Mobile Netw Appl (2013) 18:60–80
DOI 10.1007/s11036-012-0422-y

MobiSens: A Versatile Mobile Sensing Platform for Real-World Applications

Pang Wu · Jiang Zhu · Joy Ying Zhang

Published online: 29 November 2012

Abstract We present the design, implementation and evaluation of MobiSens, a versatile mobile sensing platform for a variety of real-life mobile sensing applications. MobiSens addresses common requirements of mobile sensing applications on power optimization, activity segmentation, recognition and annotation, interaction between mobile client and server, motivating users to provide activity labels with convenience and privacy concerns. After releasing three versions of MobiSens to the Android Market with evolving UI and increased functionalities, we have collected 13,993 h of data from 310 users over five months. We evaluate and compare the user experience and the *sensing efficiency* in each release. We show that the average number of activities annotated by a user increases from 0.6 to 6. This result indicates the activity auto-segmentation/recognition feature and certain UI design changes significantly improve the user experience and motivate users to use MobiSens more actively. Based on the MobiSens platform, we have developed a range of mobile sensing applications including Mobile Lifelogger, SensCare for assisted living, Ground Reporting for soldiers to share their positions and actions horizontally and vertically, and CMU SenSec, a behavior-driven mobile Security system.

P. Wu (✉) · J. Zhu · J. Y. Zhang
Carnegie Mellon University, Silicon Valley Campus,
NASA Research Park, Bldg. 23 (MS 23-11), P.O. Box 1,
Moffett Field, CA 94035, USA
e-mail: pang.wu@sv.cmu.edu

J. Zhu
e-mail: jiang.zhu@sv.cmu.edu

J. Y. Zhang
e-mail: joy.zhang@sv.cmu.edu

Keywords Mobile sensing · Activity recognition · Lifelogger · Anomaly detection · Human behavior modeling

1 Introduction

Today's smart phones come equipped with a rich range of sensors including GPS, accelerometers, WiFi, Bluetooth, NFC, microphone etc. Combined, this contextual information can tell us a great deal about a user's current activity: what is the user doing at which location and for how long. When logged, this data can provide important information about the user's behavior patterns. Caregivers can design effective and personalized plans to improve the user's health and well-beings. If we aggregate this kind of information across thousands of volunteers in a city, it can also tell us a great deal about that city, for example, average waiting times for buses, commuting patterns using public transportation systems, how public and private places are used, and so on. This kind of data collection and analysis offers a way to understand human behavior at large scale, which can have positive impact in a number of domains, including health care, traffic planning, urban design, and social network analysis.

Smart phones are more affordable to most of the users and are less intrusive compared to other sensing approaches such as "Smart Home" [22, 26, 28] and wearable sensing platforms [15, 20, 27]. They are easy to carry, free with sensor deployment, and commonly accepted by many users. Smart phones have built-in network connections for data transmission and have large local storage to save sensor data on-device. Smart phone operating systems such as iOS and Android support device independent hardware/OS resource abstraction which make it much easier for developers to write applications for different devices as compared to developing applications for device-specific embedded systems. Given these merits of smart phones, we consider smart phones ideal sensing platforms for human behavior/social research.

In this paper, we present MobiSens, a full-blown mobile sensing platform designed for a range of real-world mobile sensing applications using novel algorithms and user interface. MobiSens addresses common challenges in mobile sensing including robust sensor data sampling, power optimization, indexing and understanding the sensor time-series and interaction with users.

After releasing MobiSens on Android Market for five months, we have collected 13,993 h of sensor data submitted from 310 users for the mobile lifelogger project. Several other research projects have been using MobiSens platform to collect users' data to develop behavior-based anomaly detection [33], future activity prediction [19] and remote elderly care [30] and behavior-driven passive authentication [32].

The rest of the paper is organized as below. In Section 2, we describe several mobile sensing applications of different kind and discuss their common requirements. We analyze the challenges in mobile sensing (Section 3) and present the design of the MobiSens platform in Section 4. In Section 6, key features of the MobiSens platform are evaluated. We conclude the paper in Section 8 after reviewing related work in Section 7.

2 Mobile sensing applications

Table 1 shows a list of mobile sensing applications we have developed based on the MobiSens platform. These applications differ in their functions yet share common requirements.

2.1 Lifelogger

CMU's mobile lifelogging system [5] constantly records sensor readings from the mobile phone users carry at all time to "log" a user's daily life. Sensor data is uploaded to the lifelogger server, where information is indexed and processed so that the user can browse, search and summarize his/her daily activities from the web. On the mobile side, activities are automatically segmented so that users can annotate the segmented activities. We also developed a lightweight adaptive activity recognition algorithm running on the mobile device to label segmented activities.

A key challenge in developing a real-world mobile lifelogging system is that we can not *foresee* activities users need to log their lives. Instead of several predefined low-level activities such "sitting", "walking" and "running", users have a wild range of activities that they may find useful to their lifelogs. For example, a user might want to find information from his/her lifelog such as "How much time did I spend on grocery shopping last month?" and "Where did I park my car in the garage 5 days ago at the airport?" To answer these questions, lifelogging systems need to be able to learn and recognize user-defined activities based on (usually) very few labeled instances as training data. We developed a novel *adaptive activity recognition* algorithm in MobiSens to incrementally train activity recognition models through online learning.

On the server side, sensor data is further processed with computationally intense algorithms including the hierarchical activity segmentation and hierarchical activity recognition. Figure 1 shows an example of a user's lifelog. For that particular day, this user drove from home to a small airport and took a 30 min discovery flying lesson and then drove to work. The Lifelogger service applies the hierarchical segmentation on the time-series sensor data and creates a nested calendar view. The activity labels are either from automatic activity recognition such as "driving", or annotated by the user, such as "Flying".

When the user clicks on the sub-event of "Flying" (10:37–11:08, marked as $*$ in the figure), the map displays the corresponding GPS-trace. The "Flying" event is further segmented into 1: "taxing", 2: "ascending leg" and 3: "descending leg" which the user can click on to review (Fig. 2).

Table 1 Characteristics of different mobile sensing applications

Application	Server connection	Storing raw sensor data	Activity annot./recog.	Activity sharing	Msg. from server
Lifelogger	Yes	Yes	Yes	No	No
Senior care	Yes	Yes	Yes	Yes	Yes
Mental health	Yes	No	Yes	No	Yes
Ground reporting	Yes	Yes	Yes	Yes	No
Behavior-driven security	No	No	No	No	No
Social sensing	Yes	No	Yes	No	No

