Towards Mobile Phone Localization without War-Driving

Ionut Constandache
Duke University
Durham, NC, USA
Email: ionut@cs.duke.edu

Romit Roy Choudhury
Duke University
Durham, NC, USA
Email: romit@ee.duke.edu

Injong Rhee
North Carolina State University
Raleigh, NC, USA
Email: rhee@ncsu.edu

Abstract—**This paper identifies the possibility of using electronic compasses and accelerometers in mobile phones, as a simple and scalable method of localization without war-driving. The idea is not fundamentally different from ship or air navigation systems, known for centuries. Nonetheless, directly applying the idea to human-scale environments is non-trivial. Noisy phone sensors and complicated human movements present practical research challenges. We cope with these challenges by recording a person's *walking patterns*, and matching it against possible *path signatures* generated from a local electronic map. Electronic maps enable greater coverage, while eliminating the reliance on WiFi infrastructure and expensive war-driving. Measurements on Nokia phones and evaluation with real users confirm the anticipated benefits. Results show a location accuracy of less than $11m$ in regions where today's localization services are unsatisfactory or unavailable.**

I. Introduction

Recent years have witnessed the explosion of location based applications (LBAs). The iPhone App Store now features 3,000 LBAs. The Android community already lists 400 services, with the number growing rapidly every month [1]. Localization technology is projected to play a critical role in the future, ushering in applications such as location-based advertising, friend-tracking, micro-blogging, etc. At the outset of this explosion, GPS was primarily used for localization. However, Place Lab [2] and Skyhook [3] identified problems with GPS, including poor indoor operations, short battery life, and long acquisition time. Alternative solutions were proposed to exploit pre-existing infrastructure for localization. The basic idea is to *war-drive* an area and create a map of existing WiFi/GSM access points – this map is then made available to mobile devices. As a mobile device enters a mapped area, it computes its location by detecting WiFi/GSM access points, and searching for them in its stored radio map. Localization became feasible even in indoor environments while the location acquisition time reduced significantly. Overall, it has been a valuable enhancement to GPS based localization. However, the system leaves room for considerable improvement.

(1) *War-driving is an expensive calibration operation, but offers limited localization coverage.* Skyhook currently employs 500 drivers who continuously war-drive to create WiFi/GSM maps of new regions and update the existing ones[1][4]. Still, a large portion of space remains uncovered, including walking paths in university campuses, shopping plazas, apartment complexes, theme parks, etc. War-driving on these walking paths is impractical while war-walking is intolerably time consuming. Moreover, the recurring financial cost of war-driving is excessive, and its impact on environment is undesirable. Complementary solutions are necessary that are cheap, environment-friendly, but scale to regions where Skyhook cannot reach.

(2) *Independence from infrastructure.* WiFi based localization is a useful idea in urban regions covered with dense AP deployments. However, large portions of the world do not have WiFi coverage, especially rural regions in US and many developing regions. Cell tower based localization produces poor localization accuracy while on-phone GPS has serious energy ramifications discussed next. An infrastructure-independent solution would be ideal for global scalability.

(3) *Energy consumption with GPS and WiFi based localization.* Our prior research showed that GPS and WiFi pose a serious tradeoff between localization accuracy and energy [5], [6]. While GPS offers high accuracy (about 10m), it drains a (Nokia N95 8GB) phone's battery within around 10 hours. Skyhook's solution lasts for 16 hours although at the expense of a degraded accuracy of $30m$ (and a high variance). With continuous usage of location services, energy-efficiency is an important concern.

In this paper, we present a simple scheme, called *CompAcc*, to address the above deficiencies in today's localization systems. Our target is to enable energy-efficient localization over walking paths (outside the purview of Skyhook) without relying on war-driving or WiFi infrastructure. The main idea is to leverage the mobile phone's accelerometer and electronic compass to measure the walking speed and orientation of the mobile user. These readings can produce a *directional trail* that is matched against *walkable path segments* within a local area map. The local map is downloaded based on a rough

[1]Updates are necessary because WiFi access points change over time as people shift in/out of apartments, homes and offices.

location of the phone, easily available from the cell tower. The path segment with the best match yields the phone's approximate location. Using only infrequent Assisted-GPS (AGPS) readings, the phone can periodically recalibrate its location, and use it as a reference for subsequent position estimation. We have implemented CompAcc on Nokia phones, and have run live experiments in the Duke University campus. Evaluation results demonstrate that CompAcc achieves average localization accuracy of around $11m$, even in areas without WiFi. This is in contrast to Skyhook's accuracy of $70m$, computed on Duke campus with dense WiFi coverage. We also observed 23.4 hours of continuous operation with CompAcc, a 40% battery life improvement over Skyhook.

CompAcc is among the early attempts to jointly utilize the phone's accelerometer and compass for infrastructure-independent localization. Our system is not yet ready for wide-scale deployment. We are currently planning large scale testing for detailed parameter tuning, particularly for (unpredictable) phone orientations inside pockets, bags, holsters. Nevertheless, results in this paper are adequately promising to justify this large-scale experimentation effort. The promise is particularly pronounced because CompAcc is complementary to existing localization solutions. While Skyhook targets urban regions near roads and streets, CompAcc is focused on areas not close to drivable roads or devoid of WiFi infrastructure. We believe that in conjunction with Skyhook, CompAcc may be an important step towards a complete localization technology.

II. System Design

Figure 1 presents the overall CompAcc architecture. We sketch an overview first, followed by the description of the three main components: (i) generating path signatures, (ii) generating directional trails, and (iii) matching signatures with trails.

