Pervasive and Mobile Computing 13 (2014) 52–66

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

RRACE: Robust realtime algorithm for cadence estimation

Idin Karuei*, Oliver S. Schneider, Bryan Stern, Michelle Chuang, Karon E. MacLean

Department of Computer Science, University of British Columbia, Canada

A R T I C L E I N F O

Article history:
Received 30 August 2012
Received in revised form 3 July 2013
Accepted 16 September 2013
Available online 25 September 2013

Keywords:
Cadence
Realtime
Empirical evaluation
Smartphones

A B S T R A C T

We present an algorithm which analyzes walking cadence (momentary step frequency) via frequency-domain analysis of accelerometer signals available in common smartphones, and report its accuracy relative to the published state-of-the-art algorithms based on the data gathered in a controlled user study. We show that our algorithm (RRACE) is more accurate in all conditions, and is also robust to speed change and largely insensitive to orientation, location on person, and user differences. RRACE's performance is suitable for interactive mobile applications: it runs in realtime (∼2 s latency), requires no tuning or *a priori* information, uses an extensible architecture, and can be optimized for the intended application. In addition, we provide an implementation that can be easily deployed on common smartphone platforms. Power consumption is measured and compared to that of the current commercially available mobile apps.

We also describe a novel experiment design and analysis for verification of the best-optimized RRACE's performance under different conditions, executed outdoors to capture normal walking. The resulting extensive dataset allowed a direct comparison (conditions fully matched) of RRACE variants with a published time-based algorithm.

We have made this verification design and dataset publicly available, so it can be re-used for gait (general attributes of walking movement) and cadence measurement studies or gait and cadence algorithm verification.

1. Introduction

Contemporary smartphones carry a wealth of sensors which can be used to estimate aspects of a user's context and activities that are of value in a multitude of applications. One notable example, walking cadence ("the beat, time, or measure of rhythmical motion or activity"—Merriam-Webster; used hereafter to refer to the step frequency as estimated in realtime), has broad utility for applications that support fitness, rehabilitation, gaming, navigation, and context awareness. But available cadence detection methods require unrealistically specific placement and sensor calibration to achieve viable performance. There is a need for realtime cadence detection that is robust to carrying method.

Current realtime mobile cadence detection methods are largely based in the time domain, detecting timing of individual footfalls which themselves are estimated when an accelerometer signal exceeds a threshold. This threshold dependency is not ideal from a usability standpoint because the threshold is specific to many parameters—for example, Melanson et al. show that the threshold-based pedometer accuracy changes dramatically by age, weight, and height [1]. Detection accuracy consequently necessitates device (or additional sensor) placement in a location known to the algorithm, on one of a small number of body sites with highly regular movement – *e.g.*, the pocket, on the hip, or the leg [2] – at a specific orientation, and with user-specific calibration to adjust for weight, height, and body shape of the user. This invokes a harsh tradeoff between the reliability and usability [3].

* Corresponding author. Tel.: +1 7789904346.
E-mail addresses: idin@cs.ubc.ca, idin@karuei.com (I. Karuei), oschneid@cs.ubc.ca (O.S. Schneider), maclean@cs.ubc.ca (K.E. MacLean).

1574-1192/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.pmcj.2013.09.006

Frequency methods for cadence detection have received little attention to date, yet in contrast to acceleration thresholds, there is substantial qualitative commonality in frequency profiles as a function of position in various body locations [4]. Because the frequency and wavelength of the acceleration depend on the time interval between footfalls and the wave's shape and amplitude depend on the individual and location on the body, theoretically the major frequency component of the acceleration should be more robust to the individual-, location- and model-specific amplitude concerns which make time-based thresholds so problematic.

In this paper, we describe an algorithm (RRACE, for "Robust Realtime Algorithm for Cadence Estimation") to analyze cadence through a frequency-domain analysis of movement, and report its accuracy based on the data gathered in a user study. RRACE's basic structure is a computationally efficient moving window that is subjected to a spectral analysis followed by an analysis of frequency peaks. Empirically, we found that the performance peaks at a window length of 4 s, producing about 2 s latency including computational delay. This algorithm is extendable, allowing for improvements with advanced filtering or harmonic analysis, and can be used to provide spectral information for classification of gait (general attributes of walking movement) and other gait analysis applications.

While others have reported using frequency-based approaches [4,5], our approach's exceptional robustness is due in part to its ability to utilize the non-uniformly sampled data (the most readily available) and in part to the reliance on acceleration vector magnitude (the component unaffected by orientation) to determine cadence without knowledge of the placement of the device on the user's body.

Our contributions are (a) a cadence detection algorithm that can work across many body locations, is robust to change of orientation, and does not require calibration; (b) an experimental setup for assessing the accuracy of a gait detection method across many body locations, outdoors and under normal, unconstrained walking conditions; (c) performance data examining the effects of body location and speed on the algorithms we tested; (d) a thorough comparison between our frequency-based gait detection method and the highest-performing published time-based acceleration threshold method, hereafter referred to as the time-based method; and (e) an implementation that can be easily deployed on common smartphone platforms.

After discussing related work, we describe the RRACE algorithm and present our pilot and main validation experiment with RRACE running in realtime on a smartphone. We then compare RRACE to the time-based method, and conclude with a discussion of our findings and plans for future work.

2. Related work

To ground the presentation of our algorithm and evaluation, we first discuss realtime gait and cadence detection and its applications, then examine the state-of-art in time-domain and frequency-domain methods of cadence detection.

2.1. What is realtime cadence detection good for?

Gait and cadence information is relevant to many current and future mobile applications. Often attributed to Thomas Jefferson [6], the modern pedometer has long been a fitness tool for dedicated walkers and runners. Today's ever-expanding lineup of smartphone app versions further support logging, mapping, calorie burning estimates and social media [7–9].

Kavanagh and Menz point out the popularity of the accelerometer-based systems for human gait measurement and give a broad overview of accelerometer-based gait measurement systems with suggestions on optimal use conditions, reliability, and applications [10].