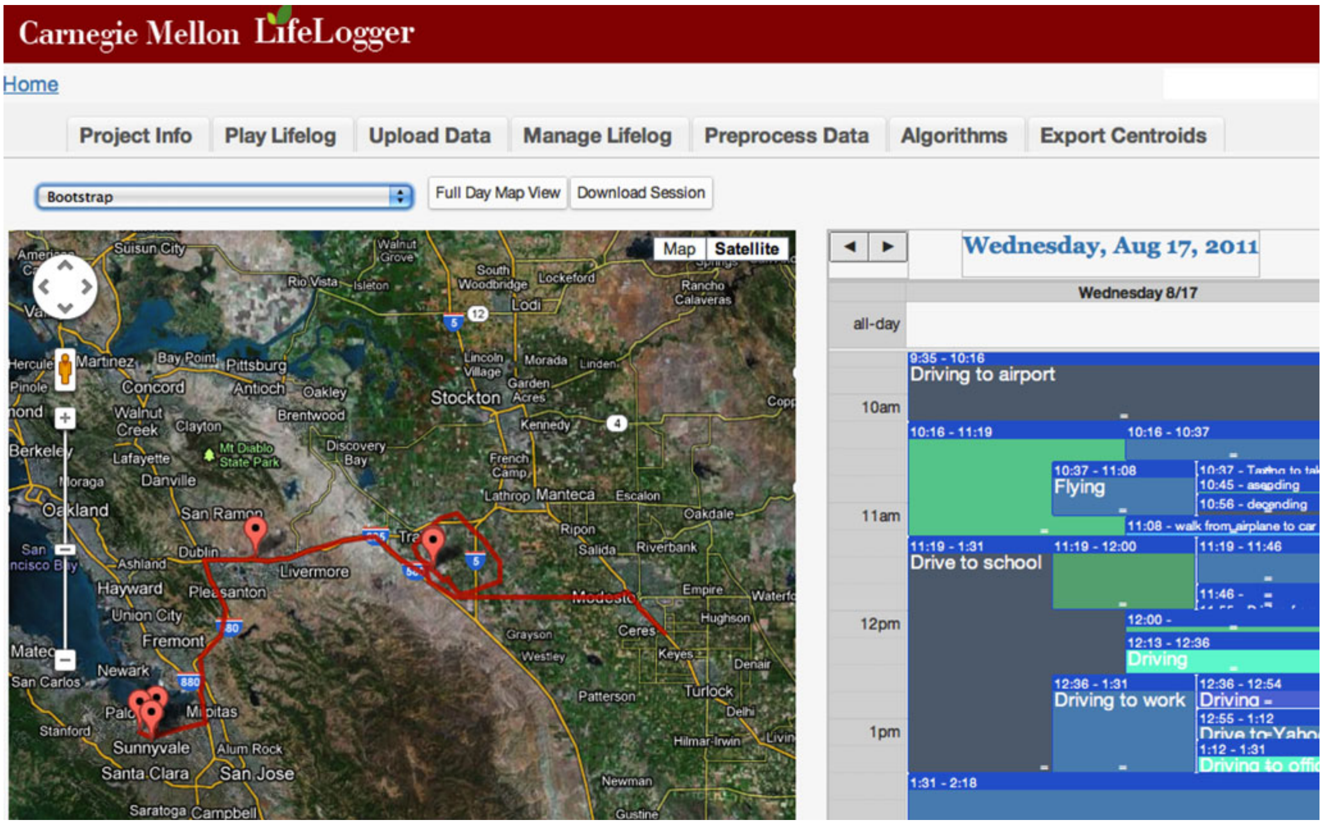


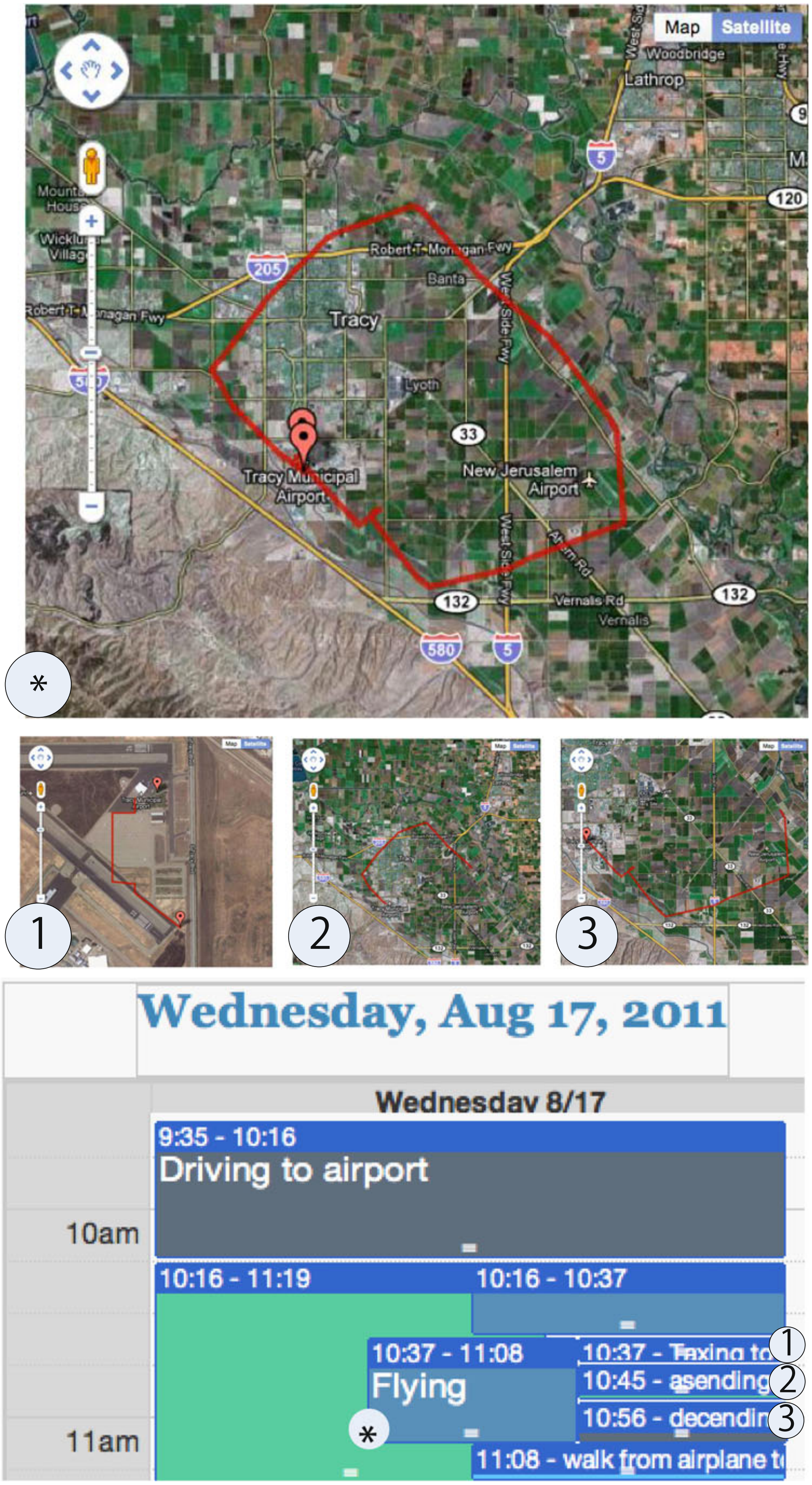
Fig. 1 The web interface of the CMU lifelogger system. Shown here is an example of a user's one day event of "driving from home to a small airport and taking discovery flying lesson and then drive to work." The whole event is segmented by an unsupervised activity segmentation algorithm into a nested calendar tree shown in the *right*. Annotations of the segmented activities are added by the automatic activity recognizer trained from user's previous annotated activities. Users can correct the automatic activity recognition label and provide labels to those unknown to the system. Figure 2 shows three examples when the user clicks on one of the sub-activities in the event

2.2 Senior care through mobile sensing

In 2008, the first wave of baby-boomers have reached their retirement age. Increasing number of seniors choose to age-in-place by living in their own homes with caregivers visiting them on a regular basis. The SensCare system [30] uses smart phones carried by the elderly to sense their activities and (1) to detect anomaly such as accidental falls or getting lost due to Alzheimer disease. When such an anomaly is detected, the system will notify the caregivers or family members for prompt response; (2) to display the current and past activities of the elderly so that authorized users such as caregivers and remote family members are informed on what the elderly is doing; and (3) to generate activity summary of the elderly for doctors to diagnosis the health/well-being issues.

2.3 Mental health monitor and intervention

The 2005 National Co-morbidity Survey-Replication study reports that lifetime prevalence estimates for mood disorders is around 20.8 % in the population aged 18 or older; including, but not limited to, major depression and dysthymia [14]. In particular, an estimated 300,000 veterans among the nearly 1.7 million who have served in Iraq and Afghanistan are battling depression or post-traumatic stress disorder. More than half of these people, are slipping through the cracks in the bureaucratic system, going without necessary treatment according to the study by RAND [1]. One of the main drawbacks in the current psychiatric clinic practice is that doctors' diagnoses are based on the interview during the patient's office visit. Mental issues, however, require constantly monitoring to understand their causes and to predict potential high risk of episodes. With the constant sensing capabilities provided by mobile phones, we are developing a mobile system that constantly monitors the user's activities and his/her behavior to predict the risk of psychological disorder. Working with the doctors, the system can also intervene the user's behavior by sending messages to user's mobile device with suggestions such as "please consider go take a walk with your family instead of going to the bar tonight."



Wednesday, Aug 17, 2011

Wednesday 8/17

10am

11am

9:35 - 10:16
Driving to airport

10:16 - 11:19

10:16 - 10:37

10:37 - 11:08
Flying

10:37 - Taxing to

10:45 - asending

10:56 - decendin

11:08 - walk from airplane t

Fig. 2 When the user clicks on the subevent "Flying[10:37–11:08]" (*) which corresponds to the activity when the user got on the airplane, the geo-trace is updated on the map. The "flying" event is further segmented into three sub-events: (*1*) "Taxing to take off [10:37]". (*2*) "Ascending leg [10:45]"; and (*3*) "Descending leg [10:56]"

2.4 Ground reporting

In battle fields and similarly fire fighting scenes, police raids and disaster response efforts, soldiers, firefighters, policemen and rescuers more or less battle in the fog without knowing their colleagues' locations and activities. Though modern communication and collaboration technologies allow the team members to talk to each other and even collaborate on a common operating map, the cognitive load required for them to concentrate on their tasks usually prohibits them to report their positions and current activities. The ground reporting system uses mobile sensing techniques to detect the user's current position and activity and share the information horizontally to his team members and vertically to the commander. Logged information of the mission can later be used for debriefing after the mission is over and for work-related knowledge extraction to transfer field experience to others.

2.5 Behavior-driven mobile security

Most users carry their mobile phones at all time. With mobile phones sensing their locations and activities, we can build the user's behavior model which captures the regularity of a user's behavior. Similar to biometrics, such behavior model is user-dependent and can be used as a passive authentication method. In our past work, we build a user's mobility model from her outdoor GPS-trace [4], indoor Wi-Fi trace [33] or gripping pattern [32]. We investigated and evaluated a number of inexpensive and easily acquired passive factors, rigorously examined how well these factors differentiate between people including for example motion, location, running applications, etc. We developed methods for fusing these passive factors and model people's behavior in a manner that is effective, robust, and reliable. Our experiments show that this mobile security system can archive 70∼80 % accuracy in user identification using the passive sensors on the mobile device with only a few days of training data.

We also observed that different applications on a mobile device may have different sensitivities towards the threats due to loss or being stolen. One-thing-for-all approach in authentication schemes may be either too loose for some applications, which expose them to risks, or too tight for others, which causes usability problems. Using the sensory data collected on the mobile device, we can build user behavior models to reflect the contexts under which the mobile device is used, including motion, location, running applications, etc. From these contexts, the device can calculate *certainty score* that the system is *at risk*. A security application on the mobile device constantly monitors the certainty score and compares it with the *sensitivity* levels for different mobile applications. Once the system is deemed "uncertain" for the mobile application's "sensitivity", SenSec activates different active authentication mechanisms such as password verification, face recognition, voice recognition or finger print scanning to verify user's identity before an application is executed.

2.6 Social sensing

If we can aggregate the mobile sensing information across thousands of participants in a city, we can infer the *behavior* of that city, for example, where do people live and work, how they commute from home to work, average time spent waiting for buses, how public and private places are used, what residents typically do, and so on. This kind of data collection and analysis offers ways to understand human

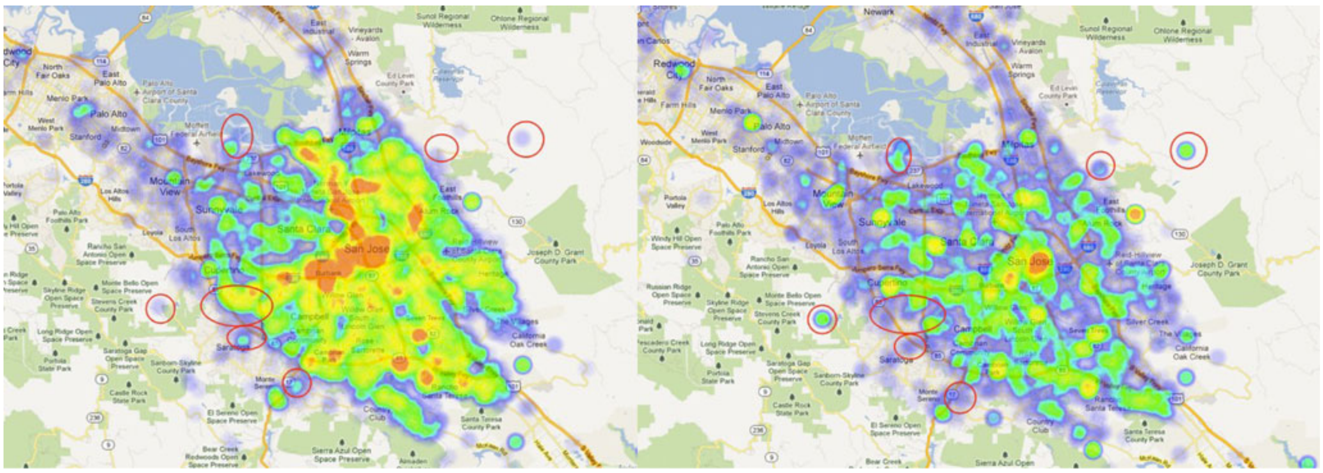


Fig. 3 The aggregated check-in "heat map" of Silicon Valley. The *left hand side figure* shows the population of southern bay area in working hours while the *right hand side figure* shows the population of the same area after work. *Red circles* highlighted the residential regions (places with small population during working hours but larger population after working hours)

behavior at large scale, which can have positive impact in a number of domains, including health care, traffic planning, urban design, and social network analysis.

For social sensing, we do not need the raw sensor information from each individual user. The aggregated information (e.g., Fig. 3) such as the accumulated geo-trace *n*-gram frequency suffices as sufficient statistics to infer people's social behavior. Activity recognitions and annotations provided by users are also helpful to the social sensing application because it is impossible for system developers to foresee all activities that would be of interest to the user and the social sensing problem such as "shopping at Target" and "take kid to baseball practice."

