Finding 9-1-1 Callers in Tall Buildings

Wonsang Song
Columbia University
wonsang@cs.columbia.edu

Jae Woo Lee
Columbia University
jae@cs.columbia.edu

Byung Suk Lee
Columbia University
bsl@ee.columbia.edu

Henning Schulzrinne
Columbia University
hgs@cs.columbia.edu

ABSTRACT

Accurately determining a user's floor location is essential for minimizing delays in emergency response. This paper presents a floor localization system intended for emergency calls. We aim to provide floor-level accuracy with minimum infrastructure support. Our approach is to use multiple sensors, all available in today's smartphones, to trace a user's vertical movements inside buildings.

We make three contributions. First, we present a hybrid architecture for floor localization with emergency calls in mind. The architecture combines beacon-based infrastructure and sensor-based dead reckoning, striking the right balance between accurately determining a user's location and minimizing the required infrastructure. Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module addresses three core challenges that make it difficult to accurately derive displacement from acceleration. Third, we present the stairway module which determines the number of floors a user has traveled on foot. Unlike previous systems that track users' foot steps, our stairway module uses a novel landing counting technique.

Keywords

Floor localization, Indoor vertical location, Smartphone-based dead reckoning

1. INTRODUCTION

The emergency call systems in the United States and elsewhere are undergoing a transition from the PSTN-based legacy system to a new IP-based system. The new system is referred to as the Next Generation 9-1-1 (NG9-1-1) system [32] in the US. We have previously built a prototype NG9-1-1 system [18] based on the Session Initiation Protocol (SIP) [28].

The most important piece of information in the NG9-1-1 system is the caller's location. The location is first used for routing the call to a proper call center. The emergency responders then use the caller's location to pinpoint the caller on site. Therefore, it is essential to determine the caller's location as precisely as possible to minimize delays in emergency response. Delays in response may result in loss of lives.

In the NG9-1-1 system, GPS can provide a user's location accurately when the user makes an emergency call outdoors using a mobile phone. Indoor positioning, however, presents a challenge because GPS does not generally work indoors. Moreover, unlike outdoors, vertical accuracy is very important in indoor positioning because an error of few meters will send emergency responders to a different floor in a building, which may cause a significant delay in reaching the caller. The importance of vertical positioning makes GPS not a good solution even if GPS signals can somehow reach indoors, since the altitudes reported by GPS are usually inaccurate [21, 26].

Ladetto and Merminod [19] proposed a barometer-based solution for vertical positioning. Barometers, however, have a critical limitation when they are used in a vertical positioning system intended for emergency situations. Firefighters use a technique called positive pressure ventilation (PPV) [16], which means blowing air into a burning building in order to clear out smoke. PPV will result in pressure changes in the building, which will in turn cause large fluctuations in barometer readings. In addition, parts of some buildings are intentionally pressurized for various reasons [11], which will also affect barometer readings.

This paper presents a proposal to augment our previous NG9-1-1 prototype system with floor localization. We aim to provide floor-level accuracy with minimum infrastructure support. Our approach is to use multiple sensors, all available in today's smartphones, to trace a user's vertical movements inside buildings.

When a user enters a building, the user's smartphone receives the information about the building and the current floor from a beacon deployed at the entrance. The smartphone starts tracking the user's vertical movements when she rides elevators or walks on stairs. Additional beacons deployed sparsely throughout the building provide periodic corrections to the user's location.

Our design is largely driven by the requirements for emergency calls. First of all, a positioning system intended for emergency calls must be immune to transient conditions or on-going changes inside the building. For example, interfering electromagnetic signals, rearranged equipment and furniture, or the number of current occupants should not affect the system's operation. Because of this requirement, we had to rule out wireless fingerprinting, an effective technique used in many other indoor location systems [8,30,33]. Secondly, the infrastructure should be reduced as much as possible because an extensive infrastructure requirement hinders wide adoption. We chose a hybrid design, combining beacon-based infrastructure and sensor-based dead reckoning, in order to fill the gap between sparsely deployed beacons. Lastly, in an emergency call system, a partial failure must not result in a complete system failure. In our system, partial failures caused by power outage or structural damage in the building result in gradual degradation of performance.

In this paper, we make three contributions. First, we present a hybrid architecture for floor localization with emergency calls in mind. The architecture strikes the right balance between accurately determining a user's location and minimizing the required infrastructure.

Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module calculates the elevator's displacement by double-integrating vertical acceleration. Double integration is considered too noisy for tracking human movements in general. However, we show that the constrained movement of an elevator enables a number of error correction techniques, making double integration a viable method.

Third, we present the stairway module which determines the number of floors a user has traveled on foot. Previous proposals counted a user's steps on stairs [25, 38]. This approach has a critical limitation that it cannot account for a user walking up multiple stairs in each step. Instead, our stairway module counts landings, the level areas either at the top of a staircase or in between flights of stairs.

This paper is organized as follows. Section 2 presents our overall architecture. Section 3 describes the design and algorithms of our three analysis modules and the activity manager. Section 4 describes implementation details. Section 5 provides our evaluation results. Section 6 discusses related work. Lastly, we conclude and discuss future work in Section 7.

2. ARCHITECTURE OVERVIEW

Figure 1 shows the overall architecture of our vertical positioning system. We describe each component in detail in the following subsections.

2.1 Sensor array

The sensor array includes different kinds of sensors available in most of today's smartphones. The Inertial Measurement Unit (IMU) integrates a three-axis accelerometer, a three-axis gyroscope, and a three-axis magnetometer. Thus, the IMU provides motion sensing with a total of nine degrees of freedom. The accelerometer measures linear accelerations along the three spatial axes. The measured accelerations can be used to detect whether a user is moving, and if so, the user's velocity or traveled distance can be derived from them. The gyroscope measures the angular velocities of rotations around the three spatial axes. The orientation of the device can be derived from the gyroscope measurement. The magnetometer is a digital compass that measures the strength of the Earth's magnetic field. The compass provides the heading of the device. Heading refers to the angle which the device forms with the magnetic north on a level plane.

GPS provides the device's location in the geographic coordinates using satellite signals. GPS cannot be used indoors but it can help detect when a user moves from outdoors to indoors.

2.2 Analysis modules

The analysis modules collect data from the sensor array and compute a user's location. There are three analysis modules in our architecture: the elevator module, the stairway module, and the escalator module.

The elevator module calculates the vertical displacement of an elevator by measuring its linear acceleration. The linear acceleration is measured using the device's accelerometer. Integrating the linear acceleration twice with respect to time yields the distance that the elevator has traveled.

The stairway module determines the number of floors a user has traveled by counting the number of landings in stairways. Our landing detection algorithm is based on an intuitive fact that there is less vertical movement on landings than on steps. The stairway module utilizes the accelerometer, the gyroscope, and the magnetometer. We describe the details in Section 3.2.

The escalator module also calculates the vertical distance that a user has traveled by double-integrating the vertical component of the acceleration measurements, as we did in the elevator module. In both escalator and elevator modules, the vertical distance is converted to the number of floors by looking up the floor-to-floor heights. The user's smartphone receives the floor height information from the infrastructure components. We describe the infrastructure in Section 2.4.