A. *Architectural Overview*

CompAcc initializes by obtaining the phone's location through either GPS or AGPS (we discuss the pros and cons later). The location is sent to a remote CompAcc server which then sends back a map of the small area around that location – we call this a *map tile*. A map tile is expected to include all walking paths in that region. Now, if the phone is detected to be stationary based on its accelerometer readings, the phone's location is naturally known from the initial GPS reading. If the phone begins to move, CompAcc acts on the accelerometer readings to estimate the user's displacement. The estimation algorithm exploits the rhythmic nature of human walking patterns, and computes the number of steps walked. The distance traversed can be derived by multiplying the step count with the user's step size possible to approximate based on user's height and weight [7]. Alongside accelerometer readings, the phone's compass

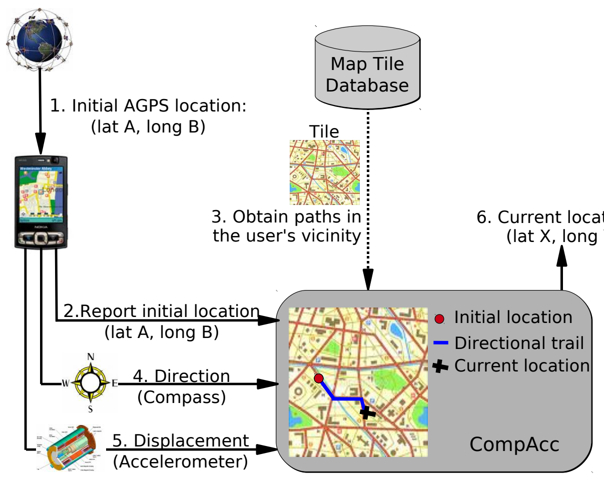

1. Initial AGPS location: (lat A, long B)
Map Tile Database
Tile
3. Obtain paths in the user's vicinity
6. Current location (lat X, long Y)
2.Report initial location (lat A, long B)
4. Direction (Compass)
5. Displacement (Accelerometer)
N
W
E
S
Initial location
Directional trail
Current location
CompAcc

Fig. 1. Flow of operations in CompAcc.

orientations are also recorded. Combining these time-indexed $< distance, orientation >$ tuples, CompAcc creates a *directional trail* of the user. Distance is expressed in *meters*, and orientation in *clockwise angular degree* with respect to magnetic north. With the starting point at the known AGPS location, this sequence of tuples presents a natural opportunity to estimate the user's position.

With a perfect compass and accelerometer, the user's location can be trivially computed over time. Unfortunately, electronic compasses and accelerometers are highly noisy [8] (unless extremely expensive). The user's movement variability and the phone often jiggling in pockets and backpacks further add to this noise. Localization becomes erroneous when the device noise drowns the changes in motion and orientation, particularly along soft turns on curved paths.

CompAcc approaches this problem by "matching" the directional trail with possible walking paths around the phone's known location. The paths are extracted from Google Maps[9], and suitably formatted for matching. The best matching path is declared to be the path of the user. Matching continues, allowing CompAcc to identify when the user turns at an intersection or moves on a curved path. Even though the approximate step count may accumulate error over time, *direction changes reset the error*. Put differently, CompAcc recognizes the user making a direction change, and thereby learns the user's location with better accuracy. Of course, error can still arise due to walking up/down stairs, detours from walking paths, and user's leaps and jumps. CompAcc occasionally falls back on AGPS to reacquire its current location.

B. *Generating Path Signatures and Map Tiles*

Matching the user's movement against electronically generated paths eliminates the need for war-driving. The approach scales globally because the path generation is a faster process at the server's side. We describe how a map tile is constructed from path segments, and is sent to the mobile user.

Digital maps, including Google Maps, represent roads and paths as polylines. In computer graphics, a polyline is defined as a continuous line composed of one or more linear segments. The polylines are superimposed on the map to display directions. Figure 2 shows an example polyline, with segment ends marked with large blue icons. The latitude and longitude for each of the segment markers are available through the Google Maps API. The intermediate coordinates within each segment is computed based on the end points. The curvature of the earth disallows cartesian formulas for distance computation. CompAcc employs the Haversine formula [10] known to be accurate for small distances (equations omitted in the interest of space). Figure 2 shows the intermediate points with small black icons. The separation between the small icons is $20m$ for visual clarity, but is actually $1m$ in our implementation.

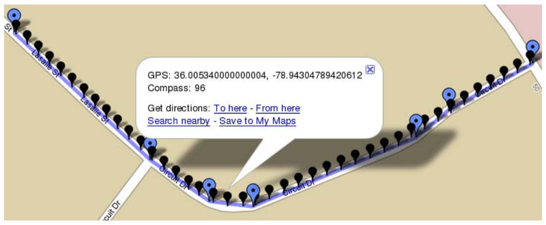

GPS: 36.005340000000004, -78.94304789420612
Compass: 96
Get directions: To here - From here
Search nearby - Save to My Maps

Fig. 2. Large blue icons marks the piecewise segments of the polyline, while black icons are computed intermediate points. Each location associated with the latitude, longitude and the angle from magnetic north.

Given the end coordinates of any path segment, the precise path orientation can be computed using the formulas in [11]. A path running east-west will result in an orientation of $90°$ or $270°$. An ideal mobile phone compass is expected to reflect this value while the user is walking on this path segment.

When the mobile phone sends its AGPS location, the CompAcc server computes a *map tile* with that location as the center. The tile includes all the path segments in a data structure, each segment consisting of its end markers, intermediate coordinates, and orientations (a path signature). The tile is downloaded to the phone, and used to match against the user's directional trail. If Internet connectivity is a concern, a large tile may be downloaded in advance.

C. Generating Directional Trails

Modern phones come equipped with accelerometers, mostly used to rotate pictures between landscape and portrait mode. The accelerometer is always on and CompAcc exploits it for computing the user's displacement. The natural computation method is to double-integrate acceleration, $\int\int a(t)dt$, where the first integration computes speed, and the second computes displacement. Unfortunately, several factors cause fluctuations in acceleration, resulting in erroneous displacements [8]. Departing from this approach, CompAcc identifies a rhythmic acceleration-signature in human walking patterns. The rhythm is evident in the raw measurements in Figure 3; each spike in the negative direction, roughly corresponding to a step. Around -300 units of acceleration is due to gravity, g. In

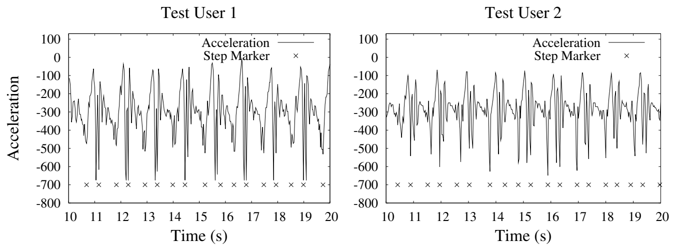


Fig. 3. Accelerometer readings from two users (steps marked with a cross)

addition to g, we observe two spike patterns. The small ones (at around -500) correspond to the foot being lifted off the ground; the large ones (at around -700) are caused by the leg settling back on the ground. We observed this rhythm across 15 test users, only the height and periodicity of the spikes varied. CompAcc computes the number of steps using this signature, and multiplies it with the step size of the individual. The individual's step size can be approximately derived from the individual's height and weight [7].

