AutoGait: A Mobile Platform that Accurately Estimates the Distance Walked

Dae-Ki Cho, Min Mun, Uichin Lee†, William J. Kaiser‡, Mario Gerla
UCLA Computer Science Department, †Alcatel-Lucent Bell Labs, ‡UCLA Electrical Engineering Department
{dkcho, bobbymun, gerla}@cs.ucla.edu †uichin.lee@alcatel-lucent.com ‡kaiser@ee.ucla.edu

***Abstract*—AutoGait is a mobile platform that autonomously discovers a user's walking profile and accurately estimates the distance walked. The discovery is made by utilizing the GPS in the user's mobile device when the user is walking outdoors. This profile can then be used both indoors and outdoors to estimate the distance walked. To model the person's walking profile, we take advantage of the fact that a linear relationship exists between step frequency and stride length, which is unique to individuals and applies to everyone regardless of age. Autonomous calibration invisible to users allows the system to maintain a high level of accuracy under changing conditions. AutoGait can be integrated into any pedometer or indoor navigation software on handheld devices as long as they are equipped with GPS. The main contribution of this paper is two fold: (1) we propose an auto-calibration method that trains a person's walking profile by effectively processing noisy GPS readings, and (2) we build a prototype system and validate its performance by performing extensive experiments. Our experimental results confirm that the proposed auto-calibration method can accurately estimate a person's walking profile and thus significantly reduce the error rate.**

I. Introduction

Accurate estimation of the distance walked is the key enabler for numerous ubiquitous mobile applications, ranging from pedometers to indoor navigation systems. Despite its high energy consumption, the Global Positioning System (GPS) can provide a very useful means of distance estimation as long as mobile users are outdoors. For indoor coverage, there are two popular techniques, namely indoor localization using RF-based fingerprinting (e.g., WiFi/GSM/Bluetooth-based fingerprinting methods [1]) and pedestrian dead reckoning using wearable sensors (e.g., accelerometer- or pressure-sensor-based pedometers). The latter is more preferable for ubiquitous mobile applications [2], because the former requires infrastructure and its performance heavily depends on the fingerprint database or signal conditions [3].

Existing sensor-based pedestrian dead reckoning systems typically use accelerometers to count steps and compasses and gyroscopes to measure changes in walking directions [4]. Then, the total distance walked is calculated by simply multiplying the measured step count by the average stride length.[1] However, these systems suffer from inaccuracy, namely distance overestimation at slower speeds and underestimation at faster speeds. While one possible cause is the sensor problems such as accelerometer measurement errors and sensor misalignment [5], the major problem is that these systems assume a constant stride length while estimating the distance walked. Most pedometer systems ask the user to manually calibrate the average stride length; e.g., after walking a known distance, a user finds the average stride length by dividing the distance by the measured step count. This makes the distance estimation severely biased towards dominant walking patterns. Considering the fact that human walking patterns are not uniform and the stride length is variable, such manual calibration may result in severe errors. Recall that for indoor navigation, even a small error in estimating the distance walked (e.g., $10m$) can account for location misprediction. Hence, it is very important to characterize the variable stride length by profiling walking patterns.

In physiology literature, it has been shown that there is a linear relationship between step frequency and stride length – the stride length linearly increases with the step frequency, which is unique to individuals and applies to everyone regardless of age [6], [7].[2] For instance, the stride length while walking slowly (low step frequency) is shorter than the one while walking fast (high step frequency). Given that the same walking profile applies both indoors and outdoors, we can find a linear walking profile using GPS when a user is walking outdoors, and this profile can then be used for estimating the distance walked for both indoor and outdoor environments. The profile can be re-calibrated occasionally, because walking is a time-varying activity that is affected by several factors such as mood, body shape, and weight [8].

The main challenge is to calibrate a walking profile using commercial off-the-shelf (COTS) GPS in a typical mobile device. In this paper, we address this challenge and propose *AutoGait*, a mobile platform that autonomously discovers a user's walking profile and accurately estimates the distance walked. The following are the key contributions of the paper:

- The proposed auto-calibration method effectively processes noisy GPS readings by searching for straight-line walking patterns, where heading changes of a mobile user are bound within a certain threshold degree. Finding such segments is not difficult because users walk on the pedestrian roads. Further, the distance of a straight-line segment can be accurately measured, because the impact of GPS errors becomes negligible as the distance walked increases. This allows us to find sample data points for linear regression by calculating the average step frequency and stride length from each line segment. Thus, our system can accurately calibrate the linear model even with noisy GPS readings. Moreover, AutoGait periodically triggers auto-calibration to

[1] *Stride length* is the distance between the heel of the front foot and the heel of the back foot at the point where they are farthest apart.

[2] *Step frequency* is the inverse of the time taken by a user to move a stride.

978-1-4244-5328-3/09/$25.00 ©2009 IEEE

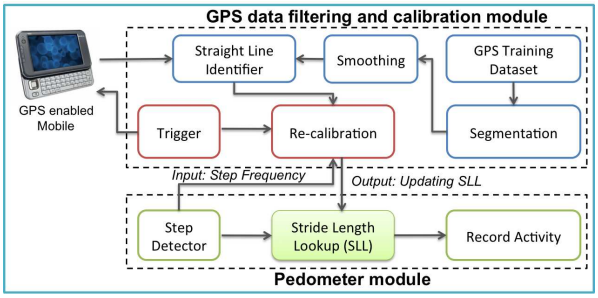


Figure 1. High-level system view of AutoGait

detect whether one's gait pattern has changed over time.

- We implement the AutoGait system based on Nokia N810 and Smartshoe platforms and validate its accuracy via extensive experiments in both controlled (treadmills) and outdoor environments. AutoGait is sufficiently generic to be paired with any pedometers or indoor navigation systems as long as they are equipped with GPS and requires minimal modifications to their software. Our experiment results show that AutoGait performs within error rates of less than 1.5% in the tested scenarios; this is comparable to the performance of differential GPS (DGPS)-based schemes [9].