3 Challenges in mobile sensing

Though collecting sensor information may sound trivial, developing a reliable mobile sensing platform that can be used for various mobile sensing applications in real world settings is challenging:

- **Low-level activity recognition vs. high-level activity understanding**. The majority of the existing activity recognition work is to train classifiers to recognize primitive human activities such as "walking", "running" or "sleeping" whereas in social sensing we are more interested in high-level activities such as "grocery shopping" or "pick up kids from school".
- **Ad-hoc vs. generalized approaches**. Training a classifier to recognize predefined activities from pre-selected sensors in an ad-hoc fashion limits the flexibility of the sensing platform. The fundamental problem with the ad-hoc approach is that data representation of the sensor data and processing algorithms are entangled. We envision that by separating the data representation from processing algorithms, mobile sensing can become a more generalized framework on top of which sensing applications can be developed.
- **Lacking of labeled data**. For some mobile applications such as anomaly detection, there is no need to annotate the recognized activities with meaningful labels such as "walk in the park" or "shopping". For many other applications such as mental health monitoring, however, we do need meaningful labels to interpret the sensed results. These applications do require training data with annotation. Annotating sensory data is time consuming and tedious and humans are not good at recalling what happened to them if no other contextual information is provided in parallel. Certain incentives need to be provided to motivate users to annotate their activities.
- **Sensibility of mobile sensors**. The sensors available on smart phones may not have enough information to differentiate activities such as "driving" vs. "in a car as passenger". The sensibility of mobile sensors also depend on the positions where phones are stowed, e.g., phones in hand bags vs. in pockets or mounted on arms show different sensitivities to human activities.
- **Uncontrolled data collection process**. When deployed, we have very limited control on how the system is used by real users. One such example is the mounting position of the mobile device. In controlled environment like laboratories, we can ask all participants to stow mobile phones in their pockets where as in real life a mobile phone can be placed at any positions: pocket, handbag, held in hand or laid on table.
- **Privacy**. As mobile sensing collects a lot of data from the user, privacy is a legitimate concern. In some situations such as in senior care, users may sacrifice some degree of privacy for the benefits of being monitored constantly. To general public, however, a mobile sensing system without convincing privacy protection will not be popular among users.
- **Power consumption**. If all sensors are turned on, the battery of a typical smart phone can be drained in just a few hours, mainly due to the high frequency of CPU being activated to sample sensors. Such short battery life will drive all users away from using the mobile sensing applications. Due to our experience, we can reduce the sampling rate of certain sensors to achieve power saving when they are not a significant data modality for the current activity. For example, if we found a user staying in one location for a certain period of time, we can slow down the GPS sampling rate or even turn GPS off. We can rely on other more power efficient sensors like accelerometer and WIFI to detect the possible activity change and reboot the GPS sampling.
- **Non-disturbance and robustness**. The mobile sensing application should run quietly in the background and does not affect users' normal usage of the phone function. It should also be robust to handle different situations such as no data connection, low storage space, phone rebooting etc. without affecting phone's normal operation.

4 MobiSens platform

The MobiSens platform is a client/server system consisting of two major components: an Android mobile Application and a two-tier back-end system. Figure 4 illustrates the system architecture.

4.1 Mobile client framework

The MobiSens mobile client has four components: Data Widgets, Data Aggregator, Device Controller and Activity

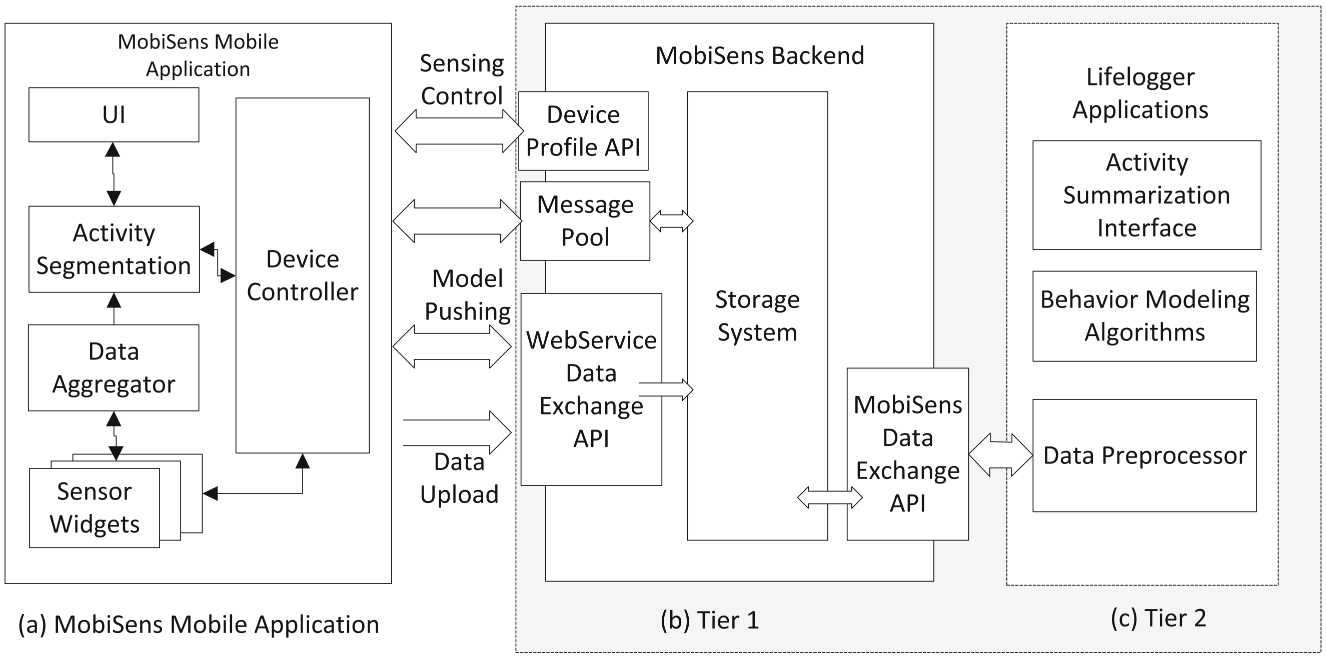


Fig. 4 MobiSens mobile application and back-end system architecture. MobiSens is a client/server system with Android App as client and two-tier server systems. Mobile client collects various sensor data from users' phones, apply activity segmentation, lightweight adaptive activity recognition and directly interacts with the user. Sensor data is uploaded to the MobiSens server to index and process the sensor data using heavy-weight algorithms. Second-tier (application and service) servers use information processed by MobiSens server to provide application specific services such as lifelogging, senior care and ground reporting etc

Recognition Module. The client is designed as data-centric and message-driven, i.e., components are both message receivers and broadcasters. They register as listeners of a certain set of messages and then process data passed with these messages and broadcast the processing result as new messages.

4.1.1 Data widgets

In MobiSens, we consider all information sources as "sensors". Sensors include "hardware sensors" such as accelerometers and GPS, and "soft sensors" such as time, date, calendar, phone charging status, ringtone settings etc. Data Widgets are collectors of sensory data. Data Widgets collect raw sensor readings (Table 2) and broadcast the information to data consumer components using Android's inter process communication mechanism (IPC) if the sensor sampling rate is low or using RAM directly for high-frequency sensors such as the accelerometer, magnetometer and gyroscope.

4.1.2 Data aggregator

Mobile devices are not always connected to the Internet. We can not stream the sampled sensor data to the MobiSens Backend at all time. MobiSens first stores all the data locally and upload them once the network connection is available and the phone is being charged. Two components on the client side manage the data storage and upload process:

Table 2 Examples of sensor data collected by MobiSens client

Sensors	Information sensed
3-axis accelerometer	Motion
Magnetometer	Azimuth value of heading direction
GPS coordinates	Outdoor locations and trace
Rotation matrix	Orientation of the phone
Ambient light level	Lighting level
Shuffled sound recording	Environmental acoustic features
Charging state	Whether the phone is being charged or not
Temperature sensor	Environment temperature
WiFi RSS	Indoor location and WiFi provider
Nearby bluetooth IDs	Whether the user is in a crowded place
Battery level	Power consumption of the device
Process list	Which application/service is running
Network stats	Network traffic to/from applications

All these sensory data are recorded with time-stamps. For *rotation matrix*, we used Motorola Droids which do not have gyroscope sensors. The orientation of the phone is estimated by the Android SDK based on the accelerometer readings and the gravity g-value

– Data Aggregator: a local data storage manager. Data Aggregator receives all raw readings from different Data Widgets and saves them to multiple files. It also

pushes the raw data file queue to Data Uploader and notifies the Uploader which files are ready to be uploaded.
– Data Uploader uploads and removes the uploaded file to free local storage space for incoming sensor data.

4.1.3 Sensing profile pulling

There are hundreds of parameters controlling the mobile client's behavior. For example, the sampling interval and sampling rates of each sensor impact the sensitivity of the activity recognition and the battery life of the whole system. Different mobile sensing applications and different mobile devices require different sensing profile. For instance, mobile lifelogging systems need to run at least 12 h without charging for users who leave home in the morning and come back from work in the late afternoon. For behavior-driven security system (Section 2.5) using user's gripping pattern, the sensing only needs to be activated for a few seconds after the screen is unlocked.

On the other hand, different Android devices have different CPU frequencies (which significantly impact the power consumption rates) and battery capacities, MobiSens clients need to adjust their sampling profiles to ensure the desired battery life on different devices. To avoid publishing new versions of MobiSens on Android Market each time when we want to adjust the client's parameter settings, MobiSens mobile client has a feature called *profile pulling* which dynamically updates its configurations from the backend server. As the core component of this functionality, Device Controller periodically *pulls* configuration profile from the backend MobiSens server, including list of sensors need to be sampled, sensor sampling rate and sampling strategy, data push interval, etc.

Client profile can be customized for individual devices by specifying different configuration files. This allows the system to adjust the sensing configuration for each user or each type of devices if needed.

4.1.4 Activity recognition module

Activity Recognition Module monitors process events from Data Aggregator and performs simple activity modeling task, such as motion-based activity segmentation and recognition. The recognized activities are shown on the mobile client for the user to view and annotate. Details of activity recognition is discussed in Section 5.

4.2 MobiSens backend and mobile sensing application servers

The two-tier back-end server contains the first-tier MobiSens Backend and the second-tier Mobile Sensing Application servers. MobiSens Backend communicates with mobile devices directly and provides three key functions: device control, data storage and data access management.

MobiSens Backend receives the sensor feature data pushed/uploaded from mobile devices and stores them in a file system. The data can then be fed via MobiSens Data Exchange API to Mobile Sensing Application servers to derive and infer meanings for sensing applications, such as lifelogger activity summarization and social behavior aggregation. Results are pushed back to MobiSens Backend which can be accessed by mobile devices such as for users to check their activity summary on their smart phones.

Mobile Sensing Application Servers access data processed by MobiSens Backend and provide high-level services with new analytical components. The actual functionalities of *lifelogger* service and *social sensing* etc. are provided by these servers.

5 Activity segmentation, recognition & annotation

During the past half decade, much efforts have been devoted to sensor-based human activity understanding. The majority of activity understanding work is based on supervised learning [2, 6]. Supervised learning builds statistical models from a large amount of labeled data and apply the model to classify unseen activities. In reality, labels for real-life human activities are very difficult to obtain. Very few users have the motivation to label their daily life at fine granularities with starting/ending time and descriptions.

Most past research is done in lab-settings by recruiting participants to "perform" some predefined activities such as "walking", "sitting" in order to create the labeled training data set. Such lab-setting supervised activity recognition systems is not practical for real-life applications:

1. It is impossible to enumerate all possible activities that are of interest to end users.
2. Supervised activity recognition can not answer structured questions such as "What costs me most of the time this morning" or "How many hours did I spend on the road to grocery stores last month?"
3. Activities are personal to users. One user's activity of "going to work" means " driving 2 h from home to work" while for another user it means "taking bus then subway for 30 min" to office.

Observing all the problems described above, we propose a different framework of learning, training and recognizing users' activities called *adaptive activity recognition* to make activity recognition practical for real mobile sensing applications. Adaptive activity recognition builds activity models through online learning and incorporates users' input and feedback to adapt the model over time.

As shown in Fig. 5, adaptive activity recognition works as the following:

1. Heterogeneous sensors data are sampled from mobile sensors.
2. Applying the unsupervised activity segmentation algorithm on the sensor streams to chop the data into activity segments.
3. Users see their activities segmented and recognized on the mobile device. They can correct activity labels if they are wrongly recognized by the system.
4. Newly labeled activities from user are used by the system to (1) update the known activity database and (2) adjust the modality weights for each activity type for more effective sensor fusion in the future.

The following subsections describe the unsupervised activity segmentation algorithm and the adaptive activity recognition based on our early work [30].