2.3 Activity manager

The activity manager coordinates the interactions between the sensor array and the analysis modules. The

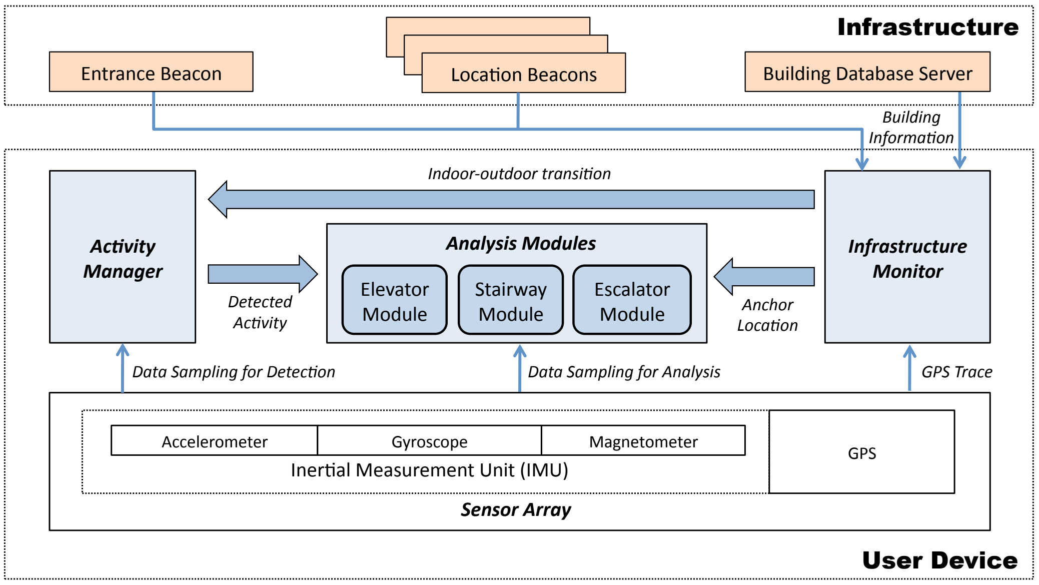


Figure 1: Architecture overview.

activity manager monitors the sensors to detect changes in a user's activity, such as indoor-outdoor transitions, riding an elevator, or walking on a stairway. Once the user's activity is identified, the activity manager selects the proper analysis module to process the data from the sensor array.

2.4 Infrastructure

As we will show in Section 5, the elevator, stairway, and escalator modules perform well within limited ranges, but the modules cannot reliably capture the user's movement over longer vertical distances. Moreover, the sensor-based components can only report *relative location*, i.e., the number of floors that the user has traveled. Therefore, the initial anchor location must be provided in order to obtain the *absolute location*.

These problems can be solved by deploying an infrastructure for indoor positioning. Densely deployed infrastructure, such as beacons installed every floor and every entrance, can provide accurate location, but the high cost of such installation is a hindrance to ubiquitous deployment, which is an important consideration for an emergency call system. On the contrary, sparsely deployed infrastructure will not be able to provide the required level of accuracy.

Our architecture combines sensor-based dead reckoning with minimum and practical beacon-based infrastructure. First, the infrastructure includes location beacons deployed at each entrance of a building. The beacons provide the location of a user's entry to the building. The floor of entry becomes the anchor for all subsequent calculations of the user's vertical location. In addition to the floor of entry, the beacons also provide other building information which is needed by the analysis modules. The additional building information includes the floor-to-floor height and the number of landings between each pair of floors. User devices include the infrastructure monitor, which interacts with the location beacons.

Second, for the buildings that are not equipped with these beacons, we propose that central authorities such as local governments maintain well-known building database servers. When a user enters a building not equipped with the beacons, the infrastructure monitor sends the last known GPS location to the building database server to retrieve the same building information that the location beacons would have provided. This GPS-based entrance detection is not as reliable as the beacon-based approach, especially in urban canyons. Thus, we only use it as a fallback.

Lastly, the limited range of the sensor-based components can be overcome by sparsely deploying location beacons at the edge of the range. For example, if the location tracked by the elevator module is reliable up to 20 floors, beacons can be placed at elevator entrances every 20 floors.

One advantage of our hybrid architecture is that partial failures caused by power outage or structural damage result in gradual degradation of performance rather than a complete system failure. If an entrance beacon fails, the smartphone will not have the initial anchor location and other building information, but it can still keep track of the user's relative location. If some location beacons are unavailable to provide periodic corrections, the system simply produces less accurate locations. This is an important characteristic of an emergency call system because even incomplete information can be helpful to first responders.

3. SYSTEM DESIGN AND ALGORITHMS

3.1 Elevator module

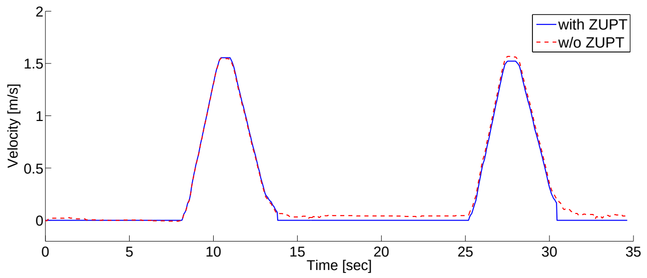
There are three challenges in accurately measuring the vertical distance that a user has traveled in an elevator. The three challenges are how to extract the vertical component in the accelerometer measurement, how to subtract Earth's gravitational acceleration, and how to eliminate noise and errors.

The accelerometer returns linear accelerations along the three axes. Those three axes are not aligned with the world coordinate system. Instead, they are aligned with the frame of the device. Thus, the axes in the device coordinate system keep changing as the orientation of the device changes. One way to extract vertical acceleration is to combine the accelerometer measurement with the gyroscope measurement. In fact, we do this in the stairway and escalator modules. In the elevator module, however, we take advantage of the fact that, in the elevator, the dominant movement of the device is in the vertical direction. We simply assume that the measured acceleration is close to vertical, and approximate the vertical projection with the vector itself. Thus, the vertical acceleration is calculated as follows:

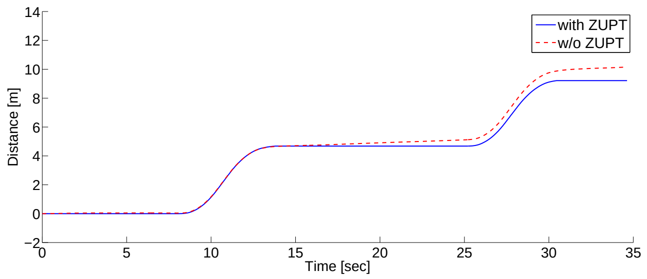
$$a_{vertical} \approx |\overrightarrow{a}| = \sqrt{x^2 + y^2 + z^2} \qquad (1)$$

where x, y, and z are three-axis accelerometer measurements. We do not need a gyroscope in this calculation. We justify our approach by making the following two observations. First, a user's sudden movements in the elevator will be filtered out by the low-pass filter, which we will describe shortly. Second, users typically stand still in the elevator, and when they move, the accelerations of the movements are small compared to the vertical acceleration of the elevator. The consequence of this approximation is that whenever there is non-vertical acceleration, we overestimate the vertical acceleration by $\frac{1}{\cos\theta}$, where θ is the angle that the measured acceleration vector makes with the vertical axis. This overestimation is small, and we compensate it by applying zero velocity update (ZUPT), which we describe later. Our measurement shows that the approximation does not affect the resulting distance calculation.

The vertical acceleration calculated above includes the gravitational acceleration (g), which we need to subtract before computing the traveled distance. In theory, g should be constant at $9.8\,\mathrm{m/s^2}$, but we found slight variations in our experiments. We measured g by sampling the accelerations of smartphones sitting still on a desk. The measured values deviated slightly from g, and moreover, the variations were different on different devices. Smartphone SDKs provide APIs returning g-free acceleration, but they exhibited the same deviation. We eliminate the effect of the deviation in g as follows. We take advantage of the fact that, if we take g out of the acceleration, the integral of the acceleration taken over the duration of the trip must be zero because the elevator is not moving at the end of the trip. Thus we can deduce that the value of g measured by the device is the mean of the acceleration samples taken over the trip.



(a) Velocity with and without ZUPT.



(b) Distance with and without ZUPT.

Figure 2: Comparison of the distance calculations with and without ZUPT.

