to make a trade off between recognition accuracy and energy consumption. Based on Section 7.1.1, to guarantee a good activity recognition accuracy (e.g., no less than 90%), the sampling rate of the sensor should be equal to 20Hz at least. Besides, when the sensors work with 20Hz, they will not incur much extra energy consumption of sampling and data processing, according to Section 7.1.2 and Section 7.1.3. However, if we continue increase the sampling rate, it will incur more energy



(a) Confusion matrix for seven activities (b) Accuracies of different users (c) Accuracies of different phones

Fig. 24. Activity recognition accuracy.

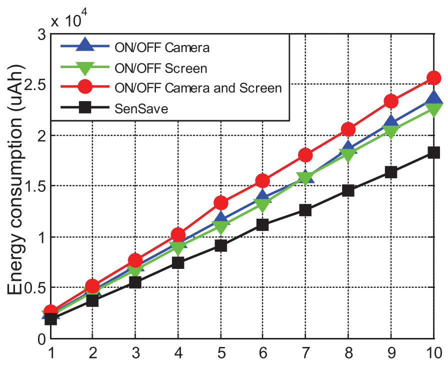
consumption, as shown in Figure 23(d). Therefore, we set the sampling rate of sensors to 20Hz in this article, while considering the recognition accuracy and energy efficiency.

7.2 Recognition Accuracy

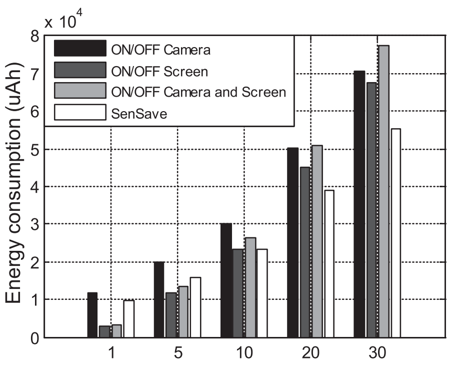
7.2.1 Recognition Accuracy of Each Activity. In Figure 24(a), we show the confusion matrix for the seven activities in terms of recognition accuracies. In the experiment, for the user who participates in the experiment, there is an observer who records the process with a video. By reviewing the video, we can obtain the ground truth of activity recognition. In Figure 24(a), each row represents the actual activity performed by the user, while the corresponding column of the row represents the recognized activity by our scheme "SenSave." For example, the third row represents the user's activity "lift up the arm," while the second column in the third row means that "lift up" is recognized as "body movement." That is to say, row i represents the actual activity i, while column j represents the recognized activity j. Therefore, each element in row i and column j of the matrix denotes the probability that activity i is recognized as activity j. According to Figure 24(a), the activity usually can be recognized with a high accuracy, that is, larger than 90%. Overall, the average recognition accuracy of the activities can achieve 95.5%. It indicates that we have selected suitable parameters for our scheme SenSave and it can recognize the activities accurately.

7.2.2 Recognition Accuracies for Different People. Considering that different people have different behavior habits, we invite five users to evaluate the feasibility of our scheme SenSave. In the experiment, all the users use the same phone, that is, Samsung Galaxy S5, to take 10 photos in 5min. During the process, each user takes photos according to his/her habit and he/she may perform any activity in three levels. The average recognition accuracy in the whole process of each user is shown in Figure 24(b). It indicates that although the user behavior has some effect on the recognition accuracy, SenSave still has a good recognition accuracy, which is usually larger than 85% and can achieve 90%. Therefore, SenSave has a good feasibility and it can work well for different users.

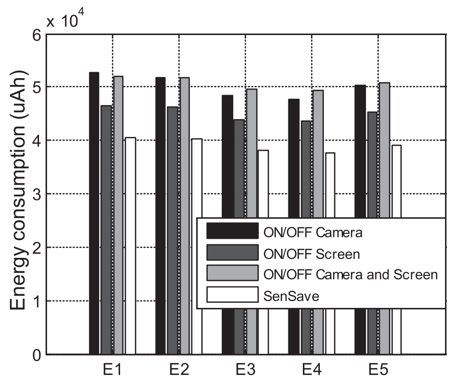
7.2.3 Recognition Accuracies with Different Phones. To verify that SenSave can work on different phones, we use three phones, that is, Samsung Galaxy Nexus, Samsung Galaxy S5, Samsung Galaxy Note4, to test the activity recognition accuracy of SenSave. As shown in Figure 24(c), whatever the phone is, the recognition accuracy can achieve 89% or more. It means that SenSave has a good activity recognition accuracy, while not relying on the specific phone. It can work well on common phones, which are usually equipped with the sensors like accelerometer, gyroscope, and so on.