A number of fitness applications and products focus on automaticity, personalization, and direct feedback to increase motivation. As early as 2008, UbiFit used persuasive technology in its visual displays (using a metaphor of a garden's healthiness) of activity and goal achievement [11]. The Nike+iPod Nano measures distance, speed, and energy expenditure and can be programmed to play a motivational song when necessary [12]. Endomondo, claimed to be the most highly-reviewed activity monitoring Android app, uses a GPS signal to track speed, distance, duration, and calories burnt for running, cycling, and other sports [13]. Runtastic Pedometer, another Android app, also uses accelerometer data to count steps and measure calories burnt [14].

MPTrain (later extended to be TripleBeat) goes further by selecting and playing music with specific features to support pace goals like speeding up and slowing [8,15]. Garmin produces the Forerunner 910XT, a multisport watch that can be used for running, biking, and swimming [16]. It can detect walking steps and swimming and cycling strokes using its 3-axis accelerometers, measure elevation by a barometric altimeter, and be paired with a heart rate monitor; as for many tools, users can plan their workouts and analyze their activity through a number of metrics. The burgeoning area of exercise games could benefit from realtime knowledge as well; the previous work has linked game performance to step count [17] and overall physical activity [18].

All-day wearable activity monitoring currently includes successful commercial products like the Nike+Fuelband [19] and the FitBit [20] and mobile apps such as Endomondo [13] and Runtastic Pedometer [14]. These products, using 3-axis accelerometers along with various extras like GPS and ambient light sensors, aim to track and support goal achievement including steps taken, calories burned, and hours of sleep, providing a global view of activity levels as distributed over the day, week, and longer periods of time. Based on our informal measurements, Fuelband (worn on the wrist) appears to be less precise in measuring steps, while we saw Fitbit's error remain within a 5% bound and Runtastic Pedometer's within a 10% bound when counting steps.

These devices are representatives of the current market selection, which is rapidly moving. Popularity and supported price points (presently \$100–200 USD) highlight the growing consumer interest in holistic, conveniently acquired perspectives on their activity.

Meanwhile, it is possible to fuse cadence estimates with other data to identify more complex user states. When GPS is unavailable, they can augment the navigation algorithms through dead-reckoning [21,5]. Accurate cadence information provides a valuable feature for mobile fitness games and detailed guidance tasks that require higher-resolution data (*e.g.*, skipping, hopping, "turn here") in addition to GPS and biometrics [3]. Context-aware applications benefit from discerning walking, running, or sedentary states by using gait along with posture, auditory and other data to optimize notification timing [22–24]. Cadence can also supplement interior GPS and localization systems [25]. In all of these examples, accuracy and convenience of mobile collection are paramount.

A number of specialized, commercially available devices record and analyze human movement with good accuracy for medical purposes such as clinical, biomechanical, physical therapy, and movement disorders research, as well as athletic tuning. *Movement Monitors* by APDM Movement Monitoring Solutions are watch-sized Inertial Measuring Units, or IMUs, intended to be worn on multiple sites simultaneously (wrist, ankle, belt, and sternum straps), using accelerometers, gyroscopes, and magnetometers [26]. Industrially, IMUs are a valuable diagnostic and research tool for industrial vibration or movement monitoring, inertial guidance, virtual reality, or any application where precise monitoring of subtle movement is required. However, in these specialized situations it is feasible to wear or install a potentially expensive specialized device, precisely calibrated and location constrained. This is not the case for most potential consumer uses of cadence or gait detection.

2.2. Sensor type

Traditional pedometers identify individual steps using mechanical or piezoelectric sensors. Purely mechanical sensors detect a step if acceleration surpasses a threshold, measured when a sensor element strikes a surface. Piezoelectric sensors vary in form and sophistication. Like their mechanical cousins, many operate on an acceleration-threshold principle while more sophisticated devices compare an acceleration time series to a model of a step. The variants found in contemporary smartphones and IMUs, however, typically rely on 3D accelerometers. Their output can be processed in the same way as a piezoelectric or mechanical sensor, but also give rise to new algorithmic possibilities as described below.

There has been some sensor-based improvement in pedometer accuracy observed for piezoelectric relative to mechanical sensors, in particular at very slow walking speeds, likely due to increased sensitivity. A 2004 treadmill-based analysis of mechanical and piezoelectric pedometers found error reduced with speed for slow walking: 29% ($<$2.0 mph), 9%–26% (2.0–3.0 mph), and 4% ($>$3.0 mph) for mechanical pedometers. In a second variant, at speeds between 1.8 and 2.0 mph, the piezoelectric's error was $<$3% as compared to the mechanical sensors's 5%–48% [1]. The lowest error reported (0.3%) is for a piezoelectric sensor worn on the ankle [2]. In these studies, all pedometers were tested while worn on their optimal and calibrated specified location with specific orientation.

Accelerometer-based instruments are sampled using an embedded CPU, and accessed by an application through its operating system. Thus, their usable accuracy is both due to the sensor itself, and the quality, rate and latency of access to its output permitted by the operating system; these parameters all vary widely and their relative impact is not generally discussed. Current overall performance levels are discussed below.

2.3. Estimating cadence

2.3.1. Time domain

Whether standalone or in an iPhone app, an algorithmic (programmed) cadence estimate derived in the time domain is based on thresholds and step model parameters. These are in turn generated from the user-supplied information such as body mass and height, as well as the sensor's known or constrained location-on-person (LOP), and details of the hardware platform. Without this, accuracy is very poor (as we will demonstrate in Section 5.2.1), and this need for substantial context and/or limitations in where and how they can be worn is their major drawback.

The most straightforward solution for cadence estimation of any type is to analyze the acceleration in the time domain and detect individual footfalls. This requires just a single axis of acceleration, and produces algorithms that are computationally lean. Time domain approaches can be quite effective when the context information is known or constrained, being simple and reasonably accurate. A brief peak-detection algorithm (such as the one we compare later in this paper) delivers a latency equivalent to the last two steps.

Yang et al. sampled a waist-mounted tri-axial accelerometer module with built-in low pass filter, and computed autocorrelation in the time domain to measure cadence in realtime [27]. They reported a mean absolute percentage error of 4.89% when comparing their results with cadence measurements from the synchronized video. Their algorithm used a 3.5 s window.