5.1 Unsupervised activity segmentation based on motion

Activity segments are the basis units to train activity recognition models. MobiSens first segment the sensor time-series into a sequence of activities in a unsupervised manner without any prior knowledge of users' activities, The automatically generated activity segments reduce the cognitive load of users as it is very challenging for a user to remember the start and end time of an activity but relatively easy to recall what they did during a time interval.

MobiSens uses motion information for activity segmentation. Intuitively, when one changes his/her activities, motion usually changes [16]. Motion is also the most distinguishing modality for most of the human activities. Table 3 shows the activity recognition accuracies using single-modality. Using motion-only information, we can achieve 72 %, much higher than other modalities such as location (GPS) (Table 4).

MobiSens' activity segmentation algorithm operates on the motion sensor (accelerometers) time-series.

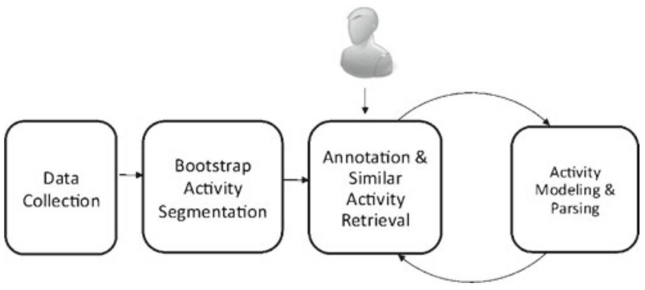

Data Collection
Bootstrap Activity Segmentation
Annotation & Similar Activity Retrieval
Activity Modeling & Parsing

Fig. 5 The input–output flow of adaptive activity recognition. The system starts with no training data and learning everything incrementally based on users' annotation and feedback

Table 3 Sensors and their sampling rate used for the single-sensor classification experiment

Sensor	Sampling rate
Accelerometer, ambient light	20 Hz
GPS, speed	Every 2 min

5.2 Behavior text representation

Similar to the approach used by Bao et al. [2], we apply a sliding window (width = 24 samples, step size = 6 samples) over the accelerometer time-series and extract motion features from each window to represent the motion characteristics at each time. With the sampling rate at 4 Hz, each window corresponds to 6 seconds. We extract 6 features from each window including the *mean* and *standard deviation* of each of the three axises of accelerometers. During the offline training process, motion feature vectors from many users' data are clustered into V classes using K-Means clustering. During the runtime, we map an incoming motion feature vector to its closest class and use the class label to represent this motion feature vector for the follow up process. The input accelerometer reading stream is thus converted to a sequence of cluster labels which we refer as *"behavior text"* in later sections.

Needless to say, such clustering and quantization process loses information and precision compared to using the raw motion feature vectors. The benefit of discretizing feature vectors to a symbol is the convenience to model the sequential and structural patterns in the motion time-series.

Number of clusters V is the *vocabulary size* of the behavior text. The selection of V affects the overall system performance. If V is too small, quantization loses too much information which decreases the segmentation accuracy. If V is too large, the symbolic representation loses generalization and it is difficult to capture the patterns in activities. Also, the quantization process needs more computation to calculate the distance between the input feature vector with each of the V centroids and drains the battery fast. In a working system, V needs to be manually optimized based on the

Table 4 Activity classification accuracy using single sensor modality

Data modality	Accuracy
Accelerometer (motion)	0.72
GPS (location)	0.41
Speed	0.32
Light	0.17

J48 decision tree is used in this example. Motion is the most informative single sensor in this experiment system requirements, i.e. the minimum granularity of activity and battery life. In our experiments, we set $V = 200$ which can balance the power consumption and segmentation accuracy. The detail of parameter selection is discussed in Section 6.3.

5.3 Change of activities

When a user changes his/her activity at time t, there should be a significant change from behavior text string $[t - w, t - 1]$ to string $[t, t + w]$. Here w is the window size that controls the granularity of detected activities. We refer w as *segmentation window size*.

To compare the similarity between two behavior strings, we use the classic Vector Space Model (VSM) [24]. Each behavior string is represented by a vector over the whole vocabulary $\mathbf{b} = (b_1, b_2, \ldots, b_i, \ldots, b_V)$ where b_i is the weight of word i (term-weight) in documents. Here we want to partially model the temporal patterns between symbols. In addition to words, we also consider consecutive words (n-grams) in the string. Each behavior string is then represented as a vector in the n-gram space and b_i is the normalized frequency over 1-gram, 2-gram and up to N-gram.

The similarity of two behavior text vectors $\mathbf{b}$, $\mathbf{q}$ is calculated by their cosine distance:

$$S(\mathbf{b}, \mathbf{q}) = \frac{\mathbf{b} \cdot \mathbf{q}}{||\mathbf{b}||||\mathbf{q}||} \tag{1}$$

where $S(\mathbf{b}, \mathbf{q})$ (Eq. 1) is the n-gram similarity between string B and Q. The lower the value of $S(\mathbf{b}, \mathbf{q})$, the more likely the user changes his/her activity in $[t - w, t + w]$. A threshold Θ was defined to determine the activity change. For single level segmentation on the smartphone, we empirically set $\Theta = 0.8$.

5.4 Hierarchical activity segmentation

An advantage of using this activity segmentation framework is that the window size w can be used to control the granularity of segmented activities. Since the length of activities varies in real life, we can use different window sizes to generate hierarchical activity segmentation: After getting the segmentations in level one with a large segmentation window size, a smaller window will be used to identify fine-grain activities recursively on each of the generated segment (Fig. 6). The hierarchical activity segmentation will enable the user to view and annotate activities in different granularities, he/she can "zoom" in or out to change the granularity of activities.

In our experiment, we implemented a four level hierarchical activity segmentation using segmentation window sizes of 30, 20, 10 and 5 min.

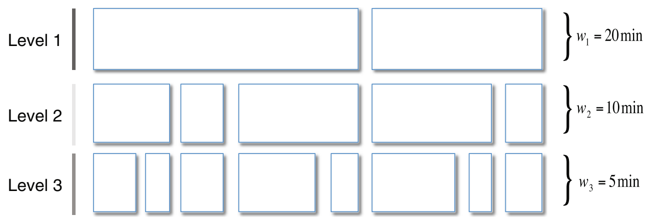


Fig. 6 An example of 3-level top-down hierarchical activity segmentation with $w_1 = 20$ min, $w_1 = 10$ min and $w_1 = 5$ min

We discover that the activity change "peaks" (similarity lower than average) identified by larger segmentation windows are also peaks identified by smaller segmentation windows but not vice versa; and activity changes over larger windows are smoother than smaller windows so setting one threshold for all levels is not a good approach. Instead of setting a hard threshold, we obtain threshold T_l for level l dynamically using the following equation:

$$\Theta_l = 0.9 \cdot (\max(\{S\}) - \min(\{S\})) + \min(\{S\}) \tag{2}$$

where $\{S\}$ are the similarities obtain from two neighboring segmentation windows of level l.

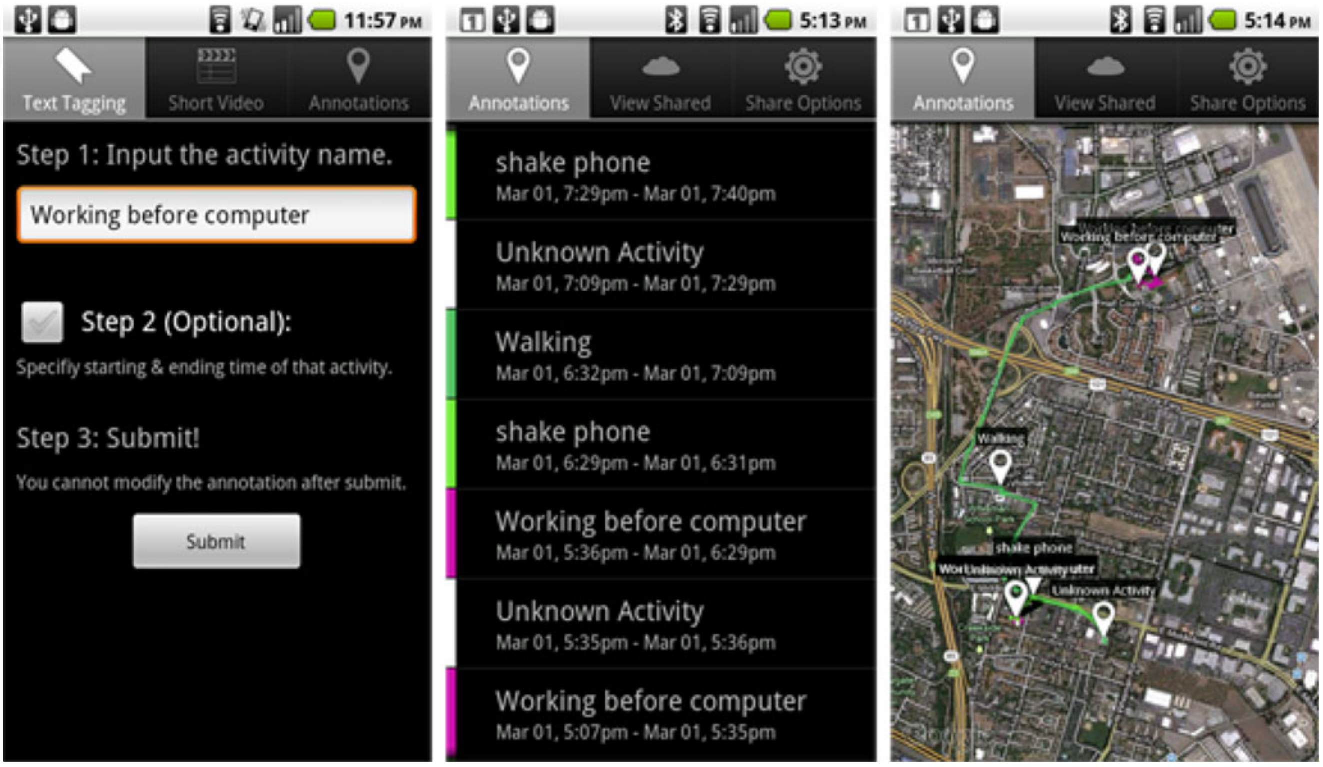
5.5 User activity annotation

One of the key challenges with smartphone-based mobile sensing is how to motivate the user to provide annotations of their activities. The traditional experiment method utilizes video and audio to help subjects or researchers recall the captured activities and label event boundaries. This method not only requires extra devices and power to store the video/audio and also relies heavily on human effort which raises many practical issues when sensing applications are targeted at real world users.

In our study, we found that besides incentives such as micro payment, *cognitive load* and *user-interfaces* are critical to users' involvement and participation. We have explored different approaches and will discuss them in the following sections.

5.5.1 Activity annotation through self-report

Our first attempt for scalable activity annotation is based on a naive self-report approach (Fig. 7). In the self-report experiment, MobiSens App pops up a text box for user to input activity name, and a time picker to select the starting and ending time of the activity. There is also menu to record a short video, in case the user choose to annotate the activity in the future and will use video/audio to help them to recall what has happened then.



Step 1: Input the activity name.

Working before computer

Step 2 (Optional):

Specify starting & ending time of that activity.

Step 3: Submit!

You cannot modify the annotation after submit.