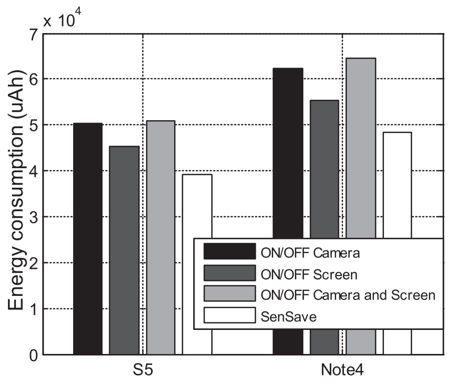
(a) Energy consumption vs. different number of consecutive shots



(b) Different number of shots in 5 minutes



(c) 20 shots in 5 minutes in 5 different environment



(d) 20 shots in 5 minutes with two phones

Fig. 25. Energy consumption vs. number of shots, environments, phone types.

7.3 Energy Consumption

To evaluate the energy efficiency of SenSave, we use the Monsoon power monitor (Monsoon Solutions Inc. 2015) to measure the energy consumption of the phone and compare SenSave with three common schemes, that is,"Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen," as mentioned in Section 2.2.2. Unless otherwise specified, we repeat each experiment 50 times and average the experiment results.

7.3.1 Energy Consumption Comparison with Other Schemes. In this experiment, the user continuously takes ten photos. We compare SenSave with three common schemes, that is, "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen." In the experiment, the user keeps in the motionless state. Each time, he/she lifts up his/her arm to take a photo, then he/she lays down the arm. After that, the user lifts up the arm again to repeat the photographing process. For each scheme, the user takes ten photos. Figure 25(a) shows the energy consumption of each scheme in Samsung Galaxy S5. As the number of shots (photos) increases, the common schemes "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen" are energy-consuming, due to the frequent user interaction with the phone. Our "SenSave" outperforms the three common schemes, because it can automatically recognize the user's activity and adopt a suitable energy-saving strategy without user interaction. When the user has taken ten photos, "SenSave" can, respectively, reduce the energy consumption by 22.6%, 19.1%, and 28.5%, when compared to "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen."

Considering the actual situation in photographing, the user usually does not take photos in the above way. Therefore, in the following experiment, the user randomly takes 1, 5, 10, 20, and 30 photos in 5min. During the process, an observer records the user's behavior, which will be repeated in each scheme, that is, "Turn ON/OFF Camera," "Turn ON/OFF Screen," "Turn ON/OFF Camera and Screen," and "SenSave." The user performs the above experiment in the same environment. Figure 25(b) shows the energy consumption of the phone under each scheme. When the number of shots in 5min is very small, for example, 1 photo, "Turn ON/OFF Screen" and "Turn ON/OFF Camera and Screen" have good performances. Because they only take one photo and then turn off the screen for energy consumption. Our SenSave needs to detect the user's activities with linear accelerometer and gyroscope, thus SenSave consumes more energy. However, as the number of shots increases, SenSave outperforms the common schemes, because SenSave adopts the energy-saving strategies without user interactions. If the user takes 30 photos in 5min, then SenSave can, respectively, reduce the energy consumption by 21.9%, 18.2%, and 28.8%, when compared to "Turn ON/OFF Camera," "Turn ON/OFF Screen," "Turn ON/OFF Camera and Screen." In addition, if users take photos in a longer time instead of 5min, SenSave can adaptively turn off the camera, screen, or sensors for further energy saving.

7.3.2 Energy Consumption Under Different Environments. To verify that SenSave can adaptively select a suitable strategy under different environments, the user takes 20 photos in 5min, as mentioned in Figure 25(b). The five different environments are shown in Table 2. Accordingly, we call the environments as $E1, E2, E3, E4, E5$, respectively. As shown in Figure 25(c), "SenSave" outperforms the common schemes "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen" in each environment. When the environment changes, SenSave can detect the variation of the ambient light and select a suitable energy-saving strategy for photographing. Under the five environments, when compared to "Turn ON/OFF Camera and Screen," "SenSave" can reduce the energy consumption by 22.2%, 22.1%, 23.2%, 23.3%, 22.9%, respectively.

7.3.3 Energy Consumption with Different Phones. To verify that our scheme SenSave can work well on different phones, the user takes 20 photos in 5min, as mentioned in Figure 25(b). The user uses two different phones, that is, Samsung Galaxy S5 and Samsung Galaxy Note4. As shown in Figure 25(d), whatever the phone is, SenSave outperforms the common schemes. When compared to "Turn ON/OFF Camera and Screen," Samsung S5 and Samsung Note4 running "SenSave" can reduce the energy consumption by 22.9% and 25.0%, respectively.

7.4 Efficiency of the Extend Markov Chain

In Section 5, we introduce the extended Markov chain into SenSave to predict the next activity state in advance, aiming to adopt the energy-saving strategy in time. Then, we enhance "SenSave" as "SenSave-MC." In the following experiments, we invite three users to test the efficiency of "SenSave-MC." Before using the extended Markov chain to predict the next state, we first collect 500 history activity states. Then, the user moves around our campus for about 30min to verify the efficiency of SenSave-MC.