II. Overview of AutoGait System

The AutoGait system consists of two modules: a pedometer module and a GPS data filtering/calibration module (see Figure 1). The pedometer module detects steps and keeps track of the step history as well as the distance walked. When calibration is required, the GPS data filtering/calibration module first filters noisy COTS GPS readings via smoothing and then searches for straight-line walking segments where heading changes of a user are bound within a certain angular threshold. We only consider straight lines, because smoothing tends to distort the shape of the corners owing to noisy GPS readings. Moreover, it is possible to accurately measure the distance of a straight-line segment, because the impact of GPS errors becomes negligible as the distance traveled increases.

AutoGait exploits the fact that there is a linear relationship between stride length and step frequency [6], [7]. We call this linear relationship the Stride Length Lookup (SLL) function, namely $s = \alpha \cdot f + \beta$; i.e., for a given step frequency f, we can find the corresponding stride length s. The rest of the parameters, α and β, can be found using straight-line segments as follows: For each straight-line segment, the system calculates the average step frequency and the average stride length. The two values form a sample point, and the system runs a linear regression on such sample points obtained from multiple straight-line segments. Therefore, our system can accurately calibrate the linear walking profile or SLL even with noisy GPS readings. AutoGait periodically triggers auto-calibration and detects whether one's gait pattern has changed over time. Given that we have a set of measured step frequencies $f_1, \cdots, f_s$, we can estimate the distance using SLL as follows: $d_{total} = \sum_{i=1}^{s} \alpha \cdot f_i + \beta$.

III. Mechanism for GPS Data Filtering, Calibration, and Triggering

Whenever a calibration is required, AutoGait *opportunistically* calibrates the SLL while a user is walking outdoors. After collecting a sufficient number of GPS readings outdoors, the system performs GPS data filtering and detects straight-line segments. It then calibrates the SLL using the straight-line segments. In this section, we detail GPS data filtering, calibration, and triggering mechanisms.

A. GPS Data Filtering

Instead of using highly accurate DGPS to calibrate the linear relationship between stride length and step frequency by measuring the exact position of the feet [9], we utilize COTS GPS in mobile devices, which have the error range of 5 to $10m$. Therefore, the distance between two consecutive GPS coordinates is not always the same as the actual distance that a user has walked. To reduce the overall error rate, we propose the following filtering processes: segmentation, smoothing, and straight-line identification. These processes filter noise and curvilinear walking GPS data and produce only straight-line GPS data segments.

1) Segmentation (Pre-process): We segmentize GPS data points into different groups using the following criteria:

- *Immobility Detection*: If the time interval between two consecutive GPS readings is considerably larger than the values of all of the recorded intervals, it is likely that the mobile user has stopped walking. If the time interval is greater than the sum of the mean and three times of the standard deviation of the time intervals as a whole, we divide it into different segments.
- *Unrealistic Movement Detection*: Poor GPS satellite visibilities due to environmental obstacles often generate sudden jumps in GPS traces (see Figure 2). We compute the speed by dividing the distance between two consecutive GPS readings by the time interval. If the estimated speed is far greater than the values of the others; i.e., if it is greater than the sum of the average and two times of the standard deviation of the recorded GPS readings, we divide it into different segments.

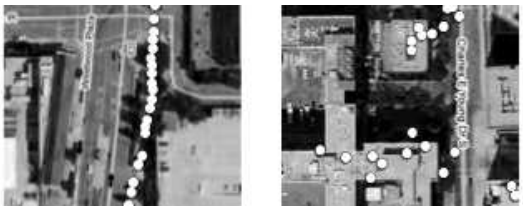
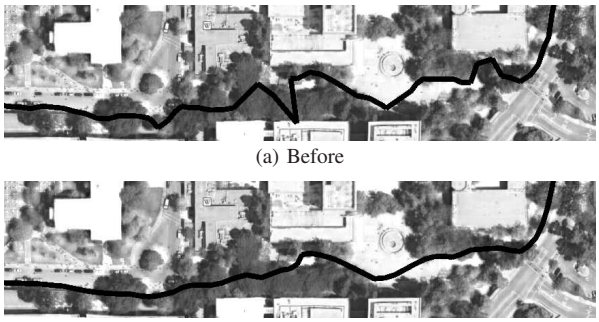


Figure 2. GPS coordinates collected in widely opened area (left) and near tall buildings (right)

The current prototype collects a GPS reading for every ten steps, which is approximately 5 to $10m$ in distance. When we consider that the GPS error range is 5 to $10m$, the error rate of the distance estimation for a given segment can be higher than 100% if it has only a small number of GPS readings. In general, a longer segment is required to effectively reduce the error rate. In our prototype, we remove segments that have less than ten GPS data points.



(a) Before

(b) After

Figure 3. Visualization of the Smoothing: Noisy GPS trace (a) becomes smoother after the convolution (b).

2) Smoothing: Even when a user walks in a straight line, GPS errors may make raw GPS traces jagged or curved, as illustrated in Figure 3(a). Such data fluctuations cause distance misprediction. Therefore, we smooth the pre-processed GPS segments using convolution, a linear filtering method that is similar to cross-correlation [10]. This process flattens the jagged or curved lines of GPS traces. As shown later, it also helps identify a straight-line segment. Other filtering methods such as autoregression [11] are not suitable for our system, because we are not predicting the current position from the previous coordinates. In the convolution, each GPS coordinate in the segment is smoothed with only its neighbors as follows:

$$f(t) * g(t) \equiv \int_0^h f(\tau)g(t-\tau)\,d\tau$$

where f is an array of (latitude | longitude) floats, g is $\frac{1}{a}$ for $0 \leq t < a$ and 0 otherwise, and h is the number of GPS position data in a segment.

The output of the convolution indicates the amount of overlap of a function g as it is shifted over another function f. It thus blends smoothed data segments with the pre-processed segments. Because a GPS coordinate consists of latitude and longitude values and because these two elements are independent, we separate them into two arrays and independently filter them. We use the continuous uniform distribution function g, because the values within a window should be equally weighted. An example of the output of a convolution is illustrated in Figure 3(b).

3) Straight-Line Identifier: The smoothing process straightens a jagged trace. However, when a user makes turns, it rounds off GPS readings sampled at the corners, making sharp corners dull and round. Moreover, some of the noisy GPS data still remain even after the smoothing process, thus severely distorting the actual path traveled. To remove the erroneous estimation caused by turns, we focus on walking patterns in straight-line roads where a user walks in a near straight line. Thus, we can exclude curves and GPS noise by considering only *near* straight-line walking patterns over noisy GPS readings. To this end, we propose a heading change (HC)-based filtering algorithm. The idea is to find a series of GPS coordinates where each heading is formed by two consecutive GPS readings points toward a similar direction.

