WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2016; **16**:2557–2571
Published online 4 July 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.2706

RESEARCH ARTICLE

Scalable floor localization using barometer on smartphone

Haibo Ye[1], Tao Gu[2*], Xianping Tao[1] and Jian Lu[1]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2] School of Computer Science and IT, RMIT University, Melbourne, Victoria, Australia

ABSTRACT

Traditional fingerprint-based localization techniques mainly rely on infrastructure support such as GSM and Wi-Fi. They require war-driving, which is both time-consuming and labor-intensive. With recent advances of smartphone sensors, sensor-assisted localization techniques are emerging. However, they often need user-specific training and more power intensive sensing, resulting in infeasible solutions for real deployment. In this paper, we present Barometer-based floor Localization system (B-Loc), a novel floor localization system to identify the floor level in a multi-floor building on which a mobile user is located. It makes use of the barometer on smartphone. B-Loc does not rely on any Wi-Fi infrastructure and requires neither war-driving nor prior knowledge of the buildings. Leveraging on crowdsourcing, B-Loc builds the barometer fingerprint map, which contains the barometric pressure value for each floor level to locate users' floor levels. We conduct both simulation and field studies to demonstrate the accuracy, scalability, and robustness of B-Loc. Our simulation shows that B-Loc can locate the user fast and the field study in a 10-floor building shows that B-Loc achieves an accuracy of over 98%. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
mobile phone localization; floor localization; barometer; crowdsourcing

***Correspondence**
Tao Gu, School of Computer Science and IT, RMIT University, Melbourne, Victoria, Australia.
E-mail: tao.gu@rmit.edu.au

1. INTRODUCTION

With the increasing pervasiveness of mobile phones, we have experienced an explosive growth of location-based applications, in which the location of a mobile user has to be known. In a multi-floor building environment, knowing the floor level of a mobile user is particularly useful for a variety of location-based applications. For example, in a fire emergency, locating the floor level of a user quickly and accurately is critical to life saving. In a shopping mall or an airport environment, a navigation service such as Google maps can prompt a mobile user with the floor map by knowing her/his current floor level. This is known as the *floor localization* problem, which we aim to determine the floor level in a multi-floor building on which a mobile user is located.

Indoor localization [1–3] has been well studied in the literature, and they can be used for floor localization. The fingerprint-based approach leveraging on Wi-Fi or GSM appears the most. SkyLoc [4] appears the first work for floor localization using GSM fingerprints; however, the accuracy is far from real use (i.e., three floor levels). RADAR [5] uses Wi-Fi signal. The idea is to war-drive the entire building to create a radio map between a physical location and its Wi-Fi fingerprint measured from nearby access points and base stations. Users can then pinpoint their locations by comparing their measured signal strength in the map. However, the main drawback is that war-driving is both time-consuming and labor-intensive for large indoor areas. Some recent approaches such as LiFS [3] use crowdsourcing to reduce the war-driving cost to some extent, but it involves a complicated training process. In reality, many mobile users may not turn on Wi-Fi all the time for energy saving, limiting the effectiveness of crowdsourcing. In addition, in many developing countries, many buildings have no or sparse Wi-Fi coverage, which is not dense enough for localization. Even, in developed countries, study [6] shows Wi-Fi may not be fully available in many buildings. Therefore, a cheap and scalable solution, which does not need any infrastructure support, is desirable.

The advancement of embedded sensors in smartphones has motivated a sensor-assisted localization approach

[1,7,8]. The accelerometer and compass have been used to measure the walking distance and direction of a mobile user. The user's location can be easily obtained by comparing the user moving trace and the map. However, these sensors are highly noisy [9]. The computed trajectory will increasingly diverge from the actual one. Hence, careful calibration is needed, for example, through fixed beacons [1] or landmarks [2]. Crowdsourcing has been also used to reduce the war-driving effort [2,10]. These works rely on detecting user activities using sensors such as accelerometer. However, to ensure reliable detection, they typically require user-specific training, which is costly, and the high sampling frequency, which may drain the battery power quickly. In addition, the detection may be often interrupted by users making or receiving phone calls.

The increasing availability of barometer embedded in smartphones (e.g., Galaxy Nexus and Nexus 4) has motivated us to go beyond the existing work by building a simple, sensor-based, battery efficient solution for floor localization. Muralidharan's most recent paper [12] studies on the properties of mobile-embedded barometers across a number of buildings. He concludes that it is difficult to use the barometer to determine the actual floor on which a user is located. In this paper, we overcome these challenges. We propose a novel Barometer-based floor Localization system (B-Loc). B-Loc does not rely on any Wi-Fi infrastructure; it requires neither war-driving nor any prior knowledge of the buildings. In addition, it is more energy efficient as compared with other sensor-assisted approaches [2,3]. B-Loc leverages on barometer sensing and crowdsourcing to build barometer fingerprints, which contain the barometric pressure value for each floor level, and locate users' floor levels by looking up the map.

An intuitive solution may work as follows. Let us assume that the barometric pressure (a.k.a. atmospheric pressure) at the ground floor of a building is p_0. Given the barometric pressure decreases by 0.12 hPa for going up every 1 m in the vertical direction, we can easily calculate the altitude of a smartphone user by $h = (p_0 - p)/0.12$ where p is the barometer reading. If each floor has the same height of h_0, we then know the floor level is $\lceil h/h_0 \rceil$. Unfortunately, in reality, it does not work in this way. First, p_0 is usually not accessible, and h_0 varies for different buildings. Second, the barometer reading p from smartphone is not accurate because of sensor drift. Such as a drift for the same floor level can vary from one to three levels, which is about 2 hPa (16.7 m). In addition, the most critical issue is that the barometric pressure at the same floor of a building keeps changing in a day because of different weather conditions and time [11,12].

In another typical solution, proposed by Wang in [13]. They track the user using barometer readings. First, they assume that the user's initial floor level is f_0. When the user changes floor levels, the barometer reading change Δp can be detected, and the user's new floor level is computed as $f_0 + \Delta p/(0.12 * h_0)$. In reality, h_0 varies from building to building, and each floor may have different height. So the floor height of each floor of every building is needed, limiting the scalability of this approach. More important, the initial floor f_0 is difficult to know, because users may not always enter into a building from the ground floor and they may start using the localization service at any floor level. Furthermore, a miss or wrong detection of Δp will cause serious errors in the latter localization.

In B-Loc, we propose several novel solutions to address these issues. First, we design a scalable, transitive calibration algorithm to automatically calibrate different smartphone users' barometers. The calibration makes use of user encounters and crowdsourcing, and it is carried out in a transitive way with more users involved. Second, to solve the issue that the barometer reading at the same floor changes over time, we propose time-based projection to project the barometer readings collected from different floors at different time to a common timestamp. It is based on the observation that although the barometer reading at the same floor may change over time, the difference of the barometer readings between any two floors keeps constant. If we obtain the barometer reading of a floor at a timestamp, the readings of any other floor can be estimated by computing the difference. Third, leveraging on crowdsourcing, we cluster the barometer readings for each floor to generate a barometer fingerprint map, which contains the barometer readings of every floor at any time.

In summary, we make the following contributions:

(1) We propose a novel barometer-based approach for floor localization. B-Loc makes use of barometer on smartphone only and does not require any infrastructure and the prior knowledge of buildings.
(2) We design several novel techniques to calibrate barometers for different smartphone users in a scalable way, project barometer readings to a common timestamp, and cluster crowdsourced barometer readings to generate real-time barometer fingerprints.
(3) We conduct both extensive simulations and field studies to analyze the performance of B-Loc. We deploy B-Loc in a real situation to demonstrate its superiority over existing solutions.

The rest of this paper is organized as follows. Section 2 gives an overview, followed by the detailed design. Section 3 discusses the system robustness, and Section 4 gives the theoretical analysis. Our evaluation is reported in Section 5. Section 6 discusses the related work, and finally, Section 7 concludes the paper.