As shown in Figure 26(a), if we do not adopt the extended Markov chain, the activity recognition accuracy of "SenSave" is about 90%. If we only use the extended Markov chain to predict the next state (i.e., "SenSave-Prediction"), while not calibrating the prediction result with the final sensor data of next activity, then the recognition accuracy decreases a little. This is because the user takes photos according to his/her needs, we cannot guarantee that he/she often takes photos

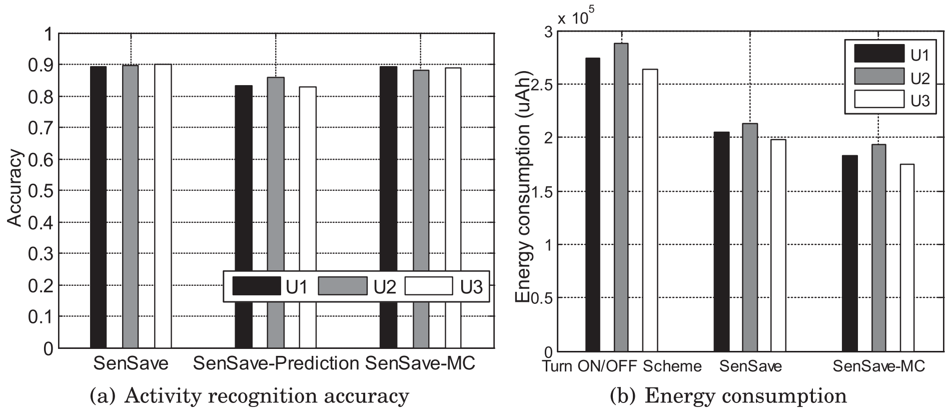


Fig. 26. Activity recognition accuracy and energy consumption of SenSave-MC.

in a fixed pattern, although enough history data can assist for predicting the next activity state. Therefore, we also use the recognized activity to calibrate the prediction result, and then update the transition matrix and the energy-saving strategy, as shown in Figure 20. With the calibration, the activity recognition accuracy of the extended Markov chain "SenSave-MC" increases. The accuracy of "SenSave-MC" is good and it is almost equal to that of "SenSave." Take user 1 as an example, the recognition accuracies of "SenSave," "SenSave-Prediction," and "SenSave-MC" are 89.4%, 83.1%, and 89.3%, respectively. It indicates that SenSave-MC can predict the next state in advance, while still guaranteeing a high recognition accuracy.

In regard to the common energy-saving schemes, the user can "Turn ON/OFF Camera," "Turn ON/OFF Screen," or "Turn ON/OFF Camera and Screen" at random, thus we do not distinguish the three schemes. Instead, we use "Turn ON/OFF Scheme" to represent the combination of "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen." Sometimes, the user may not adopt any energy-saving strategy, according to his/her convenience. This case is also contained in "Turn ON/OFF Scheme." In the experiment, we compare our SenSave and SenSave-MC with the "Turn ON/OFF Scheme," as shown in Figure 26(b). When compared with "Turn ON/OFF Scheme" and the proposed scheme "SenSave," "SenSave-MC" can further reduce the energy consumption. Take user 1 as an example, "SenSave-MC" can respectively reduce the energy consumption by 33.2% and 10.7%, when compared to "Turn ON/OFF Scheme" and "SenSave," respectively. This is because "SenSave-MC" utilizes the extended Markov chain to predict the next activity state in advance and adopts the energy-saving strategy timely.

7.5 User Experience

To verify that SenSave provides a good user experience, we first measure the time latency of activity recognition. Then, we conduct the user study to test the user experience of using SenSave.

7.5.1 Time Latency. In Figure 27, we show the time latency of recognizing an activity in Samsung S5 and Samsung Note4. We repeat each experiment 50 times and average the experiment results. According to Figure 27, the activity recognition time is usually less than 50ms, which is small enough and it is below human response time (Wang et al. 2014; Yin et al. 2016). That is to say, our SenSave can do activity recognition in real time. In addition, with the recognized result, we can start up the camera view within 50ms, that is, we can use camera in real time.

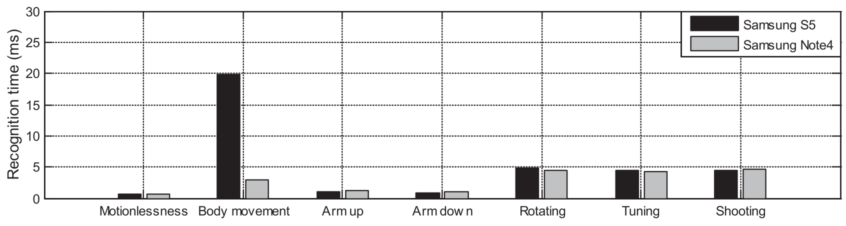


Fig. 27. Recognition time of each activity.

