Sherlock: Micro-environment Sensing for Smartphones

Zheng Yang, *Member, IEEE,* Longfei Shangguan, *Student Member, IEEE,*
Weixi Gu, *Student Member, IEEE,* Zimu Zhou, *Student Member, IEEE,*
Chenshu Wu, *Student Member, IEEE,* and Yunhao Liu, *Senior Member, IEEE*

Abstract—Context-awareness is getting increasingly important for a range of mobile and pervasive applications on nowadays smartphones. Whereas human-centric contexts (e.g., indoor/ outdoor, at home/in office, driving/walking) have been extensively researched, few attempts have studied from phones' perspective (e.g., on table/sofa, in pocket/bag/hand). We refer to such immediate surroundings as *micro-environment*, usually several to a dozen of centimeters, around a phone. In this study, we design and implement Sherlock, a micro-environment sensing platform that automatically records sensor hints and characterizes the micro-environment of smartphones. The platform runs as a daemon process on a smartphone and provides finer-grained environment information to upper layer applications via programming interfaces. Sherlock is a unified framework covering the major cases of phone usage, placement, attitude, and interaction in practical uses with complicated user habits. As a long-term running middleware, Sherlock considers both energy consumption and user friendship. We prototype Sherlock on Android OS and systematically evaluate its performance with data collected on fifteen scenarios during three weeks. The preliminary results show that Sherlock achieves low energy cost, rapid system deployment, and competitive sensing accuracy.

1 Introduction

In mobile systems, context-awareness is a computing technology that incorporates information about the current environment of a mobile user to provide more relevant services to the user. It is a key component of ubiquitous or pervasive computing and has attracted many research efforts in the past decade.

Most context-aware applications (via mobile phone sensing) are human-centric, recognizing contexts from users' perspective (e.g., indoor/outdoor [9], at home/in office, driving/walking [2]). Such information supports services according to *users'* situation. For example, when a mobile phone detects that its user is driving, it automatically blocks phone calls if its user is holding it in hand for safety [1]. When a user enters a building, it is unnecessary to keep his phone's GPS working to save energy. Similarly, WiFi is usually unavailable in the open countryside and should be turned off there [9].

While human-centric contexts have been extensively utilized, few works study from *phones'* perspective. We refer the immediate surroundings (i.e., several to a dozen of centimeters around a phone) as *micro-environment*. Similar to human-centric environments, being aware of micro-environments is directly beneficial to a broad range of phone applications. For example, if a mobile phone is in a bag or pocket, it is useless to light up the screen when a phone call is coming. In addition, if a phone is placed on a sofa rather than on a desk, it is better to turn up ring volume to avoid missing calls. Given accurate micro-environment information, a phone can adapt its behaviour automatically and properly.

In this paper, we design *Sherlock*, a micro-environment sensing platform that automatically records sensor hints and characterizes the immediate surroundings of smartphones. It runs as a daemon process on a smartphone and provides finer-grained environment information to upper layer applications running on the smartphone.

To implement such a platform, difficulties are triple. First, previous context-aware solutions (especially the algorithms and metrics) are assisted by human intuition; however, the micro-environments are less sensible for people. Second, the usage, placement, attitude, and interaction of smartphones vary across time and users, thus complicating timely and accurate micro-environment detection. Third, distinguishing similar micro-environments relies on systematic collaboration among multi-modal sensors.

We build the framework of Sherlock upon an investigation of phone usage and user habits. The framework covers the majority of phones' states, and consists of 3 core modules: phone placement detection, phone interaction detection, and backing material detection. Phone placement refers to the location of a smartphone along with its user, and we consider the situations of in bag, in chest pocket, in pants, and in hand. Whether a user is concentrating on his smartphone is another key judgement for micro-environment sensing. At last, backing material detection analyzes the hardness of the stuff that touches (or holds) the phone.

We implement Sherlock on 3 types of Android smart-

• *Z. Yang, W. Gu, C. Wu and Y. Liu are with School of Software and TNList, Tsinghua University, Beijing, China*
E-mail: {yang, guweixi, wu, yunhao}@greenorbs.com
• *L. Shangguan and Z. Zhou are with the Department of Computer Science & Engineering, The Hong Kong University of Science & Technology, Hong Kong*
E-mail: {longfei, zhouzimu}@greenorbs.com

phones and run it as a background service. Other Apps can obtain the current micro-environment information from the platform via programming interfaces and use it accordingly. We evaluate the platform with 8 volunteers in 15 scenarios during 3 weeks, mainly in campus areas during the normal period 7:00 to 23:00. Preliminary results demonstrate that Sherlock achieves average detection error of below 17%, with 11.4% additional energy cost.

In summary, the key contributions of this paper are: First, Sherlock is a unified micro-environment sensing framework. Although some previous works have implemented part of similar functionality for simple environments, they cannot be directly combined to an applicable level for practical use with complicated phone situations and user habits. Second, as a middleware run on smartphones, Sherlock is both energy optimized and user friendly. We design a hierarchical architecture and a set of efficient algorithms for multi-stage micro-environment detection to reduce working time and the types of sensors. In addition, sensors, especially actuators[1], are carefully selected for the purpose of effectiveness and non-intrusiveness. For example, Sherlock won't trigger vibrator or speaker when a smartphone is carried by its user.

The rest of the paper is organized as follows. We provide the motivation and architecture overview in Section 2. System design and implementation details are interpreted in Section 3. We present evaluation results in Section 4. Section 5 reviews related work and Section 6 concludes this paper.

2 Motivation and Overview

2.1 Target Applications

The aim of micro-environment sensing on smartphones is to provide a more general primitive for novel human-centric applications, especially in healthcare and behavior monitoring. For example, it is important to ensure that the healthcare monitors are attached to the target user during his daily life, and emerging trends arise to perform such tasks via smartphones [13], [14]. A micro-environment perceivable smartphone, therefore, would remind its user if it is not carried by its user via, e.g. its built-in speaker, and further informs him of its location.

Identifying the phone's micro-environment also opens new possibilities to perform fine-grained context-aware energy saving strategies, which is essential for battery-powered smartphones. On detecting being placed in the drawer, for instance, it is reasonable for the phone to infer that it will not be used in the near future, and can switch to certain power saving mode and turn off unnecessary sensors and software.

In addition, Sherlock enables more accurate inertial based localization and navigation. In most of these schemes, a key parameter is the count of the user's footsteps, which is then multiplied by the average length of one footstep to estimate trace distance. Empirical studies [10] have shown that the accuracy of step counter is sensitive to phone placement. For instance, the counter usually generates accurate step count (i.e. consistent with the ground truth) when the phone is held in hand, while often doubles the output count when the phone is placed in chest pocket. Hence knowing the phone's placement assists the step counter to eliminate erroneous output.