The phone's compass records the user's orientation, expressed as an angle with respect to magnetic north. With Nokia 6210 Navigator models, this angle is the direction faced by the phone's back, when the phone is held vertically. Ideally, while the user is moving in a straight line (say towards East), the readings should be $90°$. However, as with accelerometers, measurements yield noisy data with random fluctuations for sharp movements and slow responses to soft turns. CompAcc caches a window of previous N compass readings, utilized in constructing the user's directional trail.

Forming the Directional Trail:
The directional trail is defined as a series of last N compass readings and an associated set of displacements between them. The compass readings are collected with a time separation of 1 second. Denote the compass trail by a set $C_N = \{c_0, c_{-1}, c_{-2}...c_{-N}\}$. Here, c_0 is the compass reading at the user's *estimated* current location, c_{-1} is the previous reading, and so on. Also, let the accelerometer based displacements between compass readings be denoted by a set $D = \{\delta_0, \delta_{-1}, ...\delta_{(-N+1)}\}$. Thus, the user walked a distance of δ_j between two compass measurements, c_{j-1} and c_j. The user covered a total displacement of $TD = \sum_{j=0}^{-N+1} \delta_j$ in the past N seconds.

Similarly, a sequence of compass readings is also obtained from the path signatures in the tile. Denote this sequence by $P_N = \{p_0, p_{-1}, p_{-2}...p_{-N}\}$ where p_0 is the actual orientation (known from the tile) at the user's estimated current location. Similar to the compass trail, the values of p_k and p_{k-1} are also separated by distance δ_k. The goal of CompAcc is to slide the P_N window forward and backward, and find the maximum similarity between P_N and C_N. Figure 4 shows the trail and the (sliding) path signature. The details on matching are explained next.

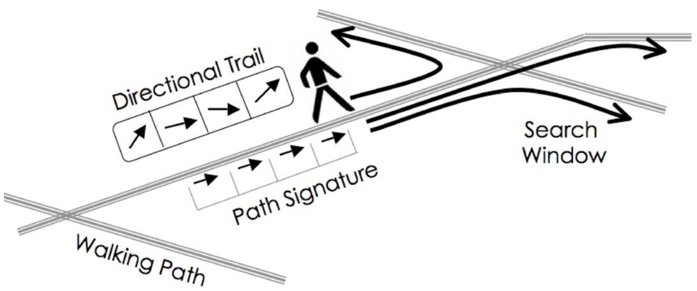


Fig. 4. CompAcc slides the path signature to a maximum of *Search-Window* steps, forward and backward. The sliding needs to turn into all possible directions within this window. For every step-wise slide, the "Dissimilarity" is computed between the given directional trail and the path signature. The path signature with minimum *Dissimilarity* yields the user's current location.

D. Matching Path Signatures with Directional Trails

Matching is triggered only when CompAcc finds that the user's estimated location is within a threshold distance from a possible direction change. Direction change is possible not only at path intersections, but also at the end of linear segments in a polyline. When near a segment end, CompAcc records the current compass trail C_N and the corresponding path signature P_N. The "Dissimilarity" ϕ is computed between the two sequences using the following simple equation:

$$\phi_0 = \sqrt{\frac{\sum_{i=0}^{-N}(c_i - p_i)^2}{N}} \tag{1}$$

Of course, P_N is computed based on the user's estimated current location. This estimate can be erroneous (due to step count inaccuracies), and the user may either be ahead or behind the estimated location. If the user is actually ahead, then a *slide-forward* version of P_N is likely to better match C_N (and the vice versa). We use P_N^{+j} to represent a forward slide of j steps on P_N. Similarly, P_N^{-j} represents a backward slide of j steps on P_N. The maximum extent of the forward/backward slide is called the *Search Window* where W is the size of this window. CompAcc computes

$$k : \phi_k = min\{\phi_j\} \quad \forall j \in [-W, W] \tag{2}$$

Given the value of k, CompAcc computes P_N^k, and the corresponding value of p_0^k. The location corresponding to p_0^k is the new estimate of the user's location. The accelerometer based step count is reset, and restarted from this point. The next section discusses parameter choice in CompAcc. Evaluation results are reported thereafter.

E. Fallback Mechanisms

CompAcc continuously estimates the user's location. In certain scenarios, the difference between the estimated and the actual location (i.e., the localization error) can create confusion. Consider Figure 5(a) where CompAcc believes that the user is located between the 5th and 7th Street intersection, and walking northward. Now, assume that the actual user has taken a turn, hence, the compass values reflect it. Since CompAcc will search both forward and backward, the trail

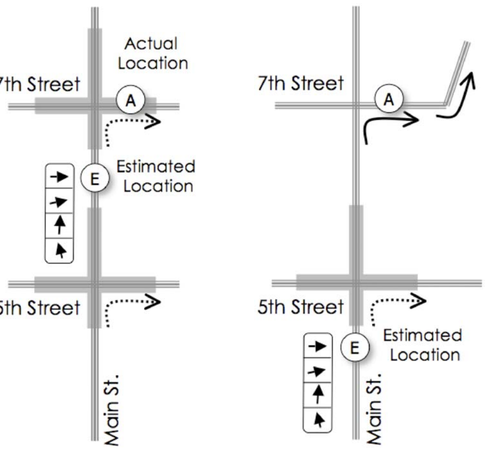


Fig. 5. "E" is the CompAcc-estimated user location, and "A" is the user's actual location. The scenarios lead to possible confusion (Search Window shown in gray).

may match "a right turn" onto both 5th and 7th Streets. The actual user could be ahead or behind the estimated location, and hence, there is no way of telling the difference.