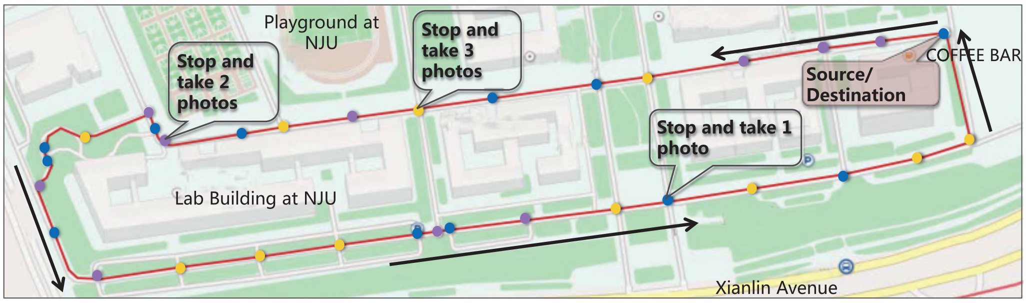


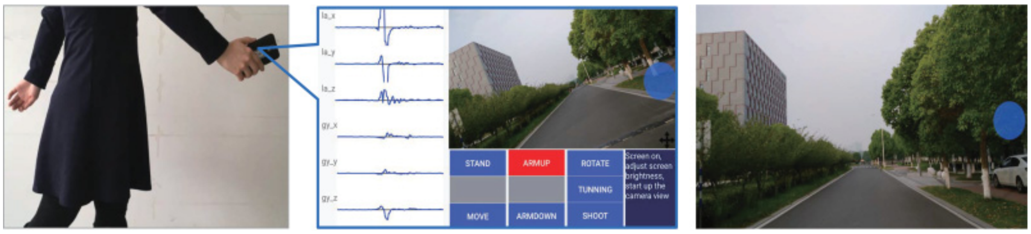
Fig. 28. Experimental scene in the campus.

7.5.2 *User Study.* We invite 20 volunteers to perform a double-blind study to test whether SenSave degrades the user experience. Before the experiments, we adjust the user interface of SenSave and make sure that SenSave and the built-in camera of the phone have the same user interface. Then, we select 10 phones and turn on their own built-in cameras. For the other 10 phones, we turn on SenSave, which has the same user interface with the built-in camera. After that, we invite the 20 volunteers to use the built-in camera or SenSave for photographing. When a user receives the phone, the camera is turned on, thus the user does not know whether the camera belongs to built-in camera or SenSave.

During the experiment, the user takes photos with the camera. To guarantee that the user does not know which app the current camera belongs to, the user will not turn ON/OFF the camera manually. For each user, he/she uses the camera to take photos for 10min freely. Then, we ask him/her to report his/her user experience in the following aspects: the guess of app he/she uses (i.e., built-in camera or SenSave), the score of overall user experience, which is rated on a scale from 1 ("Poorest") to 5 ("Best") with a 3 as neutral. Among the 10 users who use SenSave for photographing, seven of them report that it is like to use the built-in cameras, while eight of them report the good user experiences, that is, the score of the overall user experience is 4 or 5.

8 CASE STUDY

To show the efficiency of the proposed energy-saving schemes "SenSave" and "SenSave-MC," we test our solution in a real world environment. In the experiment, the user moves (e.g., walks, jogs, etc.) around our campus and takes photos along the way, as shown in Figure 28. The user departs from the Coffee Bar (Source) and moves counterclockwise around the campus, then he/she arrives at the Coffee Bar (Destination). Along the way, the user takes photos around the appointed places, as the blue, yellow, and purple points in Figure 28. Here, blue point means that the user takes one photo around the place, yellow point means the user takes three photos around the place, while purple point means the user takes two photos around the place. Along the way, from any place



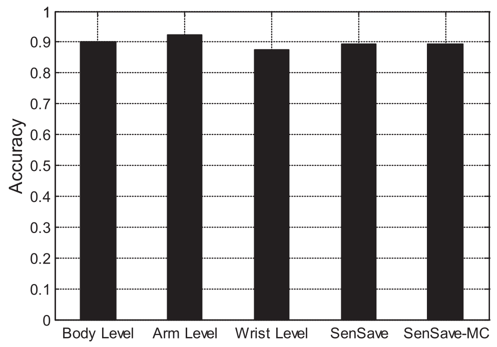
(a) Activity sensing and energy saving with SenSave (b) Full screen

Fig. 29. The APP of SenSave.