The specifics of most commercial pedometer algorithms such as Runtastic Pedometer [14] are unavailable. However, an MPTrain publication [8] identifies its step detection as an adaptive accelerometer threshold with a low-pass filter, and reports its accuracy as comparable to standard piezoelectric pedometers [8]. We further describe the MPTrain algorithm in Section 5.1, as we use it for comparison with our algorithm.

2.3.2. Frequency domain

Frequency analysis has been instrumental in revealing interesting characteristics of gait (*e.g.*, discriminating the steps of the left and right foot [28] or comparing acceleration frequency content of two devices to determine if they are carried by the same person [4]). Zhao et al. use gait detection in assisted GPS systems [5], and identify the uncertainty of the sensor location as an important issue. Their solution is to classify the sensor location by extracting time and frequency domain features, then choosing a dead-reckoning algorithm according to the classification result.

Unlike cadence estimation in the time domain, where each footfall is recorded and time-stamped, a frequency-based algorithm looks at a bigger picture: it identifies the signal's major frequency components during a given window of time. At the cost of not detecting single footfalls and (typically) a larger delay to collect multiple samples, a frequency-based algorithm is far less dependent on signal shape and amplitude. This is because a frequency-based algorithm can distinguish between the major frequency component of the signal influenced by the repetition intervals (*i.e.*, step duration), and the harmonics influenced by the placement of the sensor and subject's unique walking pattern and noise. A frequency-based cadence estimation algorithm is thereby *theoretically* more robust to individual differences and sensor location than a time-based approach.

Thus, in applications where the exact time of footfalls is not required but ease and flexibility of use is valued, a frequency-based algorithm seems a promising approach, and is the one we took here. For related reasons, autocorrelation is another avenue that deserves attention, although it is beyond the scope of this paper. For either, achievable latency and accuracy then becomes the crucial question, and necessitated a development plan that included careful validation. Of published algorithms, the frequency-based ones state that they depend on proprietary info such as placement on the body or adjustments to the parameters to compensate for user differences. Kavanagh and Menz note the necessity of user-specific calibration procedures and errors caused by change of orientation [10]; they present an elaborate list of accelerometer attachment methods from past research with every one of them using a single location for the placement of the sensors. Zijlstra and Hof for example, like the majority of other researchers, placed accelerometers on the lower trunk [28]; specifically, they fixed the position of accelerometers at the dorsal side of the trunk with a fixed orientation. To our knowledge no realtime frequency-based methods have been reported for measuring cadence that uses the built-in sensors of a commodity smartphone and works out-of-the-box (*i.e.*, without calibration).

2.4. Performance assessment of cadence estimation algorithms

Published performance data is a rarity in cadence and gait estimation. Schneider et al. elaborate on the challenge of comparing the performance of such algorithms for real-time gait classification and accelerometer-based activity recognition [29], which are partly due to the large number of possible parameters and settings, and format of testing. The sheer logistical effort of precise validation may be an even more significant problem: natural walking is best done outdoors, and the technique in question must be compared with one or ideally two additional, independent and highly accurate 'gold standard' methods that are sampled at the same time. As can be seen in the following pages, this entails a considerable commitment in setup and data collection that most published works have omitted.

Furthermore, many of the examples we have cited are proprietary algorithms in commercial products released within a fast-moving market, with minimal or zero information available about their function or performance. Without easy access to their internal realtime data streams (for example, FitBit must compute the realtime step frequency, but does not share it with the consumer even post-hoc) it is difficult for a 3rd party to independently verify their accuracy and other parameters.

Much of this difficulty would disappear with the availability of standardized datasets: published trajectories of carefully collected and documented acceleration data, ideally as streams obtained simultaneously from multiple points on the body during a range of walking conditions. Different algorithms can then easily be compared. This practice is common in other communities, such as machine learning, but there is no standard dataset for gait detection that we know of. For this reason, we are making our own dataset available, as detailed in Section 4.

3. Approach: the RRACE algorithm

To support other research in our lab, we required a reliable, outdoor-ready cadence detection method that was both unconstrained in body location, and does not require the users to acquire or wear a specialized hardware.

Our goal was therefore to develop an algorithm that measures cadence at the same rate of accuracy as the best on record [30] (5% error) or better, that works on typical smartphones and is independent of orientation, placement on the body and individual wearer's physiology, and works out-of-the-box and in realtime. We also predicted that due to their growing ubiquity, a highly usable, smartphone-ready cadence detection algorithm would enable many new possibilities beyond our immediate needs. We chose a frequency-based approach for the reasons cited above, and developed an implementation that solved a number of inherent complexities as described below.

3.1. Overview

Our cadence-detection algorithm, Robust Realtime Algorithm for Cadence Estimation (RRACE), performs a spectral analysis on a four-second window of sampled 3-axis accelerometer data. Our approach has three characteristics that make

```
function RRACE():
  (timestamps, xs, ys, zs) := get_accelerometer_values(from 4s ago to now);
  n := length(timestamps);
  magnitudes := new array of length n;
  for i := 1 to n do
    magnitudes[i] := sqrt(xs[i]^2 + ys[i]^2 + zs[i]^2);
  size := 128*n;
  hifac := 0.25;
  ofac := 4;
  frequencies := fasper(timestamps, magnitudes, size, ofac, hifac);
  cadence := most_powerful(frequencies);
```

Fig. 1. RRACE Pseudocode.

it appropriate for the realtime cadence estimation on mobile phones: (a) it is independent of the body location and subject differences (as discussed before), (b) it is robust to orientation, and (c) it is robust to sampling irregularities.

Without published details or even the identity of other frequency-based algorithms, it is difficult to compare our approach to others' on theoretical grounds. However, all the frequency based algorithms of which we are aware report using fixed-rate sampling (*e.g.*, [31]), and, as some of them point out, for smartphone signals this would likely be a source of considerably reduced accuracy.

3.2. Implementation details

Supporting orientation-invariant information: to estimate overall movement from this measure, we use the magnitude (Euclidean or L-2 norm) of the three accelerometer axes (x, y, z) as our signal, as in [4]. This is a simple path to orientation invariance which we later show to be effective.