Like GPS which helps to estimate user's coarse-grained macro-environment, Sherlock deduces phone's fine-grained micro-environment. It serves as a light-weighted middleware for upper layer applications.

1. An actuator here stands for a type of motor that converts the energy, typically electric current, into motion or mechanical operation. In mobile phone, the actuator includes vibrator, camera and microphone.

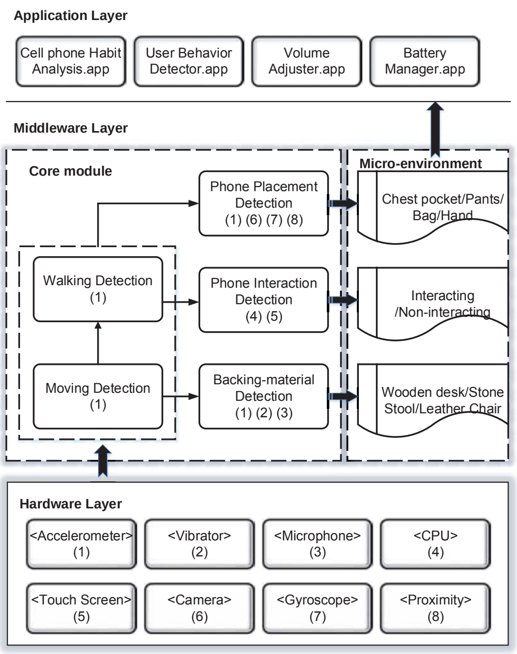


Fig. 1. System architecture of Sherlock.

2.2 System Overview

As Figure 1 shows, Sherlock runs as a daemon process in the middleware layer. It employs sensors in the physical layer to record nature events and provides fine-grained environment information to upper layer applications. As a long-term middleware on smartphones, Sherlock optimizes energy consumption via a hierarchical, multi-stage architecture. Sensors, especially actuators, are carefully selected and logically triggered. Accelerometer, for example, is solely awaken to detect simple environment semantics, after which more sensors are triggered for complex environment classification. In what follows, we describe each architectural module in turn, specifying a high-level view of how the system works.

Moving & Walking Detection. As a first step, Sherlock looks into the acceleration trace and identifies specific features in time domain. These features are then utilized to determine whether the phone is in motion. There are plenty of moving detection schemes that can successfully

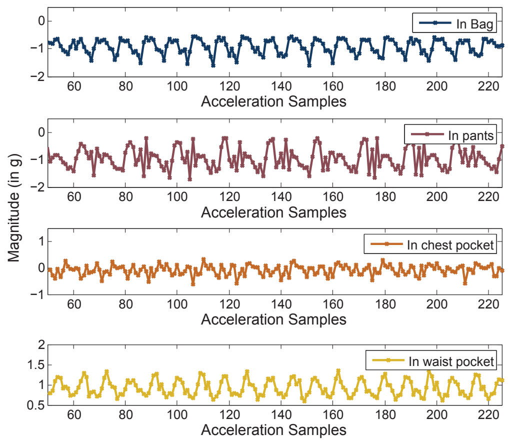


Fig. 2. An illustration of repetitive pattern of walking.

detect human motions (whether the user is moving and further if he is walking) and we use one from [8]. This scheme first detects whether the user is in motion by looking into acceleration variance. if yes, it then takes advantage of the *repetitive* nature of walking (Figure 2) and applies an auto-correlation based detection method to successfully mine the walking pattern of the user.

The above components characterize the coarse-grained environment around smartphones and benefit further sensing processes. If a user is detected walking, for example, then determining phone's placement (e.g., in chest pocket or bag) is more important than knowing its backing material. If the phone is detected stationary, it is more likely that it is out of its user's perception (e.g., fell onto a wooden desk or a leather chair). In this scenario, detecting the backing material of smartphone, and further alerting the phone user are more preferable. In what follows, we detail 3 key modules of our micro-environment sensing platform.

Local Placement Recognition. This module determines daily on-body phone placements such as in-hand, in-packet, in-bag, etc. Sherlock provides a simple yet effective classification scheme with light and inertial sensors. Specifically, the system first detects whether the phone is in hand by referring ambient illuminative conditions around the phone. If not, then, Sherlock characterizes the unique moving patterns of phones in different local placements with in-built accelerometer, by exploiting a Dynamic Time Warping (DTW) based time series matching scheme to recognize the specific local placement. i.e., in pants, in chest pockets or in bag.

Phone Interaction Detection. This module identifies whether the user is actually using the phone, like browsing. Although such interaction often occurs when the phone is *in-hand*, the phone interaction detection module emphasizes more on the semantic perspective. Sherlock exploits common screen-lock on smartphones and process transition on OS to identify whether the user is actually interacting with his phone.

Backing Material Detection. This module differentiates hard/soft material via smartphone-generated vibration patterns. Sherlock focuses on two aspects of the

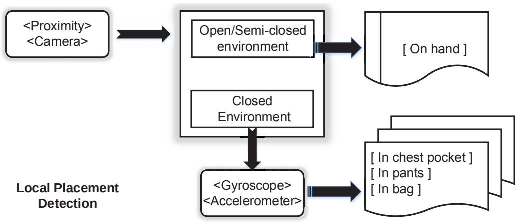


Fig. 3. Workflow of Local Placement Recognition.

vibration patterns: 1) the phone's mechanical motion and 2) the acoustical features, which can be captured by embedded accelerometer and microphone, respectively. To this end, Sherlock extracts a series of lightweight features from acceleration/acounstic traces in both time and frequency domain, and classifies backing materials like leather chair, wood desk or glass table.

3 System Design

3.1 Local Placement Recognition

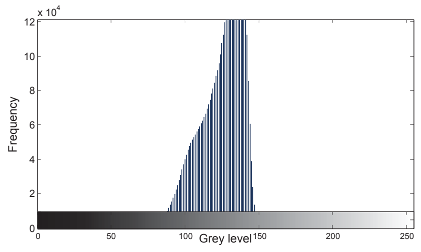
We develop a simple yet effective local placement classification scheme with light and inertial sensors. The key insights are twofold.

- When carried by a user, the phone is mostly placed in either *semi-closed/open* environments like in-hand, or *closed* environments such as in-pocket and in-bag. The extent of covering leads to different illuminative conditions for the phone, which can be captured by its built-in camera.
- Different local surroundings offer distinctive spatial degree of freedom, which is magnified when the user is moving. For instance, a phone is likely to experience fiercer movements when put in pants than inside a handbag. These unique movement patterns can be perceived by the accelerometer.