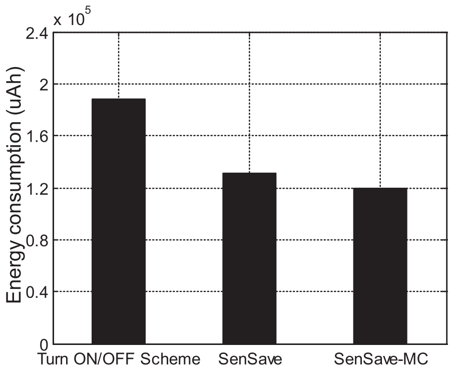
P_A to another place P_B, the user can walk, run or keep motionless. When the user moves from the source (Coffee Bar) to the destination (Coffee Bar), an observer will record the movement behavior of the user. In each experiment, the user keeps the same movement from place P_A to place P_B. In regard to taking photos, there is no strict requirement of the user, who can take photos according to his/her habit. In the experiments, the user respectively adopts "Turn ON/OFF Scheme," "SenSave," and "SenSave-MC" to take photos, while moving from source (Coffee Bar) to destination (Coffee Bar) every time. As mentioned in Section 7.4, "Turn ON/OFF Scheme" means the combination of "Turn ON/OFF Camera," "Turn ON/OFF Screen," and "Turn ON/OFF Camera and Screen." Besides, sometimes the user may not adopt any energy-saving strategy, according to his/her convenience. This case is also contained in "Turn ON/OFF Scheme." We show the activity recognition accuracy and energy consumption of each scheme in the experiments, to verify the efficiency of the proposed schemes.

Before describing the experiment results, we show how SenSave works, that is, doing activity recognition first, and then adopting energy-saving strategies, Figure 29 shows the modules of SenSave on the screen. As shown in Figure 29(a), when the user lifts his/her arm, SenSave gets the sensor data of the activity and recognize the activity as "Lifting up the arm." After that, SenSave automatically adopts a corresponding energy-saving strategy, as shown in the bottom right corner of Figure 29(a). In Figure 29(a), we also show the data of linear accelerometer and gyroscope to describe the movement features of the current activity. Here, Figure 29(a) is used to illustrate the working principle of SenSave. In fact, displaying the sensor data and other modules on the screen will incur extra energy consumption. Therefore, when SenSave works, we remove the unessential modules from the screen, then the screen of the phone will be like that of an ordinary camera, as shown in Figure 29(b).

In Figure 30, we show the activity recognition accuracy of our schemes SenSave and SenSave-MC, and the energy consumption of three schemes, that is, "Turn ON/OFF Scheme," "SenSave," "SenSave-MC." Figure 30(a) shows the activity recognition accuracy of SenSave in body level, arm level, wrist level, and the overall accuracy of SenSave, SenSave-MC, respectively. No matter which level the activity belongs to, the recognition accuracy can achieve above 87%. For all the activities, the overall recognition accuracy is 90%. Therefore, our scheme "SenSave" can efficiently classify an activity into one of the three levels and recognize the activity in a level accurately. In regard to "SenSave-MC," it utilizes the extended Markov chain to predict the next activity state and uses the recognized activity to calibrate the prediction result. The recognition accuracy of SenSave-MC is close to that of SenSave. For energy consumption, we compare "SenSave" and "SenSave-MC" with the "Turn ON/OFF Scheme," as shown in Figure 30(b). When compared with "Turn ON/OFF Scheme," "SenSave" can reduce the energy consumption by 30.0%. While "SenSave-MC" can, respectively, reduce the energy consumption by 36.1% and 8.8%, when compared with "Turn



(a) Activity recognition accuracy



(b) Energy consumption

Fig. 30. Activity recognition accuracy and energy consumption during photographing.

ON/OFF Scheme" and "SenSave". Therefore, our proposed schemes "SenSave" and "SenSave-MC" outperform the common scheme "Turn ON/OFF Camera and Screen" in terms of energy saving in photographing, while guaranteeing a better user experience without user interaction.

9 RELATED WORK

9.1 Activity Sensing

With the development of smart phones, more and more sensors are integrated into smart phones. By utilizing the built-in sensors, smart phones can be used for modeling and monitoring human behaviors (Yang et al. 2014; Ren et al. 2015b; Bulut et al. 2015; Li et al. 2016) and recognizing human activities (Lane et al. 2010; Khan et al. 2013). According to the number of sensors used in activity sensing, we can classify the approaches into single-sensor-based activity sensing and multi-sensor-based activity sensing.

For single-sensor-based activity sensing, many people tend to choose the acceleromter to recognize activities. Miluzzo et al. use the built-in accelerometer of the phone to recognize the activities like sitting, standing, walking, and running, when the user carries the phone (Miluzzo et al. 2008). Sun et al. use accelerometer-embedded mobile phones to to recognize the physical activities, while the mobile phone's position and orientation are varying (Sun et al. 2010). Kwapisz et al. use the labeled accelerometer data to recognize the user's daily activities such as walking, jogging, upstairs, downstairs, sitting and standing, by applying four machine learning algorithms (Kwapisz et al. 2011). Lee et al. use accelerometers with hierarchical hidden markov models to distinguish the daily actions (Lee and Cho 2011). Ren et al. propose a user verification system, which leverages unique gait patterns derived from acceleration readings to detect possible user spoofing in mobile healthcare systems (Ren et al. 2015a). To reduce the battery consumption for activity sensing, Akimura et al. apply the compressed sensing theory to activity sensor data gathering and reduce the acceleration data without heavy computation costs (Akimura et al. 2012). In addition to the accelerometer, the built-in microphone can also be used to detect the events that are closely related to sleep quality, including body movement, cough, and snore (Hao et al. 2013; Ren et al. 2015b).