Accommodating non-uniform sampling (FASPER): most smartphones supply accelerometer data that is not sampled at a constant rate (*e.g.*, 25 ± 5 Hz); our data indicate that irregularities in accelerometer sample intervals are endemic. For example, the variance in the data analyzed for this paper is:

- sampling period: mean = 40.0 ms, median = 31.0 ms, sd = 37.7 ms;
- sampling frequency: mean = 127.7 Hz, median = 32.3 Hz, sd = 231.5 Hz.

Spectral analysis of such irregular data is not possible with FFT (Fast Fourier Transform), which computes a Fourier decomposition under the assumption that samples are equispaced. Attempts to 'repair' the data, *e.g.*, with interpolation, obviously introduce new sources of uncertainty, and this renders the most common spectral analysis methods inappropriate.

However, the Lomb–Scargle periodogram approach (also known as least-squares spectral analysis), derived by Lomb [32] and later validated with a mathematical proof by Scargle [30], accurately handle non-equispaced data by, effectively, fitting a sine wave and estimating its frequency spectrum.

In particular, the FASPER implementation (''Fast Calculation of the Lomb Periodogram'') [33] employs four parameters: the vector time series along with the time coordinate of each sample, an output gain and an oversampling parameter to control resolution of the computed spectrum. FASPER computes the significance level for each of a discrete set of frequencies.[1]

RRACE uses FASPER to find the spectrum of the overall movement of the device. We then make the key assumption that cadence is the most significant frequency peak in the spectrum for a given computational window. We define our algorithm's latency as half the window length—*e.g.*, a 4 s window has a latency of 2 s.

3.3. Pseudocode

Pseudocode for our implementation is shown in Fig. 1. We obtained our best results using 0.25 and 4.0 for FASPER's output gain (''hifac'') and oversampling (''ofac'') parameters, respectively.

3.4. Android-based validation platform

The results of Section 4 are based on the data from up to six simultaneously-worn Google Nexus One smartphones running Android OS version 2.3.4 (Gingerbread). Our main application was implemented in Java, the primary programming language for Android development. Numerical programming algorithms (including FASPER) were implemented in C for speed benefits and because of readily-available implementations [33]. We used the Java Native Interface (JNI) to connect the two languages.

[1] Although it may be possible to improve the performance of our algorithm through signal processing, *e.g.*, by employing a smoothing filter, for the current analysis we did not use any filter or other processing components other than the parts we describe here. This permitted us to make the fairest comparison possible to other algorithms, since we were not aware of what optimization they had undergone.

4. Experimental validation of RRACE

In laying out RRACE's formal validation, we first summarize a pilot study which informed our subsequent methodology. We then describe our full study's walking task, apparatus, and measurement; and its design, metrics, analysis, and subjects. Next, we present the results of the analysis for the optimally configured (4 s window) RRACE and compare it with other RRACE variants. Finally, we measure the power consumption of our algorithm and compare it with similar Android apps in the market.

Our dataset is available at: http://www.cs.ubc.ca/labs/spin/data/.

4.1. Treadmill-based pilot validation

Before conducting our full outdoor experiment, we built confidence in our general approach (algorithm and smartphone implementation) with a preliminary study based on four participants (one female) who volunteered without monetary compensation out of interest in the research. The setup consisted of a treadmill, the three Google Nexus One smartphones available at the time, and a PC x86-64 for manual logging of footfalls. Smartphones were synchronized with the PC; each phone recorded accelerometer signals (average sampling frequency = 24.8 Hz) and estimated cadence using RRACE in realtime.

Subjects walked for 15 min at a selection of speeds chosen to represent slow to fast walking based on similar studies and pedestrian speeds [2,34], while wearing smartphones on 3 of the 6 location-on-person (LOP) sites at a time, randomly selected each trial. They then walked for another 15 min wearing the phones on the other three locations. For both segments, they were instructed to adjust their walking speed to keep up with the changes in treadmill speed, but given no instructions as to step frequency.

We assessed the accuracy of cadence measurement relative to the manually recorded step interval (T_m). Our primary metric ("Error Ratio" or *ER*) was thus the ratio of RRACE's measurement "error" (the difference between the frequency measurement produced by RRACE, F_a, and the reference frequency, F_r) to the reference frequency:

$$ER = \frac{|F_a - F_r|}{F_r} \tag{1}$$

where

$$F_r = \frac{1}{T_m}.$$

Pilot study results: in an ANOVA on *ER*, our independent variables were LOP (6 sites), Speed (10 speeds ranging from 0.45 m/s to 1.65 m/s), and Window Size (2 lengths: 4 s and 8 s). We used a significance level of 0.05, applying a Bonferroni correction to counteract the multiple comparisons problem.

Location had a significant effect on *ER*. *Front pocket* ($mean = 6\%, sd = 11\%$), *belt* ($mean = 14\%, sd = 30\%$), *arm* ($mean = 15\%, sd = 30\%$), and *bag* ($mean = 15\%, sd = 32\%$) were much more reliable than *back pocket* ($mean = 30\%, sd = 38\%$) and *hand* ($mean = 31\%, sd = 27\%$). Speed also had a significant effect on *ER*. RRACE had a lower Error Ratio at higher speeds across all LOPs except *hand*. *Arm*, *bag*, *belt* and *front pocket* reached their low Error Ratio at a much lower speed than did *back pocket*. Surprisingly, *hand* did better at lower speeds. The impact of window size was statistically significant but of a small numerical value, with 8 s *window* outperforming 4 s *window*.

The pilot study confirmed general accuracy for our approach and suggested a better choice of factors for use in a more naturalistic outdoor-walking study. Specifically, because 4 s and 8 s window sizes both produced good performance with minimal difference, we concluded that the accuracy of window sizes larger than 4 s is not worth the extra latency and we decided to try smaller window sizes for comparison. We reduced the number of speed levels to five.

4.2. Primary outdoor walking task and measurement apparatus

The primary experiment to validate RRACE was run on a concrete sidewalk in an open area on a university campus, with no nearby buildings to block GPS signals.[2] Subjects were asked to walk twice at each of five different speeds, and instructed with the definitions provided in Table 1. We further instructed all subjects that 'leisurely' walking speed meant their slowest normal walking speed, and 'typical' walking speed is their usual walking speed. Allocation of walking speed order was randomized.