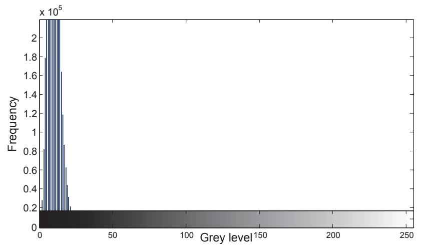
As illustrated in Figure 3, the local placement recognition module is triggered once the 'Walking' state is determined, and it works as follows: 1) the front-mounted proximity sensor and the back-mounted camera cooperate to identify *semi-closed/open* surroundings, i.e., the 'in-hand' state. 2) For *closed* environments, accelerometer is employed to automatically deliver sensory data for fine-grained placement identification. e.g., in pants, chest pockets or bags. We detail the processes as follows.

3.1.1 Phone-in-hand Identification

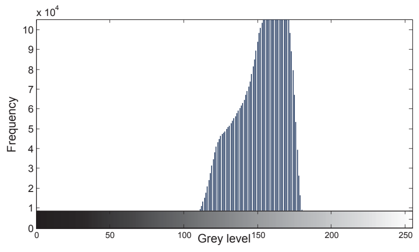
Intuitively, the 'in-hand' state differs from on-body placements in that the phone is not completely covered by surrounding objects. Although the front-mounted proximity sensor can perceive sheltering in front, the phone is unaware of that backwards. Thus with proximity sensor alone, it is likely to miss some 'in-hand' cases, e.g., when the user is making a phone call with his ear covers the front end of the phone. Therefore we also employ the back-mounted camera for proximity perception backwards. The rationale is that the global contrast of a photo taken in a closed environment (e.g.



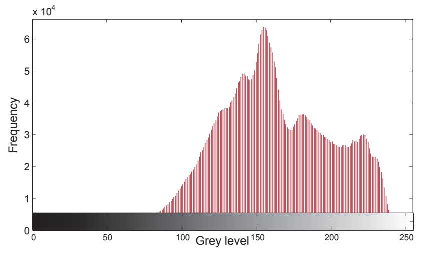
(a) Histogram of Closed case 1#



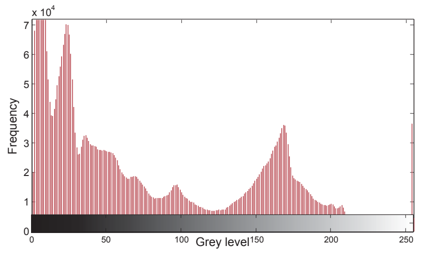
(b) Histogram of Closed case 2#



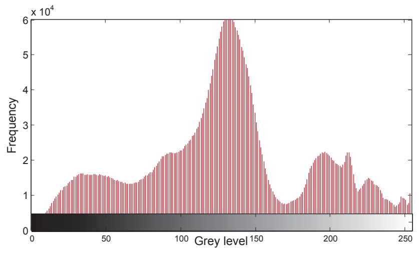
(c) Histogram of Closed case 3#



(d) Histogram of Open case 1#



(e) Histogram of Open case 2#



(f) Histogram of Open case 3#

Fig. 4. Gray-scale histogram of closed/open environments.

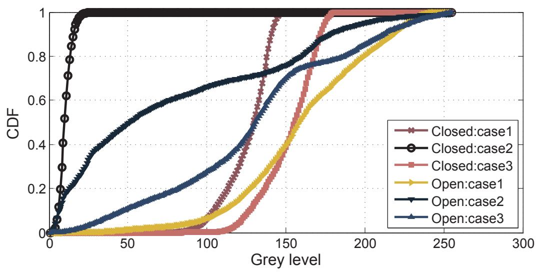


Fig. 5. CDF of gray-scale histograms under various phone placements.

in-pocket) is usually low, which is reflected in the gray-scale histogram of the photo.

As a motivating experiment, we collect photos taken by a background photographing application for various phone placements in diverse scenarios, including chest pocket, pants, bags and hands in supermarkets, cafes and streets. Figure 4 demonstrates the gray-scale histogram distribution of photos in 6 different conditions. The upper 3 correspond to closed environments including in bags, chest pockets and pants, while the lower 3 are in-hand situations. In general, the histogram distributes more spread-out when the phone is held in hand than placed in closed environments, indicating higher global contrast due to better lighting conditions. To quantitatively measure the extent of dispersion of the gray-scale histogram, we calculate the average slope of gray-scale pixels between two quantiles q_1 and q_2 in its CDF. Figure 5 plots the CDF for the 6 situations. As is shown, the histogram CDFs w.r.t. closed environments climb steeply up to 85% within a short range of gray-scales, while those w.r.t. 'in-hand' states experience a sluggish growth spanning a large portion of the gray-scale range. Putting it all together, we develop the following Image-based Phone-in-hand Detection Scheme.

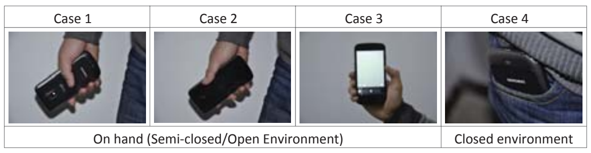


Fig. 6. In-hand states vs. Closed environments.

Image-based Phone-in-hand Detection Scheme (IPDS). Firstly, the proximity sensor identifies the sheltering condition in front, and returns either $blocked$ or $unblocked$. Meanwhile, the camera is triggered to take a photo. After calculating the CDF of its gray-scale histogram, we record the portion of pixels (e) within 2 empirically optimized quartiles of $q_1 = 0.2$ and $q_2 = 0.85$. If $e > \hbar$, where $\hbar$ is a predefined threshold, the global contrast tends to be high and the camera is not blocked by other objects. We test a range of thresholds and find 50 optimal for IPDS. By jointly considering the front-end blocking and backward lighting conditions, four distinct cases follow:

- Case 1: $[e > \hbar \wedge blocked]$: only front is blocked;
- Case 2: $[e \leq \hbar \wedge unblocked]$: only back is blocked;
- Case 3: $[e > \hbar \wedge unblocked]$: neither front nor back is blocked;
- Case 4: $[e \leq \hbar \wedge blocked]$: both front and back are blocked;

These four cases are illustrated in Figure 6, with Cases 1-3 corresponding to different in-hand states, while Case 4 indicates a closed environment.