For multi-sensor-based activity sensing, we can combine the sensors like accelerometer, gyroscope, microphone, and so on, to recognize activities. Johnson et al. utilize the gyroscope, accelerometer, magnetometer, GPS, and video to recognize the driving style, which is concerned with man's life (Johnson and Trivedi 2011). Shahzad et al. propose a gesture-based user authentication scheme for the secure unlocking of touch screen devices. It makes use of the coordinates of each touch point on the screen, accelerometer values, and time stamps (Shahzad et al. 2013). Chen et al. take advantage of features as light, phone situation, stationary, and silence to monitor user's sleep (Chen et al. 2013b). Bo et al. propose a framework to verify whether the current user is the legitimate owner of the smartphone based on the behavioral biometrics, including touch behaviors and walking patterns. These features are extracted from the phone's built-in accelerometer and gyroscope (Bo et al. 2013). Being different from these prior work, we aim to propose an energy-saving scheme for photographing. We aim to recognize the user's activity and reduce unnecessary energy cost during photographing. The scheme does not need hardware modifications and does not need user interaction, to guarantee a good user experience.

9.2 Energy Saving

Prior work on energy saving of smart phones can be classified into three categories: energy consumption of hardware, power consumption models and energy-saving schemes for specific applications.

For hardware, Chen et al. analyze the power consumption of AMOLED displays in multimedia applications and reveal that camera recoding incurs high power cost (Chen et al. 2013a). LiKamWa et al. report the experimental and analytical characterization of CMOS image sensors and reveal two energy-proportional mechanisms for energy saving (LiKamWa et al. 2013). Carroll et al. measure the overall system power, and the exact breakdown of power consumption by the device's main hardware components (Carroll and Heiser 2010).

For models, Balasubramanian et al. present a measurement study of the energy consumption characteristics of 3G, GSM, and WiFi. Then, they develop a model for the energy consumed by network activity for each technology (Balasubramanian et al. 2009). Dong et al. propose Sesame, with which a mobile system constructs an energy model of itself without any external assistance (Dong and Zhong 2011). Xu et al. propose a new way called V-edge to generate power models based on battery voltage dynamics (Xu et al. 2013). Min et al. propose PowerForecaster (Min et al. 2015), a system that provides the power use of sensing apps at pre-installation time.

For specific applications, Yan et al. introduce an activity-sensitive strategy for continuous activity recognition, where the choice of the accelerometer's sampling frequency and the classification features are adjusted in real time, to reduce the energy overhead (Yan et al. 2012). Dietrich et al. detect the game's current state and lower the processor's voltage and frequency whenever possible to save energy (Dietrich and Chakraborty 2013). Hu et al. analyze the power consumption during video streaming by considering user skip and early quit scenarios and then introduce an online solution to save energy (Hu and Cao 2015). Zhao et al. propose an energy-aware approach for web browsing in 3G-based smartphones (Zhao et al. 2015). He et al. present a flexible dynamic resolution scaling system for smartphones (He et al. 2015). The system adopts an ultrasonic-based approach to accurately detect the user-screen distance at low-power cost and makes scaling decisions automatically for maximizing user experience and power saving. While considering the energy cost made by human-screen interaction such as scrolling on the screen, Han et al. (2013) and Yu et al. (2015) propose an energy-efficient engine (E^3), which automatically tracks the scrolling speed and adaptively adjusts the frame rate according to user preferences. In addition, there are some other energy-saving schemes in smart phone applications. Ra et al. explores the energy-delay trade-off in delay-tolerant, but data-intensive, smartphone applications (Ra et al. 2010). They formulate the link selection problem as an optimization formulation, which minimizes the total energy expenditure while keeping the average queue length finite. Nath et al. propose a middleware ACE (Acquisitional Context Engine), which supports continuous context-aware applications while mitigating sensing costs for inferring contexts (Nath 2012). Hu et al. propose a Mobility-Assisted User Contact detection algorithm (MAUC) (Hu et al. 2013). They utilize the accelerometer of the phone to detect user movements for energy-saving, and Bluetooth scans only when user movements have a high possibility to cause contact changes.