Submit

shake phone
Mar 01, 7:29pm - Mar 01, 7:40pm

Unknown Activity
Mar 01, 7:09pm - Mar 01, 7:29pm

Walking
Mar 01, 6:32pm - Mar 01, 7:09pm

shake phone
Mar 01, 6:29pm - Mar 01, 6:31pm

Working before computer
Mar 01, 5:36pm - Mar 01, 6:29pm

Unknown Activity
Mar 01, 5:35pm - Mar 01, 5:36pm

Working before computer
Mar 01, 5:07pm - Mar 01, 5:35pm

Fig. 7 UI and design evolution. From *left to right*: (*1*) The self-report interface of release I. (*2*) Activity detection & summarize in Release II. (*3*) Activity annotation, the map was used to help users to recall the activity

We had hoped the self-reporting scheme can scale to make MobiSens a practical mobile sensing platform. After the release on Android Market, 6 people installed the application and uploaded data in the first week. Five of them provided 21 labels in total, which is 0.6 labels per-person-per-day. None of these users provided any short video for their captured activity which indicates that using video and audio recording as a way to provide annotation or annotation reminder is not practical for large-scale social sensing.

We analyzed the boundaries and the submitting time of each annotation. Although the data set is small with only 21 labeled activities, the result still shows some interesting trends. We found that 71 % (15 out of 21) of the annotations were done while the labeled event was actually happening, such as "Working before computer", "Meeting with boss", "Shopping at Target" etc. 29 % of the activities (6 out of 21) were labeled shortly after they had ended, for example, "Driving", "Having lunch with friends". Some activities were always annotated later, like "Driving". To our surprise, none of the activities were labeled before the user actually started executing the activity.

We also studied the nature of annotated activities. All of the labels contain verbs, e.g., "Walking", "Meeting". Four labels (19 %) mentioned the activities' location as well as the action: "Walking around house", "Shopping at Target", "Shopping at grocery store" and "Driving to office". Though we though there would be location-only labels such as "at home", none of the submitted annotations are location-only.

Another interesting finding is that users seldom annotate the same activity more than once. Only 4 out of 21 labels are duplicate labels. Most of them were activities that last for more than 30 min, like "Meeting" and "Working before computer". We hypothesize that users have the perception that the system should be intelligent enough to learn from the first annotation. One user's feedback is that: "I have told the system already when I was walking, why do I have to teach the system again? Isn't it suppose to learn that already?"

Based on the analysis from the self-report experiment, we believe that people are good at reporting what they are *doing* rather than recalling what they *did*. On the other hand, location information was used by people to describe some activities which means the social sensing systems do need to incorporate the location information for automatic activity recognition. The third information delivered is that our data collection platform should be "smart" enough to recognize similar activities. Users will lose their interests in the application if it keeps asking them to label the same thing again.

5.5.2 Activity annotation after motion-based automatic activity segmentation

Learning from the previous experiment, we designed a motion-based activity segmentation on the mobile client. Activity segmentation first segments the incoming data to activities based on the change of motion and tries to recognize the segmented activity based on learned activity model. Users can choose to annotate the unlabeled activities on the mobile device. Since segmented activities already contain the starting/ending time and the corresponding geo-trace displayed on a map, users' cognitive load is much lower compared to the pure self-report approach described in the previous section.

Based on our previous work in [5], we develop an online unsupervised activity segmentation and activity recognition system on MobiSens App. Through unsupervised activity segmentation, MobiSens detects the boundaries of activities automatically based on the value of "activity change" estimated at all times. For each identified activities, MobiSens uses k-Nearest-Neighbor algorithm to check if it is "similar" to one of the annotated activities and label it with the proper activity name. If the activity is not similar to any known activities that have been annotated by the user so far, it will be labeled as "unknown activity". If the user is interested in providing a label to this activity, she can do so by replacing the label with a meaningful name. In addition with providing the chronicle boundary of activities as reference, we also provide the user a UI with GPS trajectory. When the user selects an activity segmentation, the map will show the locations that the user have been to in that period.

After the sensor stream is segmented automatically, the average number of activity labels provided by a user increases from 0.6 (self-reporting system) to 5 per day. This supports the intuition that once the activity is segmented, it is much easier for users to recall what they did during a time interval as compared to asking a user to recall both starting and ending time and their activities.

Figure 7(2) illustrates the UI of Release II of activity annotation based on segmentation.

5.6 Activity-aware power consumption optimization

We studied the power consumption by applying flat sampling rates on different sensors. We examined the battery life under three sensing configurations. All these settings are tested on a phone that doesn't have a phone/data plan to minimize the power consumed by other phone and application usage. The first configuration has only accelerometer sensors turned on; the second configuration has both accelerometer and WiFi scanning turned on; the third configuration turns on all sensors including GPS, WiFi and accelerometer.

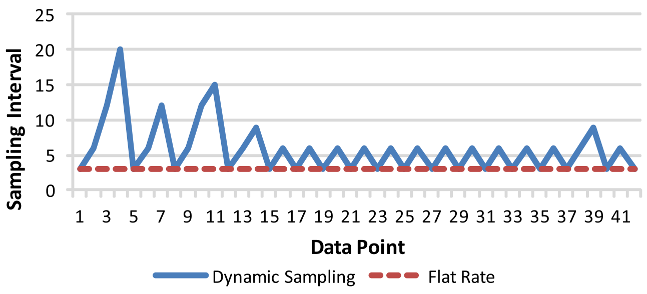


Fig. 8 Sampling intervals of dynamic sampling algorithm vs. flat rate algorithm. Numbers on *horizontal axe* are the index of collected GPS data points. The area under the *curve* is the total time corresponding to the number of GPS data points recorded

Experiments show that the phone can only work for roughly 25 h without charging if all sensors were turned on. After turning off GPS scanning, the battery can last for 60 h. If the WiFi scan is turned off as well (WiFi scans at the same rate as GPS in our experiment), the battery can last for more than 80 h[1].

Power consumption is particularly critical for GPS sensors on mobile sensing platform. Using a high flat request rate for GPS will drain the battery fast. However, if we lower the GPS sampling rate to too low, we might lose user's location change information between the gap of two samples. If the user goes out and back to the original place within a sampling cycle, the location change won't be recorded. We call this "*Sampling Gap*".

We developed an activity-aware dynamic GPS sampling scheme based on two major observations: (1) When the user is not moving, we should reduce the sampling rate or even turn the GPS off. (2) When the user transits from stable to moving, his/her motion usually changes. For example, it's impossible to switch from activity "Working in Office" to "Driving Home" without any motion changes. So we can leverage the activity change detection feature to reboot the GPS sampling cycle.

The GPS sampling scheme works as follows: The algorithm starts with a sampling seed interval (2 min in our experiment). If the system detects the user is not moving, the sampling interval will be doubled in each cycle until it reaching a predefined ceiling, then the GPS will be turned off.

When the Activity Detection Widget discovered an motion change and the sampling rate is smaller than the seed value, MobiSens will reset the GPS sampling rate to its initial value. Figure 8 illustrates the dynamic sampling scheme.

[1]The data reported are based on the average performance of three Motorola Droid "Milestone" used in our experiment, the performance may vary between phones from different manufacturers.

The experiment result shows that dynamic GPS sampling rate saves 25–40 % power comparing to the flat rate solution.

6 Experiments and evaluation

6.1 Experiments setup

The MobiSens mobile client runs on a *Motorola Droid*. Table 5 shows sensors used and their sampling rates. Although video and audio data could help, we decided not to use them in MobiSens for three reasons. First of all, the media stream will create a large data storage and transmission overhead, which greatly reduces the battery life. Another reason is privacy concerns, especially for recording video. In addition, we realized that capturing video on Android requires the application stays on foreground, which will prevent the user using other functionalities of the phone.

We collected 36 h of real life data in 5 continue weekdays from two graduate students to verify and analyze the system. The data collection process lasts about 7 h each day. Table 6 summarizes activities done by the user.

Users carried two phones during the data collection stage (Fig. 9). One was tied to the user's right arm and another phone was used in the normal way: most of the time the phone was in user's pocket and from time to time the user took it out to make phone calls or check emails.

By the end of each day, the user will use the web interface to annotate his activities during the day in two settings:

- Without looking at the unsupervised segmentation, the user listens to the recorded audio and creates from scratch his daily activity summary on the calendar. This segmentation/annotation is used as the ground truth in our experiments.
- Based on the unsupervised segmentation (starting/ending time information) and activity clustering (cluster similar activities and visualize using the same color) label segmented activities.

The goal of the experiments is to evaluate (1) whether the automatic activity segmentation matches the ground truth, and (2) whether the similar activity coloring helps the user to recall what has happened before.

Table 5 Sensors on the mobile client and their sampling rate

Sensor	Sampling rate
Accelerometer, magnetometer	
Ambient light, temperature	20 Hz (every 50 ms)
Microphone	8 KHz
Camera	
GPS, WiFi	Every 2 min

Table 6 Activity instance count and their average time

Activity	Instance count	Avg. time per instance (minutes)
Walk	7	50.29
Working on computer	15	66.07
Cooking	4	19.75
Eating	9	20.78
Washing dishes	2	4.5
Cycling	2	21.5
Video gaming	2	47.5
Presentation	2	29
Having class	2	79
Meeting	2	69.5
Talking to somebody	5	15
Driving	3	8.67
Printing paperwork	1	44

The power consumption tests are conducted by running real time activity segmentation algorithm on a phone multiple times with different configurations. To isolate power consumption from other factors, the phone functions are turned off (i.e., no connections to cell towers) and the user does not use any other applications on the phone (no web browsing, no emails etc). At the beginning of each testing run, the phone is fully charged. Each test runs lasts for 24 h and we report the percentage of power consumed by MobiSens.

6.2 Evaluation metric

We evaluate the accuracy of the activity segmentation and recognition by calculating the F1-Score of user's annotation with the ground truth (Fig. 10).

Each identified activity in system's annotation A is a triple of $< s, e, l >$ where s is the starting time of the activity, e is the ending time and l is the label of the activity such as "walking". Similarly, we can represent each activity in the ground truth G also as a triple of $< s, e, l >$.

For each activity A in the system's annotation, if there exists a ground truth activity G such that $A_l = G_l$, i.e., the two activities have the same label, and $A_s = G_s \pm \Delta$, $A_e = G_e \pm \Delta$ where two activities have roughly the same starting time and ending time within the allowed margin Δ, then we consider A matches the ground truth activity G and is a true positive (Fig. 11). With the precision and recall value calculated for each activity type, we can estimate the harmonic mean and report the F1 score. High F1

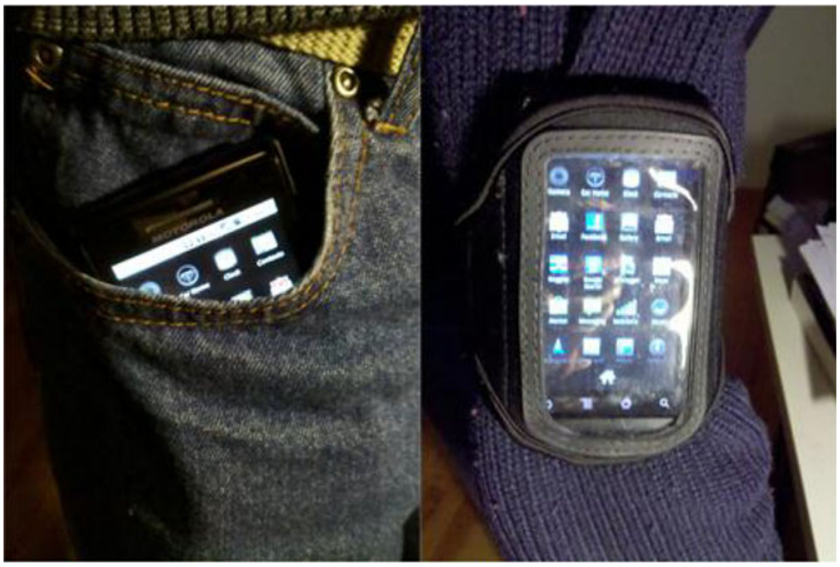


Fig. 9 Two mounting positions of phones in experiments: stowed in pocket (*left*) and mounted on upper arm (*right*)

scores indicate the system's segmentation/label matches the ground truth.