2. SYSTEM DESIGN

We give an overview of B-Loc in this section, as shown in Figure 1. The system operates in two phases. In the first phase, B-Loc builds the barometer fingerprint map

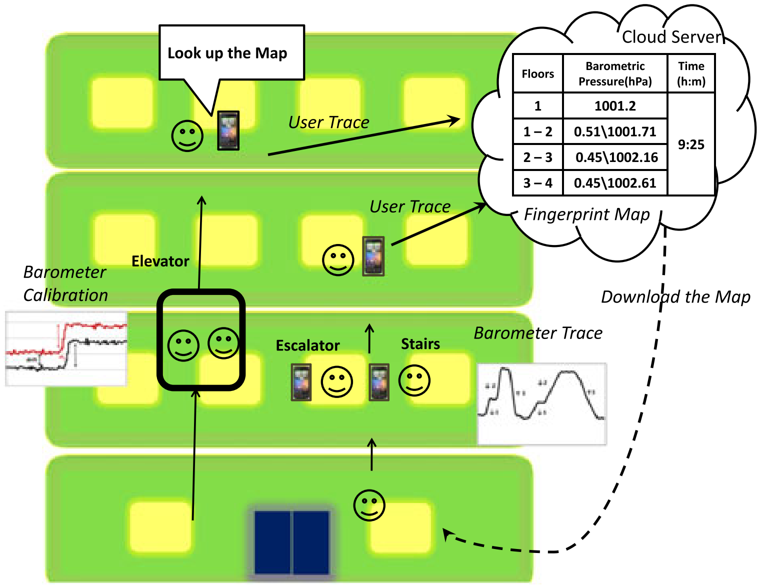


Figure 1. Overview of Barometer-based floor Localization system.

automatically. When a user travels up and down in the building (e.g., taking elevators/escalators and climbing stairs), as illustrated by the solid line arrows in Figure 1, the mobile client software running on the phone collects barometer readings in real-time. The activities of changing floors are detected and captured by our activity recognition algorithm. We design a robust technique to recognize such activities using barometer on smartphone. The recognized activities, together with real-time barometer readings, will be uploaded to the cloud server as a user trace. Different traces may contain the barometer readings of different floor levels, and these readings have to be calibrated before we make use of them. We calibrate barometers on different smartphones based on user encounter, which can be detected when two users enter in an elevator. The calibration is carried out in a transitive way with more users involved and eventually propagated to all possible users in a scalable way. After calibration, barometer readings in each trace will be projected to a timestamp t_0, making the readings from different users comparable. In the end, we cluster the barometer readings using a clustering algorithm based on CURE [14] to generate the barometer fingerprint map, which contains the barometer readings of each floor at t_0. By projecting the readings in the map from time t_0 to the current time t_{now}, we obtain the real-time barometer fingerprints, which will be used in the second phase.

In the next phase, a mobile user first downloads the barometer fingerprint map of the building and the calibrate information from the cloud server and then scans the barometer reading around. B-Loc calibrates the readings and compares the reading with the barometer fingerprints. The nearest barometer reading in the map will conclude the right floor level for the user.

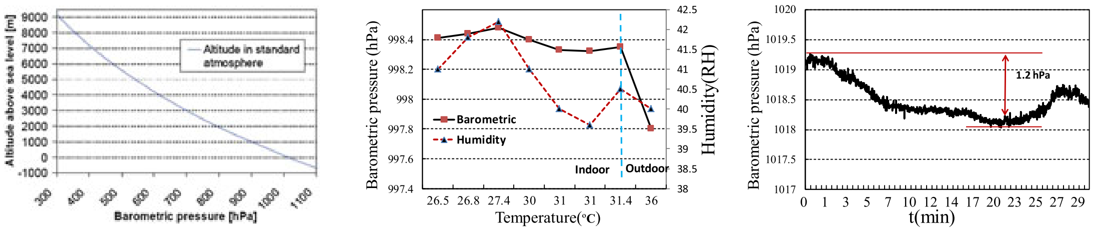
2.1. Background of barometric pressure

In this section, we give the background of barometric pressure and describe our preliminary studies. Barometric pressure is the force per unit area exerted on a surface by the weight of air above that surface in the atmosphere of Earth [11]. As altitude increases, barometric pressure decreases. Figure 4(a) illustrates this relation with a temperature of 15°C and a relative humidity of 0%. Although the pressure changes with weather, National Aeronautics and Space Administration has averaged the conditions for all parts of the earth year-round. Using this figure, one can calculate the altitude at a given barometric pressure. At low altitudes above the sea level, the pressure decreases by 0.12 hPa for going up every 1 m. For higher altitudes within the troposphere, the formula that relates with barometric pressure p to altitude h is as follows.

$$h = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right) \quad (1)$$

In an environment with different temperature and humidity, Formula 1 is not accurate enough to calculate the altitude from barometric pressure for distinguishing different floor levels. To investigate the phenomenon of barometric pressure in an indoor environment, we conducted two studies as follows.

We used a professional digital pressure gauge to measure the barometric pressure at a fix location in an office building over a period of half an hour. Figure 2(c) plots the result. From the figure, we observe that the barometric pressure measurements change with a variation of 1.2 hPa, which is equivalent to about 10 m in altitude. This



(a) Variation in atmospheric pressure with altitude (b) Barometric pressure and humidity in rooms with different temperatures (c) Barometric pressure changes by time

Figure 2. Barometric pressure in the building. (a) Variation in atmospheric pressure with altitude, (b) Barometric pressure and humidity in rooms with different temperatures and (c) Barometric pressure changes by time.

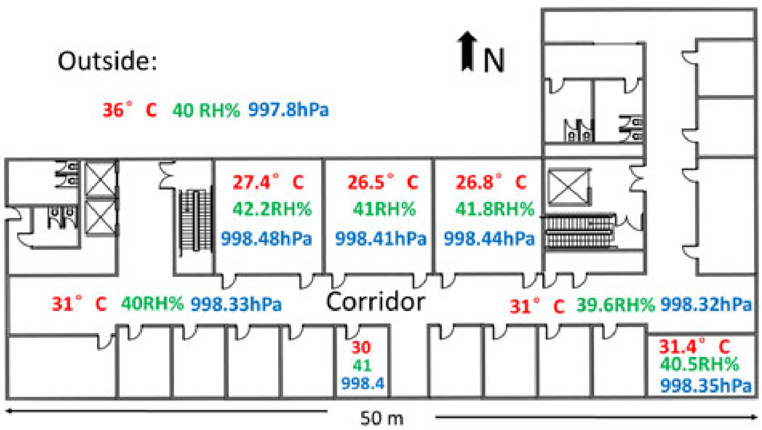


Figure 3. The barometric pressure, temperature, and humidity in different parts of the floor plan

Table I. Barometer sensor parameters.

Property	BMP180/182	LPS331AP
Absolute accuracy	$-4.0... + 2.0hPa$ $(-33... + 17m)$	$-3.2... + 2.6hPa$ $(-27... + 22m)$
Relative accuracy	$\pm0.12hPa(\pm1m)$	$\pm0.2hPa(\pm1.7m)$
Noise	$0.06hPa(0.5m)$	$0.06hPa(0.5m)$
Used in smartphone	Galaxy Note 2/3, Xaiomi M2,Sony Ericsson Active, Nexus 3/4	Galaxy S3,S4

variation may result in a detection error ranging up to three floor levels. Hence, directly applying the barometric formula to calculate the floor level is not feasible. Next, we studied the barometric pressures at different indoor locations at the same floor of the office building. We randomly selected seven indoor locations as illustrated in Figure 3. We used several digital pressure gauges, which have been calibrated to measure the barometric pressure in each of these locations at the same time during the summer, and Figure 2(b) plots the readings, including the reading at an outdoor location. The temperatures vary from room to room; however, the humidity in each room shows little difference. The result shows that the maximum variation is 0.2 hPa, which is equivalent to 1.6 m in altitude. Through this study, we observe that although the temperature and humidity vary, but the barometric pressures at the same floor level keep relatively constant.

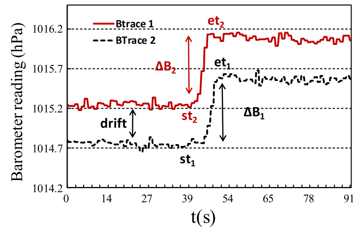
2.2. Barometer on smartphone

We now move to study the barometer sensor on smartphones. Barometer sensor has become increasingly popular on smartphones today. Most commonly used barometer sensors are BMP180/182 and LPS331AP. Table I gives their technical specifications. From the table, we observe that while the absolute accuracy[†] is about ± 20 m (which is low) and the relative accuracy[‡] is high. This implies that the barometer sensor has a high level of sensitivity, and it is good enough to detect the change of the barometric

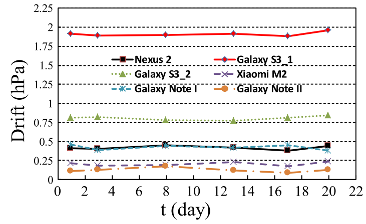
[†]The accuracy of a sensor reading compares with the real barometric pressure.

[‡]The accuracy of the change of a sensor reading compares with the change of real barometric pressure.