One way to resolve this is to ensure that CompAcc's Search Window covers at most one intersection at any given time. Let us denote the minimum distance between two intersections in a map tile as L_{min}. This value can be an attribute of the map tile, known to CompAcc. If CompAcc chooses a Search Window, $SW < \frac{L_{min}}{2}$, then the confusion in Figure 5(a) will not happen. As long as the location error is not too large, CompAcc will always search over the same intersection around which the user actually is located.

If the location error, however, grows large, the SW parameter choice will not be sufficient to prevent confusion. Consider Figure 5(b) where the user has actually turned onto 7th Street, but CompAcc's estimated location is before the 5th Street intersection. Observing that 5th Street is imminent, CompAcc will match its directional trail with path signatures, and is likely to find a good match with a "right turn on 5th Street". Clearly, the estimated location will be on a parallel path to the actual user, resulting in error.

We propose an AGPS based fallback mechanism to recover from such situations. Observe that the actual user's directional trails are unlikely to continuously match the path taken by the estimated user. The actual user may take a left turn while the estimated user has no left turn within its search window. The *minimum matching Dissimilarity* (ϕ) (between the directional trail and sliding path signatures) will be large in such situations. Figure 6 computes these distributions for the Duke University map tile. Essentially, the "Dissimilarity" is low when the actual directional trail is along the actual path signature, but much higher when the matching is against an incorrect path turn. Therefore, when CompAcc observes a large value of ϕ (45 in the case of the Duke University tile), it triggers an AGPS reading. The estimated user is relocated close to the actual location, step count is reset,

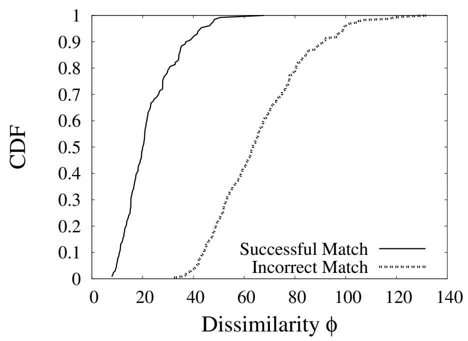


Fig. 6. *Dissimilarity* is high when the user's directional trail is matched against an incorrectly estimated path signature. This is a trigger for AGPS.

and the operation continues. This ensures that the localization error does not grow (except perhaps in rare cases such as a Manhattan grid). Results on walking paths in the Duke University campus did not require any AGPS reading due to the inherent (directional) diversity in the walking paths.

III. Experimentation and Evaluation

A. *Experimental Methodology*

We implement CompAcc on Nokia N95 and 6210 phone models using Python as the programming platform. Two phone models were necessary because N95 does not have a compass while 6210 does not have WiFi. Newer phones (like iPhone 3Gs and Android G1) have all the required sensors, but were not available when we started the project. Hence, we pretend that the two Nokia phones are the same device, gather data from each of them, and analyze the consolidated traces for localization.

We assume GPS to be the global truth, and obtain location readings through an high end Garmin 60CSx handheld GPS receiver. N95 and GPS were placed in the test-users' pockets, while the 6210 was carried vertically, either in hand or in trouser pockets. Map tiles of the Duke University west campus were generated from Google Maps. Google Maps already contain a few of the walking paths, so the rest were manually marked by clicking on start and end points of different polyline segments (Figure 7). Marking the paths was easy through Google's Satellite view, and took less than a minute to cover a 300m x 300m tile. This projects to less than 5 hours to cover the Duke University campus (2.9 sqkm). We note that this is orders of magnitude less expensive (in time and money) compared to war-walking *all* paths within the campus.

To evaluate CompAcc, 5 students walked on random paths in the tile, carrying the equipment described earlier. We collected 25 traces in total covering all segments of the tile. We use two metrics for evaluation, namely, *Instantaneous Error* (IE) and *Average Localization Error* (ALE), defined as:

$$IE(t_i) = distance(CompAcc(t_i), GPS(t_i)) \quad (3)$$

$$ALE = \sum_{i=1}^{T} \left(\frac{IE(t_i)}{T} \right) \quad (4)$$

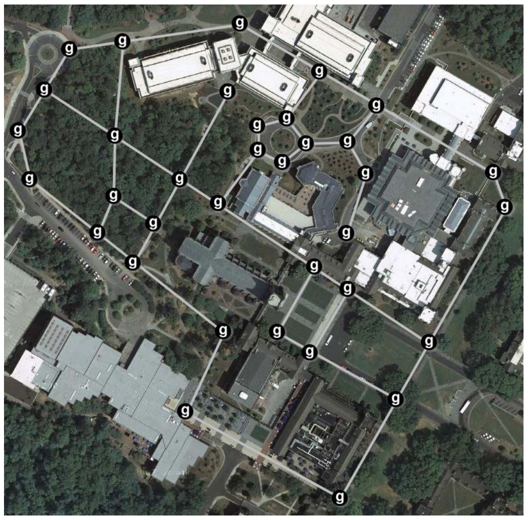


Fig. 7. A *map tile* from the Duke University west campus, used for evaluation. Light gray lines show walking paths.

Here $CompAcc(t_i)$ and $GPS(t_i)$ represent the user locations reported by CompAcc and GPS, respectively. A trace has T equally spaced discrete time-points, and $t_i - t_{i-1} = 5s$ (i.e., we sample the user's location every 5s).

B. *Design Decisions and Parameter Choices*

This subsection justifies the parameter choices in CompAcc.

Computing Initial Location through AGPS
CompAcc occasionally needs the phone's accurate location as a resetting mechanism. This ensures that undetected direction changes (or other unanticipated glitches) can be recovered. GPS is the natural solution, since it is most accurate. Unfortunately, GPS can experience a large delay for acquiring satellite locks for localization. This behavior can be attributed to a variety of factors, including cheap GPS sensors, cloudy skies, or a "cold start". Assisted GPS (AGPS), has a faster acquisition time while retaining comparable accuracy. Table I compares the location acquisition time between GPS and AGPS for 10 experiments on 10 different days. Evidently, AGPS averages 13.1 seconds, in contrast to GPS's 98.6 seconds. All the experiments were obtained in summer with clear skies; our experience with GPS is worse in winter. This motivates the use of AGPS in CompAcc for infrequently calibrating the user's location, as well as for detecting deviations from walking paths.