10 CONCLUSION

In this article, we propose a context-aware energy-saving scheme SenSave for smart camera phones based on activity sensing. We take advantage of the activity features and maintain an activity state machine to do activity recognition. Then we adopt a suitable energy saving strategy based on the result of activity recognition. Besides, by introducing an extended Markov chain to predict the activity state in advance and adopt the energy-saving strategy timely. We enhance SenSave as SenSave-MC. Experiment results show that SenSave can recognize the user's activities with an average accuracy of 95.5% and reduce the energy consumption during photographing by 30.0% for smart camera phones. By introducing the extended Markov chain, SenSave-MC can reduce the energy consumption during photographing by 36.1%. Therefore, our schemes SenSave and SenSave-MC can save energy consumption during photographing, while guaranteeing a good user experience without user interaction.

REFERENCES

Daito Akimura, Yoshihiro Kawahara, and Tohru Asami. 2012. Compressed sensing method for human activity sensing using mobile phone accelerometers. In *Proceedings of INSS*.

Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of ACM SIGCOMM*.

Frank Bellosa, Andreas Weissel, Martin Waitz, and Simon Kellner. 2003. Event-driven energy accounting for dynamic thermal management. In *Proceedings of COLP*.

Cheng Bo, Lan Zhang, Xiang-Yang Li, Qiuyuan Huang, and Yu Wang. 2013. Silentsense: Silent user identification via touch and movement behavioral biometrics. In *Proceedings of ACM MobiCom*.

Muhammed Fatih Bulut, Murat Demirbas, and Hakan Ferhatosmanoglu. 2015. LineKing: Coffee shop wait-time monitoring using smartphones. *IEEE Trans. Mobile Comput.* 14, 10 (2015), 2045–2058.

Aaron Carroll and Gernot Heiser. 2010. An analysis of power consumption in a smartphone. In *Proceedings of the USENIX Annual Technical Conference*, Vol. 14. Boston, MA.

Xiang Chen, Yiran Chen, Zhan Ma, and Felix CA Fernandes. 2013a. How is energy consumed in smartphone display applications? In *Proceedings of ACM HotMobile*.

Zhenyu Chen, Mu Lin, Fanglin Chen, Nicholas D. Lane, Giuseppe Cardone, Rui Wang, Tianxing Li, Yiqiang Chen, Tanzeem Choudhury, and Andrew T Campbell. 2013b. Unobtrusive sleep monitoring using smartphones. In *Proceedings of IEEE PervasiveHealth*.

Benedikt Dietrich and Samarjit Chakraborty. 2013. Power management using game state detection on android smartphones. In *Proceedings of ACM MobiSys*.

Mian Dong and Lin Zhong. 2011. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of ACM MobiSys*.

Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. In *Proceedings of ACM SIGARCH*.

Yuanyuan Fan, Lei Xie, Yafeng Yin, and Sanglu Lu. 2015. A context aware energy-saving scheme for smart camera phones based on activity sensing. In *Proceedings of IEEE MASS*.

Haofu Han, Jiadi Yu, Hongzi Zhu, Yingying Chen, Jie Yang, Guangtao Xue, Yanmin Zhu, and Minglu Li. 2013. E^3: Energy-efficient engine for frame rate adaptation on smartphones. In *Proceedings of ACM Sensys*.

Tian Hao, Guoliang Xing, and Gang Zhou. 2013. iSleep: Unobtrusive sleep quality monitoring using smartphones. In *Proceedings of ACM Sensys*.

Songtao He, Yunxin Liu, and Hucheng Zhou. 2015. Optimizing smartphone power consumption through dynamic resolution scaling. In *Proceedings of ACM MobiCom*.

Wenjie Hu and Guohong Cao. 2015. Energy-aware video streaming on smartphones. In *Proceedings of IEEE INFOCOM*.

Wenjie Hu, Guohong Cao, Srikanth V Krishanamurthy, and Prasant Mohapatra. 2013. Mobility-assisted energy-aware user contact detection in mobile social networks. In *Proceedings of IEEE ICDCS*. IEEE, 155–164.

Derick A. Johnson and Mohan M. Trivedi. 2011. Driving style recognition using a smartphone as a sensor platform. In *Proceedings of IEEE ITSC*.

Wazir Zada Khan, Yang Xiang, Mohammed Y. Aalsalem, and Quratulain Arshad. 2013. Mobile phone sensing systems: A survey. *IEEE Commun. Surv. Tutor.* 15, 1 (2013), 402–427.

Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. 2011. Activity recognition using cell phone accelerometers. *SIGKDD* 12, 2 (2011), 74–82.

Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. 2010. A survey of mobile phone sensing. *IEEE Commun. Mag.* 48, 9 (2010), 140–150.

Young-Seol Lee and Sung-Bae Cho. 2011. Activity recognition using hierarchical hidden markov models on a smartphone with 3D accelerometer. In *Proceedings of Springer HAIS.*

Qiang Li, Qi Han, and Limin Sun. 2016. Collaborative recognition of queuing behavior on mobile phones. *IEEE Trans. Mobile Comput.* 15, 1 (2016), 60–73.

Robert LiKamWa, Bodhi Priyantha, Matthai Philipose, Lin Zhong, and Paramvir Bahl. 2013. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceedings of ACM MobiSys.*

Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. 2008. Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proceedings of ACM SenSys.* ACM, 337–350.