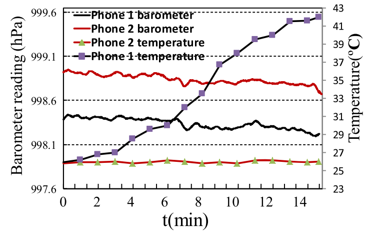
Wirel. Commun. Mob. Comput. 2016; **16**:2557–2571 © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm



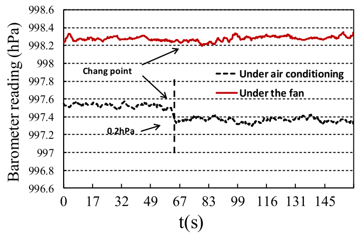
(a) The barometer reading drift between smartphones



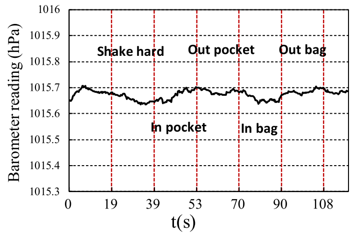
(b) The drift with real barometric pressure in different smartphones



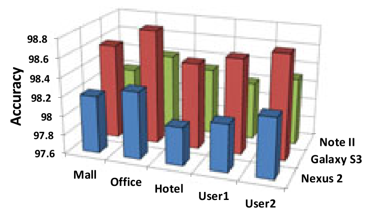
(c) The relation between smartphone temperature and barometer reading



(d) The relation between wind and barometer reading



(e) The barometer readings in different usage scenarios



(f) Floor-change detection accuracy

Figure 4. The properties of smartphone's barometer. (a) The barometer reading drift between smartphones, (b) The drift with real barometric pressure in different smartphones, (c) The relation between smartphone temperature and barometer reading, (d) The relation between wind and barometer reading, (e) The barometer readings in different usage scenarios and (f) Floor-change detection accuracy.

pressure when users go up or down in a building. Motivated by this observation, we use barometer to detect the activities when users change their floor levels.

We sampled the barometer readings of two smartphones of the same type at the same indoor location. Figure 4(a) shows a constant drift of sensor readings, which may result in an error ranging up to three floor levels. It is clear that appropriate calibration needs to be carried out. In another study, we compare the barometer readings from different smartphones, and we are interested to know if the drift changes over time. We used six different smartphones and recorded the drift to the real barometric pressure every 4 days. The results show that the sensor drift in Figure 4(b) keeps stable and the variation is negligible. This study shows the feasibility of using barometer on smartphones for floor localization.

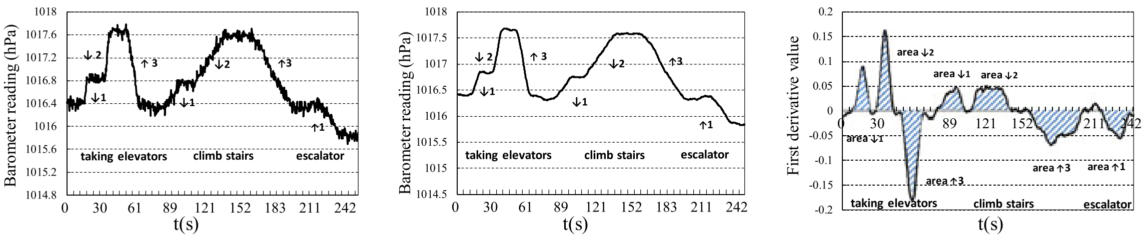
We then conducted more experiments to further study barometer sensor properties under different usage scenarios—(i) the smartphone gets hot; (ii) the smartphone is under the wind; (iii) the smartphone shakes; and (iv) the smartphone is in pocket or bag. Figure 4(c) shows the barometer readings of two smartphones at the same location for 15 min—one with a constant temperature, and the other with a growing temperature (e.g., when continuously running a computation intensive application). The result shows that the temperature does not affect the barometer reading. Figure 4(d) shows the barometer readings of the two smartphones with and without the wind (e.g., a fan and the air conditioner are used to generate the wind). The result shows that the barometer readings keep unchanged under the fan and vary in a small range of 0.2 hPa under air conditioning, it shows that the wind and temperature changes have limited impact on the reading. Figure 4(e) shows the barometer readings when shaking the phone, putting in and taking out from the pocket or bag. The result shows that the barometer readings remain constant under these scenarios.

From these studies, we conclude that barometer on smartphones, with proper calibration, is capable of determining users' floor levels.

2.3. Floor-change activity detection

We first present a novel technique to recognize the activities of changing floors using barometer. We represent a barometer sample P by $B = \{t, Baro\}$, where t is the time for sampling and $Baro$ is the barometer reading at time t. The barometer samples arriving in time order form a barometer trace, which is represented by $BTrace = \{ID, B_1, B_2, ..\}$, where ID is the identity of the user. Users typically change their floor levels by taking elevators/escalators and walking up or down the stairs. The barometer sensor is inherently noisy. Figure 5(a) shows the raw barometer readings, which apparently contain many spike noise. In B-Loc, we first filter these noise and then smooth the values with a reasonable window size of 1000 ms (i.e., the value at time t is the average value from $t-500$ to $t+500$ ms), as shown in Figure 5(b). In our previous study, we observe that barometer readings on smartphones do not change much in a short period of time unless users change their floor levels. Hence, the change of barometer

DOI: 10.1002/wcm



(a) Raw barometer readings (b) Smoothed barometer readings (c) First derivative value of the readings

Figure 5. Floor-change activity detection by barometer readings. (a) Raw barometer readings, (b) Smoothed barometer readings and (c) First derivative value of the readings.

readings can be used to recognize the floor-change activities. To do this, we extract the first derivative of the barometer readings, and the resulting curve is shown in Figure 5(c). We can see from the figure that the change of barometer readings is transformed to crest when going up and trough when going down. The crest and trough are sharp when taking elevators and smooth when taking escalators and stairs. The start and end time of the activity is the time of the left and right edge of each crest or trough.

To detect these activities, we calculate the area size of each crest or trough. If it meets certain conditions, a change-floor activity is detected. In detail, each area is defined as a continuous and closed region formed by the x-axis and the curve. The region is located below or upon the x -axis, which should meet the following conditions: (i) lasted time between 3 and 120 s, (ii) area size bigger than 1.0. Figure 5(c) shows the areas of different ways of floor changing. In B-Loc, we do not impose any constraint on the ways users carry or use their smartphones. A smartphone can be held on hand, placed into a pocket or bag, or used to make/receive a phone call. This certainly offers a great advantage over the accelerator-based activity recognition [10]. We define an floor-change activity as $A = \{STime, ETime, SBaro, EBaro\}$, where $STime$ is the start time of an activity, $ETime$ is the stop time of an activity, and $SBaro$ and $EBaro$ are the barometer reading at $STime$ and $ETime$, respectively. The user's moving trace can be then defined as $MTrace = \langle ID, A_1, A_2, \ldots \rangle$, where ID is the identity of the user. The detection is carried out in the smartphone, and the resulting $MTrace$ will be uploaded to the cloud server. We conducted experiments with two users using three different smartphones under real-life situations in three different buildings. Figure 4(f) shows the accuracy of detecting floor changes. The results show that the average accuracy using barometer is about 98.3%.

It is worth knowing that barometer readings at this stage are used for real-time activity detection on smartphone. We will show at a later stage how barometer readings are used for generating the barometer fingerprint map in the cloud server.

2.4. Transitive calibration algorithm

The objective of barometer calibration is not to calibrate each smartphone's barometer to the real barometric pressure but use any smartphone's barometer as a reference point and find the drift between each of other user's barometer and the reference. Before we introduce our calibration algorithm, we first introduce the property of the drift between sensors. We define $drift_{AB}$ as the drift between barometers A and B and $drift_{AB} = Baro_A - Baro_B$, where $Baro_A$ and $Baro_B$ is the reading from barometers A and B, respectively, under the same barometric pressure. The barometer drift holds the following properties: (i) $drift_{AB} = -drift_{BA}$ and (ii) $drift_{AB} + drift_{BC} = drift_{AC}$. These properties clearly demonstrate the transitive relationship. We define barometer calibration as follows: (i) For two smartphones, they are calibrated when the drift of the two barometers is known by the cloud server. (ii) For more than two smartphones, they are calibrated when the drift between every two barometers is directly known by encounter in elevator or indirectly known by the transitive relationship.

2.4.1. Calibration for two barometers.

Calibration is carried out by analyzing barometer traces. The idea is to calibrate users' barometers when they encounter each other. User encounter can be easily detected in elevators because elevator is very common in buildings and users often encounter each other in elevators. We observe that if users encounter each other in an elevator, the time and value of their barometer change are the same. In an other word, if we detect two floor-change activities from both users' barometer traces, these activities start and end at the same time, and the barometer reading change is the same, we conclude that the two users encounter in the same elevator and Figure 4(a) is an example. This is formalized as follows.