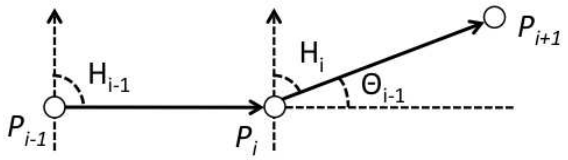


Figure 4. Definition of Heading Change (θ_{i-1})

The HC algorithm first finds a heading (or bearing) between two consecutive GPS coordinates. The algorithm begins by transforming the coordinates to a Cartesian space [12]. We then calculate the vertical (y) and the horizontal (x) distances between two consecutive points using:

$$y = \sin(P_{i+1}.Lon - P_i.Lon) \cdot \cos(P_{i+1}.Lat)$$
$$x = \cos(P_i.Lat) \cdot \sin(P_{i+1}.Lat) - \sin(P_i.Lat) \cdot \cos(P_{i+1}.Lat) \cdot \cos(P_{i+1}.Lon - P_i.Lon)$$
$$\overrightarrow{P} = [P_1, P_2, ..., P_h] \quad \text{where } P_i = (Lat_i, Lon_i)$$

Given x and y, we calculate the heading values $\overrightarrow{H} = (H_1, H_2, ..., H_{h-1})$, where atan2($y$,$x$) is the arctangent of y/x.

$$H_i = \frac{180}{\pi}[\text{mod}(\text{atan2}(y,x), 2\pi)], \quad 1 \leq i < h \qquad (1)$$

We then compute the heading changes, $\overrightarrow{\theta} = (\theta_1, \theta_2, ..., \theta_{h-2})$, using the heading values $\overrightarrow{H}$ from (1). The heading change of two consecutive heading directions is visualized in Figure 4.

$$\theta_i = H_{i+1} - H_i \quad 1 \leq i < h-1, -180^\circ \leq \theta < 180^\circ$$

Given the heading change values, $\overrightarrow{\theta}$, we can find the straight-line segments as follows: The algorithm calculates cumulative heading changes, $\overrightarrow{c}$ where c_i is the angle difference between H_1 and the heading angle of the edge from P_1 to P_{i+2} (see Figure 5).

$$\overrightarrow{c} = [\sum_{t=1}^{1}\theta_t, \sum_{t=1}^{2}\theta_t, ..., \sum_{t=1}^{k}\theta_t]$$

where k is length of $\overrightarrow{\theta}$.

Next, the algorithm looks for the maximum i such that c_i is less than the end-point threshold (ET) and c_j (where j is between 1 and $i-1$) is less than the angular threshold called the middle-point threshold (MT). Figure 5 visualizes ET and MT. If the GPS coordinates, $P_1, \cdots, P_{i+2}$, are within the boundary of the double line (=) where i is greater than a certain number (i.e., to minimize the error, each segment should have a sufficient number of GPS readings), we assume that $P_1, \cdots, P_{i+2}$ are on a straight-line segment. This follows re-entering the rest of the heading change values, $\overrightarrow{\theta_{i...h}}$ (excluding the straight-line segment), to find more straight-line segments. If the process finds no straight-line segment, it discards the first heading change value (i.e., $\overrightarrow{\theta_{2...h}}$). This process repeats until a straight-line segment is found. The current prototype uses 10^o and 35^o for ET and MT, respectively.

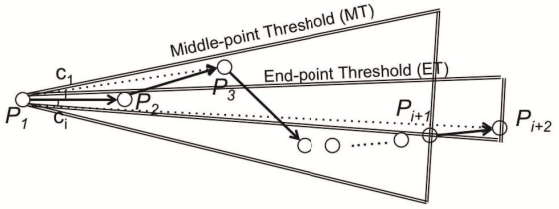


Figure 5. End-point Threshold (ET) and Middle-point Threshold (MT)

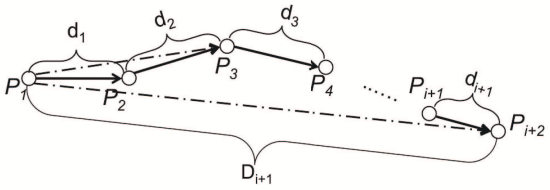


Figure 6. A Straight-Line Segment. d_i: Distance Between Two Consecutive Points (i.e., P_i and P_{i+1}), D_{i+1}: Distance Between Two Endpoints of the Segment

B. Walking Profile Calibration

1) Updating Stride Length Lookup (SLL): Once we discover a straight-line segment from the above process, we average the step frequencies that belong to the segment. We then estimate the stride length using the following two approaches:

- *end-to-end*: D_{i+1}, Distance between two endpoints of the segment (see Figure 6)
- *sum up*: $\sum_{j=1}^{i+1} d_j$, Cumulative distance of two consecutive points in the segment (see Figure 6)

The average stride length is then estimated by dividing the distance (i.e., D_{i+1} and $\sum_{j=1}^{i+1} d_j$) by the number of edges in the segment and the number of steps in an edge (i.e., ten). This sample from the segment is then used for updating the SLL.

Given the sample from the above process, the system updates the SLL using a linear least squares fitting with existing samples. The linear least squares fitting is known as the most common method of linear regression, which provides a solution of finding the best fitting line through a set of points. Recall that SLL is given as $s = \alpha \cdot f + \beta$ where s is the stride length, f is the step frequency, and α and β are constants. When the system takes a sample from the *Straight-Line Identifier*, it recalculates two coefficients, α and β, that are newly used for calculating the stride length.

2) Calibration Termination Condition: The system continuously detects the straight-line segments to update the SLL when the calibration is necessary. As the number of samples increases, the linear equation, given by the least squares method, converges. The system then stops the learning process after a certain threshold and turns off the GPS module to conserve energy. The termination criterion is given as:

$$\forall i, \left| \arctan\left(\frac{\alpha_i - \alpha_{i-1}}{1 + \alpha_i \cdot \alpha_{i-1}}\right) * \frac{180}{\pi} \right| < \gamma^\circ \qquad (2)$$

where i is between k and $k+m$, that is, the angle change between the slope of the new equation (α_i) and the previous slope (α_{i-1}) is smaller than γ° over m calibration periods. In our experiment, we set m as 5 and γ as 1^o, which is discussed in Section V.