The output from the accelerometer contains a significant amount of noise. We apply two existing techniques to tackle this problem. First, we apply a low-pass filter to the accelerometer output. This filters out the user's sudden movements and the accelerometer's inherent noise which we refer to as drift. Second, we apply a technique called zero velocity update (ZUPT) [14] to eliminate accumulated errors. Integrating the acceleration yields the velocity of the elevator. We reset the velocity to zero during the period when the acceleration is zero and the velocity is within a predefined threshold. The threshold value we choose is small compared to the speed of the elevator, so that we do not mistakenly zero out the velocity of an elevator moving at a constant speed. The accuracy of the distance calculation is improved in that, at each stop, ZUPT has an effect of wiping out the accumulated errors due to the drift and the user's non-vertical movements.

Figure 2 demonstrates the effectiveness of ZUPT. We compare the computed velocities and distances when an elevator traveled from the first, to the second, and then to the third floor. Without ZUPT, the accumulated acceleration errors result in non-zero velocities when the elevator is at the second and the third floor. This in

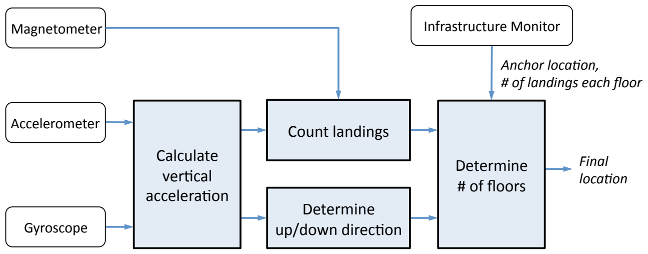


Figure 3: Overview of the stairway module.

turn results in an error of approximately one meter in the distance calculation at the end.

In general, double integration is considered too noisy for tracking human movements. In our case, however, an elevator moves only in the vertical axis, making it easy to extract the vertical component of the acceleration. An elevator also comes to a zero velocity when it stops at a floor, making it possible to apply ZUPT to eliminate the accumulated errors.

3.2 Stairway module

The stairway module determines the number of floors a user has traveled using our landing counting algorithm. To the best of our knowledge, landing detection has not been used for vertical positioning systems.

Figure 3 illustrates how the stairway module works. First, the stairway module calculates vertical acceleration from the accelerometer and gyroscope measurements. Unlike an elevator's movement, a user's movement on a stairway is more complex. A gyroscope is needed to transform the acceleration in the device coordinate system to the world coordinate system. We convert the accelerometer measurements in the device coordinate system to the world coordinate system using a rotation matrix as shown below:

$$\overrightarrow{a}' = R\overrightarrow{a} \tag{2}$$

where $\overrightarrow{a}'$ is the acceleration in the world coordinate system, $\overrightarrow{a}$ is the acceleration in the device coordinate system, and R is the rotation matrix. Most smartphone platforms provide an API to obtain R. We then take the resulting z-axis acceleration in the world coordinate system and subtract g from it. We calculate g in the same way as in the elevator module.

The landing counting algorithm compares the amplitude of vertical acceleration between steps and landings. The algorithm is based on the intuitive fact that the amplitude of the vertical acceleration is much smaller on landings than on steps because there are less vertical movements on landings.

Figure 4(a) shows a measurement of a user's vertical acceleration when she walks down four floors passing eight landings. The amplitude difference between steps and landings is clearly observed. Figure 4(b) is

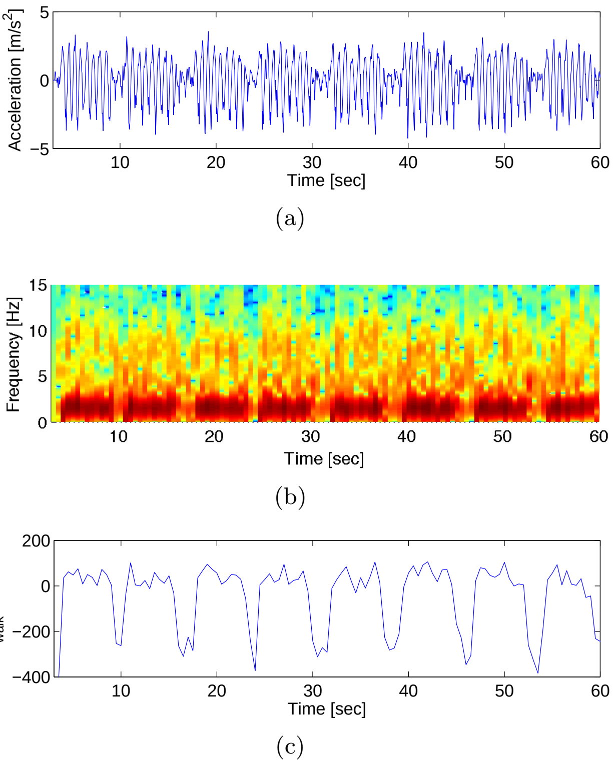


Figure 4: (a) Vertical acceleration measurement; (b) Spectrogram of vertical acceleration; (c) $p_{walk}(t)$ extracted from (b)

the magnitude spectrogram $|X(t, f)|$ in dB scale, transformed from Figure 4(a)'s acceleration data. The regions of small amplitude in Figure 4(a) manifest as reduced magnitude in the frequency range between 0.5 to 2 Hz, which corresponds to human walking.

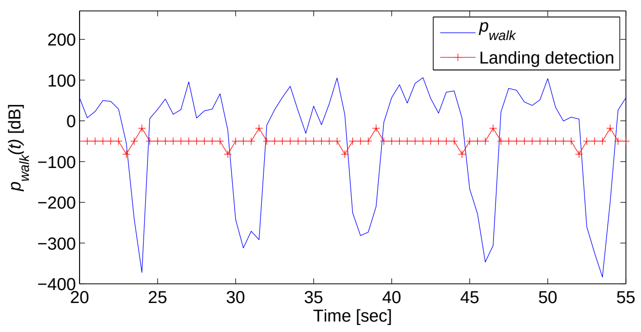
We define $p_{walk}(t)$ to extract human walking activity from the magnitude spectrogram:

$$p_{walk}(t) = \sum_{0.5\,\text{Hz}<f<2\,\text{Hz}} 10 \log_{10} |X(t, f)| \tag{3}$$

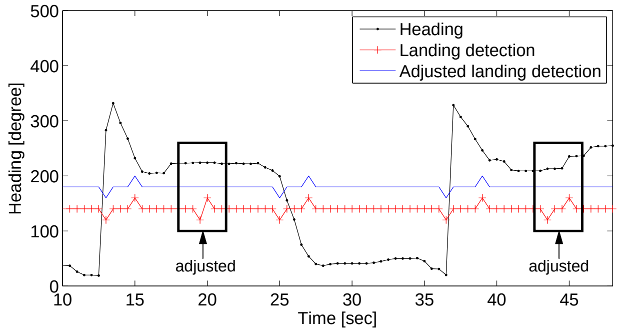
where t is time and f is frequency. Figure 4(c) shows $p_{walk}(t)$, where we can clearly observe the dips at landings.

Our landing counting algorithm traces the p_{walk} level shown in Figure 4(c) to count the number of landings. Figure 5(a) illustrates this process. Each landing is characterized by a dip below its mean value. The fall and rise of the level crossing the mean value indicate the beginning and end of a landing, respectively. The beginning and end of a landing are shown as the bumps of the "Landing detection" line in Figure 5(a).

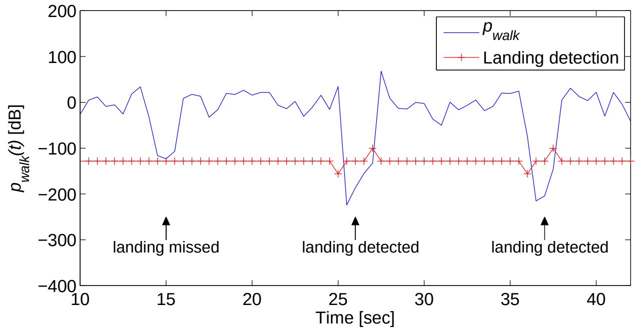
In addition to vertical acceleration, the stairway module uses heading information from the magnetometer to improve the accuracy of landing detection. Most of the time, users turn around 180 degrees on landings. We use such heading changes to correct errors in landing



(a) Case 1: no error.