(1) I_1: $A_i.STime = A_j.STime$;
(2) I_2: $A_i.ETime = A_j.ETime$;
(3) I_3: $A_i.SBaro - A_i.EBaro = A_j.SBaro - A_j.EBaro$; and
(4) I_4: $A_i = A_j$;

 Wirel. Commun. Mob. Comput. 2016; **16**:2557–2571 © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm

where A_i and A_j is the floor-change activity for users i and j, respectively. The rule is then formulated as follows.

$$R_1 : I_1 \wedge I_2 \wedge I_3 \rightarrow I_4.$$

If we have $A_i = A_j$, the drift is then calculated by the following formula.

$$drift_{ij} = \frac{\sum_{t=A_i.STime}^{A_i.ETime}(B_i(t) - B_j(t))}{n} \quad (2)$$

where $B_i(t)$ and $B_j(t)$ are the barometer readings of users i and j, respectively, and at time t, n is the total sample size.

We analyze a case of when two users enter into different elevators at different floors and experience a floor-change activity with the same barometer change at the same time. The previous rules will conclude they are in the same elevator. To handle this case, we first observe that when users encounter in an elevator, they often experience more than one floor-change activity together. For example, users i and j encounter each other at the ground floor and go up to the 8th and 10th floor, respectively. Before arriving at level 8, the elevator stops at levels 3 and 5. In this scenario, users i and j experience three floor-change activities (i.e., from 1 to 3, 3 to 5, and 5 to 8). Based on this observation, we detect consecutive floor-change activities between two users to minimize the probability of this fault case. We formalize it as follows.

(1) I_5: $\exists A_1, A_2, .., A_k \in MTrace_i$;
(2) I_6: $\exists A_1, A_2, .., A_k \in MTrace_j$;
(3) I_7: $A_{m+1}.STime - A_m.ETime < 30$ holds in $MTrace_i$ and $MTrace_j$;
(4) I_8: $MTrace_i.A_m = MTrace_j.A_m$; and
(5) I_9: users i and j are in the same elevator and the confidence is k.

The rule is then formulated as follows.

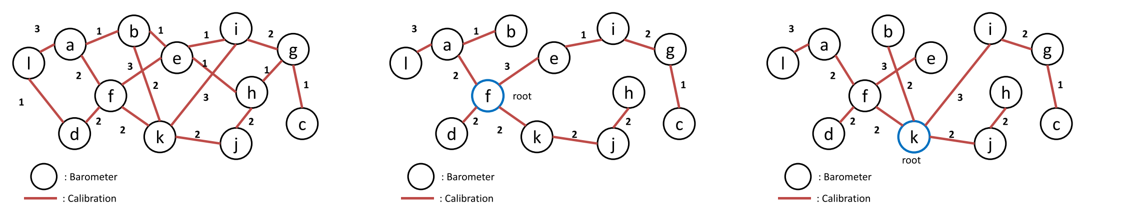
$$R_2 : I_5 \wedge I_6 \wedge I_7 \wedge I_8 \rightarrow I_9.$$

where $MTrace_i$ and $MTrace_j$ are the trace of users i and j, respectively, and k is the confidence that users i and j are in the same elevator.

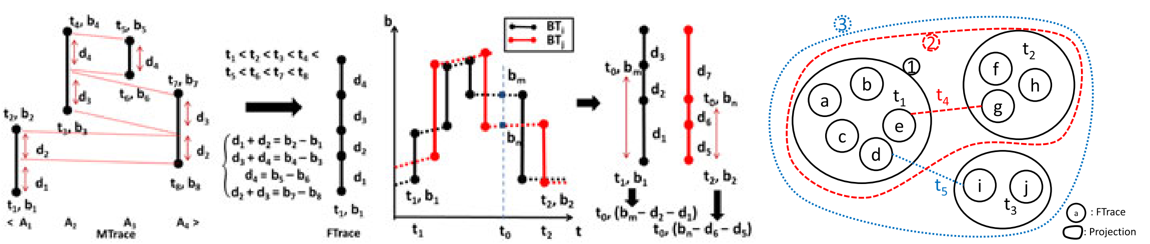
2.4.2. Calibration for all barometers.

In the previous section, we present barometer calibration for two smartphones. To calibrate all smartphones' barometers in a building, while the same principle will be applied, the calibration propagates from phone to phone in a transitive way. We model this process using a graph shown in Figure 6(a). In this graph, each barometer is represented by a node. If two barometers are calibrated by an encounter in elevator, we draw an edge between the two nodes, and the weight of the edge represents the confidence value of the calibration. Because there may be more than one calibration done between two users (e.g., two users may encounter each other multiple times), we choose the calibration with the highest confidence value. Because barometer calibration is transitive, in theory, any two barometers can be calibrated if this graph is connected. To select a root barometer, a trivial approach is to randomly choose a node as root and find a spanning tree from the graph as shown in Figure 6(b). Any node in the spanning tree can be calibrated following the path from the node to the root. For example, to obtain the drift of barometer j, we find a path between nodes f and j, $f - k - j$, and obtain the drift by $drift_{fj} = drift_{fk} + drift_{kj}$.

There are two factors affecting the accuracy of our calibration algorithm. The first one is the confidence values of the edges in the path. The other one is the length of the path. The confidence determines the probability of correct



(a) The calibration graph (b) An example calibration tree with an arbitrary root (c) Computed calibration tree



(d) Extract $FTrace$ from $MTrace$ (e) Project two $FTrace$ to a timestamp t_0 (f) Project all $FTrace$ to a timestamp t_5

Figure 6. Calibration and projection. (a) The calibration graph, (b) An example calibration tree with an arbitrary root, (c) Computed calibration tree, (d) Extract *FTrace* from *MTrace*, (e) Project two *FTrace* to a timestamp t_0 and (f) Project all *FTrace* to a timestamp t_5.

calibration. The length of the path determines the calibration accuracy because errors may be accumulated along the path. This turns out to be an optimization problem—find a spanning tree, choose a root to minimize the sum of nodes' depths, and maximize the weight of the edges in the path. It has been proven to be an NP-complete problem [15]; hence, finding this spanning tree is not realistic. In B-Loc, we propose a heuristic solution based on the observations that the confidence of edges in the path is important and the calibration errors accumulate slowly (which will be further analyzed in Section 4.2). We first find a maximum spanning tree, which maximizing the edge confidences in the tree, and we then choose a node as the root, which minimizes the average depth of all other nodes. As an example, we run the algorithm on the graph shown in Figure 6(a), and the resulting graph is shown in Figure 6(c). The complexity of the algorithm is *O(NlogN)*.

2.5. Time-based projection

After calibration, we update all the barometer readings in *MTrace* by adding their drifts. The barometer readings in *MTrace* are collected by different users at different time, and each trace may only contain a partial view of barometer fingerprints of the building. To have a complete view, we have to combine them. To do so, we first project *MTrace* to a timestamp. We define a new data structure called *FTrace*. A *FTrace* contains the barometer reading distance between some floors and a reference barometer reading point extracted from a *MTrace*. As shown in the left-hand side of Figure 6(d), in the *MTrace*, there are floor-change activities of going up and down. Because the barometer reading distance between every two floor levels is constant, we can extract the barometer distance between floors from *MTrace* as shown in the right-hand side of Figure 6(d). For example, we know b_4 and b_5 are the barometer reading at the same floor (we assume no miss detection), and $b_4 - b_3 = b_5 - b_6 + d_3$; we can then infer that b_6 is scanned on a higher floor than b_3, and d_3 is the barometer reading distance of the two floors. In this way, we extract all the barometer reading distance between floors from *MTrace*. The structure of *FTrace* is defined as $\{\langle d_1, d_2, ..d_{k-1}\rangle, \langle t, b\rangle\}$, where d_i is the barometer distance of two floors (which floors still unknown at this stage). b is the barometer reading at timestamp t in the lowest floor of *FTrace*, and $\langle t, b\rangle$ is called the reference point of *FTrace*.

Next, we project two *FTrace* to the same timestamp, making them comparable. Consider two *FTrace* FT_i and FT_j from user i and j, respectively, b_i and b_j are not comparable because $t_i! = t_j$. As shown in Figure 6(e), we first choose a reference time t_0 in the overlap time zone of the two *BTrace* from user i and user j and obtain sample $\langle t_0, b_m\rangle$ in BT_i and sample $\langle t_0, b_n\rangle$ in BT_j. For every sample, we obtain the barometer reading distance between the floor level of the sample and the floor level of the reference point; for example, $\langle t_0, b_m\rangle$ is sampled at a higher floor level than t_1, b_1, and the distance is $d_2 + d_1$. We can then infer that, at time t_0, the barometer reading at the floor of the reference point is $b_m - d_2 - d_1$, and the barometer reading of the reference point of FT_j at t_0 is $b_n - d_6 - d_5$. Therefore, both the references of FT_i and FT_j are projected into the same timestamp t_0.

We present Algorithm 1 to project all *FTrace* into the same timestamp, as illustrated in Figure 6(f).

Algorithm 1 Project all *FTrace* into the same timestamp.