Chulhong Min, Youngki Lee, Chungkuk Yoo, Kang Seungwoo, Sangwon Choi, Pillsoon Park, Inseok Hwang, Younghyun Ju, Seungpyo Choi, and Junehwa Song. 2015. PowerForecaster: Predicting smartphone power impact of continuous sensing applications at pre-installation time. In *Proceedings of ACM SenSys.*

Suman Nath. 2012. ACE: Exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of ACM MobiSys.* ACM, 29–42.

Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. 2011. E-gesture: A collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *Proceedings of ACM SenSys.* ACM, 260–273.

Huimin Qian, Yaobin Mao, Wenbo Xiang, and Zhiquan Wang. 2010. Recognition of human activities using SVM multi-class classifier. *Pattern Recogn. Lett.* 31, 2 (2010), 100–111.

Moo-Ryong Ra, Jeongyeup Paek, Abhishek B. Sharma, Ramesh Govindan, Martin H. Krieger, and Michael J. Neely. 2010. Energy-delay tradeoffs in smartphone applications. In *Proceedings of ACM MobiSys.* ACM, 255–270.

Dinesh Rajan, Russell Zuck, and Christian Poellabauer. 2006. Workload-aware dual-speed dynamic voltage scaling. In *Proceedings of IEEE RTCSA.*

Yanzhi Ren, Yingying Chen, Mooi Choo Chuah, and Jie Yang. 2015a. User verification leveraging gait recognition for smartphone enabled mobile healthcare systems. *IEEE Trans. Mobile Comput.* 14, 9 (2015), 1961–1974.

Yanzhi Ren, Chen Wang, Jie Yang, and Chen Yingying. 2015b. Fine-grained sleep monitoring: Hearing your breathing with smartphones. In *Proceedings of IEEE INFOCOM.*

Samsu Sempena, Nur Ulfa Maulidevi, and Peb Ruswono Aryan. 2011. Human action recognition using dynamic time warping. In *Proceedings of IEEE ICEEI.* IEEE, 1–5.

Muhammad Shahzad, Alex X. Liu, and Arjmand Samuel. 2013. Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you cannot do it. In *Proceedings of ACM MobiCom.*

Lin Sun, Li Bin Zhang, Daqing, Bin Guo, and Shijian Li. 2010. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. *Ubiq. Intell. Comput.* 6406 (2010), 548–562.

Google Inc. 2016a. Android APIs. Retrieved from http://developer.android.com/reference/android/hardware/SensorEvent.html#values.

Google Inc. 2016b. Camera Parameters. Retrieved from https://developer.android.com/reference/android/hardware/Camera.Parameters.html.

Google Inc. 2016c. Sensor types. Retrieved from https://source.android.com/devices/sensors/sensor-types.html.

KS Mobile Inc. 2014. Top 10 Battery Draining Apps for Android. Retrieved from http://www.businesswire.com/news/home/20140227005449/en/Clean-Master-Announces-Top-10-Android-Vampire.

Monsoon Solutions Inc. 2015. Monsoon Power Monitor. Retrieved from https://www.msoon.com/LabEquipment/PowerMonitor/.

Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous keyboard for small mobile devices: Harnessing multipath fading for fine-grained keystroke localization. In *Proceedings of ACM MobiSys.* ACM, 14–27.

Fengyuan Xu, Yunxin Liu, Qun Li, and Yongguang Zhang. 2013. V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics. In *Proceedings of NSDI.*

Zhixian Yan, Vigneshwaran Subbaraju, Chakraborty, Dipanjan, Archan Misra, and Karl Aberer. 2012. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Proceedings of ISWC.*

Zheng Yang, Longfei Shangguan, Zimu Gu, Weixi amd Zhou, Chenshu Wu, and Yunhao Liu. 2014. Sherlock: Micro-environment sensing for smartphones. *IEEE Trans. Parall. Distrib. Syst.* 25, 12 (2014), 3295–3305.

Yafeng Yin, Qun Li, Lei Xie, Shanhe Yi, Edmund Novak, and Sanglu Lu. 2016. CamK: A camera-based keyboard for small mobile devices. In *Proceedings of the IEEE INFOCOM*. IEEE, 1–9.

Jiadi Yu, Haofu Han, Hongzi Zhu, Yingying Chen, Jie Yang, Yanmin Zhu, Guangtao Xue, and Minglu Li. 2015. Sensing human-screen interaction for energy-efficient frame rate adaptation on smartphones. *IEEE Trans. Mobile Comput.* 14, 8 (2015).

Bo Zhao, Wenjie Hu, Qiang Zheng, and Guohong Cao. 2015. Energy-aware web browsing on smartphones. *IEEE Trans. Parall. Distrib. Syst.* 26, 3 (2015), 761–774.

Received January 2016; revised November 2016; accepted April 2017