TABLE I
Location acquisition delay: AGPS avg. is 13.1s, GPS is 98.6s.

Delay(s)	1	2	3	4	5	6	7	8	9	10
A-GPS	6	14	15	16	16	12	15	11	13	13
GPS	21	216	143	108	117	57	97	53	103	71

How to cope with different walking styles?
The accelerometer signature of human footsteps is characterized by rhythmic oscillations. We parameterized the oscillation by two parameters, h and Δt. Briefly, h is the

TABLE II
CALIBRATING Δt BY MEASURING THE TIME INTERVAL BETWEEN SPIKES.

Test	1	2	3	4	5	6	7	8	9	10
count	10	50	56	64	70	71	77	89	113	645
$0.3s$	12	53	62	64	98	80	101	112	111	661
$0.4s$	11	53	61	61	77	72	77	88	111	618
$0.5s$	11	50	60	47	68	62	75	73	85	522

minimum height of the oscillation, and Δt is the minimum periodicity. With comparable noise in the accelerometer and variations in walking styles, the values of h and Δt need to be chosen empirically. Error in the parameter value will result in step count error, however, CompAcc can tolerate part of this error through signature matching. Thus, an approximate user-displacement is acceptable.

We inspected 10 traces collected from 5 student test users (*different from those who will evaluate CompAcc later*). After compensating for gravity and simple noise reduction, we consistently found that step-induced deceleration drops below -50, i.e., $h \leq -50$. The step count was more sensitive to parameter Δt, and hence, we monitored its behavior for different values. Table II shows the results only for $\Delta t = 0.3s$, $0.4s$ and $0.5s$. $\Delta t = 0.4$ was found to exhibit less than 10% error and proved robust in tests with other arbitrary users. Thus, CompAcc increments the step count only when the oscillation has a magnitude less than -50 and a time separation greater than $0.4s$ (from the previous spike). Ongoing work is attempting to learn user-specific parameters based on step count and a few consecutive AGPS readings.

What if step count error keeps increasing?
As discussed earlier, the step count error is likely to get reset when users take turns. However, if the user walks on a long straight segment, the error continues to grow. An AGPS reading resolves this problem ensuring that the error is lowered before it exceeds a user-specified threshold. To meet this threshold, say δ, CompAcc must trigger the AGPS when the accumulated error is close to δ. We calibrate the accumulated error due to step count by requesting test users to walk along a long straight path. Figure 8 shows the variation. Evidently, for any value of δ (on the x-axis), we can find a corresponding value of step count, Y_δ, on the Y axis. CompAcc takes an AGPS reading if the user's compass orientation does not show a turn for Y_δ consecutive steps. In our evaluations, we used $\delta = 40m$, corresponding to around 472 steps (approximately $400m$ in physical distance). If users pause in between, CompAcc is aware of it, and does not increment the steps unnecessarily. For our tests in the Duke campus, the longest straight line path was around $300m$, so AGPS was never triggered. As we shall see later, this is desirable for energy conservation.

What happens if the phone shakes in pockets?
An ideal compass in ship or aircraft navigation is expected to track the device's orientation through turns and straight paths.

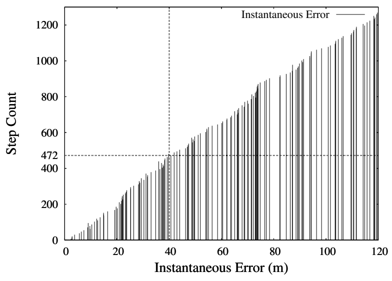


Fig. 8. The growing error from Step Count observed over a straight walk.

However, with noisy compasses, swaying human movements, and the phone jiggling in pockets, the compass output is less ideal. Figure 9(a,b) show an example recording from a trace with 7 turns. The compass readings are modulo 360 (i.e., -5 and 355 are along same northward directions). The ground truth is pre-computed from the map and shown as a solid black line. Other readings are from three placements of the phone, namely, handheld, in a trouser pocket, and in a running shorts (the phone display facing inward). Carrying the phone in shorts' pockets induces wide variations (segments III, V and VI show heavy fluctuations). For our evaluations in this paper, we assume that the phone is handheld or in a trouser pocket. Our future work will address the problems with carrying the phone inside shorts-pockets.

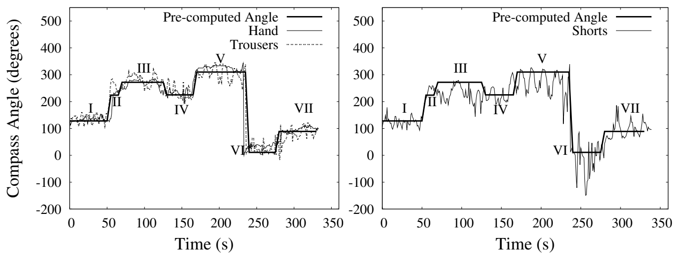


Fig. 9. Compass readings can be tolerated when phone is handheld or in trouser pockets. The phone jiggles excessively in shorts pockets.

How large is the Directional Trail?
We select the value of $|C_N|$ based on error measurements for different window sizes (Figure 10). We include 4 out of 25 traces for visual clarity. ALE (Average Localization Error) is consistently large for small values of SW. This is because small windows are more susceptible to compass noise, resulting in degraded location quality. For window sizes 8 or larger, the compass shows a clearer (statistical) trend, and the ALE value drops sharply. Twenty one other traces show similar trends, guiding us to assign $|C_N|$ to 10.