3.1.2 On-body Placement Recognition

On-body placement recognition classifies *closed* environments into finer-grained on-body placements such as in chest pocket, pants, bags, etc. As previously discussed, the module takes advantage of human mobility induced inertial patterns, which potentially limits its usage to

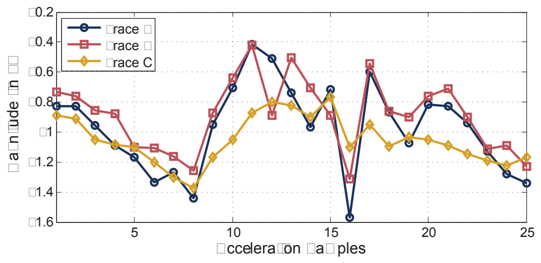


Fig. 7. Stability of acceleration trace over time.

truly 'mobile' phones. To compensate for this weakness, we propose a backing material recognition scheme (Section 3.3) based on phone induced vibrations specially designed for 'immobile' phones when the phone-holder stays still, or even when the phone is placed off-body (e.g. left on a sofa). The two modules are complementary and can cooperate to further enhance the recognition performance. For 'mobile' cases, though, the on-body placement recognition module proves to be sufficient for normal phone placement identification.

As human induced mobility is mainly perceived by inertial sensors, we take a careful scrutiny on acceleration traces with different phone placements. Revisiting the acceleration traces with different phone placements in Figure 2, it is obvious that acceleration samples within a single footstep demonstrate unique pattern across different phone placements, while walking leads to a regenerative process of these acceleration patterns, indicating viability to take the acceleration traces as fingerprints for different phone placements. However, it remains unsettled whether the patterns of the same phone placement stay similar when taking device and user diversity into consideration.

Figure 7 plots the acceleration traces sampled from 3 types of phones (Samsung Galaxy S2 I9100, Samsung Nexus3 I9250, Motorola MT788), which are put in different users' pants. The 3 traces roughly share common variation trends, indicating stable patterns across users, whereas certain lags are also notable. The second peak in trace A, for example, appears at the 15^{th} sample while it occurs at the 13^{th} sample in trace B, i.e., 2-unit lags after trace A. These lags are due to different walking speed. In general, given a fixed sampling rate, a rush stride tends to shrink the trace pattern, while a stroll at leisure stretches the pattern and induces random deformation as well. Therefore, a robust and speed-independent similarity metric is needed to compare and classify the measured acceleration traces.

DTW-based Trace Matching (DTM). Dynamic Time Warping (DTW) [15] is a dynamic programming based similarity measure for sequences which may vary in time or speed. In DTW, the two sequences are first reconstructed by non-linear "warping" in the time domain to compare their similarity independent of non-linear temporal variations. Therefore, DTW based trace matching is able to eliminate the effect of different walking speed.

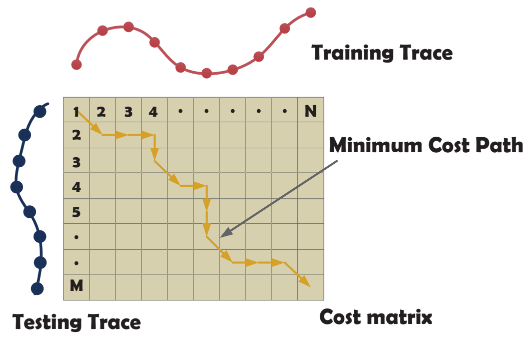


Fig. 8. DTW based matching.

Given two acceleration profiles A and B with lengths of M and N samples, DTW first constructs a distance matrix d[M×N], where

$$d(i,j) = (a_i - b_j)^2 \tag{1}$$

and a_i and b_j are the i^{th} and j^{th} elements in A and B, respectively. Taking d[M×N] as input, DTW returns a warping path P=$\{p_1, p_2, p_3, \ldots, p_k\}$, where $p_i = (x, y) \in [1:M] \times [1:N]$ for $i \in [1:k]$.

Figure 8 illustrates the matching process. To generate the warping path, DTW constructs a cost matrix C[M×N] which stands for the minimum cost to reach any point (i, j) in the matrix from (1, 1) in a dynamic programming fashion. For instance, (i, j) can be reached from ($i-1$, $j-1$), (i, $j-1$) and ($i-1$, j). The algorithm picks the one with minimum cost. Formally:

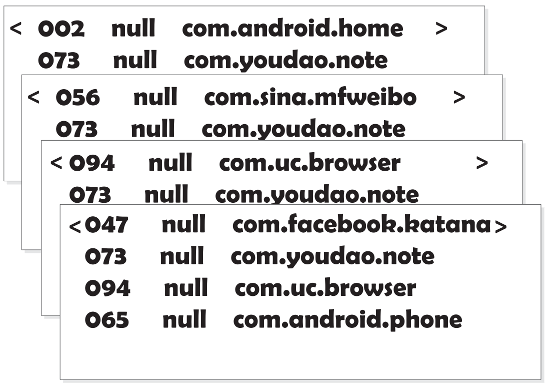
$$C(i,j) = d(i,j) + min(C(i-1,j-1), C(i,j-1), C(i-1,j)) \tag{2}$$

A measured acceleration trace is compared with all pre-stored traces collected with different phone placements based on DTW and output the corresponding minimum costs. The phone placement w.r.t. the smallest cost is then identified as the phone placement for the measured acceleration trace. For example, if C(M,N) = 25 for pants, C(M,N) = 36 for chest pocket and C(M, N) = 19 for bag. Then our scheme classifies this acceleration trace into the category of *in-bag*.

The on-body phone placement recognition scheme does not rely on the *closed* environment, and thus is orthogonal to the 'in-hand' detection in Section 3.1.1. Therefore the on-body placement detection scheme also serves as a double verification to improve the robustness of the in-hand detection scheme. This is useful when the IPDS suffers from low global contrast background like white walls all around or gloomy lighting conditions.

3.2 Phone Interaction Detection

Phone interaction detection identifies whether the user is using the phone, e.g. browsing, texting, playing games, etc. Although such interaction often occurs when the phone is 'in-hand', which can be identified as in Section 3.1.1, the phone interaction detection scheme in this section emphasises more on the semantic perspective.



```
< 002   null   com.android.home      >
  073   null   com.youdao.note

< 056   null   com.sina.mfweibo      >
  073   null   com.youdao.note

< 094   null   com.uc.browser        >
  073   null   com.youdao.note

< 047   null   com.facebook.katana   >
  073   null   com.youdao.note
  094   null   com.uc.browser
  065   null   com.android.phone
```

Fig. 9. An illustration of process queues in different scenarios.