(b) Case 2: false positives get fixed by magnetometer.



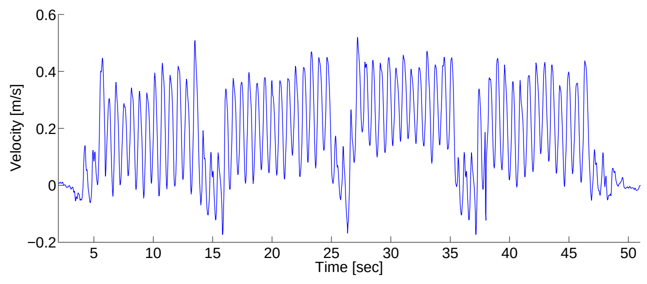
(c) Case 3: missed one landing.

Figure 5: Three landing detection cases.

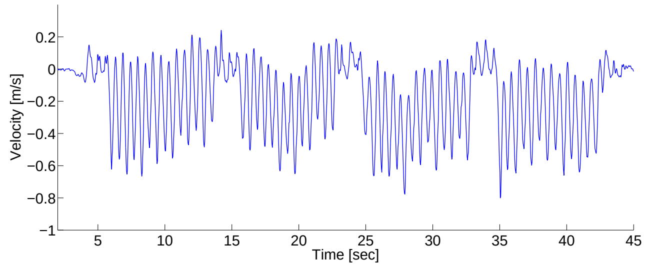
detection, specifically to remove incorrectly identified landings. Since we are only interested in 180 degree turns, our magnetometer reading does not require calibration.

Figure 5(b) shows a case where our algorithm removes two incorrectly identified landings using the heading information from the magnetometer. The dotted line labeled "Heading" shows the heading changes reported by the magnetometer. The heading largely stays the same from 15 sec to 25 sec, and changes from 220° to 40° in the next two seconds. This 180° turn, combined with the bumps on the landing detection line, confirms a landing. Note that the seeming discontinuity in the heading from 20° to 330° at 37 sec is in fact a steady change from 20° to −30°, wrapping around. The two rectangles in the figure highlight two incorrectly identified landings being removed because the heading did not change during the period.

This heading-based verification of landings makes it unlikely that our algorithm produces false positives. If the acceleration-based landing detection misses a landing to begin with, however, the heading information does not help recover it. Figure 5(c) shows this case. Therefore, our algorithm produces a conservative estimate of the number of landings.



(a) Walking up

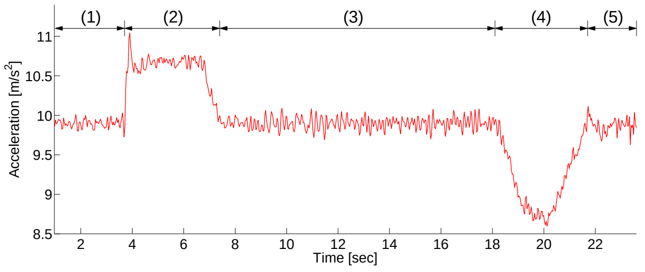


(b) Walking down

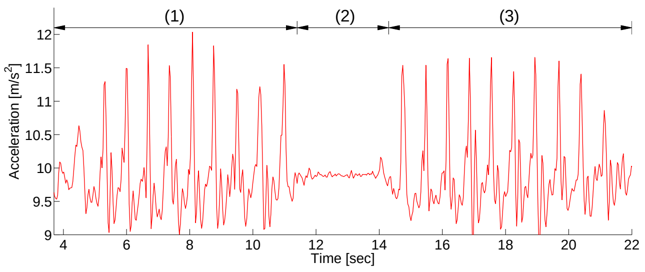
Figure 6: Velocity measurements of a user walking on a stairway.

We determine whether a user is moving up or down by comparing the average vertical velocity on steps and landings. Figure 6 shows the vertical velocity measurements when a user walks up and down two floors passing four landings. The figure clearly illustrates the difference in the velocity patterns between the up and down cases. We determine that the user is ascending if the velocity on steps is higher than the velocity on landings, and vice versa. In theory, the average vertical velocity should be zero on landings, positive when the user walking up steps, and negative when walking down. But in practice, the velocity values can shift due to the noise and errors that have been introduced while extracting vertical acceleration and subtracting g.

The stairway module returns a relative location which is the number of floors the user has traveled from the initial floor. Like the elevator module, the stairway module relies on the information from the infrastructure monitor to get the initial anchor location. The infrastructure monitor also provides the number of landings between each pair of floors. There are typically two landings per floor but the number can vary depending



(a) Elevator



(b) Walking

Figure 7: Acceleration of elevator and walking.

on the design of a building. In some buildings, for example, there are more landings between the lobby and the second floor.

3.3 Escalator module

The escalator module combines the elements of both elevator and stairway modules. The escalator module uses double integration like the elevator module. However, the user's movement on an escalator is not vertical, so we use the gyroscope measurements to extract the vertical component from the measured acceleration, as we did in the stairway module.

3.4 Activity manager

The activity manager classifies a user's movements as one of the following activities: elevator riding, walking, and standing. The classification is based on the user's vertical acceleration. The current version of the activity manager does not identify escalator riding. Vertical acceleration does not work well for escalators because of the complexity of a user's movement. We are investigating other ways to detect escalators, such as the technique of magnetic field variance proposed by Wang *et al.* [34].

Figure 7(a) depicts the pattern of a user's vertical acceleration when she is riding an elevator. The elevator starts with zero acceleration (1), accelerates to a steady velocity (2), moves at a constant speed (3), decelerates (4), and stops (5). When the activity manager detects this pattern, the sensor measurements during that period are passed to the elevator module.

Figure 7(b) shows a user's vertical acceleration when the user walks a few steps, stops for a bit, and then resumes walking. The activity manager detects human steps by identifying local extrema of amplitude in vertical acceleration. One step contains exactly one maximum and one minimum in a short time interval. The period (1) and (3) in Figure 7(b) contain human steps so they are walking periods.

The period (2) in Figure 7(b), where the vertical acceleration is under a threshold, is classified as a standing period. The activity manager uses the standing period to partition the sensor measurements.

Each walking period, separated by the standing periods, is passed to the stairway module. A single walking period can be either walking on the stairway (stairway walking) or walking on the same floor (same-floor walking). Ideally, the activity manager should detect all periods of same-floor walking, and filter them out, so that they do not get passed to the stairway module. The current version of our activity manager does not implement this filtering. Thus, the stairway module needs to handle not only stairway walking (as described in detail in Section 3.2), but also same-floor walking.

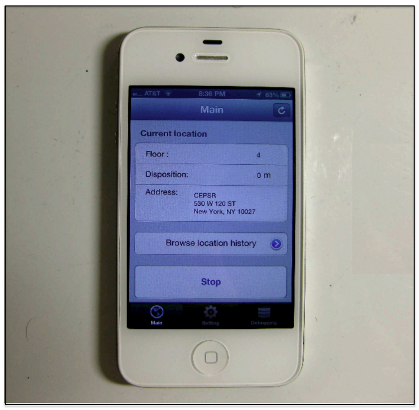
To detect same-floor walking, the stairway module calculates the total distance that a user has traveled during the walking period, and see if the distance turns out to be close to zero. If so, the walking period is considered to be an instance of same-floor walking.

Normally, a single walking period does not contain both stairway and same-floor walking. A user would typically stop to open a door to the stairway, producing a standing period which separates them into two walking periods. It is possible, however, that both stairway and same-floor walking are included in a single walking period if the user avoids stoppage in the middle. In this case, the same-floor walking portion will be detected as a landing by the stairway module, assuming that the user has made a significant change in the heading during the same-floor walking. The only case that the stairway module will not be able to handle is the one where the user walks a long straight corridor between two flights of stairs without stoppage. Our stairway module will not identify this as a landing due to the lack of any heading change.