C. *Performance Evaluation*

This section presents the performance of CompAcc in comparison to Skyhook and WiFi-War-Walk. We downloaded the *Skyhook* software from the Internet, and installed it on a Nokia N95 model (equipped with all the necessary sensors). Since Skyhook relies on war-driving to create the radio map, its performance within university campuses and other walking paths reflects today's achievable performance. *WiFi-War-Walk* is a customized scheme that captures the

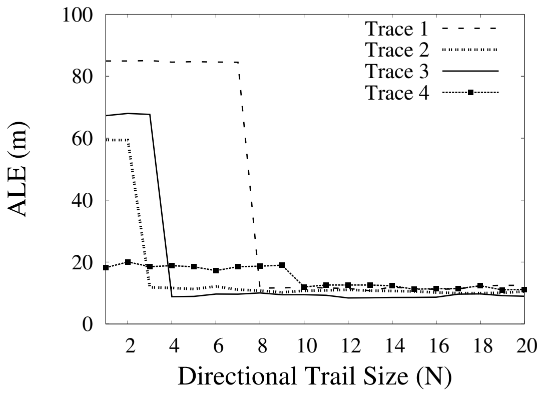


Fig. 10. Variation of localization error with the size of the directional trail.

improvements if a Skyhook-like scheme was augmented by war-walking the footpaths. Although war-driving is expensive and time-consuming, this provides an estimate of Skyhook's performance. We war-walked the Duke University west campus, and implemented Place Lab's localization algorithm on the WiFi map (Skyhook's algorithm is proprietary, but similar to Place Lab). The metrics of interest are Average Localization Error (ALE), and energy consumption due to localization. We also qualitatively contrast localization coverage through Google Maps.

Figure 11 presents the per-trace ALE from all 3 schemes. The average across all traces for each scheme is included in the caption. CompAcc outperforms Skyhook by a factor of 6.4; the improvement over WiFi-War-Walk is 2.8. Further, we note that Duke University is densely covered with WiFi, and hence, similar comparisons in off-campus areas are expected to yield wider performance gaps. Figure 12 zooms into the CompAcc performance and shows the Instantaneous Error (IE) over time. Two arbitrary segments from distinct traces are shown – vertical lines denote instants when the user turned. The trend shows the error increasing as the user walks down a path segment, but drops at turns. The error does not become zero because compass and accelerometer noise miscalculates the user's precise position. Nevertheless, it prevents the error from accumulating.

Figure 13(a,b) visualizes the comparison between CompAcc and Skyhook on Google Satellite Maps. GPS locations are shown with a "g", Skyhook with "s", and CompAcc with "c". The light gray lines joining g–s or g–c denote the location errors. Figure 14 shows the variation of Instantaneous Error (IE) over time from 3 randomly selected traces. The differences between the schemes are visually apparent. Nevertheless, we make a few important observations:

(1) *Skyhook's localization is biased towards roads and streets (regions A, B and C).* Users walking within the campus were localized to the nearest street. Often, a street had a dead end within the university, and a large number of location readings were found to be clustered at these dead ends. This is an outcome of Skyhook war-driving only on streets, resulting in

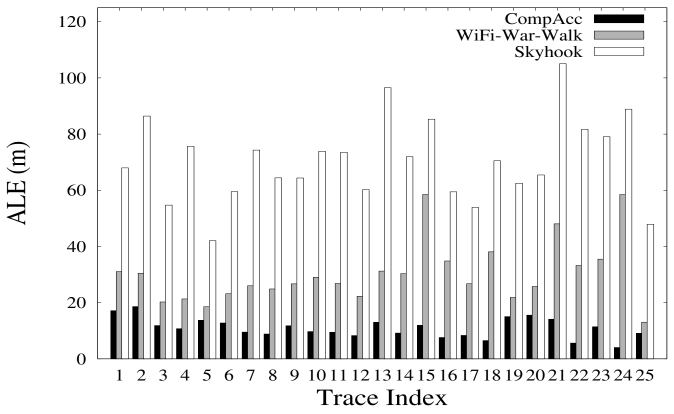


Fig. 11. ALE for CompAcc, WiFi-War-Walk, and Skyhook. Average localization errors across all traces and schemes are: CompAcc $10.95m$, WiFi-War-Walk $30.24m$ and Skyhook $70.6m$.

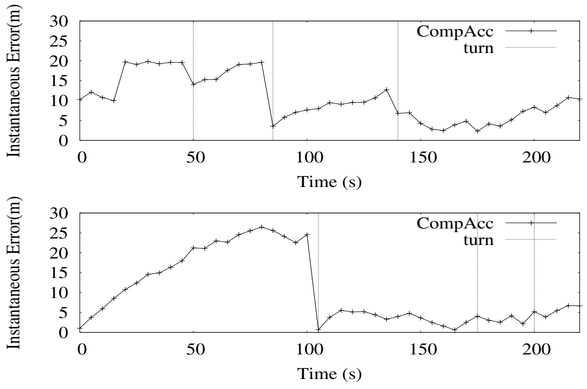


Fig. 12. CompAcc's Instantaneous Error (IE) drops at path turns.

poor localization on walking trails. CompAcc's independence to war-driving is evident in the consistent accuracy.

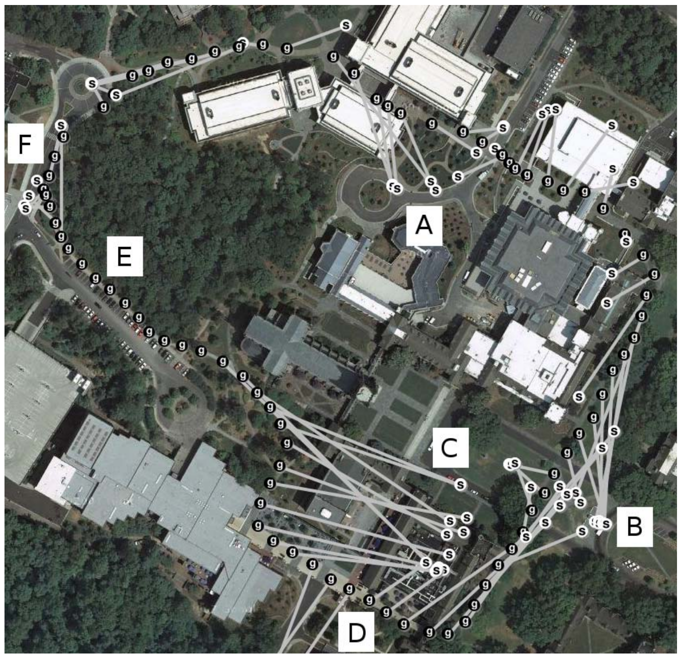
(2) *Skyhook relies heavily on WiFi availability (region E).* A popular campus walking path bordering a wooded region does not have WiFi connectivity. Skyhook was unable to localize the phone on this path. Although GSM localization should have been triggered, we suspect that the time to detect WiFi failure was too long. CompAcc, however, shows near-accurate localization.