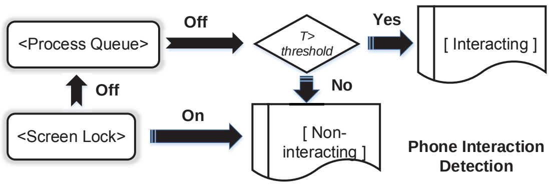


Fig. 10. The work-flow of phone interaction detection module.

An intuitive indicator for interaction detection is the screen-lock on touch screen smartphones. The touch screen is typically unlocked on an 'interaction active' phone. Nevertheless, the opposite is not always true. According to a questionnaire we conducted on 500 students in Tsinghua University, around 420 lock their phones in 'non-interactive' states, while the other 80, for ease of operation, would like to keep their phones unlocked all the time. Therefore, with the screen-lock alone, we would result in high false-positive for interaction detection.

To obtain a more accurate usage detection scheme, we utilize the phone's process queue. The on-executing process, in general, is on top of the process queue. Therefore, if the phone is running an application, thus 'interaction active', a corresponding process ought to be running and take up the 1^{th} position of process queue. This leads us to identify the 'interaction active' state by checking the current on-executing process. Figure 9 lists the process queues of 4 specific application scenarios: non-interactive, browsing Weibo[2], searching online, and logging in facebook. As is shown, a process $com.android.home$ is running on the foreground (denoted by $< 002\ null\ com.android.home >$) in non-interactive state. In all the other scenarios, the top on-executing process is always the process corresponding to the specific application. Therefore, we propose the following phone interaction detection scheme.

Integrated Interaction Detection Scheme (IIDS). As illustrated in Figure 10, IIDS first queries the screen-lock to check whether the screen is locked. If the screen is locked, IIDS confirms a 'non-interactive' state and suspends; otherwise it further refers to the process queue. If the on-executing process is $com.android.home$, and the execution duration (T) is shorter than a predefined threshold δ, IIDS announces a 'non-interactive' state as well. For other cases, IIDS returns an 'interaction-active' state. Here δ is set to be 17 seconds, which is slightly longer than the normal sleeping interval of touch-screen.

2. Weibo is a Chinese microblogging website. Akin to a hybrid of Twitter and Facebook.

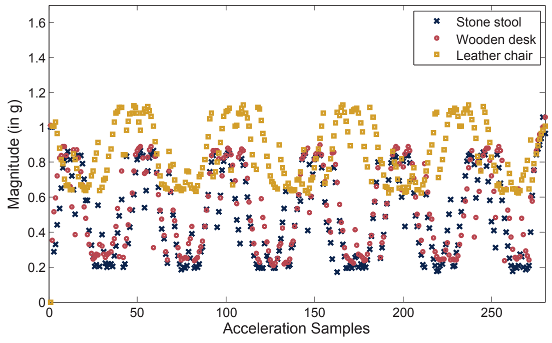


Fig. 11. Acceleration samples over time domain.

3.3 Backing Material Recognition

This subsection aims at distinguishing hard/soft material via smartphone-generated vibration patterns. We mainly focus on two aspects of the vibration patterns: 1) the phone's mechanical motion and 2) the acoustic features, which can be captured by embedded accelerometer and microphone, respectively. More concretely, with a phone placed on a backing surface, the vibration pattern of the phone-surface system driven by the internal phone motor varies with the stiffness of the backing material. The physical underpinning is that the more rigid the material is, the smaller phone-driven deformation and shorter recovery time it would experience, and hence less energy absorption. Consequently, the acceleration values detected on harder material would demonstrate larger amplitudes of fluctuations due to more notable mechanical motions, while the magnitude of dominant frequency of the vibration sound would be higher as well. We detail our recognition schemes as follows.

3.3.1 *Acceleration-based Recognition*

In the normal course, smartphones are placed on fiber, wood, stone or metal material. As a proof-of-concept experiment, we select 3 representative material with different stiffness: leather chair (soft), wooden desk and stone stool (hard). Our scheme easily extends to other materials with a bit extra calibration. In each scenario, the phone motor is triggered to vibrate for 7 seconds, and the acceleration readings and sounds are recorded.

The acceleration-based recognition is performed first to roughly distinguish hard and soft material, i.e. the leather chair or wooden desk/stone stool in our case. Figure 11 illustrates the acceleration traces of the z axis sampled at 40Hz on the three surfaces. The amplitudes of both the envelope and the embedded short pulses tend

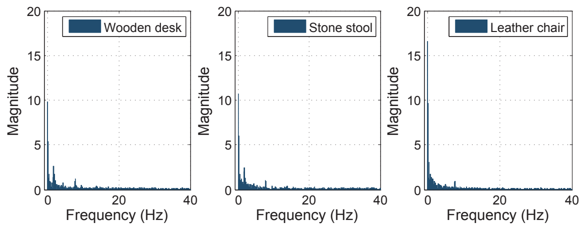


Fig. 12. Acceleration samples over frequency domain.

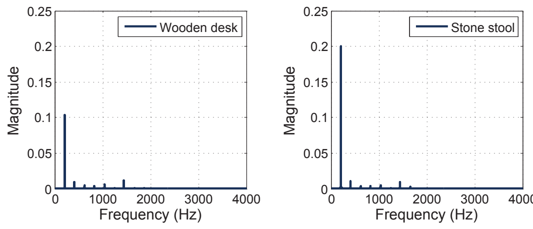


Fig. 13. Acoustic spectrogram of phone vibrations on hard surfaces.

to be larger on hard material. This distinction is clearer in the frequency domain. Figure 12 portrays the FFT amplitudes of the corresponding temporal traces. As shown in the two subfigures on the left, two notable peaks approximate the dominant frequencies of the mechanical vibrations, while in the rightmost figure (i.e., on the leather chair), the peaks at the corresponding frequencies are almost invisible. One counter-intuitive observation, however, is that the DC component (i.e. the amplitude at 0Hz) is strongest in the soft material case. Note that the acceleration readings have included the gravity as well. Hence, when removing the effect of gravity, the DC component for soft material cases approaches zero, which resembles a simple harmonic motion. On hard material, in contrast, the phone is more likely to leave and fall back to the surface periodically, which, on average, induces larger and shorter accelerations upwards, and thus non-zero DC component. Due to the irregular inclination angles when leaving the surface, tough, the acceleration readings of the z axis do not necessarily correspond to the vertical direction. Therefore it is non-trivial to extract accurate DC component to distinguish hard and soft material. We leave careful calibration of DC component in future work, and for this paper, we adopt a multi-feature based classifier with less weights on the DC component related features.