3) Triggering and Re-calibration: The system can estimate the distance walked using the SLL discovered in the calibration process. However, the step frequencies and the stride lengths can change depending upon one's mood, body shape, and weight; e.g., a regression model constructed this week could be different from the one generated two weeks ago. One simple method is to compare the new SLL generated with the previous values periodically (say a month). The triggering frequency can be adjusted on the basis of the differences between the SLLs. If the user is experiencing dynamically changing SLLs, the system can trigger the re-calibration process more frequently, say once a week. Additionally, we can configure the system to automatically measure the changes in step frequencies to trigger auto-calibration. If the system finds any significant changes, it asks users whether they would like to perform re-calibration.

IV. AutoGait Prototype Implementation

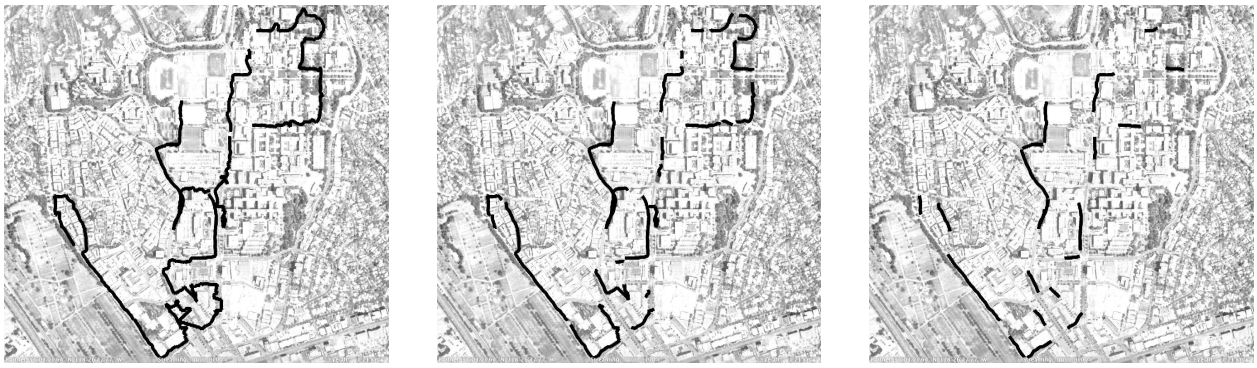
We implemented AutoGait on the Nokia N810 using Linux Python (version 2.5) and designed our own pedometer using low-power force sensors in the SmartShoe platform to avoid the low-acceleration sensing problem found in accelerometer-based pedometers. In this section, we briefly describe the SmartShoe platform and illustrate how we integrate AutoGait into the pedometer.

A. SmartShoe Platform

The SmartShoe platform has 1) low-cost sensors that measure motion, force, and pressure signals, and 2) a MicroLEAP [13] computing unit that acquires sensor data and transfers it using Bluetooth to a mobile user's handheld device. MicroLEAP is a mote-size wireless wearable sensor platform. It includes a low-power embedded processor, the Texas Instruments (TI) MSP430, for basic data processing, a high-resolution analog-digital converter, data storage, a Bluetooth module, and MEMS sensors. The sensor signals are digitized with 16-bit resolution or higher. More details of SmartShoe can be found in [14].

B. Pedometer Implementation

We implemented the pedometer in Linux Python (version 2.5), which is integrated with the calibration module as shown in Figure 1. Two MicroLEAPs (from the left shoe and the right shoe) are separately connected to the Nokia N810 using Bluetooth. We use threads to guarantee reliable data collection for both connections. When the program starts, it creates socket connections using Bluetooth to both MicroLEAPs. Once the connections are successfully established, the system issues a command to both MicroLEAPs, configuring a sampling rate, the streaming data format, and the number of channels. It then creates threads for data collection. The threads loop until the system detects a keyboard



(a) Raw GPS Data (b) Effect of Segmentation and Smoothing (c) Heading Change based Filtering

Figure 7. The Performance of the Straight-Line Identifier

interrupt command. Each thread (Bluetooth connection) runs separately to count steps. The thread periodically receives five pressure data samples from a MicroLEAP. Each sensor value is then converted to a Boolean value, indicating pressure or no pressure, respectively. Normal walking motion follows unique sequential patterns of stance and swing. In the stance motion, the ground contact force on the foot shifts from the heel to the toe, whereas in the swing motions, there is no pressure on the foot. Thus, by analyzing the sequential Boolean values, the system recognizes that a step has been taken. When the pedometer detects a step while the calibration is running, the pedometer passes its step frequency to the calibration module. Every ten steps, the module reads the GPS to find a straight-line segment. Once a straight-line segment is found, the SLL runs a linear regression to update the SLL.

V. Experiments

In this section, we describe the experiments of our proposed system.

A. Linear Relationship Verification (Treadmill)

Our first experiment was performed on a treadmill to illustrate the linear relationship between stride length and step frequency. One participant walked on a treadmill while changing walking speeds from 1.0 mph to 4.5 mph as follows: at every two hundred steps, the speed was increased by 0.5 mph. The system computes the step frequency for every step. The step frequencies are averaged, and thus, each result represents the average frequency for a given speed.

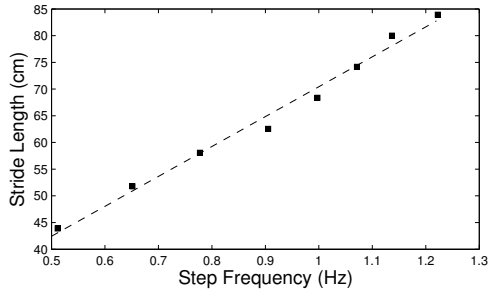


Figure 8. Linear Relationship Verification (Treadmill): Sample points (■) are used for estimating the slope (−−) using linear least squares.

We also recorded the total distance walked for each speed that was given by the treadmill. The distance was divided by the total number of steps taken to calculate the average stride length. Figure 8 clearly validates that there is a linear relationship between the stride length and the step frequency.