Input:
The set of all FTrace, C; An empty set, C';
Output:
The set of all projected FTrace, C';
1: Find a biggest subset C_1 of C, where the intersection of their time intervals is not empty. Remove C_1 from C, and put C_1 to set C';
2: Choose a timestamp t from the intersection time interval of C_1, project all *FTrace* to timestamp t;
3: Repeat 1 and 2 until C is empty;
4: Choose two biggest set C'_1 and C'_2 from C', choose a *FTrace* from C'_1 and C'_2 respectively, project them to timestamp t_1, and project all *FTrace* in C'_1 and C'_2 to t_1, union set C'_1 and C'_2;
5: Repeat 4 until C' become an one element set;
6: **return** C'.

2.6. Barometer reading clustering

After calibration and projection, we are now able to compare barometer readings in *FTrace* and relay them to each floor level in the building. As shown in Figure 7, we first transform *FTrace* from $\{\langle d_1, d_2, d_3, ..d_{k-1}\rangle, \langle t, b\rangle\}$ to barometer reading points $\{b, b + d_1, .., b + d_1 + .. + d_{k-1}\}$, where each element represents the barometer reading of a floor level. Each *FTrace* contains some barometer readings at different floors of the building at the same timestamp t. Ideally, for a n-floor building, we should have n different barometer readings. However, errors may be introduced during calibration. To eliminate errors, we use clustering. We apply the hierarchical clustering algorithm named CURE [14]. Initially, each barometer reading is a cluster. The CURE algorithm merges two closest clusters in

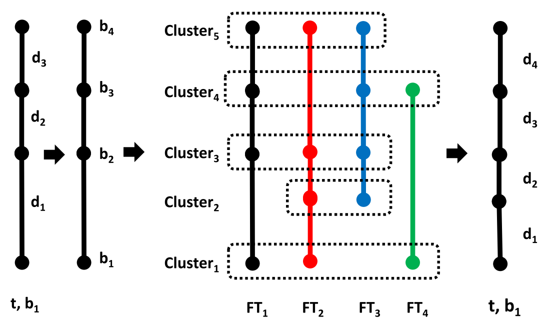


Figure 7. Clustering barometer readings.

each step until a certain number of clusters are formed. CURE fits well in this situation because it is less sensitive to outliers. However, we cannot apply CURE directly because the resulting number of clusters f (should be the same to the floor numbers n) is unknown. In B-Loc, we adapt the CURE algorithm by designing the distance function and determining when to stop clustering. We use the Euclidean distance as distance function to calculate the distance between two clusters of samples. In each cluster, we choose m median samples to calculate the distance. The distance function between cluster C_i and C_j is computed as follows.

$$Distance(C_i, C_j) = \sqrt{\sum_{k=1}^{m}(B_{ik}.b - B_{jk}.b)^2} \qquad (3)$$

where $B_{ik}.b$ is the barometer value of the kth middle value sample of cluster C_i.

The clustering algorithm stops when the distance of the nearest two clusters is smaller than a threshold. We set the threshold to the two-third (0.3 hPa) of the minimum barometer distance between floor levels in *FTrace* (the one-third 0.15 hPa is for tolerating the error of the barometer reading). After clustering, we obtain a set of clusters. For each cluster, we compute the average of their m median samples as the value of this cluster. We then order all the clusters by this value from high to low. The ordered sequence has an one-to-one mapping to floor levels. The highest value maps to the ground floor, and the lowest value maps to the top floor level. In the end, we obtain the barometer reading of each floor at timestamp t and also the barometer reading distance of every pair of floor levels. A barometer map with a reference point $\langle t, b\rangle$is generated, and it is defined as $Map = \{\langle d_1, d_2, d_3, ..d_{n-1}\rangle, \langle t, b\rangle\}$, where d_i represents the barometer distance of floor i and $i + 1$, n represents the floor number, and b represents the barometer reading at the ground floor at timestamp t. Using the barometer fingerprint map, we can know the barometer readings at each floor. For example, the barometer reading at floor level 3 is $(b + d_1 + d_2)$.

2.7. Real-time barometer fingerprint map

After obtaining the barometer fingerprint map at timestamp t with the reference point $\langle t, b_1\rangle$, we now convert the reference point to the current time (i.e., t_{now}), in the first scenario, when the users are still in the building. Our idea is to find a *BTrace*, which contains readings at both time t and t_{BTrace}. We first obtain the floor level f_1 of user at time t by comparing the barometer reading at time t with the barometer fingerprint map. We then detect if the floor-change activities occurred between time t and t_{BTrace} and obtain the floor change, which is Δf. The current floor level of the user is $f_2 = f_1 + \Delta f$; we then obtain the barometer reading b_2 at time t_{BTrace}, and the reference point is now $\langle t_{BTrace}, b_2 - d_{f_2-1} - d_{f_2-2} - ... - d_1\rangle$. In this way, we convert the barometer fingerprint map from time t to t_{BTrace}.

In the second scenario, all users leave the building and arrive the building in the next day, in order to update the reference point to now (t). Because users are all calibrated, the approach is to obtain the barometer readings of the users at t and cluster the readings using our clustering algorithm. Only if there is at least one user in the ground floor, the cluster with the biggest barometer reading must be the reading of the ground floor. Hence, the reference point gets updated.

2.8. Locating users

Users can now download the barometer fingerprint map and the calibration information from the cloud server. For each barometer reading sampled from a smartphone, the reading will be adjusted based on the calibration information and look up the map to find the floor level of the user.

The barometric pressure of the ground floor may change by time, and B-Loc is able to dynamically update the reference point of the map. Our approach is to append the barometric pressure change to the reference point ($\langle t, b\rangle$). For example, the barometric pressure changed Δb after 60 s, and the reference point will be updated to $\langle t + 60, b + \Delta b\rangle$. In this way, the application does not need to download the map again if the user stay in the building. At the same time, the application will upload the reference point to the cloud server periodically, where the barometer fingerprint map is updated.

3. DESIGN FOR ROBUSTNESS

In this section, we first analyze different fault cases of floor-change activity detection. There are basically two fault cases, (i) miss detection (a floor-change activity occurs but miss detected) and (ii) wrong detection (e.g., a normal barometer fluctuation was detected as a floor-change activity). The experiments show that our floor-change activity detection has about 1.1% miss detection and 0.4% wrong detection. We are interested to know the consequences of these fault cases.

(1) Influence on calibration: The miss detection does not affect calibration because we only use detected floor-change activities. The wrong detection will cause a failure in calibration when two wrong detections occur at the same time and with the same barometer distance. However, this probability of wrong detection is very small and can be negligible.
(2) Influence on map generation: When clustering the barometer readings, the wrong detection will results in wrong barometer readings. For example, given a *FTrace,* which is $\{\langle d_1, d_2, ..d_{k-1}\rangle, \langle t, b\rangle\}$, if d_3 is a wrong detection, when we transform them to barometer readings $\{b, b+d_1, .., b+..+d_{k-1}\}$, there will have $k-3$ wrong readings. These readings, which are rare, will be viewed as outliers during clustering. Because CURE algorithm is less sensitive to outliers, they will be removed and have little impact on clustering. The

miss detection will generate a wrong barometer reading only if the floor height of the building is different. If that is the case, miss detection is equivalent to wrong detection.

(3) Influence on localization: It will not affect the localization because we locate users by barometer readings not their travel traces.

Next, we analyze the accuracy of the calibration process. The confidence values of all the edges in the path determine the probability of correct calibration. In our approach, we choose a maximum spanning tree which essentially minimizes the probability of wrong calibration. The length of the path determines the calibration accuracy because calibration errors may accumulate (will be analyzed in the next section).

4. THEORETICAL ANALYSIS

We provide the theoretical analysis of B-Loc in this section.

4.1. Modeling calibration

We model the calibration process using a random graph $\Gamma_{n,N}$ with n labeled vertices and N edges, where a vertex represents a barometer (n is the total number of barometer sensors) and an edge may exist between two vertices if they are calibrated. All sensors can be calibrated if graph $\Gamma_{n,N}$ is connected. We denote p as the probability that there exists an edge between any two vertices in $\Gamma_{n,N}$, which is equal to the probability that two users encounter in an elevator. We assume that the average number of users in elevator is k and the frequency that a user takes an elevator is f. We have

$$p = \frac{k * f * t}{n-1} \tag{4}$$

where t is the time. It is shown in [16] that the random graph $\Gamma_{n,N}$ is almost surely be connected if

$$p > \frac{1}{n}(1+\epsilon)\ln n \tag{5}$$

With a large n, the value of $(1+\epsilon)\ln n/n$ is small; therefore, a smaller p can meet the previous equation.