Soft/Hard Backing Material Classifier. We consider various candidate features, including mean, variance, Zero Crossing Rate (ZCR) [5] in the time domain, and number of peaks, sub-band energy, spectral entropy in the frequency domain. Since the differences of acceleration traces are more notable in the frequency domain, we manually put larger weights on all candidate spectral features. In addition, we leave out the DC component when extracting all the features to alleviate errors incurred by lack of calibration on gravity. We adopt Support Vector Machine (SVM) for feature classification to achieve satisfactory performance while retaining moderate computation cost. The features are scaled according to their assigned weights before input into the SVM classifier.

3.3.2 Acoustic-based Recognition

As shown in Figure 12 (a) and Figure 12 (b), the spectrum of accelerations between stone and wood resemble each other. Hence acceleration-based recognition can only distinguish hard and soft material. We thus further investigate a broader frequency range to extract unique vibration patterns. Figure 13 compares the spectrogram of sound captured by the microphone during the phone vibration on two representative *hard* surfaces. It is clear that the dominant vibration frequency peaks at around 200Hz (the same frequency as the phone's vibration motor), while the amplitude on the stone is larger than that on the wood. This accords with the phenomenon that vibration usually sounds louder when the phone is placed on a flat stone surface.

Hard Backing Material Classifier. Base on the above analysis, we employ a simple threshold based classifier to distinguish different hard material. Concretely, after performing the acceleration-based recognition scheme, we further classify hard surfaces based on the acoustic features of their vibration pattern. In our case, if the amplitude of the peak frequency (around 200Hz) surpasses a pre-defined threshold, it is classified into stone surface, and wood material otherwise. The scheme easily extends to more kinds of hard material with multi-level thresholds and a bit of extra training.

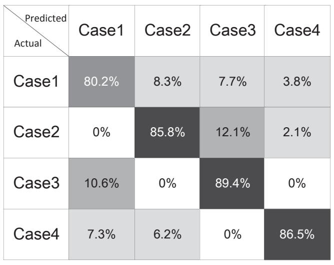
4 Evaluation

We prototype Sherlock as a daemon process that runs on Android smartphones. In this section, we detail the experiment methodology and results.

Prototype Implementation: We implement Sherlock on Android 4.0 Ice Cream Sandwich (ICS). The current version consists of about 2,150 lines of code and leverages LibSVM for phone local placement recognition. Once launched, Sherlock runs as a daemon process, providing upper layer applications with micro-environment sensing results. In the current version, we simply trigger Sherlock every 10 minutes, and we believe such interval is rigid to examine phone placement transitions.

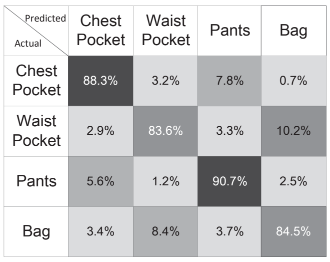
Experiment Device: We implement Sherlock APK on 3 different types of smartphones (Samsung Galaxy S2 I9100, Samsung Nexus3 I9250, Motorola MT788). All types of phones are equipped with the necessary sensors. The 3 types of smartphones all have 1GB RAM, with dual-core 1.2GHz, dual-core 1.5GHz, and single-core 2.0GHz processors, respectively. Since Sherlock is independent of platforms, we envision it to be easily extended to other mobile OS like WP8 and iOS.

Experiment Environment: we experiment with 8 volunteers (4 males and 4 females), collect sensory data from 15 scenarios, ranging from open football fields and square, to crowded supermarket and cafeteria, mainly



Actual \ Predicted	Case1	Case2	Case3	Case4
Case1	80.2%	8.3%	7.7%	3.8%
Case2	0%	85.8%	12.1%	2.1%
Case3	10.6%	0%	89.4%	0%
Case4	7.3%	6.2%	0%	86.5%

Fig. 14. Confusion matrix of in-hand states classification.



Actual \ Predicted	Chest Pocket	Waist Pocket	Pants	Bag
Chest Pocket	88.3%	3.2%	7.8%	0.7%
Waist Pocket	2.9%	83.6%	3.3%	10.2%
Pants	5.6%	1.2%	90.7%	2.5%
Bag	3.4%	8.4%	3.7%	84.5%

Fig. 15. Confusion matrix of on-body states classification.

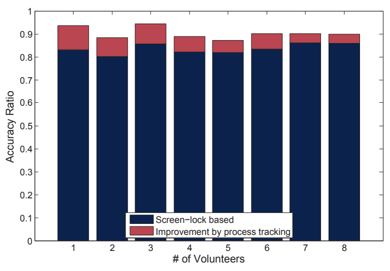


Fig. 16. An illustration of interaction detection accuracy.

TABLE 1
Backing material detection result.

Category	Wooden desk	Plastic table	Leather chair	Mattress	Stone stool	Glass table
Accuracy(Acceleration)	74.1%	70.2%	80.7%	79.4%	73.6%	75.5%
Accuracy(Acoustic)	83.2%	77.4%	\	\	86.2%	81.8%

in campus areas during the normal period from 7:00 to 23:00 in 3 weeks. The volunteers also record the ground truth using memo widget.

In what follows, we first evaluate the performance of each functional module, then measure the system overhead by trace-driven experiments.

4.1 Micro Benchmarks

4.1.1 Performance of Local Placement Detection

To evaluate the local placement detection module, we collect 8440 labeled data from 8 volunteers over 3 weeks.

Phone Posture Classification: We denote the 4 phone postures in Figure 6 as $Case1$, $Case2$, $Case3$ and $Case4$. Figure 14 details the confusion matrix of the classification result. In the confusion matrix, each column represents the instances in a predicted class, while each row represents the instances in an actual class. It is clear that the proposed local placement detection module achieves a remarkable accuracy (above 85%) for each of the 4 postures. Specifically, the detection accuracy peaks 90% in $Case3$, slightly higher than in $Case4$ (86.5%) and $Case3$ (85.8%). For the remaining case ($Case1$), however, the detection accuracy is not superior. Nevertheless, it still maintains at a reasonable level (80%+). On the other hand, Sherlock occasionally mistakes $Case1$, $Case2$, $Case3$ with each other. This might be because when holding a phone in hand, the user may unconsciously block the proximity sensor with his thumb or forefinger.

On-body Placement Classification: We further examine the performance of on-body placement classification for closed environments, i.e., in chest pocket, waist pocket, pants and bag. The corresponding confusion matrix is illustrated in Figure 15. On the whole, Sherlock achieves competitive classification result, with an accuracy of over 87% on average. More specifically, the classification accuracy for pants and chest pocket are 90.7% and 88.3%, respectively, much higher than that for the waist pocket and bag cases. This stems from the phenomenon that a person's arms often swing back and forth rhythmically while walking. This movement then drives the chest to shake continuously. The phone in chest pocket, as a result, will experience a significant repetitive pattern. In contrast, a phone in bag or waist pocket is ”too” faraway from arms or legs, resulting in less notable motion pattern, which exerts an adverse effect on the classification result.