4. IMPLEMENTATION

4.1 Hardware platform

For the implementation and evaluation, we used the Apple iPhone 4 and 4S, running iOS version 6. The iPhone 4 contains an accelerometer, a gyroscope, and a magnetometer. The accelerometer in iPhone 4 can measure acceleration from $-2\,g$ to $+2\,g$, where $1\,g$ is $9.8\,\mathrm{m/s^2}$ [29]. The sampling rate can be adjusted from 0.5 Hz to 1 kHz. We used 30 Hz for our measurements.



Main

Current location

Floor : 4

Disposition: 0 m

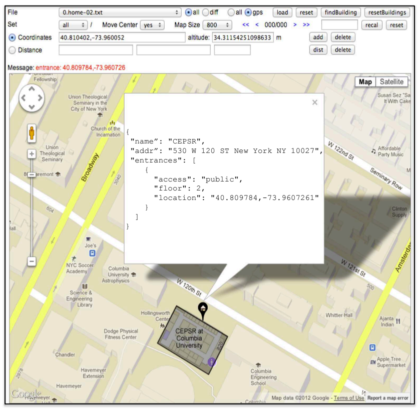
Address: CEPSR
530 W 120 ST
New York, NY 10027

Browse location history

Stop

(a) iPhone app for data collection.

(b) Foil-wrapped Mac mini as Bluetooth beacon.



File: 0.home-02.txt (● all ○ diff) (○ all ● gps) load reset findBuilding resetBuildings

Set: all / Move Center: yes Map Size: 800 << < 000/000 > >> recal reset

● Coordinates: 40.810402,-73.960052 altitude: 34.31154251098633 m add delete

○ Distance dist delete

Message: entrance: 40.809784,-73.960726

```
{
 "name": "CEPSR",
 "addr": "530 W 120 ST New York NY 10027",
 "entrances": [
    {
      "access": "public",
      "floor": 2,
      "location": "40.809784,-73.9607261"
    }
  ]
}
```

(c) Web interface to the building database server.

Figure 8: Prototype implementation.

The gyroscope measures angular velocity from -250 degree/sec to +250 degree/sec [29]. We also read the gyroscope at 30 Hz. The magnetometer is a three-axis electronic compass manufactured by Asahi Kasei [10]. According to the specification from the manufacturer, the measurement range is $\pm 1{,}200\,\mu$T.

Most smartphones based on Google's Android platform are also equipped with the same set of sensors with comparable specifications. Both iOS and Android offer APIs to access the sensors.

Data	Unit	Source
Timestamp	ms	System clock
X-axis acceleration	g	Accelerometer
Y-axis acceleration	g	Accelerometer
Z-axis acceleration	g	Accelerometer
Rotation matrix	N/A	Accel. & Gyro.
Heading	degree	Magnetometer
Latitude & Longitude	degree	GPS

Table 1: Data types collected on iPhone.

4.2 Data collection from sensor array

In our current prototype, an application running on iPhone collects data from the sensor array. Figure 8(a) shows the screenshot. Table 1 provides the list of data types we collect on iPhone. The measurements from the accelerometer and gyroscope in iPhone can be accessed using the Core Motion framework in iOS. The Core Motion framework provides APIs to retrieve the raw data such as the timestamp and three-axis accelerations shown in Table 1. The framework also provides processed motion data, such as *attitude*, which is derived from both the accelerometer and gyroscope. Attitude refers to the spatial orientation of the device with respect to the world coordinates, and can be obtained either as a rotation matrix or as a quaternion. We use the rotation matrix in our implementation of the stairway module.

The heading information from the magnetometer can be accessed using the Core Location framework in iOS. The framework provides two headings: magnetic heading and true heading. Magnetic heading points to the magnetic north pole, and true heading points to the geographic north pole. We use the magnetic heading in our implementation. Both types of heading will satisfy our need to detect a user turning around on landings, but using magnetic heading avoids additional processing to calculate the true heading from the current location, which may consume more energy.

We collect GPS traces outdoors. The Core Location framework provides an API to obtain the device's location. Normally the framework determines the locations using various sources including GPS, Wi-Fi, and cellular network, but a flag can be passed to indicate that we are only interested in GPS locations.

4.3 Data collection from infrastructure

In this section, we describe the infrastructure monitor running on the user's smartphone, the location beacons deployed in the building, and the central building database server.

We chose Bluetooth technology for location beacons because Bluetooth is available on most smartphones. The infrastructure monitor and the beacon communicate using service discovery protocol (SDP) [6]. SDP allows Bluetooth devices to discover available services and their characteristics without initiating a pairing process.

Currently, iOS does not provide APIs for Bluetooth communication. We implemented the infrastructure monitor using BTstack [2], an open source Bluetooth stack for iOS. Installing BTstack requires jailbreaking iPhone.

We used a Mac mini computer wrapped in aluminum foil to prototype a Bluetooth beacon as shown in Fig-

Data item	Default
Floor of entry	1
Number of landings between floors	2
Floor-to-floor height	3.5 m

Table 2: Predefined default values for building information.

ure 8(b). The foil wrapper decreases the Bluetooth signal strength so that it does not reach the adjacent floors. The Bluetooth beacon is written as a Java application using BlueCove [1], an open source Java library for Bluetooth.

The beacon interacts with the smartphone's infrastructure monitor in the following sequence. First, the infrastructure monitor scans for nearby Bluetooth devices by sending periodic inquiry messages. Second, the infrastructure monitor sends an SDP request to all the discovered Bluetooth devices. The request includes a unique identifier defined for location beacon service, so the request is ignored by all devices that are not location beacons. Lastly, the infrastructure monitor receives an SDP response from a location beacon. An SDP response contains the building's address, the floor where the beacon is located, and for each pair of floors, the height and the number of landings.

The infrastructure monitor falls back on a central building database server when a building is not equipped with location beacons. While a user stays outdoors, the infrastructure monitor tracks the user's location using GPS. When GPS signal is lost, the infrastructure monitor assumes that the user has entered a building, and sends the last known GPS coordinates to the building database server. The building database server finds the nearest entrance from the user's last GPS location, and returns the same information that the location beacon returns. Figure 8(c) is the web admin interface to the building database server that we have built for testing, displaying the information about a building in Columbia University campus.

Table 2 shows the default values for building information when the infrastructure monitor fails to obtain the information specific to a building. The default value for the floor-to-floor height is the average value for commercial and residential buildings reported by the Council on Tall Buildings and Urban Habitat [9].

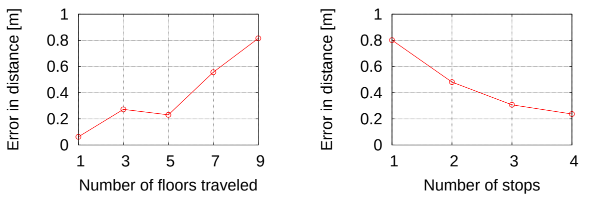
4.4 Analysis modules

The current version of our iPhone application does not include the analysis modules. The collected sensor data is sent to a central repository. Using this data, we have tested our algorithms for the analysis modules prototyped in MATLAB.

We are currently developing the analysis modules running on iPhone. It is desirable to run all analysis locally on the user's device whenever possible, so that the user's privacy is preserved as much as possible.

Building name	Reference floor height	Average error	Error-to-height ratio
CEPSR	4.65 m	0.08 m	1.6%
Mudd	3.67 m	0.06 m	1.7%
Pupin	3.48 m	0.09 m	2.7%

Table 3: Errors in one floor distance calculated by elevator module.



(a) Traveling 1, 3, 5, 7, 9 floors without stopping.
(b) Traveling 9 floors with increasing number of stops.

Figure 9: Distance errors of elevator module measured in Mudd building.