We note that, subjects' walking speed and cadence were not expected or required to be perfectly consistent (either within or between subjects) for measuring accuracy of cadence detection. Our goal was to observe walking at a larger variety of speeds for every individual, and this loosely controlled mechanism allows a more finely resolved spectrum of actual speeds; meanwhile, this allowed a large dataset, mitigating the effect of imbalances and imperfections.

Apparatus: the experimental setup consisted of six Google Nexus One smartphones, an external GPS receiver connected to one of the phones via bluetooth, our *reference* cadence measurement consisting of two shoe-mounted FSR (force sensing resistor) sensors [35] to detect footfalls and connected to a Bluetooth-enabled Arduino board [36], two laptops (one for

[2] While GPS data was collected for possible use in validation, RRACE does not use GPS data itself.

Table 1
Walking speeds during the experiment.

Label	Definition	Mean (m/s)	SD
Speed −2	Leisurely (slowest) walking speed	1.14	0.22
Speed −1	Slower than typical but faster than leisurely	1.34	0.14
Speed 0	Typical walking speed	1.52	0.08
Speed 1	Faster than typical but slower than the fastest speed	1.67	0.12
Speed 2	Fastest walking speed	1.95	0.12

Table 2
Experiment design.

Factor	Number of levels	Factor levels
Window size	4	1, 2, 4, or 8 s
LOP	6	*back pocket*, *bag* (*backpack*), *dominant hand* (*held*), *front pocket*, *hip* (*mounted on belt*), *upper arm* (*mounted*)
Condition	5	*typical* (0), *fastest* (2), *leisurely* (−2), *faster than typical* (1), *slower than typical* (−1)
Repetition	2	First time, second time

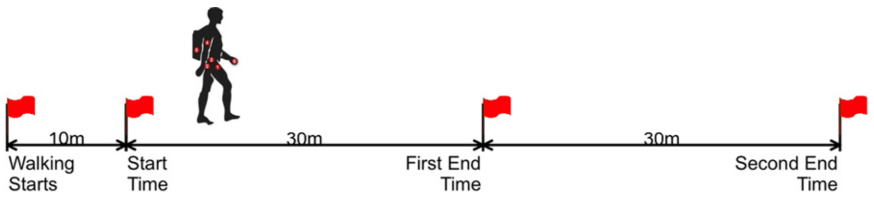

Fig. 2. Experiment walkway, start and end points.

logging trials and a second, a small netbook, to log footfalls sent from the Arduino board via Bluetooth), a backpack, a stop watch, and two flags for experimenters to send timing signals to each other. The study required three experimenters to run.

Prior to the experiment, subjects were asked to wear pants with front and back pockets but pocket locations were not controlled. The six phones and the Arduino board were synchronized with the main computer at the start of the experiment. One of the phones, the GPS receiver, netbook, and Arduino were put in the backpack (*bag*). The bag had a filled weight of approximately 2 kg. See Table 2 for general phone locations, which were chosen as the places people used most frequently for their mobile phones while commuting [37].

FSR footfall detection: we used timestamped force sensing resistor (FSR) data as our *reference* footfall detection method:

$$ER = \frac{|F_a - F_r|}{F_r} \tag{2}$$

where

$$F_r = \frac{1}{T_{\mathrm{FSR}}}.$$

FSRs are ideal for detecting changes in force or pressure [35]. We placed a force sensor inside each shoe (to measure the force exerted by subject's feet and compare it with a threshold), and connected both to the Arduino. The Arduino timestamped the FSR readings (avoiding impact of Bluetooth latency) and sent them on to the netbook via Bluetooth. The footfall detector system was calibrated and verified for each subject at the beginning of the experiment. To analyze the data, we used the median of the last three intervals of each of the two feet (T_{FSR} in Eq. (2)) to filter errors caused by false positives (extra footfall detected) or false negatives (footfall missed).

Trial length and speed measurement: we wished to collect 20 s of walking data for each trial (twice the length of our largest window size with a 25% safety margin) and compute step frequency every 200 ms. We asked subjects to walk a known distance, either 30 m or 60 m (marked by small flags along the walkway), depending on whether 20 s had elapsed by the time the 30 m point (first end time) had been reached (Fig. 2). Timespan was manually recorded via stopwatch.

4.3. Experiment design, metrics, and subjects

The design was within-subjects repeated-measures, with independent variables of *window size*, *LOP*, and *speed condition* (Table 2). The five speed conditions and their repetitions (10 trials) were randomized.

Metrics and analysis: we assessed RRACE's accuracy by comparing it to our shoe-located pressure sensor reference (Section 4.2). As in the pilot, our primary metric was Error Ratio (*ER*, Eq. (1) in Section 4.1). We conducted our analysis with Linear Mixed-Effects Models, using unpaired Z-test comparisons for post-hoc analysis and $p = 0.05$ significance, again