6.3 Impact of vocabulary size V and N-gram size

Table 7 shows the system performance with different vocabulary size.

Overall, recognition accuracies increase when V increases. It reaches the max when V is 200 and then drops slightly when $V = 300$. It could be that when V is too large, the system loses generalization and the trained activity recognition model over fits the training data.

The power consumption increases significantly when V increases. Because larger V requires more computation and CPU is one of the major sources that consumes power.

By fixing $V = 200$, we test different size of N (1, 2, 3, 4) for n-grams. Table 8 shows that when N reaches 3, activity recognition performance do not improve any more. The power consumption doesn't seem to have very significant change because the additional computation needed with larger N is trivial compared to the increase of V.

6.4 Impact of phone position in activity recognition

We run experiments to study whether the mounting position of the phone has any impact on the activity recognition accuracy. As shown in Fig. 9, participants carry two phones at the same time, one in the pants pocket and another one mounted on the upper arm. Figure 10 shows the recognition accuracy of different activities with two mounting positions. Overall, that system performs better when the phone is attached to users upper arm. For activities where motion comes mainly from hand movements, l(e.g., "cooking" and "working on computer"), the upper-arm setting performances better. For activities where motion comes mainly from legs, such as "cycling" and "walking", pocket position performs better than the upper-arm setting. It makes sense because arms are relatively stable while riding a bike yet legs are moving more frequently and regularly. For activity "walking", the motion patterns of upper arm are more similar to other activities whereas as the motion patterns of legs (pockets) are more distinguishable.

6.5 Hierarchical activity segmentation vs. smoothed HMM

Hidden Markov Models (HMM) have been widely used to model sequential data that have inherent structure as state transitions. We compare activity segmentation using the proposed hierarchical activity segmentation approach in MobiSens with a *smoothed single-layer Hidden Markov Model (HMM)*. After training a single-layer HMM using the EM algorithm over the unlabeled motion label sequence, we apply the learned HMM on the testing data (also a sequence of motion labels). During the decoding time, HMM finds the state sequence that best explains the observed sensor stream. For each input motion label, its corresponding HMM state becomes an "activity label". We apply a sliding window over the activity label sequence and use the majority label in window $[t - w/2, t + w/2]$ as the label for position t to smooth out noises. Segmentation is then done over the "smoothed" activity label sequence to identify positions where activities change.

The best configuration of HMM in our experiment has 10 states and the smoothing window size $w = 800$ which

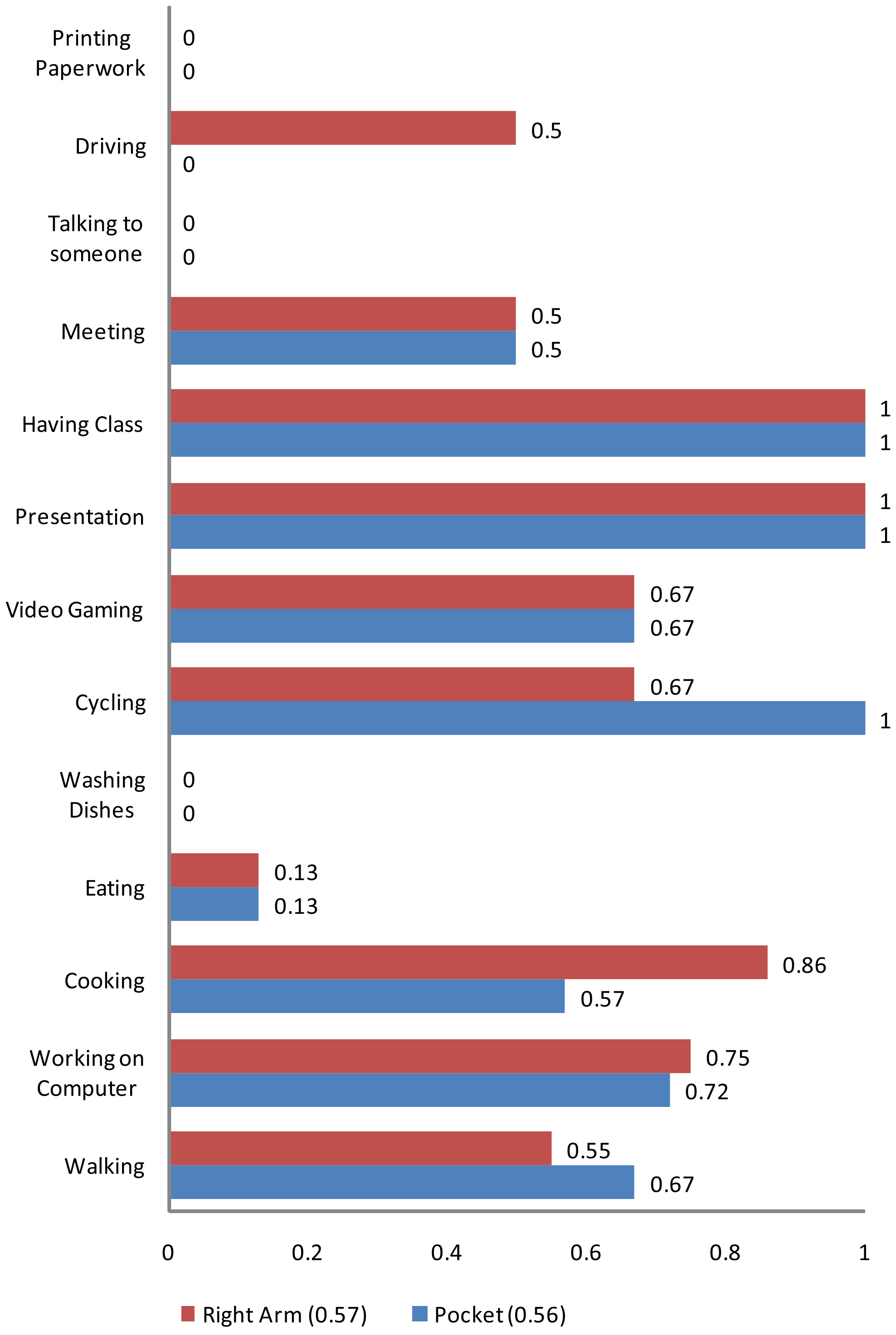


Fig. 10 $F1$ scores of activity recognition accuracies for two phone mounting positions: pants pocket and upper arm. Activity recognition uses only accelerometers data. For pocket settings, the average $F1$ score is 0.57. For upper-arm setting, $F1 = 0.56$

corresponds to 6 min in time. Averaged over all activity types, HMM performs worse than the hierarchical segmentation approach (Fig. 12). In particular, HMM performs badly on high-level activities such like *"Having Class"*, *"Meeting"*, *"Working on Computer"* and *"Presentation"*. These activities are usually composed by multiple low-level activities and have multiple motion patterns. Single-layer HMM models the lowest-level sensor readings and doesn't have the capability to merge these similar patterns to higher level activities.

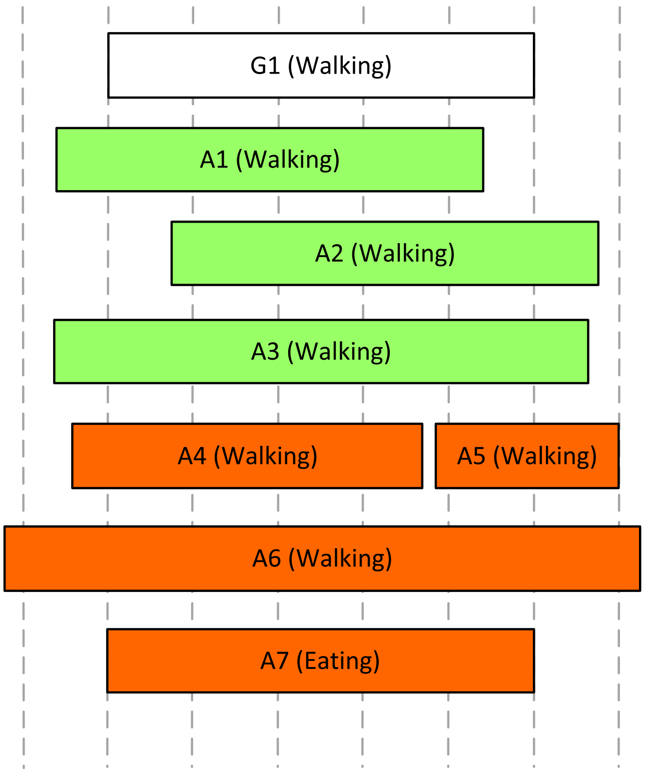


Fig. 11 Ground truth labeled as G_1 with three true annotations (A_1–A_3) and four false annotations (A_4–A_7)

6.6 Data size and power consumption

With a polling rate of 4 Hz, MobiSens collects 2 MB sensor data per hour, which is the major storage and power overhead. The following chart shows the power consumed by sensor (accelerometers, gyroscope and magnetometers) data collection (includes writing to SD card), GPS (using activity aware power optimization algorithm) and wireless sensor data upload (using WIFI). The experiment was done on a Motorola Droid running Android 2.3.

As we can see, sensor data collection (includes writing SD card) and WIFI transmission makes up 65 % of the MobiSens' power consumption. After leveraging the activity aware GPS sampling scheme, the power consumes by GPS only takes 10 % of the total power consumed by MobiSens. If we use a constant rate of 3 min per GPS request, this number will grow to more than 50 %. The computation overhead brought by activity segmentation and recognition, as well as other system-wise power consumption also make up 17 % of MobiSens' power consumption (Fig. 13).

This experiment indicates that we should reduce the wireless data transmission via the mobile device to lower power consumption. For MobiSens, we transmit raw sensor data to the server in order to keep all information for research purposes. For real-world applications though, it is more desireable to compress the sensor data before uploading or only upload sufficient statistics to the server side.

Table 7 The impact of V to the system, in terms of power consumption, overall recognition performance

$V =$	50	100	200	300
Recognition $F1$	0.33	0.45	0.57	0.49
Power (%)	41	45	51	63

N is set to 3 in these experiments

Table 8 The impact of n-gram size N to on power consumption and activity recognition accuracy

$N =$	1	2	3	4
Recognition $F1$	0.51	0.51	0.57	0.55
Power (%)	48	48	49	51

6.7 User adoption

MobiSens is designed to actively involve real-life users to participate in the sensing process. It provides not only an interactive interface for them to label their activities, but also an incentive system to motivate them doing so. Therefore, the usability of such a system becomes an important factor, which should be understood in order to guide system design and evaluation.

We have released three versions of MobiSens to Android Market. Each version contains one or two major aforementioned design improvements. Using data from randomly selected MobiSens users, we evaluate how the changes of UI design and activity segmentation/recognition algorithm affects the average number of activities annotated and shared by these users.

Our goal is to learn how the design and our activity segmentation/recognition algorithm change the amount of annotation shared by users. Instead of recruiting volunteers for this testing. We decide to evaluate our system based on those "random" users who download our application from Android Market. We believe this can be a better approximation of the real usage scenario of social sensing platforms.

During the five months' period, 310 users have downloaded and activated MobiSens. 13,993 h of data were collected. There are 68 valid users who provided data in more than two data upload cycles[2], and 22 of them have annotated and shared more than one activity.