B. Identifying Straight-Line Segments and Calibrating SLL

To show that the stride length and the step frequency can be calibrated using GPS, the prototype system was used for collecting a dataset obtained by walking near the UCLA campus. During the experiments, a participant casually walked the same route six times with the Nokia N810 in hand. The traces are plotted in Figure 7(a).

As we described in the GPS data filtering section III-A, we processed the raw GPS data with a three-step filtering method: segmentation, smoothing, and straight-line identification. The result of each step is illustrated in Figure 7. By removing outliers and short segments, the pre-processing and smoothing separated a long segment into several pieces (see Figure 7(b)). Finally, the HC algorithm found only the straight-line segments from the smoothed segments as shown in Figure 7(c).

For each segment obtained after the filtering process, two samples were obtained by two different methods, *end-to-end* and *sum up*, that we mentioned in Section III-B. Figure 9 plots the results of the *sum up* (+) and the *end-to-end* (□) methods. The results show that the *end-to-end* method underestimates the stride length when compared to

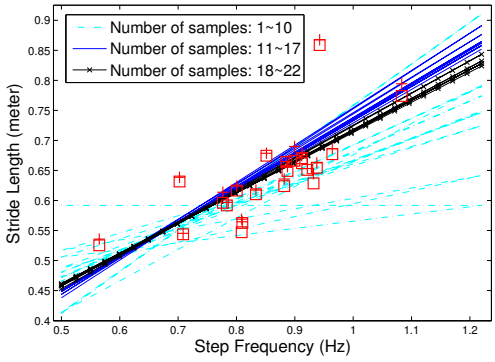


Figure 9. Least Square: As number of samples increases, the linear line converges to a line

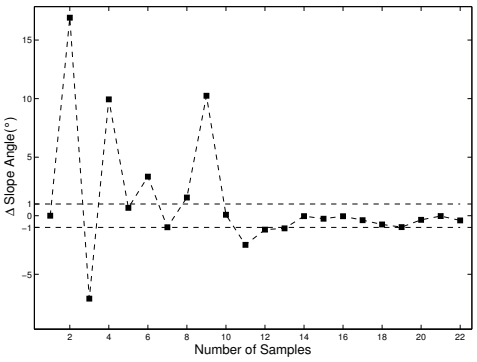


Figure 10. Angle variation between previous and current lines is rapidly decreasing as number of samples increases

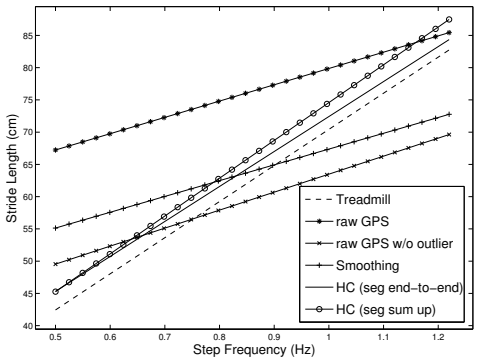


Figure 11. Discovered Stride Length Lookup (SLL): Stride Length vs Step Frequency

the *sum up* method. The system then separately updates the SLL of the *end-to-end* and the *sum up* using linear least squares fitting. The straight lines in Figure 9 show the updating progress of the SLL based on the *end-to-end* method. When there are only a few samples (1 to 10), the angle variation of the lines (see (2)) is large because of the small size of the learning set. However, when the number of samples is larger than 11, it gradually converges and shows minimal slope variations. The changes in the slope angle are plotted in Figure 10. The graph shows that the change in the angle stays within $\pm 1^o$ after the 12^{th} sample. In our implementation, we stop the discovery process when the angle change is sequentially smaller than 1^o over five time periods. Hence, the calibration process terminates after the 17^{th} sample.

C. Effectiveness of GPS Filtering

In order to show the effectiveness of our learning process, we compared our two methods with the following attributes:

- Raw GPS (RG): For every two sequential GPS coordinates, we estimate the stride length by dividing the distance between two points by the number of steps, and the step frequency by averaging step frequencies between two points. Each point (step frequency, stride length) is then used for estimating the slope using linear least squares.
- Raw GPS without Outliers (RGO): From the above process, we remove the outliers whose stride lengths are greater than the sum of the average and two times the standard deviation of all the stride lengths. The remaining samples are applied to the linear least squares fitting in order to obtain a linear equation.
- Smoothing (SM): After the smoothing process, several segments are obtained. Using the same process that is used for raw GPS data, the segments are used for estimating the linear equation.

The results are plotted in Figure 11. As the figure shows, the Treadmill and two HC methods follow similar trends; however, the lines generated by the two HC methods are above the Treadmill, because the stride length increases slightly when the user walks on the ground [15] compared to when the user walks on the treadmill, and the result indeed makes sense. The fitted lines of the other methods are by far different from those of the Treadmill or HCs. The raw GPS overestimates the stride length as compared to the others. On the other hand, the raw GPS without outliers significantly underestimates the stride length. This means that the raw GPS is very noisy, and thus, the effective filtering method is necessary. Smoothing is relatively close to two HCs, but it underestimates when the walking speed is high (i.e., step frequency is high).

D. Validation of SLL's Accuracy: Field Tests

To evaluate the accuracy of the calibrated SLL, we ran an experiment where the participant walked on both the treadmill and the ground for a mile. Because it is difficult to measure the accurate distance traveled while a user is walking on an unevenly surfaced ground, we had the participant walk four laps on a track (1 lap = .25 mile) at three different speed levels, as listed in Table I. Our system collected the step frequency for each step, and we estimated the distance walked by using the SLL profiles in Figure 11. During the test, the system missed one step on the treadmill; however, none of the steps were missed on the track.

Error rates are plotted in Figure 12, where the error rate is the percentage of distance that deviates from one mile; e.g., if the estimated distance is 1.1 mile, the error rate is 10%. From this experiment, we made the following observations. First, high error rates for RG and RGO indicate that the raw GPS data cannot be directly used for estimating the stride length. Second, SM underestimates the distance traveled (between -6% and -4.5%), because the smoothing process distorts the sharp corners and curves. This confirms that it is necessary to use straight-line segments to accurately estimate the stride length. Third, the *sum up* method performs two times better than the *end-to-end* method in the track test.