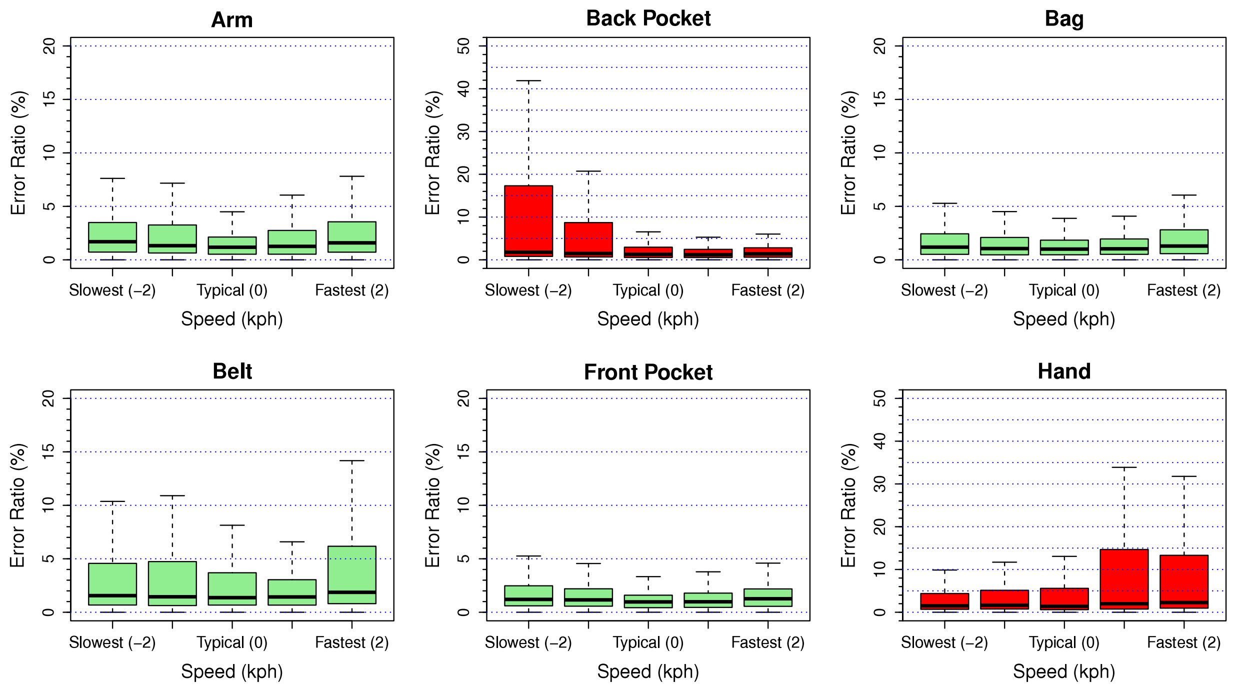

Fig. 3. Error Ratio (*ER*) as a function of speed condition for 4-s Window RRACE. Dark-red box plots have a larger range for Error Ratio. The box plot's central bar indicates sample median. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

applying a Bonferroni correction. Note that for sample sizes as large as our dataset, Z-test produces the same result as a t-test. Also, we report differences between the effect levels as z-scores, and because z-scores are normalized by standard deviation, differences between means in our analysis are analogous to Cohen's d statistics of effect size.

Subjects: eleven individuals (6 female and 5 male), aged 21–30 years (mean = 25.2, SD = 3.3), 155–179 cm tall (mean = 165.9, SD = 7.0), and weighing 46–80 kg (mean = 59.1, SD = 10.0) volunteered. No subjects had physical impairments.

Speed/Frequency relationship: as a basic check for our measurements we verified a correlation between walking speed and cadence ($r = 0.84$ using Pearson's Correlation), which is consistent with [38].

4.4. Results for outdoor validation of 4 s window RRACE

We chose the 4-s window RRACE as our analytical baseline, and describe its analysis first: both theoretically and in our pilot, 4 s is enough to detect a wide range of walking cadences. We then present the results from alternative window sizes. Because the phones were prone to dropping data (14%), we used a Linear Mixed Effect Models for its robustness to this situation.

Main effects: LOP has a significant effect on *ER*. Results were consistent with our pilot: *front pocket*, *belt*, *arm*, and *bag* (light-green box plots of Fig. 3) are much more reliable than *back pocket* and *hand* (dark-red box plots of Fig. 3). Speed condition also has a significant effect on *ER*; *ER* is generally lower at the typical and fast speed and higher at the slowest and the fastest speed.

Interaction effects: both LOP/speed condition and LOP/window size on *ER* interact significantly. *Arm*, *bag*, and *front pocket ER* remain consistently below 5% under all speed conditions, with their minimum at the middle (typical) speed. *Belt* produces lowest *ER* at the typical speed and largest *ER* at the fastest speed. *Back pocket* produces lower *ER*'s at higher speeds and *hand* produces lower *ER*'s at lower speeds (Fig. 3). As we will see in Section 4.5, the interaction between LOP and window size does not affect our general conclusions about LOPs.

4.4.1. Quantitative comparisons

Location-on-person (LOP): Table 3 compares *ER* as a function of LOP—four out of six locations have *ER* of 5% or below. *Front pocket* and *bag*, with the lowest error ratios, significantly outperform the other locations. For example, *arm* has an *ER* of 3.6% on average and is 0.3% different from *front pocket* while *bag* and *front pocket* are not significantly different from each other.

This accuracy is approximately the same as the best reported elsewhere and has proved acceptable for most applications [2,1]. As noted earlier, it is not currently possible to make a direct comparison (*i.e.*, based on running the algorithms on the same dataset, or confirmation that the datasets/experimental conditions are fully comparable) with other

Table 3
Error Ratio differences by LOP for *four-second window* RRACE through an unpaired *Z*-test. The second column contains the mean Error Ratio of each LOP; remaining cells contain the difference between two LOPs where the difference is significant. The differences are the maximum possible while maintaining statistical significance, and thus are less than the distance of *ER* means from each other. A large value means larger distance between *ER*s.

LOP	*ER* (%)	Difference from				
		Front pocket	Bag	Arm	Belt	Back pocket
Front pocket	2.8	–	–	–	–	–
Bag	3.1	Not sig	–	–	–	–
Arm	3.6	0.3	0.1	–	–	–
Belt	5.5	2.2	2.0	1.4	–	–
Back pocket	7.9	4.5	4.3	3.7	3.0	–
Hand	11.4	7.8	7.6	7.1	5.2	2.6

Table 4
RRACE Error Ratio differences by speed condition for 4 LOPs with a *four second window* (unpaired *Z*-test). *Hand* and *back pocket* – the inconsistent LOPs with more obvious reaction to speed – are excluded to focus on the effect of speed in the absence of the interaction effects and on the more similar LOPs. See Table 3 for more information.