5. EVALUATION

First, we evaluate the algorithms of our elevator, stairway, and escalator modules individually. The individual evaluation scenarios assume that the activity manager correctly identifies the user's activity and selects the proper analysis module. Then, we present a combined case where the user's travel involves multiple types of movements including riding an elevator, walking on a stairway, and walking around on the same floor, which are all detected by the activity manager.

5.1 Elevator module

We evaluated the elevator module in three different research and classroom buildings at Columbia University: CEPSR, Mudd, and Pupin. They have 10, 15, and 13 floors, respectively. Table 3 shows the reference floor-to-floor height of each building, which we measured using a tape measure, followed by the error of the result from the elevator module. The error is the difference between the reference height and the distance calculated by the elevator module when a user moves one floor in an elevator in each building. The error is an average of ten trials, five moving up and five moving down.

Errors are small in all three buildings, indicating that the elevator module can provide accurate vertical location up to a reasonable number of floors. We can extend the range by strategically deploying location beacons. For example, in the Pupin case in Table 3, the error is under 3%, so the elevator module will be accurate up to about 15 floors. Thus, location beacons can be

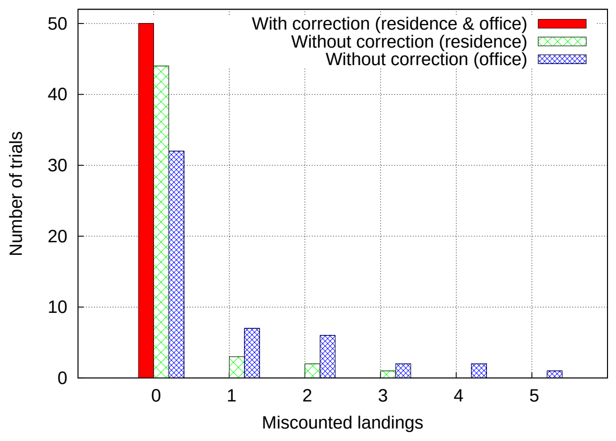


Figure 10: Stairway module measurements of 50 trials of walking four floors. The heading-based correction eliminated all miscounted landings in both buildings.

deployed conservatively in every 10 floors to cover the entire building.

Figure 9(a) shows distance errors from the elevator module as we increase the number of floors traveled in an elevator without stopping. The graph shows that the errors accumulate as the elevator travels farther. The error of 0.82 m when the user traveled nine floors is about 22% of the floor-to-floor height, which is well within the margin of error for accurately determining the destination floor.

Figure 9(b) plots the distance errors of traveling nine floors in an elevator as we vary the number of stops that the user has made during the travel. The graph shows that the error decreases as the user makes more stops. This shows the effectiveness of applying ZUPT in the distance calculation. At each stop, ZUPT eliminates accumulated errors by removing residual velocity. Therefore, if the elevator makes stops during the trip, the elevator module's distance estimation becomes much more accurate, extending the upper bound of the elevator module's distance limitation.

5.2 Stairway module

We evaluated the stairway module in two buildings. One was an office building and the other was a residential building. Both buildings have two landings between each pair of floors.

Figure 10 shows our stairway module measurements. In each building, we performed 50 trials of walking four floors. The graph compares the landing counting results with and without our heading-based correction algorithm described in Section 3.2. Our heading-based correction was able to eliminate all miscounted landings in both buildings, producing the correct landing count in all 50 trials. Without the heading-based correction, only 44 and 32 trials produced the correct landing count in the residential and the office building, respectively. The graph shows the number of trials that produced one or more miscounted landings in each building. For instance, two trials in the office building miscounted four landings, which would have resulted in an error of two floors, if the heading-based correction had not been applied.

Figure 10 also shows that, without the heading-based correction, the stairway module performs better in the residential building than in the office building. We attribute this difference to the steeper stairs in the residential building. The difference in the vertical acceleration between steps and landings is more pronounced on the steeper stairs. In general, our landing detection works better when the amplitude difference in the acceleration is pronounced. This is also in line with our observation that the waveforms are generally cleaner in the walking-down cases than in the walking-up cases. Human steps are typically a bit bouncier when walking down.

We note that, in all trials in Figure 10, the user moved at a normal walking speed. If the user walks very fast or very slowly, the amplitude difference of the accelerometer reading between steps and landings is much less pronounced. We can address this issue by giving more weight to the heading information from the magnetometer. In the extreme case, we can reverse the roles of the accelerometer and the magnetometer, i.e., instead of using the magnetometer to make adjustments to the landings identified by the accelerometer, we can use the magnetometer first to identify landings. The relative weights of the two sensors can be dynamically determined depending on how pronounced the amplitude difference is.

The iPhone's magnetometer readings, however, often showed large fluctuations in our experiments even when the user did not change direction. For this reason, we chose to use the magnetometer conservatively, i.e., only for correcting false positives. In order to see the effectiveness of the magnetometer-first approach, we conducted the same experiment with the user walking very fast and very slowly, and selected the measurements that did not contain incorrect magnetometer readings. We confirmed that the magnetometer-first approach, when the magnetometer readings are reliable, can cover a wider range of human walking speed.

5.3 Escalator module

We evaluated the escalator module in a building where the escalator connects the second and the fourth floor. We used a tape measure to obtain the reference height between the two floors: 7.3 m.

Figure 11 shows the CDFs of the error in the distance reported by the escalator module. The figure compares

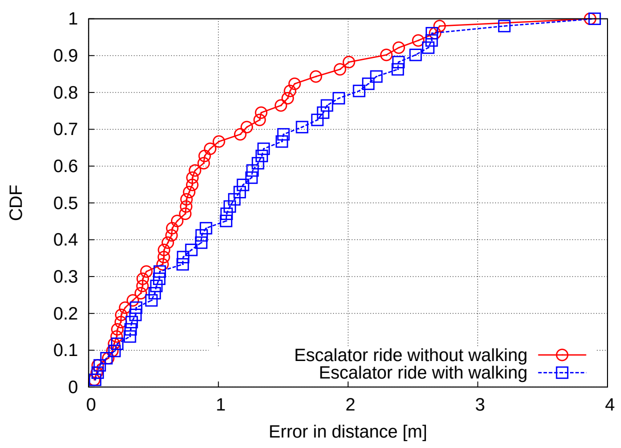


Figure 11: CDF of error in the distance measurement by the escalator module.

two cases. In one case, the user stood still while riding the escalator. In the other case, the user was also walking during the ride. For each case, the user took the escalator 50 times.[1]

In the case without walking, the 50% and 80% of the results are within 0.75 m and 1.5 m, respectively. In the case with walking, 50% and 80% are within 1.1 m and 2.1 m, respectively. As we expected, walking on the escalator generates more noise in the vertical acceleration, causing larger errors in the distance calculation.

The 2.1 m error–the 80th percentile in the case with walking–is about 29% of the reference height between the 2nd and 4th floors. This error-to-height ratio is large compared to the elevator cases. This is because, unlike an elevator where the movement starts and ends at a standstill, the user steps on and off an escalator that is constantly moving, making it harder to separate the acceleration purely due to the escalator.

However, an escalator typically covers no more than 2-3 floors, so the error in the distance measurement will still not cause an error in determining the number of floors. If a user rides a series of escalators one after another, the error from one ride will not carry over to the next one because we can apply ZUPT at landings.

5.4 Combined case

Figure 12 shows our evaluation scenario of a case involving multiple types of movement: elevator up/down, stairway up/down, and same-floor walking. The activity manager detects each activity and sends the sensor measurements to the corresponding modules.

The travel scenario consists of seven steps. First, a user takes an elevator on the 7th floor (0), and gets off on the 10th floor (1). The user then walks on the 10th floor to the door to the stairway (2). She walks down the stairway from the 10th to the 8th floor (3). On the 8th floor, she comes out of the stairway, walks to the other side of the floor to enter another stairway (4). She walks up the stairway from the 8th back to the 10th floor (5). On the 10th floor, she comes out of the stairway, and walks to the elevator entrance in the middle of the floor (6). Finally, she takes the elevator on the 10th floor and goes down to the 7th floor, back to where she started (7).