Speed		Slow	Moderate	Fast
Distance (m)		400	800	400
Lap Time (min:sec)		9:56	11:52	3:45
# of Steps (Ground truth)		718	1192	488
AutoGait	Est Dist (m)	395.9	795.4	396.3
	Error Rate	1.02%	0.58%	0.93%
Const. Stride Length ($0.7m$)	Est Dist (m)	502.6	834.4	341.6
	Error Rate	-25.7%	-4.3%	14.6%

Table I
ONE PARTICIPANT WALKED TOTAL FOUR LAPS ON A TRACK AT THREE DIFFERENT SPEED LEVELS.

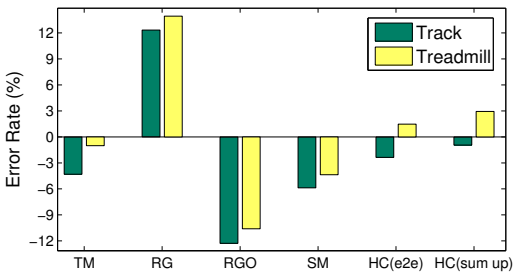


Figure 12. Error rate from Field Test: TM (Treadmill), RG (Raw GPS), RGO (Raw GPS without outlier), SM (Smoothed), HC (end-to-end), HC (sum up)

This is because the *end-to-end* method connects two end points of a straight-line segment to estimate the stride length, possibly shortening the actual path that the user walked, while the *sum up* method closely estimates the length of a real walking path by summing up all the edges within a straight-line segment. Finally, the *sum up* method works four times better than Treadmill (TM) in the track test while the TM performs three times better than the *sum up* method in the treadmill test. This implies that for a better estimation of the distance traveled, the SLL should be discovered outdoors.

On the basis of the *sum up* method, we evaluate how accurately the SLL estimates the distance walked at different speed levels. Table I lists the experiment results, which shows that the SLL-based method significantly outperforms the constant stride length-based method both at slow speeds and at fast speeds. More precisely, the error rate decreases from -25.7% to 1.02% at slow speeds and from 14.6% to 0.93% at fast speeds. This also shows that the distance walked is overestimated at slow speeds and underestimated at fast speeds when a constant stride length is assumed [5]. Even at moderate speeds, the accuracy of the SLL-based estimate is higher than that of the constant stride length-based estimate. This confirms that AutoGait performs well at different walking speeds.

E. Benchmark Studies

Setup: During the field test in Section V-D, the participant was equipped with two additional commercial products, an Omron pedometer (HJ-720ITC) and a Nokia Step Counter. The pedometer was clipped on the participant's belt, and the Nokia Step Counter was placed in the pocket. Both devices use different ways of estimating the distance traveled. The Omron pedometer estimates the distance by multiplying the number of steps by the participant's stride length (i.e., 70 cm in our experiment). The Nokia Step Counter measures the distance using a constant stride length that is calculated on the basis of a user's height and weight. We also evaluated the performance of Nike + iPod shoes. Because one participant cannot wear two different shoes at the same time, we instead had the participant walk on a track wearing the Nike+ shoes and the sports kid sensors separately. The test was performed with and without calibration. The calibration was done on a treadmill by making the participant walk $400m$ at moderate speeds as recommended by the manual. In each test, there were three trials with slow, moderate, and fast speeds. For each trial, the participant walked one lap of the $400m$ track.

Omron Pedometer and Nokia Step Counter: The benchmark study results are listed in Table II and Table III. The error rate of the Nokia Step Counter was 54.6% at slow speeds, 10.3% at moderate speeds, and 22.1% at fast speeds. The error rate was high, because the step detection was poorly done at slow speeds. The participant walked a total of 2398 steps, but the system detected only 1773 steps. The Omron pedometer missed around 10 steps, and its error rate was around -24.08% at slow speeds, -4.13% at moderate speeds, and 14.6% at fast speeds. While it outperformed the Nokia Step Counter, the constant stride length assumption kept the error rate high.

Speed	Slow	Moderate	Fast
Omron Pedometer (HJ-720ITC)			
Found Steps	709	1190	488
Est Dist (m)	496.3	833	341.6
Error Rate	-24.08%	-4.13%	14.6%
Nokia Step Counter			
Found Steps	266	1051	456
Est Dist (m)	181.7	717.8	311.4
Error Rate	54.6%	10.3%	22.1%

Table II
BENCHMARK STUDIES: ONE PARTICIPANT WALKED TOTAL FOUR LAPS ON A TRACK AT THREE DIFFERENT SPEEDS WITH OMRON PEDOMETER AND NOKIA STEP COUNTER.

Nike + iPod: We make the following observations. First, the accelerometer-based step detector could not detect the steps at slow speeds even if the sensors were mounted on shoes. Sensors were not sufficiently accelerated to generate the peak value. Note that the Nike shoes use the Speedmax algorithm [16] that finds the peak values in order to detect the steps. Second, Nike shoes use a constant stride length. As the results show, the distance estimated at moderate speeds was higher than the distance estimated at high speeds both with and without calibration. Given the fact that the stride length increases when the speed is high (i.e., a linear relationship), we conclude that the Nike shoes use a constant stride length for estimating the distance walked.

400m test	**Without Calibration**			**With Calibration**		
Speed	Slow	Mod	Fast	Slow	Mod	Fast
Lap Time	8:41	5:08	3:26	8:21	4:59	3:34
Est Dist (m)	160	460	390	290	410	360
Error Rate	-60%	15%	-2.5%	-27.5%	2.5%	-10%

Table III
NIKE+ SHOES TEST RESULT

F. Testing on Multiple Users

We tested AutoGait with multiple users in order to confirm that the system could be personalized. Three participants wore the SmartShoe-based pedometer with Nokia N810 and casually walked around the UCLA campus to calibrate the SLL by using the *sum up* method. The participants then walked four laps on a track (1 lap = .25 mile) to find the error rate of the SLL learned.

The results are shown in Table IV. The number of segments did not significantly affect the accuracy. The results show that participant B had four more segments than participant C, yet the error rate difference was minimal.

Moreover, SLL (i.e., α and β) learned by three different participants were different for individuals, which showed that the profile should be personalized in order to accurately measure the distance walked.