It is shown in [16] that, in the random graph $\Gamma_{n,N}$, the size of the greatest component of $\Gamma_{n,N}$ is, for $c = \frac{N}{n}$ with $c > \frac{1}{2}$ with probability tending to 1, approximately $G(c)n$, where

$$G(c) > 1 - \frac{x(c)}{2c} \tag{6}$$

where

$$x(c) > \sum_{k=1}^{\infty} \frac{k^{k-1}}{k!}\left(2ce^{-2c}\right)^k \tag{7}$$

The curve $y = G(c)$ is shown on Figure 8(a). This means almost all points of $\Gamma_{n,N}$ belong to either some small component which is a tree or the single "giant" component of the size $G(c)n$. From this, we imply that, if $p > \frac{1}{n-1}$, most of the nodes will belong to the same "giant" component and all can be calibrated. Figure 8(c) shows the calibration graph of 100 users in a 10 floor building in our simulation, the giant component grows soon with more user elevator trips. Figure 8(c) shows the calibration graph of 100 users in a 10 floor building in our simulation, the giant component grows soon with more user elevator trips.

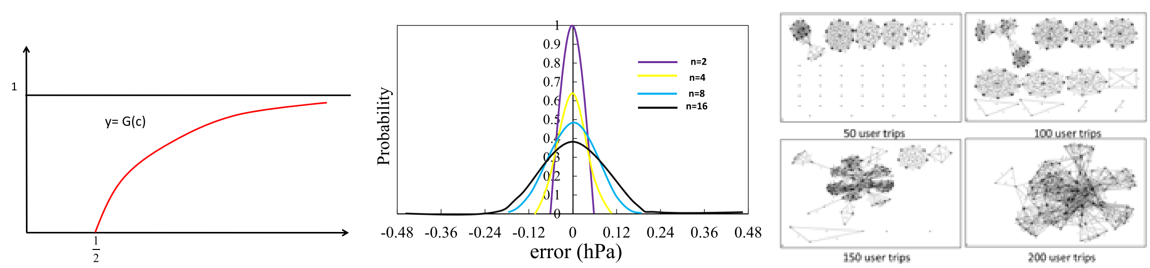
4.2. Calibration error

We model the calibration accuracy as follows.

$$drift_{ij} = drift'_{ij} + \sum_{k=1}^{n} U_k \tag{8}$$

where n is the length of the path from node i to j, $drift'_{ij}$ is the real drift, and U is the noise function with a random value between -0.06 to 0.06 hPa. The accumulated noise function is $X = \sum_{k=1}^{n} U_k$. Because U has a uniform distribution, in probability and statistics, X is a Irwin-Hall distribution [17] function. The probability density function is

$$f_X(x:n) = \frac{1}{(n-1)!}\sum_{k=0}^{\lceil x \rceil}(-1)^k \binom{x}{y}(x-k)^{n-1}; x \in [0,n] \tag{9}$$



(a) The curve $y = G(c)$ (b) Probability density function (c) The calibration graph of 100 users in a 10-floor building

Figure 8. Theoretical analysis. (a) The curve $y = G(c)$, (b) Probability density function and (c) The calibration graph of 100 users in a 10-floor building.

The curve of $f_X(x:n)$ is shown in Figure 8(b). It is shown that the error is less than 0.24 hPa with high probability when the path length is less than 16. Because the minimum barometer reading distance of any adjacent floor level is about 0.45 hPa, the error is tolerable.

5. EVALUATION

We now move to evaluate B-Loc using both simulation and field studies.

5.1. Simulation methodology

We design a simulator to evaluate the efficiency and scalability of B-Loc. In the simulation, we aim to evaluate how well B-Loc performs calibration and how fast the barometer fingerprint map can be built. The simulator models the process of user taking the elevator up and down in a multi-floor building. It works as follows. The simulation process is divided into cycles of elevator going up or down (which occurs with an equal probability). For elevator going up, each cycle simulates the process that the elevator goes up from the ground floor, with people entering and leaving the elevator from or to any levels, until the elevator is empty. We model the process of people entering the elevator from the ground floor as the Poisson distribution. The expected number of the Poisson distribution is set to 1/4 of the maximum load of a typical elevator (i.e., four persons). People on the ground floor may go up to any floor with a probability of $1/(n-1)$, where n is the number of floors of the building. From any other floor f_i, some people may enter the elevator, and go to the rest $(n-i)$ floors with an equal probability $1/(n-i)$. Each cycle starts from the ground floor, we first compute the number of people entering the elevator and which floors they are going to, the elevator goes up from the ground floor, and stops when people exiting or entering, until there are no users in the elevator. For elevator going down, every time the elevator starts from the top floor, users in every floor may enter the elevator, and will go to the rest n' floors with an equal probability of $1/2(n'-1)$, except to the ground floor which is $1/2$. When people enter or exit the elevator from a floor, the number of people on that floor gets updated, and the trace of every user is recorded. Based on our observation from real-life situations, in our simulation model, we assume that when an elevator passing a floor, the probability of a user in that floor entering the elevator is p (1% in our setting).

In the simulation of barometer readings of a building, the barometer reading of the ground floor is simulated with function $F(t) = F(t-1) + math.random() * 0.1$, where $F(0) = 1000hPa$ and t is the time, one unit of time is set to the time the elevator travels 1 floor, for simplicity, the barometer change of every floor is the same which is set to $0.45hPa$.

Given a number of floors n and a number of users u, we simulate cycles of elevator going up and down until a certain number of user-elevator trips m is reached (a user-elevator trip is defined as the process of a user entering and leaving the elevator). At the end of each simulation cycle, we combine the ground truth from the floor-change activity detection to get the *MTrace* of every user. We will evaluate how well the barometer sensor of the users can be calibrated. We then build the barometer fingerprint map using the calibrated *MTrace*, and show how fast it can be built.

The parameters of the simulator are listed as follows.

(1) *floor number*: the number of floors.
(2) *user number at each floor*: the number of users at each floor.
(3) *total trip number* : the number of user-elevator trips for all the users.
(4) *average trip*: the average number of user-elevator trips for each user.
(5) *average number of users in elevator*: the average number of users in elevator when it is moving.

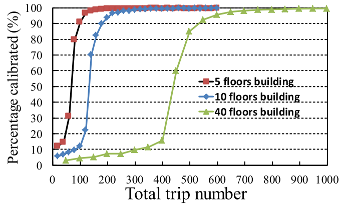
The performance metrics used in the paper are summarized as follows.

(1) *average weight*: The average weight of all edges in the calibration tree.
(2) *average hop*: The average hop of all nodes to the root in the calibration tree.
(3) *percentage calibrated*: The percentage of users who have been calibrated.

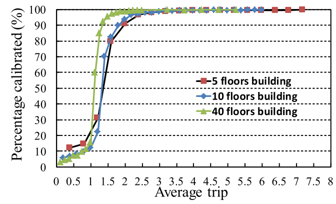
5.2. Simulation results

Figure 9(a)–(f) show the simulation results when the *user number at each floor* is 10 and the *average number of users in elevator* is 4. Figure 9(a) shows the *percentage calibrated* in three different buildings under different *total trip number*. It shows that all users can be calibrated after about 150/300/1000 trips in a 5/10/40−floor building. When the *total trip number* is changed to the *average trip* as shown in Figure 9(b). We find that in the three buildings, all users can be calibrated when each user takes the elevator for about 2.5 times on average. Figure 9(f) shows the *percentage calibrated* in the 10-floor building with different *average trip*. The different colors in each column represent the calibrated groups, from the largest to the smallest. It shows that when *average trip* grows from 1 to 1.3, the size of the largest calibrated group grows fast and almost 95 percent of users are calibrated when the *average trip* is 1.5.

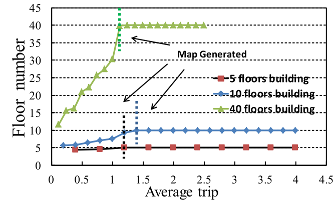
Figure 9(c) shows the number of floors found in the barometer fingerprint map with different *average trip*. It shows that B-Loc builds the map for the 5/10/40-floor buildings when the average trips of users are less than 1.5. Compare to the result in Figure 9(a) and Figure 9(f), it shows that B-Loc builds the map before all users are calibrated, and locates most of the users quickly (e.g., 95 percent of users when *average trip* is 1.5). Figures 9(d) and 9(e) show the *average weight* and *average hop* in the calibration tree, respectively. Figure 9(d) shows that the average calibrate weight is about 2/3/5 in the 5/10/40-floor building. In Figure 9(e), the hops is about 3/4.5/8 in the three buildings when all users are calibrated, which do not



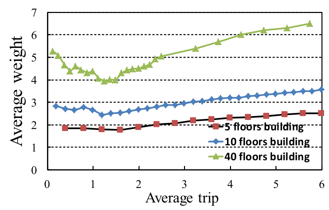
(a) The *percentage calibrated* with different *total trip number*



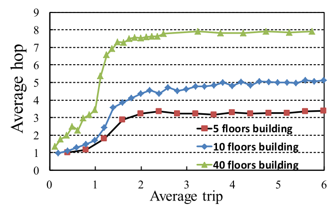
(b) The *percentage calibrated* with different *average trip*