(3) *Skyhook exhibited high error variations (regions B, C).* We are unsure why this occurs, however, open parking lots may be susceptible to signal strength variations. Labs may induce similar impacts due to electromagnetic radiations that interfere with WiFi. CompAcc remains unaffected by such environmental factors.

(4) *Skyhook performed well on a footpath beside a drivable path (region F).* CompAcc too achieved high accuracy in these regions.

Energy Consumption

We report energy consumption due to localization through the Nokia Energy Profiler (logging energy usage every 0.25 seconds). AGPS and Skyhook were made to sample the location every 5 seconds. For CompAcc, the accelerometer and compass were continuously probed. Figure 15 shows the power draw for AGPS, Skyhook, accelerometer, and compass.



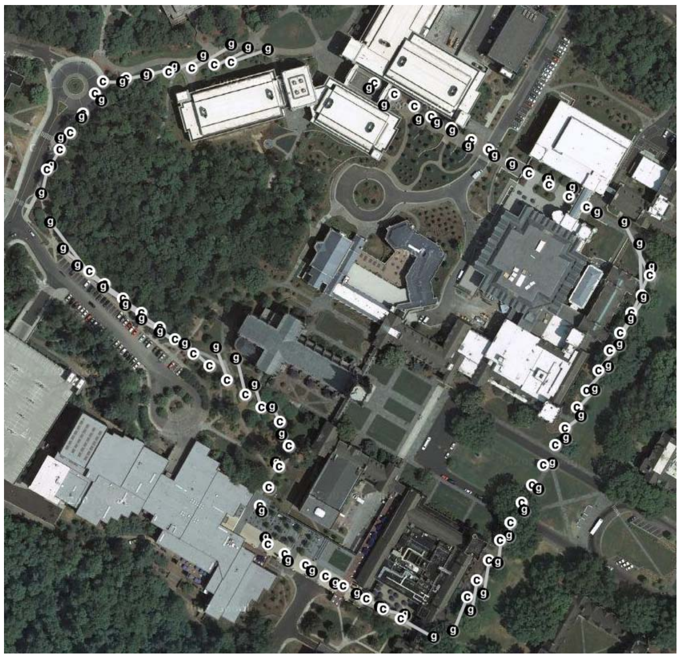


Fig. 13. Visualization of (a) Skyhook's and (b) CompAcc's performance in Duke campus. GPS locations shown by "g", Skyhook by "s", and CompAcc by "c". Straight lines joining s–g and c–g are a measure of the location errors. Some regions do not have a "s" value because Skyhook was unavailable there.

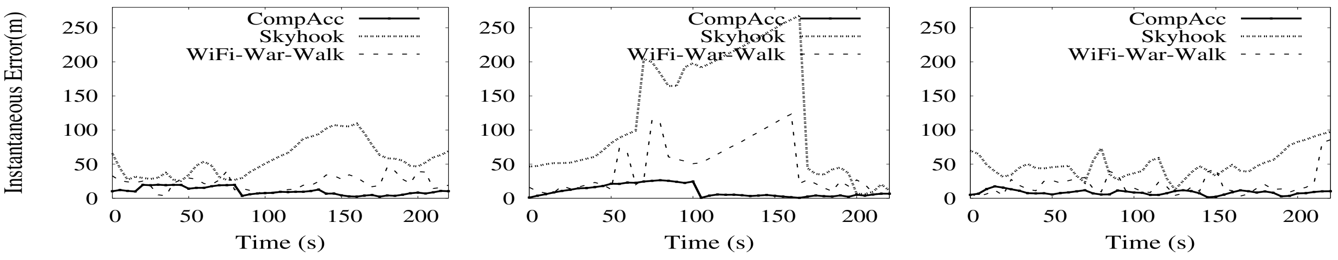


Fig. 14. Variation of localization accuracy with CompAcc, WiFi-War-Walk, and Skyhook for three test traces

The 3G network was enabled for AGPS and Skyhook (GPS was turned off for SkyHook for fairness). The compass measurements were taken from the Nokia 6210 and superimposed on this graph. The heavy energy footprint of AGPS (baseline of $0.4W$) precludes it as a stand-alone localization method. Skyhook displays periodic spikes of around $1W$, separated by smaller spikes of $0.2W$. The accelerometer and compass sensors have similar power consumption signatures, with a baseline of around $0.1W$. Table III shows the individual battery life for each sensor/localization scheme, assuming only the profiler running in the background. The compass measurements were executed on 6210 which has a battery capacity of approximately 80% of a N95. We combined their individual recordings proportionately to derive the CompAcc battery life. Results show an improved battery life with CompAcc.

IV. Limitations and Future Work

CompAcc is not yet ready for deployment. A number of issues still need to be resolved efficiently.

Map Generation. Currently, map tile generation includes a manual component of marking out the start/end points of footpaths. Although significantly less expensive than war-walking, an ideal system should automate this process. One option is to design Internet based labeling games as proposed by Vohn Ann [12]. Another option could be to launch this into Mechanical Turque [13] where many Internet users could label paths for small financial incentives. A more complex approach could be to extract the paths through sophisticated image processing on Google Satellite view. We plan to investigate the viability of these and other approaches.

Improved Signature Matching. CompAcc resets a signature mismatch through AGPS. This is required because CompAcc keeps no record of possible directions the user may follow. Location is estimated only along a unique line approximating the actual path. For example, when crossing a very wide road, CompAcc will trigger AGPS because the user trail contains many compass angles perpendicular to the current path signature. Such cases can be accommodated using particle filters [14], [15] which operate on movement probabilities in multiple directions. Employing particle filters may provide both energy and accuracy improvements.