Speed condition	*ER* (%)	Difference from			
		(0)	(1)	(−1)	(2)
Typical (0)	2.5	–	–	–	–
Fast (1)	2.8	0.04	–	–	–
Slow (−1)	3.4	0.6	0.3	–	–
Fastest (2)	4.0	1.1	0.8	0.2	–
Slowest (−2)	6.3	3.2	3.0	2.3	1.7

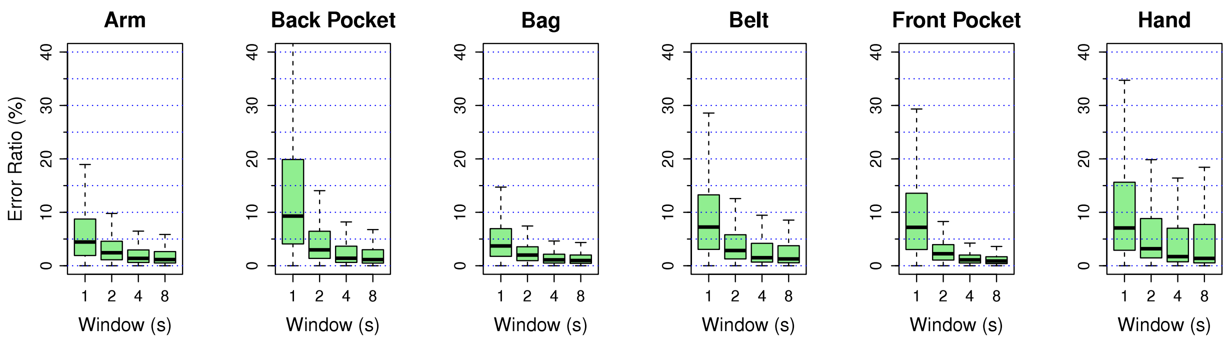

Fig. 4. Error Ratio is a function of window size per each LOP for all speed conditions lumped.

reported results given the level of implementation detail available. However, to the best of our knowledge the comparison is conservative: we asked RRACE to do the same or harder task, in that our setup was far less constrained.

Speed condition: Fig. 3 shows that Error Ratio decreases as speed increases only when the phone is placed in *back pocket* and the opposite happens when the phone is held in *hand*. However, the differences among different speed conditions are not very obvious for other LOPs in the figure. We can exclude those two LOPs and quantitatively compare speed conditions for the other LOPs; if we do so, we will see that Error Ratio is lower at the typical and fast (one level above typical) speed and generally highest at the slowest and/or fastest speed conditions (Table 4).

4.5. *Analysis of the effect of window size on RRACE*

Four-second and eight-second processing windows produced similar *ER*s, consistent with our pilot results (Figs. 4 and 5, and Table 5). While the *ER* of a one-second window is double that of four or eight seconds, the two-second window is only 1% (significant) different from four and eight-second windows, and may be usable in some circumstances (Table 5). As shown in Fig. 4, increasing window size reduces *ER* for all LOPs but has a smaller effect on locations with lower *ER* in general. By comparing Table 3 (*ER* of LOPs for four-second window RRACE) with Table 6 (*ER* of LOPs for all variations of RRACE) we see that the window size only affects the rank of *front pocket* among other LOPs; *front pocket* is not the best LOP when we choose smaller window sizes. Other five LOPs stay in the same relative order when we change the window size.

As anticipated, the effect of increasing window size on reducing *ER* is more noticeable at lower speeds (Fig. 5). Since smaller windows capture fewer steps than larger windows, with decreasing speed the chance of capturing enough steps is reduced. In effect, the increasing window size compensates for the effect of slowing down.

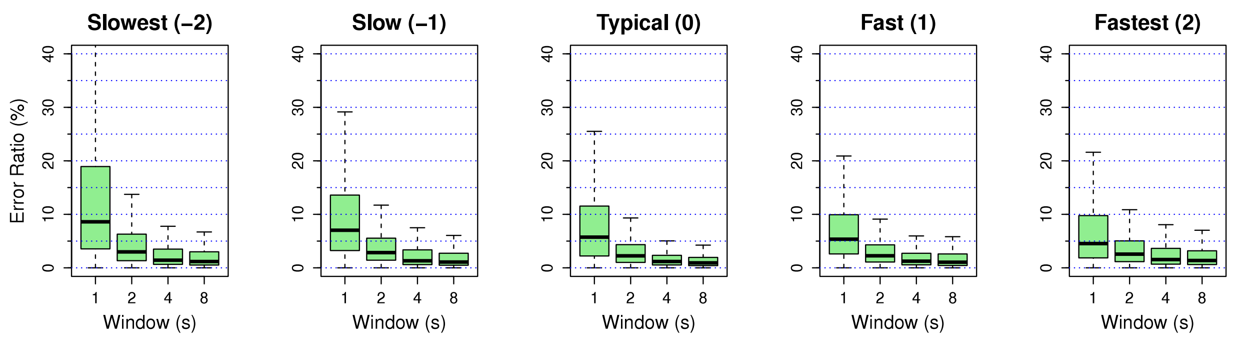

Fig. 5. Error Ratio is a function of window size per each speed condition for all LOPs lumped.

Table 5
Error Ratio differences by window sizes of RRACE, with walking speed and LOP lumped. Window sizes are ordered by increasing *ER* mean. See Table 3 for more information.

Window size	*ER* (%)	Difference from		
		8 s	4 s	2 s
8 s	5.8	–	–	–
4 s	5.8	Not sig	–	–
2 s	7.1	1.1	1.1	–
1 s	11.5	5.4	5.4	4.1

Table 6
RRACE Error Ratio differences by LOP for all window sizes and walking speeds (unpaired Z-test). Locations are ordered by increasing *ER* mean. See Table 3 for more information.

LOP	*ER* (%)	Difference from				
		Bag	Arm	Front pocket	Belt	Back pocket
Bag	4.1	–	–	–	–	–
Arm	4.8	0.5	–	–	–	–
Front pocket	5.5	1.2	0.4	–	–	–
Belt	7.1	2.8	2.0	1.3	–	–
Back pocket	10.5	6.1	5.3	4.6	3.0	–
Hand	12.8	8.3	7.6	6.9	5.3	1.9

Table 7
Power consumption.

App name	Duration (s)	Average usage (mW)	LCD usage	CPU usage
RRACE	361	772.85	528.53	244.32
Endomondo	308	555.52	529.87	25.65
Endomondo (no GPS)	369	548.78	530.08	18.70
Runtastic pedometer	322	575.47	521.74	53.73
Angry Birds	559	735.78	516.00	225.33

4.6. Power consumption

We used PowerTutor [39] to measure RRACE's power consumption on a Samsung Galaxy Nexus smartphone running the Android 4.1.1 Jelly Bean operating system, and compared it with Endomondo [13], a sport tracking app, which is claimed to be the highest rated app of its kind on Android, Runtastic Pedometer [14], a pedometer app that uses accelerometers to count steps, and Angry Birds, the famous game (Table 7).