[2]The upload cycle varies from 1 to 3 h during the whole experiment

Fig. 12 The F-Score of user annotation on hierarchical activity segmentation result vs. HMM, on right arm dataset, motion only

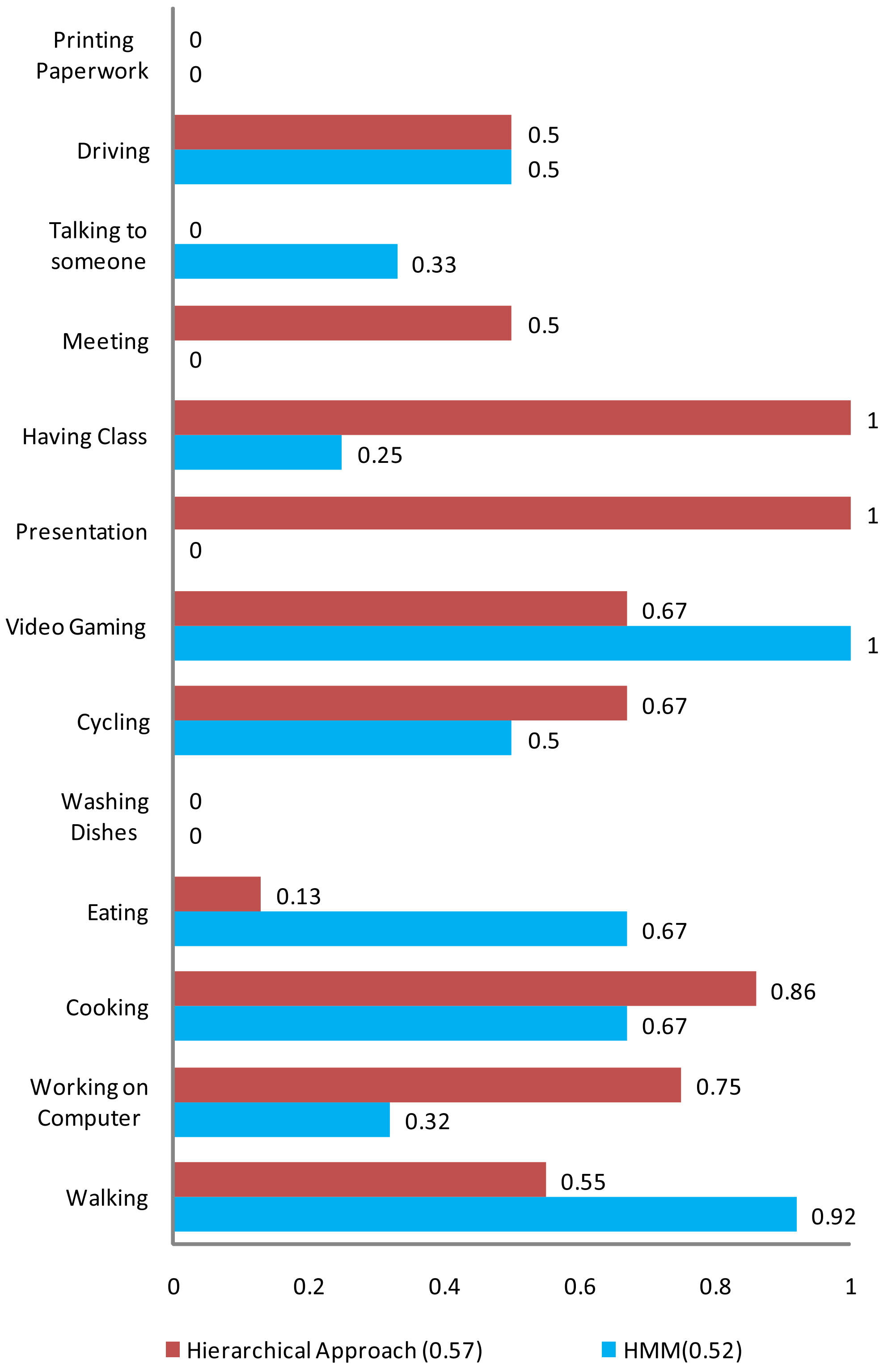


Figure 14 shows the distribution of users by their data contribution amount. There are 4 users who had contributed more than 1,000 h of data. The number of users who contributed data >100 h and $\leq 1{,}000$ h, >10 h and ≤ 100 h, >1 h and ≤ 10 h are 8, 26 and 30 respectively.

The average numbers of labels shared by each active user per day are shown in Fig. 15. It is clear that the average number of annotated activities from users increases significantly after the activity segmentation algorithm was integrated to MobiSens.

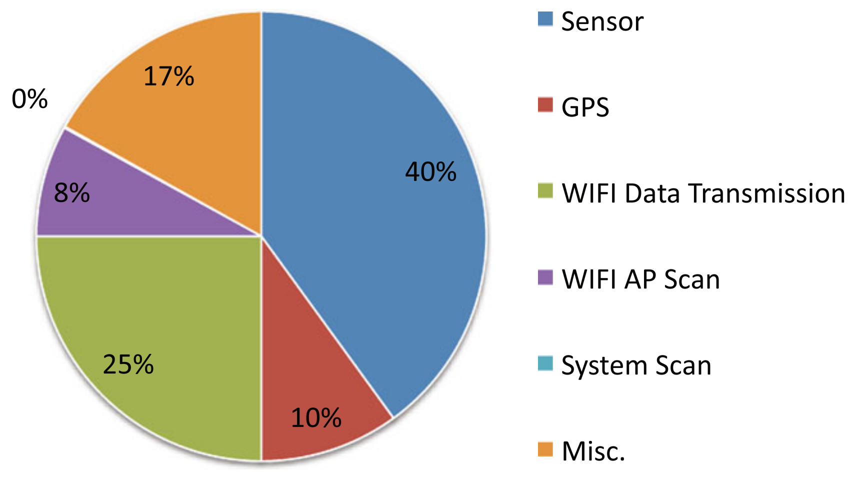


Fig. 13 The power consumed by different module/functionalities of MobiSens mobile client. "Sensor" are physical sensors including accelerometers, gyroscope and magnetometers (digital compass). "System Scan" is the scanning of running applications in the system, which consumes very little power

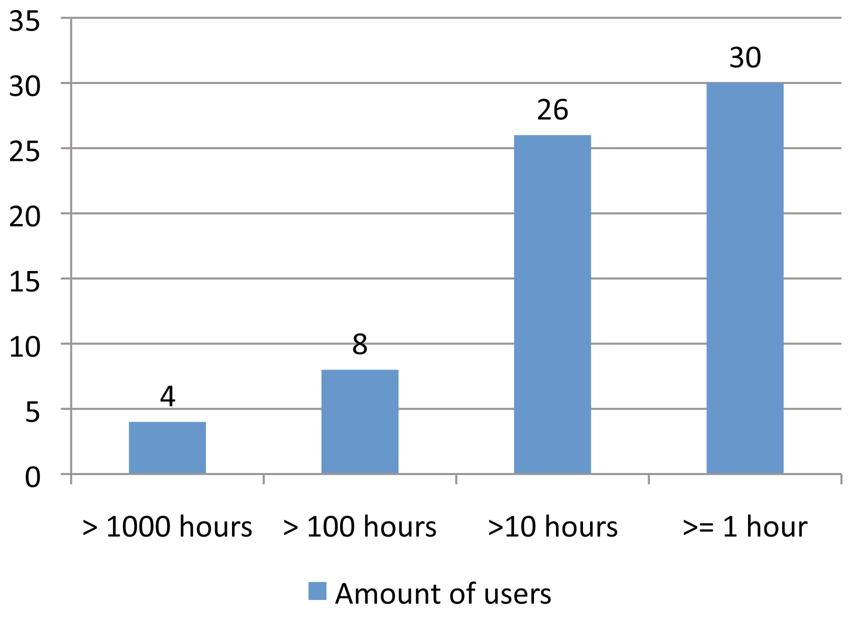


Fig. 14 The user distribution by their total data contribution time, the number is not accumulated

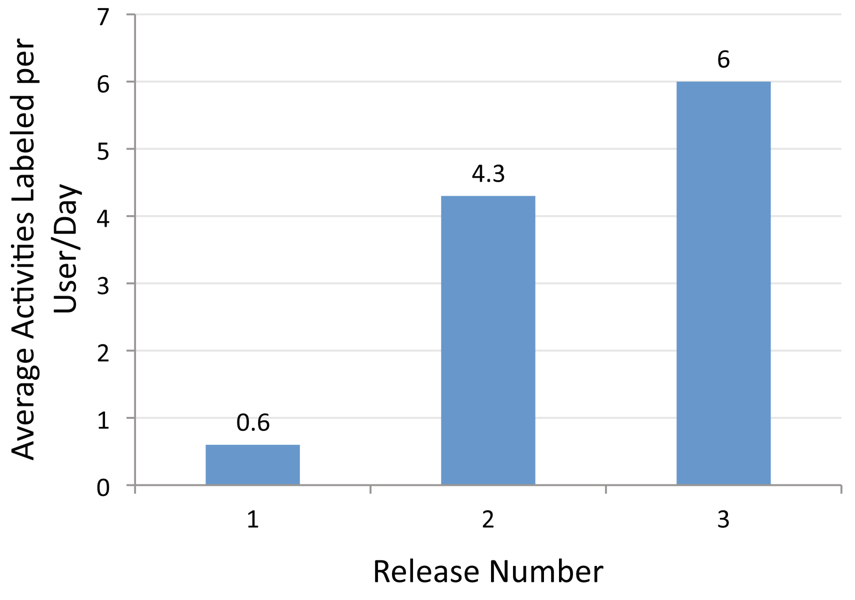


Fig. 15 The average number of activities shared by a user in one day between three MobiSens releases: (*1*) The naive self-report approach; (*2*) Integrated automatic activity segmentation and recognition to MobiSens; (*3*) After adding a map UI to help users recall their activities

7 Related work

7.1 Context-aware mobile systems

Several systems have been developed to collect information leveraging viral mobile platforms and build models to identify user activity, social patterns [8] and mobility [12]. Nathan and Pentland [8] used 100 Nokia 6600, Bluetooth-enabled smart phones, to help to collect proximity, time and location data from 75 users. Mobile user model is built in this context application. This system identifies social patterns in daily user activity, infers relationships, and models individual and collective behavior. They also developed Funf [18], an open source mobile data collection/sensing platform for context aware mobile sensing research.

The context-aware mobile systems are also widely studied under the Wireless Health scenario. Woodbridge et al. [29] studied five best selling Smart Phones in terms of their applicability to Wireless Health. In the surveyed systems, which include SmartCane [31] and SmartShoes [29], use smart phones as central controlling units that provide temporary storage, wireless transmission and data sinking functionalities. In these systems, sensing is done by external sensors that built in customized object like cane and shoes.

SystemSens [3] logs users' interactions with application and system parameters, such as battery life, screen status, OS traces and sensor network data to identify unexpected user behavior. It provides valuable insights to analyze and optimize energy consumption on the device caused by different activities [9]. It also provides a uniform and consistent method for publishing and sharing data.

CenceMe [17] is a personal sensing system that enables members of social networks to share their sensing presence with their buddies in a secure manner. Relying on a two-tier split-level activity classification, it can capture a limit set of user status in terms of activity, disposition, habits and surroundings. It also injects sensing presence into popular social networking applications such as Facebook, MySpace, and IM, allowing new levels of "connection" and implicit communication between friends in social networks. It represents the first system that combines the inference of the presence of individuals using off-the-shelf, sensor-enabled mobile phones with sharing of this information through social networking applications.