Participant	A	B	C
α (SLL)	0.453	0.064	0.539
β (SLL)	0.23	0.612	0.2156
Segments Found	15	18	14
Est Dist (m)	1577.5	1579.4	1616.9
Error Rate	-1.41%	-1.29%	1.06%

Table IV
MULTI-USER RESULTS

VI. DISCUSSION AND FUTURE STUDIES

A. Limitations of Using Accelerometer-Based Pedometers

Low-Acceleration Sensing Problem: Step misdetection or overdetection can occur because of accelerometer measurement errors and sensor misalignment. The impact of measurement errors is more pronounced when users walk slowly [5]. As accurate step detection is a significantly important factor for finding the linear relation in the AutoGait system (i.e., gathering GPS coordinates is based on the number of steps), the missing steps can cause the system to miscalibrate the linear walking profile. In this paper, we propose two potential solutions for this problem. First, by using a statistical method such as autoregression modeling [11], we can predict the number of missed steps and their step frequencies on the basis of the recorded history. The predicted data can then be used for discovering the linear relationship. Second, after we discover a straight-line segment to find a sample point (i.e., an average stride length and step frequency) for the linear regression, if the number of steps times the inverse of the step frequency is not close to the time difference between the first step and the last step in the segment, we can simply discard the segment and not use it for performing the linear regression. Thus, a straight line cannot be found if a user walks at a slow speed (because of the poor performance at slow speeds). Note that we can still use the linear regression method based on moderate and fast walking samples.

Data Processing Algorithms: There are a number of accelerometer data processing methods such as peak detection at the stance phase [17], zero cross/flat zone detection in the swing phase [18], or frequency analysis using FFT [9], [19]. Although these algorithms are suitable for finding steps from accelerometer data, not all of them can be used for real-time step detection. For instance, FFT is not suitable for our case because it requires buffering of data, which get decomposed into different frequency components in a batch, making it difficult for real-time step detection. On the other hand, peak detection and zero cross/flat zone algorithms detect steps on the basis of a threshold mechanism, which can be run in real-time.

B. AutoGait on Indoor Navigation Systems

Orientation Detection: In indoor navigation systems, orientation detection is very important for keeping track of a user's location. Generally, compasses or gyroscopes are used for detecting orientation. However, their performance depends on many factors such as the location's magnetic fields and the orientation of the sensors. Most recent Smartphones such as iPhone 3GS and Android G1 are equipped with compasses. In a future study, we will thoroughly test them to see whether they can be used for indoor navigation systems.

C. Consideration of Physiological Factors

In Case of Running: As a part of a future study, we will consider the case of running where the physiological model is different from that of walking. Separate profiles for walking and running will be maintained, and an activity detection mechanism [6] would determine which profile to use.

Impact on Age Variance: Zijlstra *et al.* showed that the linear walking profile applies to everyone, regardless of age [7]. Hence, we believe that the AutoGait system can benefit a wide range of people, from the young to the elderly. However, we have not evaluated our prototype with diverse age groups. We will validate this hypothesis in the future.

Walking Uphill/Downhill vs. Stride Length: Depending on the slope of inclination or declination, the stride length varies although the walking profile still follows the linear relationship [6], [20]. More precisely, the stride length reduces when the user walks uphill, and it increases when the user walks downhill. Unfortunately, our system does not account for this aspect yet. We will take advantage of altitude data from the mobile's GPS to estimate variations in steps when the user walks uphill or downhill.

VII. RELATED STUDIES

GPS-based pedometer products such as Garmin Forerunner and Nokia Sport Tracker [21] use GPS to offer athletes a personal training device that measures speed, distance, trace, calories burned, and pace. However, GPS does not work indoors, and continuous GPS data sampling is power hungry. Moreover, we focus on the calibration methodology of human walking profiles using GPS rather than finding the distance walked. Therefore, these devices are not suitable for our purposes.

Several studies investigated the human walking profile to compute stride lengths. Scarlett *et al.* [19] proposed an algorithm to estimate the travel distance by double integrating the raw accelerometer data from a hip-located device that requires an accurate sensor alignment. This technique has been applied to indoor localization or pedestrian navigation such that a mobile system can track a user's current position using a known starting point and a two-dimensional relative trace. Researchers showed that a linear relationship between stride length and step frequency is an important feature of a human walking profile [6]. Lee *et al.* [22] proposed a calibration method that estimates the stride length based on acceleration measurements that are carried out on a treadmill. Ladetto *et al.* [9] developed the Pedestrian Navigation Module (PNM) for Leica Vectronix AG using the linear relationship. In practice, the main drawback of these approaches is that estimating the distance using accelerometers is error-prone because of sensor misalignment and low acceleration in the case of slow walking speeds.

There were several approaches that explore auto-calibration using GPS [9], [23]. Leick *et al.* showed that accurate calibration is feasible, but the performance mainly depends on the accuracy of GPS because it requires stride-level distance measurement [23]. To the best of our knowledge, all the existing solutions are based on DGPS for accurate distance estimation. In practice, it is neither feasible nor convenient for mobile users to carry a bulky, expensive DGPS device in their everyday lives. The main departure from existing studies is that we propose an algorithm that processes noisy COTS GPS readings and calibrate the linear profile on the background.

Ultrasound sensors have been considered for measuring stride length; a source sensor can measure the time for an ultrasound signal to echo back from the destination sensor. The distance between two sensors can be estimated by multiplying the time-of-flight by the speed of sound. Koss *et al.* [24] used ultrasonic sensors at anatomical and anthropometrical points to measure human motion. In addition, a portable walking distance measurement system was developed using ultrasonic wave characteristics [25]. Yeh *et al.* elaborated the design, implementation, and evaluation of a footstep-based indoor location system by attaching the ultrasound sensors to traditional Japanese GETA sandals [26]. However, ultrasound sensors have several limitations in practice: (1) they require additional sensors such as gyroscopes to differentiate the steps taken sideways or backwards from steps taken forward because of the dynamics of human motion, (2) they may interfere with other ultrasonic devices nearby or background noise, and (3) they consume more energy for acoustic transmissions.