Table 4 shows the floor levels reported at each stage in Figure 12 when we repeated the travel ten times. In eight out of the ten trials, the correct floor was reported at every stage. In the two remaining trials, the stairway module failed to recognize the walking period between (3) and (4) as same-floor walking. The module incorrectly reported one landing instead of zero. This caused an error of $\frac{1}{2}$ floor in the subsequent stages.

The evaluation results show that our system can successfully track the user's complex movements in general. At the same time, the errors in the two trials reveal the weakness in our system: it may fail to distinguish between stairway walking and same-floor walking. This is indeed a difficult problem that remains as an active research area. At the time of this writing, other activity recognition systems have similar success rates [37].

[1]Out of the 50 walking trials, the user walked during the entire ride 20 times, only the first half 15 times, and only the second half 15 times. We do not show these cases separately because there was no significant difference between them.

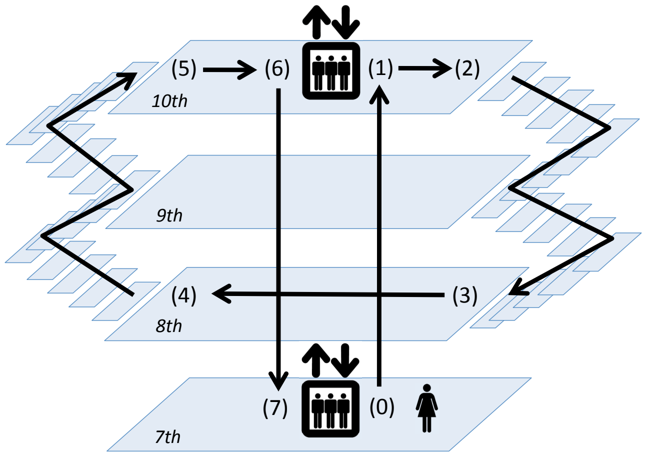


Figure 12: A person's travel between 7th and 10th floor including elevator riding, stairway walking, and same-floor walking.

# of trials	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
8x	7	10	10	8	8	10	10	7
2x	7	10	10	8	**8.5**	**10.5**	**10.5**	**7.5**

Table 4: Floor levels reported at each stage.

6. RELATED WORK

6.1 Fingerprinting

Fingerprinting identifies signals that have long-term stability at each location. During the *offline* phase, the signal strengths at different location coordinates are recorded to build a fingerprinting database. During the *online* phase, the real-time signal measurement is looked up in the fingerprinting database to find a matching location.

Many kinds of signals have been used for fingerprinting systems. RADAR [5], Place Lab [20], Horus [40], FindingMiMo [30], and Ekahau [3] use ubiquitous Wi-Fi signals. WALRUS [7] and Centaur [23] combine Wi-Fi and acoustic localization. SkyLoc [33] uses GSM signals. MoteTrack [22] is based on RF signals generated by low-power radio transceivers. There are also a number of systems that use the distortions of Earth's magnetic field caused by the steel structure of a building [4, 8, 13].

Some of the fingerprinting-based systems consider vertical location. SkyLoc [33] was in fact the first floor localization paper. SkyLoc was also motivated by the importance of floor localization in emergency situations, and it tackled the problem using GSM fingerprinting. Shin *et al.* [30] and Chung *et al.* [8] also considered floor-level vertical localization as part of their system using Wi-Fi signals and geo-magnetism, respectively.

One disadvantage of fingerprinting is the effort required to conduct offline surveys. To achieve an acceptable accuracy, signals should be sampled at every meter, and on top of that, at least toward four different directions at each location [5], which generates an enormous amount of data.

Moreover, fingerprinting-based approaches are vulnerable to transient conditions like interfering electromagnetic signals, or on-going changes like rearranged equipment and furniture. This characteristic makes fingerprinting an unsuitable approach for emergency call systems.

6.2 Dead reckoning

Systems based on dead reckoning typically measure a user's acceleration, and calculate the distance by double integration. The main disadvantage of double integration is that the accumulation of errors degrades the accuracy of the distance estimation over time. For this reason, many indoor location systems instead count human steps to estimate the distance traveled. A step-based system usually requires an initial training period to determine the length of a user's stride. Yet another way to implement dead reckoning is to measure the travel time. The traveled distance can then be calculated from the predetermined velocity, which is commonly obtained through training.

Among the systems based on double integration, two systems [12, 24] use foot-mounted IMU. Mounting a sensor on the user's foot enables ZUPT to achieve a significant reduction of errors. Xuan *et al.* [36] and Shanklin *et al.* [29] use smartphones to develop indoor positioning systems. Both systems do not reach the accuracy of the foot-mounted systems because of the lack of adequate mechanisms to handle the accelerometer drift.

Our elevator module calculates the elevator's displacement by double-integrating vertical acceleration captured by smartphones. Unlike the other smartphone-based systems, we are able to apply ZUPT and other error correction techniques because an elevator moves only in the vertical axis, and its velocity becomes zero when it stops at a floor.

Step-based systems [15, 17, 31] detect human steps by identifying the local maximum and minimum of vertical acceleration. A pair of local maximum and minimum within a short time period identifies one human step. Our activity manager uses this approach to detect the act of walking, but it does not need to count the steps.

Our stairway module similarly monitors the amplitude of vertical acceleration. The difference is that, instead of trying to identify each and every step by scrutinizing vertical acceleration, we detect landings by focusing on large amplitude changes in acceleration, which are easier to identify.

Two systems [25, 38] implement floor localization using the time-based approach. Both systems track a user's movement in elevators and on stairways. In [25], a user's current activity is classified into one of four classes, elevator up/down and stairs up/down, using the smartphone's real-time accelerometer data. The system then estimates the number of floors that the user has traveled simply by dividing the total travel time by the time it takes to travel one floor. This system requires a training period to build a classifier for each activity and to calculate the average times needed to travel one floor. FTrack [38] takes a similar approach, but uses a novel crowdsourcing technique to construct a mapping from the starting floor and travel time to the destination floor. Crowdsourcing, however, requires the willing participation of a large number of users, which may not be feasible. In addition, there is still a privacy concern during FTrack's offline map construction phase.

The main disadvantage of the two time-based approaches is that it cannot take account of speed variations. Different elevators can have different speeds. Users may walk at different speeds on stairs, or may even climb up multiple stairs in each step. Our system do not have such limitations. Our elevator and escalator module is distance-based, rather than time-based. Our stairway module works by counting landings, rather than counting individual steps or measuring the travel time.

6.3 Hybrid systems

Hybrid systems combine infrastructure and dead reckoning in order to overcome the shortcomings associated with taking a single approach. The estimated locations from dead reckoning are periodically adjusted by the information from the infrastructure, such as RFID beacons [39] or Wi-Fi fingerprinting [27, 35]. Beacons in this case can be deployed in much coarser granularity compared to the systems purely based on infrastructure.

Woodman and Harle [35] proposed a hybrid indoor localization system that uses dead reckoning and Wi-Fi fingerprinting to track a pedestrian through multiple floors. Their system tracks the user's movement using a foot-mounted IMU, and aligns the user's path with the floor plan of the building. Wi-Fi fingerprinting constrains the possible initial locations into a particular region of the building, which in turn reduces the complexity of the alignment algorithm, and resolves the ambiguity arising from the symmetries in the floor plan. Zee [27] also combines dead reckoning and Wi-Fi fingerprinting, but for a different purpose. Zee aims to eliminate the calibration effort required to build the Wi-Fi fingerprinting database.

Our system can be viewed as a hybrid system because we primarily rely on dead reckoning, but we anchor the user's location using the information from the entrance beacon. The user's location is also checked and adjusted by the location beacons sparsely deployed throughout the building.