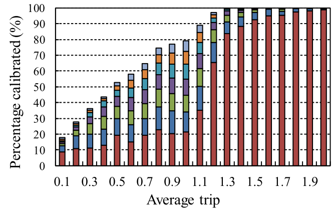
(c) Floor numbers detected with different *average trip*



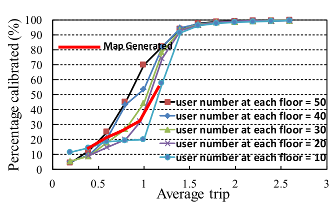
(d) The *average weight*



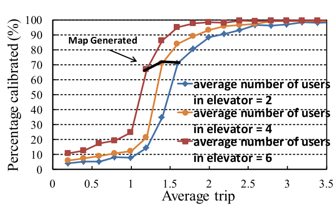
(e) The *average hop*



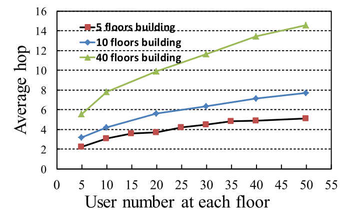
(f) The size of the calibration groups



(g) Calibration result when *user number at each floor* is different



(h) Calibration result when *average number of users in elevator* is different



(i) The *average hop* when *user number at each floor* is different

Figure 9. Simulation results. (a) The *percentage calibrated* with different *total trip number*, (b) The *percentage calibrated* with different *average trip*, (c) Floor numbers detected with different *average trip*, (d) The *average weight*, (e) The *average hop*, (f) The size of the calibration groups, (g) Calibration result when *user number at each floor* is different, (h) Calibration result when *average number of users in elevator* is different and (i) The *average hop* when *user number at each floor* is different.

affect the calibration accuracy based on our analysis in Section 4.2.

Figure 9(g) shows that when there are different number of users in the building, all users can be calibrated with nearly the same *average trip*. It implies that more users is not necessary to have more average trips. Figure 9(h) plots the calibration result with different *average number of users in elevator*. The result shows that the calibration process is faster with more average user in elevator. Figure 9(i) plots *average hop* when all users are calibrated. The result shows that *average hop* grows when there are more users in the building, this is reasonable because more users form large calibration trees.

5.3. Field study

To evaluate B-Loc under the real-world situations, we implemented a prototype system and publish it on a website [18]. We encourage users to download and try B-Loc in our 10-floor computer science building. The floor plan of the building is shown in Figure 2(b). A total number of 67 users downloaded our application to their mobile phones (e.g., Samsung, Google Nexus, and Sony Ericsson). Out of 67 mobile phones, only 28 have both barometer sensors and a mobile network data connection (i.e., GPRS or 3G). We developed a mobile application named "Talking to Strangers (up/down stairs)" which is built on top of B-Loc. The application finds users from other floors of a building for message chatting. This is similar to other chat applications such as find strangers around, but we incorporate B-Loc into our application for floor localization.

When the application runs, it continuously collects barometer readings at a rate of two samples per second, and all the samples will be logged in a data file which will be uploaded to the cloud server every 2 h. The floor-change activity detection is carried out in real-time and the *MTrace* will also be uploaded to the cloud server. The client also performs time synchronization with the sever by computing the round-trip delay time and the offset. If the barometer fingerprint map is available in the cloud server, it will be downloaded to locate the user. The indoor/outdoor detection is carried out by scanning GPS signals. Before obtaining the map, the application uses a simple tracking approach to locate the user. It has a low accuracy because it assumes users enter the building from the ground level (i.e., the initial floor is 1) and the height of each level is fix (i.e., the barometer reading change of every floor is about 0.45), floor localization is then carried out by detecting user activities of changing floors. We did not give special instructions to the users during our study. Users ran the application as they like. To get the ground truth, we first manually get the drift to the real barometric pressure for all the 28 smartphones, then placed a barometer logger at each floor to get the ground truth by comparing their readings with the ground truth. The field study ran for eight days. In each mobile client, the history of floor location is logged. In the cloud server, all calibration results and the barometer fingerprint map generated are logged. We analyze the performance based on the logged data in both the client and the cloud server.

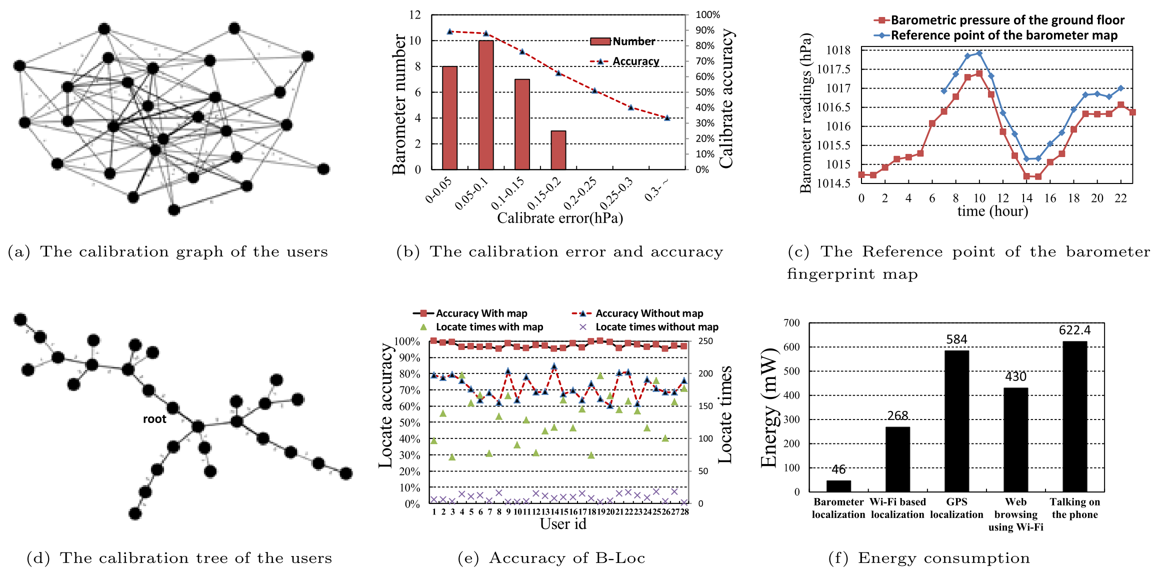
5.4. Field study results

Figure 10(a) shows the calibration graph of the users at the end of the first day. The thickness of the lines represents the confidence (i.e., weight) of the calibration between two users. The graph is connected, and users are directly calibrated (i.e., two to eight users). The users' average trips are 6 (when coming in and going out for lunch and dinner). The calibration tree is extracted and shown in Figure 10(d). The average hop of calibration is about 3; the max hop of calibration is 6, and the average calibration weight is about 1.8. The calibration error is shown in Figure 10(b). The left vertical axis shows the number of barometers in different error regions. The right vertical axis shows the accuracy of the calibration in each region. The accuracy is defined as the ratio between error and the barometer reading distance of one floor. An accuracy of higher than 50% is the necessary condition to locate the user to the right floor. Figure 10(c) shows the real barometric pressure of the ground floor and the reference point of the map in 24 h. The reference point is available from 8 AM to 11 PM when there are users in the building. The reference point updates the same way as the barometric pressure change, which shows that the map is accurate. The root of the calibration tree is not calibrated by real barometric pressure and caused a constant drift between the two curves.

We obtain the accuracy of floor localization by comparing the ground truth with the barometer loggers and the floor location history of B-Loc. It is shown on the left vertical axis of Figure 10(e), where the accuracy is more than 98% for every user when they are calibrated and the barometer fingerprint map is generated. When the users are not calibrated or at the beginning of every day when the reference point of the map is not found, using the simple tracking approach the accuracy is about 70%, which appears in about 3% of all location cases as shown in the right vertical axis of Figure 10(e).

5.5. Energy consumption

In the end, we evaluate the energy consumption of B-Loc using a Samsung Galaxy Nexus smartphone running Android 4.1 OS, and the result is shown in Figure 10(f). The power consumption is computed based on PowerTutor, a diagnostic tool for analyzing system and application power usage from the Android Market. The experiment ran for 12 h continuously. The average power consumption of B-Loc is 46 mW. For comparison, we also show the power consumption of other localization techniques and some basic mobile functions. It shows that B-Loc consumes much less energy than the traditional localization techniques.



(a) The calibration graph of the users
(b) The calibration error and accuracy
(c) The Reference point of the barometer fingerprint map
(d) The calibration tree of the users
(e) Accuracy of B-Loc
(f) Energy consumption

Figure 10. Field study results. (a)The calibration graph of the users, (b) The calibration error and accuracy, (c) The Reference point of the barometer fingerprint map, (d) The calibration tree of the users, (e) Accuracy of B-Loc and (f) Energy consumption.