7.2 Activity detection and classification

In the past decades, different methods have been applied to a variety of sensors to address the activity recognition problem. For example, the computer vision based approaches [7, 21], Context-Free Grammar [23] and some other sequential language models [10, 11].

The most successful and exhaustive work in this area is made by Bao et al. [2]. In their experiments, subjects wore 5 biaxial accelerometers on different body positions as they performed a variety of activities like walking, sitting, standing still, watching TV, running, bicycling, eating, reading etc. Decision tree classifiers showed the best performance among other classifiers, which the overall recognition accuracy is 84 %.

The sequential and temporal characteristic of activity makes dynamic models such as the variants of Hidden Markov Model (HMM) are widely used in activity recognition, [25] uses Hierarchical Hidden Semi-Markov Model to detect activity patterns in the daily life of assisted living community residents. The method requires a training stage and works on data coming from simple state-change sensors which are massively installed in the environment. The presented method is supervised in the sense that a human expert must predefine activities and determines the sensors corresponding to each activity.

7.3 Power optimization

In terms of GPS power optimization, [13] proposed a model combined learning from both GPS/WIFI and cell tower pattern for significant place detection. The system will first use GPS and WIFI access points for location discovery as well as learning the cell tower change patterns. The system will map the GPS locations to corresponding cell tower pattern once the learning procedure has been done and do not need GPS thereafter .

7.4 Contribution of MobiSens

MobiSens differs from the aforementioned efforts in six-fold. (1) Instead of merely act as the central controlling device, Smart phone is used as a combination of sensing and controlling units in the MobiSens system. (2) In addition to sensory data collection, we build an unsupervised online activity segmentation on device, which automatically segments the incoming sensor stream to a sequence of activities. Users can choose to annotate activities that they are interested in and the system learns to recognize those activities in the future. (3) We introduce a bi-directional data exchange API between the mobile device and back-server. This information from the server to the mobile client enables functionalities such as dynamic sensing profile configuration, mobile process offload and incentive-based annotation and sharing. (4) To protect users' privacy, we only perform activity recognition on those segments that user has explicitly annotated or shared. "Unknown Activities" are not touched by the system. (5) We implement an incentive scheme to motivate users to participate in the sensing process by annotating their activities and share them among friends. (6) Instead of only relying on the location information to optimize GPS power consumption, MobiSens leverage other sensor information to achieve an activity change aware GPS sampling scheme.

8 Conclusion

We present the design, implementation and evaluation of MobiSens, a mobile sensing system that offers dynamic inference and summarization of users' behavior through their participation and interactions. We introduce a bi-directional data exchange API between the mobile device and back-end servers which enables functionalities such as dynamic sensing profile configuration, mobile processing offload and incentive-based annotation and sharing. We discuss the limits for current activity recognition algorithms on mobile sensing platforms and introduce an unsupervised hierarchical activity segmentation algorithm and a adaptive activity recognition algorithm to tackle challenges in mobile activity recognition. To protect users' privacy, we only perform activity recognition on those segments that user has explicitly annotated or shared. We also implement an incentive scheme to motivate users to participate in the sensing process by annotating their activities and share them among friends.

After releasing MobiSens on Android Market for five months, we have collected 13,993 h of data from 310 users. We evaluate the effectiveness of all design changes based on the average number of activities shared by per active user in each day. Experiments show that our effort has significantly increased the user acceptance of MobiSens platform.

We have developed several applications based on the MobiSens framework including the CMU SenSec App, a behavior-driven passive authentication system and mental health monitoring system. We believe this versatile mobile sensing platform will serve as a foundation to many exciting and promising real-world applications.

Acknowledgements This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273, from the Army Research Office (ARO), Nokia research award on "Mobile Sensing and Behavior Modeling for Social Computing," Google research award on "Social Behavior Sensing and Reality Mining," and a Cisco research award on "Behavior Modeling for Human Network." The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, Nokia, Google, or Cisco.

References

1. Adamson DM, Burnam MA, Burns RM, Caldarone LB, Cox RA, D'Amico E, Diaz C, Eibner C, Fisher G, Helmus TC, Tanielian T, Karney BR, Kilmer B, Marshall GN, Martin LT, Meredith LS, Metscher KN, Osilla KC, Pacula RL, Ramchand R, Ringel JS, Schell TL, Sollinger JM, Jaycox LH, Vaiana ME, Williams KM, Yochelson MR (2008) Invisible wounds of war: psychological and cognitive injuries, their consequences, and services to assist recovery. RAND Corporation, Santa Monica, CA
2. Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data. Springer, New York, pp 1–17
3. Burke J, Estrin D, Hansen M, Parker A, Ramanathan N, Reddy S, Srivastava MB (2006) Participatory sensing, pp 117–134
4. Buthpitiya S, Zhang Y, Dey A, Griss M (2011) *n*-gram geo-trace modeling
5. Chennuru S, Chen P-w, Zhu J, Zhang Y (2010) Mobile lifelogger—recording, indexing, and understanding a mobile user's life. In: MobiCase (September 2009)
6. Choudhury T, Consolvo S, Harrison B, Hightower J, LaMarca A, LeGrand L, Rahimi A, Rea A, Bordello G, Hemingway B, Klasnja P, Koscher K, Landay J, Lester J, Wyatt D, Haehnel D (2008) The mobile sensing platform: an embedded activity recognition system. IEEE Pervasive Computing 7(2):32–41
7. Duong TV, Bui HH, Phung DQ, Venkatesh S (2005) Activity recognition and abnormality detection with the switching hidden semi-Markov model. In: Proceedings of the 2005 IEEE Computer Society conference on computer vision and pattern recognition (CVPR'05), vol 1. IEEE Computer Society, Washington, DC, pp 838–845
8. Eagle N, Pentland A (2005) Reality mining: sensing complex social systems
9. Falaki H, Mahajan R, Kandula S, Lymberopoulos D, Govindan R, Estrin D (2010) Diversity in smartphone usage. In: MobiSys '10: Proceedings of the 8th international conference on mobile systems, applications and services. ACM, New York
10. Ghasemzadeh H, Barnes J, Guenterberg E, Jafari R (2008) View-invariant modeling and recognition of human actions using grammars. In: 5th IEEE international conference on mobile ad hoc and sensor systems, 2008. MASS 2008. IEEE, Piscataway, pp 58–68
11. Guerra-Filho G, Fermuller C, Aloimonos Y (2005) Discovering a language for human activity. In: Proceedings of the AAAI 2005 fall symposium on anticipatory cognitive embodied systems, Washington, DC
12. Herrera JC, Work DB, Herring R, Ban XJ, Jacobson Q, Bayen AM (2010) Evaluation of traffic data obtained via gps-enabled mobile phones: the mobile century field experiment. Transp Res, Part C Emerg Technol 18(4):568–583
13. Jiang Y, Li D, Yang G, Lv Q, Liu Z (2011) Deliberation for intuition: a framework for energy-efficient trip detection on cellular phones. In: Proceedings of the 13th international conference on ubiquitous computing, UbiComp '11. ACM, New York, pp 315–324
14. Kessler RC, Berglund P, Demler O, Jin R, Merikangas KR, Walters EE (2005) Lifetime prevalence and age-of-onset distributions of DSM-IV disorders in the national comorbidity survey replication. Arch Gen Psychiatry 62:593–602
15. Lapinski M, Berkson E, Gill T, Reinold M, Paradiso JA (2009) A distributed wearable, wireless sensor system for evaluating professional baseball pitchers and batters. In: Proceedings of the 2009 international symposium on wearable computers, ISWC '09. IEEE Computer Society, Washington, DC, pp 131–138
16. Logan B, Healey J, Philipose M, Tapia E, Intille S (2007) A long-term evaluation of sensing modalities for activity recognition. In: Proceedings of the 9th international conference on ubiquitous computing. Springer, Berlin, pp 483–500
17. Miluzzo E, Lane ND, Fodor K, Peterson R, Lu H, Musolesi M, Eisenman SB, Zheng X, Campbell AT (2008) Sensing meets mobile social networks: The design, implementation and

evaluation of the cenceme application. In: Proceedings of the international conference on embedded networked sensor systems (SenSys). ACM Press, New York, pp 337–350
18. MIT MediaLab. Funf open sensing framework (2011) http://funf.media.mit.edu/about.html. 16 Dec 2011
19. Nguyen LT, Cheng H-T, Wu P, Buthpitiya S, Zhang Y (2012) Pnlum: system for prediction of next location for users with mobility. In: Proceedings of mobile data challenge by Nokia workshop at the tenth international conference on pervasive computing, Newcastle, UK
20. Pantelopoulos A, Bourbakis NG (2010) A survey on wearable sensor-based systems for health monitoring and prognosis. IEEE Trans Syst Man Cybern, Part C Appl Rev 40(1):1–12
21. Poppe R (2010) A survey on vision-based human action recognition. Image Vis Comput 28(6):976–990
22. Roy P, Bouzouane A, Giroux S, Bouchard B (2011) Possibilistic activity recognition in smart homes for cognitively impaired people. Appl Artif Intell 25(10):883–926
23. Ryoo MS, Aggarwal JK (2006) Recognition of composite human activities through context-free grammar based representation. In: 2006 IEEE Computer Society conference on computer vision and pattern recognition, CVPR06, vol 2, pp 1709–1718
24. Soucy P, Mineau GW (2005) Beyond TFIDF weighting for text categorization in the vector space model. In: Proceedings of the 19th international joint conference on artificial intelligence (IJCAI 2005), pp 1130–1135
25. Tapia E, Intille S, Larson K (2004) Activity recognition in the home using simple and ubiquitous sensors. In: Ferscha A, Mattern F (eds) Pervasive computing, vol 3001 of Lecture notes in computer science, chapter 10. Springer, Berlin, pp 158–175
26. van Kasteren T, Noulas A, Englebienne G, Kröse B (2008) Accurate activity recognition in a home setting. In: Proceedings of the 10th international conference on ubiquitous computing, UbiComp ’08. ACM, New York, pp 1–9
27. Varkey JP, Pompili D, Walls T (2011) Human motion recognition using a wireless sensor-based wearable system. In: Personal and ubiquitous computing. Springer, Berlin, pp 1–14
28. Virone G, Wood A, Selavo L, Cao Q, Fang L, Doan T, He Z, Stoleru R, Lin S, Stankovic JA (2006) An advanced wireless sensor network for health monitoring
29. Woodbridge J, Nahapetian A, Noshadi H, Sarrafzadeh M, Kaiser W (2009) Wireless health and the smart phone conundrum. SIGBED Rev 6(2):11:1–11:6
30. Wu P, Peng H-K, Zhu J, Zhang Y (2011) Senscare: semi-automatic activity summarization system for elderly care. In: International conference on mobile computing, applications, and services (MobiCASE)
31. Wu W, Au L, Jordan B, Stathopoulos T, Batalin M, Kaiser W, Vahdatpour A, Sarrafzadeh M, Fang M, Chodosh J (2008) The smartcane system: an assistive device for geriatrics. In: Proceedings of the ICST 3rd international conference on body area networks, BodyNets ’08. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, pp 2:1–2:4
32. Zhu J, Wu P, Wang X, Perrig A, Hong J, Zhang JY (2013) Sensec: mobile application security through passive sensing. In: Proceedings of international conference on computing, networking and communications (ICNC 2013). San Diego, CA, USA, 28–31 January 2013
33. Zhu J, Zhang Y (2011) Towards accountable mobility model: A language approach on user behavior modeling in office WiFi networks. In: Proceedings of the IEEE international conference on computer communications and networks (ICCCN 2011). Maui, Hawaii, 31 July–4 August 2011