4.1.2 Performance of Phone Interaction Detection

Figure 16 shows the detection accuracy of user-phone interaction. The dark blue bars correspond to the naive screen-lock based scheme. It achieves 82% detection accuracy on average. As for the hybrid approach where Sherlock also queries the process queue, the accuracy for all cases rises marginally, as the red bars indicate. Even the worst case still outperforms the best case in the simple screen-lock based detection scheme. Therefore, it clearly verifies that Sherlock is able to effectively distinguish interaction and non-interaction states.

4.1.3 Performance of Backing Material Detection

To examine the accuracy of backing material detection, 8 volunteers collect over 2000 acceleration/accoustic traces for different backing material, with 70% used for model training and 30% for testing. Table 1 lists the classification result on different material. In general, Sherlock successfully distinguishes hard/soft material with acceleration features alone, whereas it fails to tell hard material apart in a fine-grained fashion, i.e., with only 74.1% and 73.6% accuracies in Wooden material and Stone material detection, respectively. By adding acoustic features, the detection accuracy boosts to 83.2% for wooden material, 86.2% for stone material and 81.8% for glass material. Altogether, the chosen features prove to be sufficient for classifying different backing materials.

TABLE 2
Storage used by Sherlock.

# of Volunteers	1# (M)	2# (M)	3# (M)	4# (M)	5# (F)	6# (F)	7# (F)	8# (F)
Sensory data(Day 1)(in KB)	872	1132	695	1326	1554	1296	1441	1375
Environment Semantics(in KB)	1.65	1.92	1.47	1.98	2.01	1.94	2.01	1.95
Total(in KB)	873.65	1133.92	696.47	1327.98	1556.01	1297.94	1443.01	1376.95
Sensory data(Day 2)(in KB)	1032	1319	886	1057	1396	1449	1388	1526
Environment Semantics(in KB)	1.82	1.96	1.69	1.89	2.05	2.07	1.89	2.11
Total(in KB)	1033.82	1320.96	887.69	1058.89	1398.05	1451.07	1389.89	1528.11
Sensory data(Day 6)(in KB)	1120	1413	995	1337	1613	1579	1859	1769
Environment Semantics(in KB)	1.85	2.02	1.78	2.07	2.24	2.17	2.36	2.23
Total(in KB)	1121.85	1415.02	996.78	1339.07	1615.24	1581.17	**1861.36**	1771.23

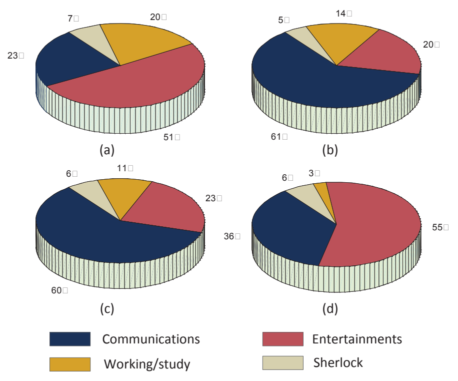


Fig. 17. An illustration of CPU share of Sherlock on different phones.

4.2 System Overhead

4.2.1 *CPU share of Sherlock*

We categorize the usage of phone into 4 groups, namely, communications, entertainments, working/study and Sherlock, and measure the CPU share of Sherlock on different phones. Figure 17 illustrates the CPU share of these four groups for 4 volunteers over a one-day study. According to the pie chart, the CPU share for communications, entertainments, working/study varies significantly from person to person, as the usage style of smartphones is highly related to the user's habits. For example, some people prefer truly a 'digital everywhere' lifestyle and are addicted to their phones for reading, shopping, gaming and enjoying music, while others would rather regard phones as a communication tool only. As a result, the CPU share of entertainments would take up the leading portion for the former groups whereas communication usages would contribute most to the CPU share for the latter. Despite such differences, the CPU share of Sherlock stays at stable share of around 6% for all the 4 volunteers. This indicates that our system incurs negligible CPU share to daily smartphone usage.

4.2.2 *Battery Consumption Overhead*

We also systematically measure the energy consumption of Sherlock. Figure 18 illustrates the lifetime of a battery with capacity of 1500mAh on a Samgsung Nexus 3 phone. We implement a battery tracing application to log the residual battery once a hour for a whole day. To fairly measure the energy consumption of Sherlock, we ask a volunteer to carry two Samgsung Nexus 3 phones. One is launching the Sherlock system and the other not. Then he is required to perform the same tasks on both phones for a whole day. The dark-blue bars illustrate the residual battery with Sherlock open, while the jacinth bars represent the energy savings without launching Sherlock. According to the bar chart, Sherlock consumes negligible energy of below 5% on average for most sampling periods. With time passing by, the cumulative energy consumption increases gradually, and finally ends up with 11.2%. We also notice two energy consumption fastigium during this 24 hour study, namely, 08:00∼09:00, 18:00∼19:00. Interestingly, these two periods accord with commuter time. Therefore it is reasonable to conclude that frequent human locomotion during commuter time triggers Sherlock to sense continuously, thus resulting in higher energy consumption. The energy cost, though, remains low as a whole, making Sherlock affordable for smartphone users, even with occasionally continuous detection.

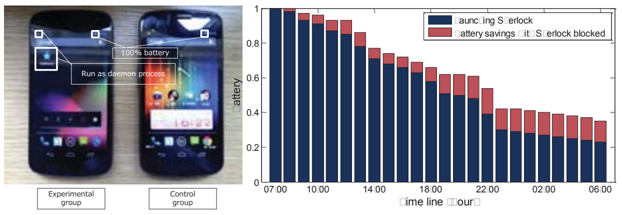


Fig. 18. An illustration of fine-grained battery usage.

4.2.3 *Storage Overhead*

We measured the storage overhead (sensory data and micro-environment semantics) by running Sherlock on a phone for 24 hours. Sherlock's overhead varies with the phone habits of different people, and thus we ran experiments with 8 volunteers (4 men and 4 women) for 1 week. Table 2 details the storage overhead for 3 randomly chosen days. As is shown, Sherlock consumes at most 1.8MB storage per day and 1.3MB on average. Recall that nowadays smartphones typically possess gigabyte of storage capacity, thus such megabyte storage overhead exerts little impact on smartphones. Further, the storage overhead is even smaller in reality, because Sherlock could delete the sensory data once the Micro-environment semantics have been deduced.