6. RELATED WORK

Many fingerprint-based techniques for indoor localization have been proposed such as [4,5,19–21]. They mainly rely on Wi-Fi signal strength, and they are capable of achieving a high accuracy in an indoor environment. However, like RADAR [5] has to war-drive the entire building in order to obtain the radio map. War-driving is very time-consuming and labor-intensive, and it may have to be carried out periodically because the Wi-Fi signature at the same location may be changed over time. Hence, this solution is not scalable. The fingerprint-based technique has been used in floor localization. SkyLoc [4] uses GSM fingerprints to locate a user's floor level in a multi-floor building. They report an accuracy of 73% for locating a user to the right floor, and 95% within 2 floors. But the GSM signals vary significantly in indoor environments, and the training process in SkyLoc is time-consuming. It has a poor scalability because war-driving and training are required for every building. Some recent approaches such as LiFS [3] use crowdsourcing to reduce the training cost to some extent, but it involves a complicated training process. In reality, many mobile users may not turn on Wi-Fi all the time for energy saving, limiting the effectiveness of crowdsourcing. Different from these systems, B-Loc makes use of the new barometer sensor appears in recent smartphones. It does not require war-driving to build the fingerprint database, B-Loc relies on crowdsourcing and intelligently build the barometer fingerprint map to locate users' floor level.

Sensor-assisted localization methods [1,7,8,22] have been proposed, making use of embedded sensors available on smartphones. These systems typically use accelerometer and electronic compass. However, careful calibration is needed from time to time due to the limitations of the sensing technology. For example, Escort [1] leverages on fixed beacons for calibration. Crowdsourcing has been also used to reduce the training effort [2,10]. These works rely on detecting user activities using sensors such as accelerometer. However, to ensure reliable detection, they typically require user-specific training which is costly, and the high sampling frequency which may drain the battery power quickly. In addition, the detection may be often interrupted by users making phone calls. Muralidharan's most recent paper [12] study on the properties of mobile-embedded barometers across a number of buildings. But failed to solve the problem of using the barometer to determine the floor of a user. In another solution proposed by Wang in [13] before the mobile-embedded barometers appear, using a barometer sensor to track user's floor level. As we discussed in the introduction, this approach need the detail information of the building and need to know a initial floor of every user, which is hard to get. Furthermore, a miss or wrong detection will cause serious errors in the latter localization. FTrack [22] using accelerometer for floor localization. The idea is to detect user activities of changing floors and track their floor levels based on their initial locations. It requires neither infrastructure nor training. With crowdsourcing, user traces containing acceleration data are collected and interpreted to generate the map for floor localization. The main problem of this approach is that they cannot handle some practical issues such as different user walking patterns and a variety of ways to carry/use mobile phones, which may affect the accuracy and limit the feasibility. While B-Loc follows the basic principle of FTrack, it is not a tracking system, and it does not rely on the previous location information. B-Loc detects user activities of changing floor by a novel barometer-based technique, and it has no assumption of users walking pattern or the ways to carry/use mobile phones.

7. CONCLUSION AND FUTURE WORK

This paper presents a novel, scalable floor localization scheme. Leveraging on mobile phone sensing and crowdsourcing, B-Loc requires neither any infrastructure nor any prior knowledge of the building. Different from similar approaches, B-Loc does not require war-driving and rely on barometer only. B-Loc provides high accuracy of activity recognition and minimum energy consumption, making it more realistic for real-world deployment. Our simulation and prototype system demonstrate the performance, scalability, and robustness of B-Loc. For our future work, we will further improve B-Loc by enhancing the calibration algorithm. We also plan to offer a full version of B-Loc as a free service to Google's play store and the Apple store for public use, and test B-Loc under real-life situations.

REFERENCES

1. Constandache I, Bao X, Azizyan M, Choudhury R. Did you see bob?: Human localization using mobile phones. In *MobiCom*, Chicago, Illinois, USA, September 20-24, 2010; 149–160.
2. Wang H, Sen S, Elgohary A, Farid M, Youssef M, Choudhury RR. No need to war-drive: Unsupervised indoor localization. In *MobiSys*, New York, NY, USA, 2012; 197–210.
3. Yang Z, Wu C, Liu Y. Locating in fingerprint space: wireless indoor localization with little human intervention. In *MobiCom*, New York, NY, USA, 2012; 269–280.
4. Varshavsky A, LaMarca A, Hightower J, de Lara E. The skyloc floor localization system. In *PerCom*, Washington, DC, USA, 2007; 125–134.
5. Bahl P, Padmanabhan V. Radar: an in-building rf-based user location and tracking system. In *INFOCOM*, Israel, 2000; 775–784.
6. *Wifi usage across the world*. http://imgur.com/gallery/ma2syfv [accessed on October 2014].
7. Constandache I, Choudhury R, Rhee I. Towards mobile phone localization without war-driving. In *INFOCOM2010*, San Diego, CA, 14-19 March 2010; 1–9.

8. Ofstad A, Nicholas E, Szcodronski R, Choudhury R. Aampl: Accelerometer augmented mobile phone localization. In *Proceedings of the 1st ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*. ACM, New York, NY, USA, 2008; 13–18.
9. Jekeli C. *Inertial Navigation Systems with Geodetic Applications*. Walter De Gruyter Inc, 2000.
10. Alzantot M, Youssef M. Crowdinside: automatic construction of indoor floorplans. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, USA, 2012; 99–108.
11. *Barometric pressure*. http://en.wikipedia.org/wiki/atmosphericpressure [accessed on October 2014].
12. Muralidharan K, Khan AJ, Misra RK, Balan RK, Agarwal S. Barometric phone sensors–more hype than hope! ACM HotMobile, New York, NY, USA, 2014.
13. Wang H, Lenz H, Szabo A. Fusion of barometric sensors, wlan signals and building information for 3-d indoor/campus localization. In *Proceedings of International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2006)*, Heidelberg, September 2006; 426–432.
14. Guha S, Rastogi R, Shim K. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record,* vol. 27, New York, NY, USA, June 1998; 73–84.
15. Franceschet M, Gubiani E. From entity relationship to xml schema: a graph-theoretic approach. In *Database and XML Technologies*. Springer, 2009, Lecture Notes in Computer Science, vol. 5679; 165–179.
16. Erdos P, Rényi A. On the evolution of random graphs. *The Mathematical Institute of the Hungarian Academy of Sciences* 1960; **5**: 17–61.
17. *Irwin-hall distribution*. http://en.wikipedia.org/wiki/irwin-halldistribution [accessed on October 2014].
18. *B-loc application and code*. http://moon.nju.edu.cn/twiki/bin/view/icsatnju/haiboye [accessed on October 2014].
19. LaMarca A, Chawathe Y, Consolvo SE. Place lab: device positioning using radio beacons in the wild. In *PerCom*, Berlin, Heidelberg, 2005; 301–306.
20. Otsason V, Varshavsky A, LaMarca A, De Lara E. Accurate gsm indoor localization. In *UbiComp 2005*, Berlin, Heidelberg, 2005; 141–158.
21. Zaruba G, Huber M, Kamangar F, Chlamtac I. Indoor location tracking using rssi readings from a single wi-fi access point. *Wireless Networks* 2007; **13**(2): 221–235.
22. Yei H, Gu T, Zhu X, Xu J, Tao X, Lu J, Jin N. FTrack: Infrastructure-free floor localization via mobile phone sensing. In *Proc. of the 10th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2012)*, Lugano, Switzerland, March 19-23, 2012.

AUTHORS' BIOGRAPHIES

Haibo Ye is a PhD student in the Department of Computer Science at Nanjing University. He received the BSc from the College of Computer and Information, HoHai University, in 2009. His current research interests include mobile and pervasive computing, wireless sensor networks, and human-computer interaction.

Tao Gu is an Associate Professor in the School of Computer Science and Information Technology at RMIT University. He received his Bachelor's degree from Huazhong University of Science and Technology, MSc from Nanyang Technological University, Singapore, and PhD in computer science from National University of Singapore. His current research interests include mobile and pervasive computing, wireless sensor networks, distributed network systems, sensor data analytics, cyber physical system, Internet of Things, and online social networks. He is a senior member of the IEEE and a member of the ACM.

Xianping Tao received the PhD degree in computer science from Nanjing University in 2001. He is currently a professor in the Department of Computer Science at Nanjing University. His research interests include software agents, middleware systems, Internetware methodology, and pervasive computing. He is a member of the IEEE.

Jian Lu received the PhD degree in computer science from Nanjing University in 1988. He is currently a professor in the Department of Computer Science at Nanjing University. He is also the director of the State Key Laboratory for Novel Software Technology. His research interests include programming methodology, pervasive computing, software agent, and middleware. He is a member of the ACM.