VIII. Conclusion

Conventional pedometers use a constant stride length for estimating the total distance walked. We argued that the constant stride length is a major source of error in accurately estimating the distance walked. To overcome this problem, we exploited the fact that there exists a linear relationship between the stride length and the step frequency and developed *AutoGait*, a mobile platform that opportunistically calibrates a user's linear walking profile using a mobile's COTS GPS while the user is walking outdoors. AutoGait uses a novel GPS filtering algorithm to calibrate a walking profile with noisy COTS GPS readings. By implementing the AutoGait prototype in the Nokia N810 and interfacing it with the SmartShoe-based pedometer, we demonstrated that the platform is applicable to any pedometer software running on Smartphones or handheld devices. Our extensive experiments confirmed that the AutoGait system outperforms existing solutions and significantly lowers the error rates, achieving more than 98% accuracy in our testbed scenarios.

Acknowledgment

The authors wish to acknowledge Myungwon Ham, Jiyeon Lee, Youngtae Noh, Sungwon Yang, and many other for their help in the experiment and thank Brian Choi for helpful comments on earlier drafts. This work is supported in part by the National Science Foundation under Grant No. CNS-0917408.

References

[1] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 1, pp. 13–32, 2009. [Online]. Available: http://dx.doi.org/10.1109/SURV.2009.090103

[2] L. Fang, P. Antsaklis, L. Montestruque, M. McMickell, M. Lemmon, Y. Sun, H. Fang, I. Koutroulis, M. Haenggi, M. Xie, and X. Xie, "Design of a wireless assisted pedestrian dead reckoning system - the navmote experience," *Instrumentation and Measurement, IEEE Transactions on*, vol. 54, no. 6, pp. 2342–2358, Dec. 2005.

[3] A. Lamarca and et al., "Place lab: Device positioning using radio beacons in the wild," in *In Proceedings of the Third International Conference on Pervasive Computing*. Springer, 2005, pp. 116–133.

[4] Z. Sun, X. Mao, W. Tian, and X. Zhang, "Activity classification and dead reckoning for pedestrian navigation with wearable sensors," *Measurement Science and Technology*, vol. 20, no. 1, p. 015203 (10pp), 2009.

[5] K. De Cocker, G. Cardon, and I. De Bourdeaudhuij, "Validity of the inexpensive Stepping Meter in counting steps in free living conditions: a pilot study," *Br J Sports Med*, vol. 40, no. 8, pp. 714–716, 2006.

[6] R. Margaria, *Biomechanics and Energetics of Muscular Exercise*. U.K. Clarendon, 1976.

[7] W. Zijlstra, "Assessment of spatio-temporal parameters during unconstrained walking," *European Journal of Applied Physiology*, vol. 92, no. 1, pp. 39–44, 2004.

[8] R. Edgeworth, B. Keen, E. Crane, and M. Gross, "Effect of speed on emotion-related kinematics during walking," *North American Congress on Biomechanics*, no. 2, pp. 164–168, August 2007.

[9] Q. Ladetto, "On foot navigation: continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering," in *ION GPS 2000, Salt Lake City, Utah, USA*, 2000.

[10] "Convolution," http://en.wikipedia.org/wiki/Convolution/.

[11] "Auto-Regression," http://en.wikipedia.org/wiki/Auto-regression/.

[12] "Cartesian coordinate system," http://en.wikipedia.org/wiki/Cartesian_coordinate_system/.

[13] L. Au, W. Wu, M. Batalin, D. Mclntire, and W. Kaiser, "Microleap: Energy-aware wireless sensor platform for biomedical sensing applications," *IEEE BIOCAS 2007*, pp. 158–162, Nov. 2007.

[14] F. Dabiri, A. Vahdatpour, H. Noshadi, H. Hagopian, and M. Sarrafzadeh, "Electronic orthotics shoe: Preventing ulceration in diabetic patients," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, Aug. 2008, pp. 771–774.

[15] H. Stolze, J. P. Kuhtz-Buschbeck, C. Mondwurf, A. Boczek-Funcke, K. Jhnk, G. Deuschl, and M. Illert, "Gait analysis during treadmill and overground locomotion in children and adults," *Electroencephalography and Clinical Neurophysiology/Electromyography and Motor Control*, vol. 105, no. 6, pp. 490 – 497, 1997.

[16] "SpeedMax White Paper, Dynastream Innovations Inc." http://www.dynastream.com/datafiles/SpeedMax%20White%20Paper%20v4_1.pdf/.

[17] R. Jirawimut, P. Ptasinski, V. Garaj, F. Cecelja, and W. Balachandran, "A method for dead reckoning parameter correction in pedestrian navigation system," *Instrumentation and Measurement, IEEE Transactions on*, vol. 52, no. 1, pp. 209–215, Feb 2003.

[18] S. Y. Cho, C. G. Park, and G. I. Jee, "Measurement system of walking distance using low-cost accelerometers," *4th ASCC*, 2002.

[19] J. Scarlett, "Enhancing the performance of pedometers using a single accelerometer," *Application Note, Analog Devices*, May. 2008.

[20] R. W. Levi and R. Marshall, "Navigation device for personnel on foot," Patent 6 813 582, November, 2004.

[21] "Nokia Sport Tracker," http://sportstracker.nokia.com/nts/main/index.do/.

[22] S.-W. Lee and K. Mase, "Recognition of walking behaviors for pedestrian navigation," *IEEE CCA '01*, pp. 1152–1155, 2001.

[23] A. Leick, *GPS satellite surveying / Alfred Leick*, 2nd ed. Wiley, New York :, 1995.

[24] "Joint kinematics and spatial-temporal parameters of gait measured by an ultrasound-based system," *Medical Engineering & Physics*, vol. 26, no. 7, pp. 611 – 620, 2004.

[25] Y. Jang, S. Shin, J. W. Lee, and S. Kim, "A preliminary study for portable walking distance measurement system using ultrasonic sensors," *IEEE EMBS 2007*, pp. 5290–5293, Aug. 2007.

[26] Yeh, Shun-Yuan, Chang, Keng-Hao, Wu, Chon-In, Chu, Hao-Hua, Hsu, and Jane, "Geta sandals: a footstep location tracking system," *Personal and Ubiquitous Computing*, vol. 11, no. 6, pp. 451–463, August 2007.