Multiplexing between Localization Methods. CompAcc needs to handoff localization to Skyhook/GPS when the

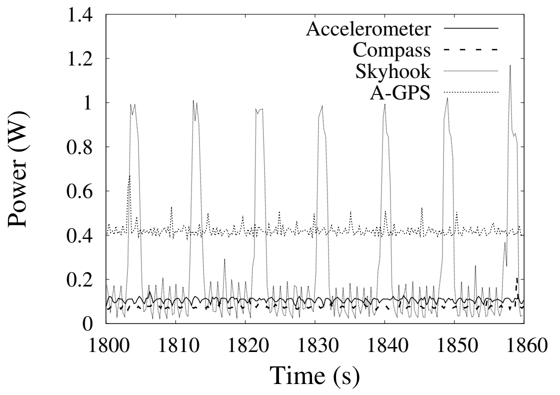


Fig. 15. Power consumed by Compass, Accelerometer, Skyhook, AGPS.

TABLE III
BATTERY LIFETIME: COMPACC (ACC, COMP), SKYHOOK, AGPS.

Sensor	Acc.	Comp.	CompAcc	Sky	AGPS
lifetime(h)	39.1	48.7	23.4	16.3	10.6

user is in a vehicle or enters indoors. The control can be regained once the user is walking again. Switching between these localization modes needs to be triggered automatically. The current system can activate handoff when the *matching Dissimilarity* is excessive (as used for fallback mechanisms). However, quicker activation is necessary.

Extending beyond Walking Paths. CompAcc targets regions that lie outside the coverage of war-driving based localization. However, the system can be extended to vehicular motions, or even towards micro-mobility within buildings, malls, stores[16]. Roads and streets' path signatures can be extracted directly from Google Maps without any manual intervention. One part of the ongoing work is focused towards these extensions. CompAcc (for vehicles) will be an effective solution for many countries in the world that do not have WiFi coverage, or remain to be war-driven.

V. RELATED WORK

Previous research on localization makes different assumptions about infrastructure and calibration effort. In general, previous localization systems rely on deployed radios (e.g. APs) or require installation of specialized hardware in the environment (radio or bluetooth devices). A calibration effort maps overheard radio signals to location coordinates. Then, overheard signals are matched with the data recorded during the calibration phase and the user location is estimated. Cricket [17], Nokia Labs, VOR [18] and Pinpoint [19] rely on specially installed hardware for indoor localization. While effective in high-budget enterprises, these systems are expensive to deploy and maintain in outdoor environments.

Radar [20], Active Campus [21] and PlaceLab [2] rely on devices already present in the surroundings for localization. The Radar system [20] operates on WiFi fingerprints, and is capable of achieving high accuracy in indoor deployments. Nevertheless, Radar needs to calibrate WiFi signal strengths at many physical locations in a building. The calibration process is time-consuming and may not scale over larger areas. PlaceLab [2] uses WiFi and GSM signals for localization. A radio map is created by war-driving car-accessible roads and mapping APs/GSM towers to GPS coordinates. The radio map is distributed to mobile devices which localize themselves by comparing overheard APs/GSM towers with those recorded in the map. Active Campus [21] is similar to PlaceLab, but assumes that the locations of the WiFi access points are known *a priori*. Unlike the previous systems, CompAcc does not require war-driving, signal calibration, or deployed infrastructure for localization. Our approach is straightforward, relying on digital maps, compasses, and accelerometers, available in modern mobile phones.

VI. CONCLUSION

The growing popularity of location based services calls for improved quality of localization, including greater ubiquity, accuracy, and energy-efficiency. Current localization schemes, although effective in their target environments, may not scale to meet the evolving needs. This paper proposes *CompAcc*, a simple and practical method of localization using phone compasses and accelerometers. CompAcc's core idea has been known for centuries, yet, its adoption to human scale localization is not obvious. We believe that CompAcc can complement current localization technology, taking an important step towards a pervasive location service for the future.

REFERENCES

[1] "Lba counts in Apple Store and Android Marketplace," http://www.skyhookwireless.com/locationapps/.

[2] Y. Chen *et al.*, "Accuracy characterization for metropolitan-scale wi-fi localization," in *ACM MobiSys*, 2005.

[3] "Skyhook wireless," http://www.skyhookwireless.com/.

[4] N. Y. Times, "Cellphone locator system needs no satellite," *http://www.nytimes.com/2009/06/01/technology/start-ups/01locate.html*.

[5] I. Constandache *et al.*, "EnLoc: Energy-efficient localization for mobile phones," in *IEEE INFOCOM Mini Conference*, 2009.

[6] S. Gaonkar *et al.*, "Micro-blog: Sharing and querying content through mobile phones and social participation," in *ACM MobiSys*, 2008.

[7] "Step size in pedometers," http://walking.about.com/cs/pedometers/a/pedometerset.htm.

[8] C. Jekeli, *Inertial Navigation Systems with Geodetic Applications*. Walter de Gruyter, 2000.

[9] "Google maps api," http://code.google.com/apis/maps/.

[10] "Haversine formula," http://en.wikipedia.org/wiki/Haversine_formula.

[11] E. Williams, "Aviation formulary," *http://williams.best.vwh.net/avform.htm*.

[12] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *ACM CHI*, 2004.

[13] "Amazon Mechanical Turk," http://aws.amazon.com/mturk/.

[14] J. Hightower and G. Borriello, "Particle filters for location estimation in ubiquitous computing: A case study," in *UbiComp*, 2004.

[15] F. Gustafsson *et al.*, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, 2002.

[16] M. Azizyan *et al.*, "Surroundsense: Localizing mobile phones via ambience fingerprinting," in *ACM MobiCom*, 2009.

[17] N. B. Priyantha *et al.*, "The cricket location-support system," in *MobiCom*, 2000.

[18] D. Niculescu and B. Nath, "VOR base stations for indoor 802.11 positioning," in *ACM MobiCom*, 2004.

[19] M. Youssef *et al.*, "Pinpoint: An asynchronous time-based location determination system," in *ACM MobiSys*, 2006.

[20] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," in *IEEE INFOCOM*, 2000.

[21] W. G. Griswold *et al.*, "ActiveCampus: Experiments in community-oriented ubiquitous computing," *IEEE Computer*, 2003.