5 Related Work

Our concept of micro-environment sensing is built on both *context sensing* and *context-awareness applications*, yet differs in its emphasis on perceiving immediate surroundings from the smartphone's perspective. In this section, we broadly review the state-of-art in both threads of research.

Context Sensing: Recent advances in lightweight sensors on smartphones have spurred enormous efforts on context sensing in a round-the-clock fashion. SoundSense [5] models sound events on mobile phones to achieve context recognition. IODetector [9] provides an indoor/outdoor detection service via collaboration of phone sensors. Jigsaw [6] constructs a general-purposed pipeline-based engine for continuous sensing applications on mobile phones. By dynamically learning the relations among context attributes, ACE [2] reports users' current states to applications in an energy efficient way. Our work falls in this category yet differs in two aspects. On one hand, previous efforts are mainly human-centric, and support targeted computing services w.r.t users' situation. Conversely, Sherlock conducts environment sensing from the phone's perspective, automatically records sensor hints and characterize the surroundings of smartphones. On the other hand, all these works perform coarse-grained environment sensing (e.g., driving, walking, riding a bus etc.), while Sherlock aims to detect immediate surroundings, usually several to a dozen of centimeters, around a phone.

Context-aware Application: Vast works also study the usage of context-aware sensing results. FALCON [3] exploits temporal and spacial characters of user behaviors to pre-load apps to speedup launch time. TagSense [4] takes advantage of sensor hints to piece together environment information about photos. Nericell [11] leverages phone sensors to monitor road and traffic conditions in developing cities. Vtrack [12] constructs an accurate, energy-aware road traffic delay estimation using smartphones. Many research efforts have also utilized context sensing result for localization. SurroundSense [7] exploits phone-equipped sensors to characterize ambient environment features for logical localization. Zee [8] uses inertial sensors to track phone users indoors. These works, in general, can provide partial indication on immediate surroundings of smartphones. However, all of them are application-oriented, thus only suitable for specific scenarios. e.g., monitoring road conditions, localizing phone users indoors. However, Sherlock provides a multi-dimensional, phone-oriented environment sensing service for upper layer applications, and is orthogonal to the efforts aforementioned.

6 Conclusion

In this paper, we present the design, implementation and evaluation of Sherlock, a simple yet practical platform for micro-environment sensing for smartphones via collaboration among built-in sensors. The platform automatically collects sensor hints and characterizes the immediate surroundings of smartphones at centimeter level accuracy, providing fine-grained environment information to upper layer applications. We conduct comprehensive experiments to evaluate our system through a prototype implementation on Android platform. Preliminary experiment results show that Sherlock achieves low energy cost, rapid system deployment, and competitive sensing accuracy.

7 Acknowledge

This work is supported in part by the NSFC Major Program 61190110, NSFC under grant 61171067 and 61133016, National Basic Research Program of China (973) under grant No. 2012CB316200, and the NSFC Distinguished Young Scholars Program under Grant 61125202.

References

[1] J. Yang, S. Sdhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser and R. Martin, Detecting Driver Phone Use Leveraging Car Speakers. In *MOBICOM'11*, 2011.

[2] S. Nath. ACE: Exploiting Correlation for Energy-Efficient and Continuous Context Sensing. In *MobiSys'12*, 2012.

[3] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In *MobiSys'12*, 2012.

[4] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Tagsense: a smartphone-based approach to automatic image tagging. In *MobiSys'11*, 2011.

[5] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *MobiSys'09*, 2009.

[6] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *SenSys'10*, 2010.

[7] M. Azizyan, I. Constandache, and R. Choudhury. SurroundSense: Mobile phone localization via ambience fingerprinting. In *MOBICOM'09*, 2009.

[8] A. Rai, K. Chintalapudi, V. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *MOBICOM'12*, 2012.

[9] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. IODetector: A Generic Service for Indoor Outdoor Detection. In *SenSys'12*, 2012.

[10] X. Zhu, Q. Li, G. Chen. APT: Accurate Outdoor Pedestrian Tracking with Smartphones. In *INFOCOM'13*, 2013.

[11] P. Mohan, V. Padmanabhan, and R. Ramjee. Rich Monitoring of Roads and Traffic Using Mobile Smartphones. In *SenSys'08*, 2008.

[12] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys'09*, 2009.

[13] C. Tacconi, S. Mellone, L. Chiari. Smartphone-based applications for investigating falls and mobility. In *PervasiveHealth'11*, 2011.

[14] J. Dai, X. Bai, Z. Yang, Z. Shen, D. Xuan. PerFallD: A Pervasive Fall Detection System Using Mobile Phones. In *PervasiveHealth'10*, 2010.

[15] S. Salvador, P. Chan, Toward accurate dynamic time warping in linear time and space, In Journal Intelligent Data Analysis, 2007.

PLACE PHOTO HERE

Zheng Yang is currently an Assistant Professor in the School of Software of Tsinghua University. He received his B.E. degree in the Department of Computer Science from Tsinghua University, Beijing, China, and his Ph.D. degree in the Department of Computer Science and Engineering of Hong Kong University of Science and Technology. He is a member of IEEE and ACM. He has been awarded the 2011 State Natural Science Award (second class).

PLACE PHOTO HERE

Longfei Shangguan is currently a Ph.D candidate in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He received the B.E. degree in 2011 from Xidian University, and M.Phil degree in 2013 from Hong Kong University of Science and Technology. He is a student member of IEEE and ACM.

PLACE PHOTO HERE

Weixi Gu is currently a master student in the School of Software, Tsinghua University. He received the B.E. degree in 2012 from the Department of Information Security of Shanghai Jiaotong University. He is a student member of IEEE and ACM.

PLACE PHOTO HERE

Zimu Zhou is currently a Ph.D candidate in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He received his B.E. degree in 2011 from the Department of Electronic Engineering of Tsinghua University, Beijing, China. He is a student member of IEEE and ACM.

PLACE PHOTO HERE

Chenshu Wu is currently a Ph.D candidate in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received his B.E. degree in 2010 from School of Software, Tsinghua University, Beijing, China. He is a student member of IEEE and ACM.

PLACE PHOTO HERE

Yunhao Liu received the B.E. degree in automation from Tsinghua University, China, in 1995, the MS and PhD degrees in computer science and engineering from Michigan State University, in 2003 and 2004, respectively. He is now a professor at TNLIST, School of Software, Tsinghua University. His research interests include wireless sensor network, peer-to-peer computing, and pervasive computing. He is a senior member of the IEEE Computer Society and an ACM Distinguished Speaker.