7. CONCLUSION AND FUTURE WORK

This paper makes three contributions toward improving vertical accuracy of indoor positioning. First, we present a hybrid architecture for floor localization with emergency calls in mind. The architecture combines beacon-based infrastructure and sensor-based dead reckoning, striking the right balance between accurately determining a user's location and minimizing the required infrastructure. Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module addresses three core challenges that make it difficult to accurately derive displacement from acceleration. Third, we present the stairway module which determines the number of floors a user has traveled on foot. Unlike previous systems that track users' foot steps, our stairway module uses a novel landing counting technique.

We recognize that there are many hurdles to overcome before our system can be deployed in the real world. For instance, our elevator module assumes that the acceleration inside an elevator is mostly vertical. This will not be the case if a user happens to pace back and forth during the ride. Similar shortcomings also exist in the stairway module. The stairway module can produce false positives in some unusual cases. For example, a user can stop in the middle of a stairway, slowly turn around 180 degrees, and walk the rest of the stairway backward. This is highly unlikely, but it illustrates the general limitation of our approach that relies on behavioral norms. As future work, we plan to study the effects of various unusual behaviors, and explore possible solutions to address them.

We also plan to improve activity detection using ambient signals. For example, an entry to a building can be detected using the RFID signals from anti-theft gates, which are typically installed at the entrances of libraries and retail stores. Identifiable magnetic signatures can be detected around elevators and escalators. Even though we have argued against relying on a barometer for vertical location, a barometer can be useful as an additional input to distinguish between stairway and same-floor walking.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 0751094.

8. REFERENCES

[1] BlueCove. `http://bluecove.org/`.

[2] BTstack: A Portable User-Space Bluetooth Stack. `http://code.google.com/p/btstack/`.

[3] Ekahau - Wi-Fi Tracking Systems, RTLS and WLAN Site Survey. `http://www.ekahau.com/`.

[4] IndoorAtlas. `http://www.indooratlas.com/`.

[5] P. Bahl and V. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. of INFOCOM*, 2000.

[6] P. Bhagwat. Bluetooth: Technology for Short-Range Wireless-Apps. *IEEE Internet Computing*, 5(3):96–103, 2001.

[7] G. Borriello, A. Liu, T. Offer, C. Palistrant, and R. Sharp. WALRUS: Wireless Acoustic Location with Room-Level Resolution using Ultrasound. In *Proc. of MobiSys*, 2005.

[8] J. Chung, M. Donahoe, C. Schmandt, I. Kim, P. Razavai, and M. Wiseman. Indoor Location Sensing Using Geo-Magnetism. In *Proc. of MobiSys*, 2011.

[9] Council on Tall Buildings and Urban Habitat. Height Calculator. `http://http://www.ctbuh.org/TallBuildings/HeightStatistics/HeightCalculator/tabid/1007/language/en-GB/Default.aspx`.

[10] S. Dixon-Warren. Motion Sensing in the iPhone 4: Electronic Compass. `http://www.memsjournal.com/2011/02/motion-sensing-in-the-iphone-4-electronic-compass.html`.

[11] R. Falke. Building Pressure Diagnostics. `http://contractingbusiness.com/enewsletters/cb_imp_9596/`.

[12] E. Foxlin. Pedestrian Tracking with Shoe-Mounted Inertial Sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[13] B. Gozick, K. Subbu, R. Dantu, and T. Maeshiro. Magnetic Maps for Indoor Navigation. *IEEE Transaction on Instrumentation and Measurement*, 60(12):3883–3891, 2011.

[14] D. Grejner-Brzezinska, Y. Yi, and C. Toth. Bridging GPS Gaps in Urban Canyons: The Benefits of ZUPTs. *Navigation*, 48(4):217–225, 2001.

[15] Y. Jin, H. Toh, W. Soh, and W. Wong. A Robust Dead-Reckoning Pedestrian Tracking System with Low Cost Sensors. In *Proc. of PerCom*, 2011.

[16] S. Kerber and W. Walton. Effect of Positive Pressure Ventilation on a Room Fire. Technical Report NISTIR-7213, National Institute of Standards and Technology, Mar. 2005.

[17] J. Kim, H. Jang, D. Hwang, and C. Park. A Step, Stride and Heading Determination for the Pedestrian Navigation System. *Journal of Global Positioning Systems*, 3(8):273–279, 2004.

[18] J. Kim, W. Song, H. Schulzrinne, A. Zacchi, A. Jain, H. Chenji, C. Magnussen, C. Norton, W. Magnussen, I. Schworer, and K. Trinh. The Next Generation 9-1-1 Proof-Of-Concept System. In *ACM SIGCOMM Demo*, 2008.

[19] Q. Ladetto and B. Merminod. In Step with INS: Navigation for the Blind, Tracking Emergency Crews. *GPS World*, 13(10):30–38, 2002.

[20] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proc. of Pervasive*, 2005.

[21] G. Lammel, J. Gutmann, L. Marti, and M. Dobler. Indoor Navigation with MEMS sensors. *Procedia Chemistry*, 1(1):532–535, 2009.

[22] K. Lorincz and M. Welsh. MoteTrack: a robust, decentralized approach to RF-based location tracking. *Personal and Ubiquitous Computing*, 11(6):489–503, 2006.

[23] R. Nandakumar, K. K. Chintalapudi, and V. N. Padmanabhan. Centaur: Locating Devices in an Office Environment. In *Proc. of MobiCom*, 2012.

[24] L. Ojeda and J. Borenstein. Personal Dead-reckoning System for GPS-denied Environments. In *Proc. of IEEE SSRR*, 2007.

[25] A. Parnandi, K. Le, P. Vaghela, A. Kolli, K. Dantu, S. Poduri, and G. Sukhatme. Coarse In-building Localization with Smartphones. In *Proc. of MobiCASE*, 2009.

[26] J. Parviainen, J. Kantola, and J. Collin. Differential Barometry in Personal Navigation. In *Proc. of IEEE/ION PLANS*, 2008.

[27] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *Proc. of MobiCom*, 2012.

[28] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, 2002.

[29] T. Shanklin, B. Loulier, and E. Matson. Embedded Sensors for Indoor Positioning. In *Proc. of IEEE SAS*, 2011.

[30] H. Shin, Y. Chon, K. Park, and H. Cha. FindingMiMo: Tracing a Missing Mobile Phone using Daily Observations. In *Proc. of MobiSys*, 2011.

[31] U. Steinhoff and B. Schiele. Dead Reckoning from the Pocket - An Experimental Study. In *Proc. of PerCom*, 2010.

[32] U.S. Department of Transportation. Next Generation 9-1-1. `http://www.its.dot.gov/ng911/`.

[33] A. Varshavsky, A. LaMarca, J. Hightower, and E. de Lara. The SkyLoc Floor Localization System. In *Proc. of PerCom*, 2007.

[34] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No Need to War-Drive: Unsupervised Indoor Localization. In *Proc. of MobiSys*, 2012.

[35] O. Woodman and R. Harle. Pedestrian Localisation for Indoor Environments. In *Proc. of Ubicomp*, 2008.

[36] Y. Xuan, R. Sengupta, and Y. Fallah. Making Indoor Maps with Portable Accelerometer and Magnetometer. In *Proc. of UPINLBS*, 2010.

[37] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-Efficient Continuous Activity Recognition on Mobile Phones: An Activity-Adaptive Approach. In *Proc. of ISWC*, 2012.

[38] H. Ye, T. Gu, X. Zhu, J. Xu, X. Tao, J. Lu, and N. Jin. FTrack: Infrastructure-free Floor Localization via Mobile Phone Sensing. In *Proc. of PerCom*, 2012.

[39] S. Yeh, K. Chang, C. Wu, H. Chu, and J. Hsu. GETA sandals: a footstep location tracking system. *Personal and Ubiquitous Computing*, 11(6):451–463, 2006.

[40] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *Proc. of MobiSys*, 2005.