Like most activity measurement algorithms, RRACE does not require the display to be on; but for consistency, all of these apps were compared with screen on. PowerTutor is able to distinguish between LCD power usage, which Table 7 shows is similar for all of them. CPU power usage varies: RRACE uses 10× the CPU power of Endomondo, 5× more than Runtastic Pedometer, and is comparable to Angry Birds.

With the screen off, our algorithm will consume considerably less power than mobile games even before improving the CPU efficiency. Until now, our development has focused on proving accuracy rather than power efficiency, so the low power consumption of other activity measurement apps is promising in terms of what RRACE can achieved with optimization, *e.g.*, with methods such as [40,41].

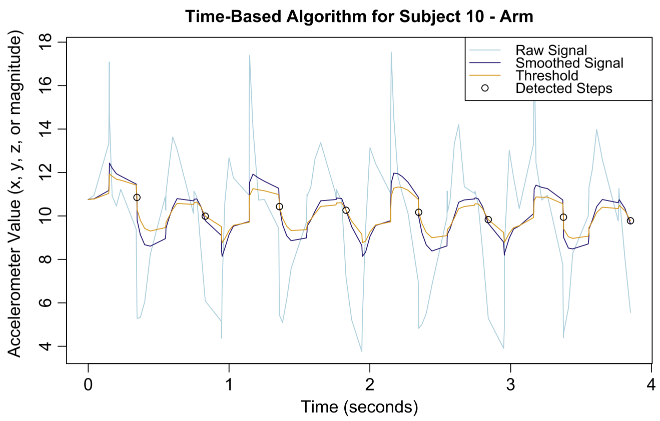

Fig. 6. Example of the MPTrain time-based step detection algorithm.

5. Comparing RRACE with a threshold-based time-domain algorithm

As detailed in Section 2, the many pedometers available commercially use proprietary algorithms that have not been released to the public. We therefore compared our frequency-based algorithm to MPTrain's algorithm [8]. MPTrain uses two low-pass filters. One removes noise in the original accelerometer signal, producing a smoothed signal; the second has a lower cutoff frequency, and its output is used as a dynamic threshold. Footsteps are detected when the smoothed signal crosses the dynamic threshold from above to below (Fig. 6). Because the MPTrain accelerometer is required to be situated on the user's torso and oriented to detect accelerations in the superior–inferior axis, it detects footsteps on both feet. Foot falls are translated to instantaneous (*i.e.*, sampled) steps-per-minute (SPM) using the following formula:

$$SPM_i = (int)\frac{60.0 \times SamplingRate}{\#SamplesSinceLastStep}. \tag{3}$$

Finally, the MPTrain algorithm applies a median filter to the instantaneous SPM to calculate estimated SPM. The MPTrain study reported a uniform sampling rate of 75 Hz for accelerometer data, achieved with an external chest-mounted sampler. The authors report that cadence measurement accuracy is comparable to those found in commercial pedometers by [1], but provide no specifics.

5.1. Implementation of time-based algorithm for comparison

We reconstructed parameterizations for the MPTrain algorithm, since details were not reported for either of the low-pass filters, and no window was given for the median filter. We also accommodated the variable sampling rate found in smartphones, and measured cadence in steps-per-second (SPS) instead of steps-per-minute (SPM) to compare it with RRACE.

Finally, given that we do not have a sensor in a known orientation, we also consider each of four different axes in our analysis: x, y, z, and m (the magnitude of the vector, *i.e.*, $m = \sqrt{x^2 + y^2 + z^2}$).

Two low-pass filters (accelerometer data smoothing and dynamic threshold) employ parameters α and β ($\beta < \alpha$), which were trained on our data (Section 5.2). For efficiency and simplicity, we implemented these as exponentially-weighted moving averages (EWMA). An EWMA is defined as follows:

$$S_i = \alpha x_i + (1 - \alpha)S_{i-1} \tag{4}$$

where S_i is the i-th smoothed (low-passed) value, x_i is the i-th raw accelerometer value, and α is the smoothing parameter ($0 \leq \alpha < 1$).

As for MPTrain, steps are detected when the smoothed signal crosses the dynamic threshold from above to below. The difference between step times is used to calculate the instantaneous cadence by the formula:

$$Cadence = 1/CurrentDifferenceBetweenSteps. \tag{5}$$

For example, if the two previous footsteps were detected at $StepTime_i = 100$ ms and $StepTime_{i+1} = 600$ ms and we wanted the instantaneous cadence at any time $t \geq 600$ ms, we would compute $1/(600-100) = 0.002$ steps per millisecond, or 2.0 SPS. Final cadence estimates were the average of each instantaneous cadence estimate and one previous estimate (*i.e.*, a 2-sample smoothing filter).

5.2. Tuning of the time-based algorithm

To compare the MPTrain time-based algorithm as favorably as possible to RRACE, we optimized the low-pass filter smoothing parameters α and β (Section 5.1) for several data subsets involving different combinations of subjects and locations-on-person (LOP):

- all data (all subjects and LOPs): 1 set
- each subject (over all LOPs): 11 sets
- each LOP (over all subjects): 6 sets
- each subject-LOP combination (*e.g.*, Subject 1, Arm) minus 9 (missing data): $11 \times 6 - 9 = 57$ sets.

This thorough search thus used 75 parameterizations of the time-based algorithm. During analysis (Section 5.2.1), data was only scored on the dataset on which it was trained. This represented a best-case scenario of an algorithm trained for a certain individual and/or LOP, which could occur in real-world use cases with one individual using a personal device in a consistent way.

Within a dataset, we used a uniform search for the best combination of smoothing parameters (α and β) with a granularity of 0.05 (*i.e.*, $\alpha \in \{0.05, 0.1, \ldots, 1.0\}$ and $\beta \in \{0, 0.05, \ldots, 1.0\}$, one of the three axes or magnitude ($\gamma \in \{x, y, z, m\}$), as well as four scaling factors ($\delta_{x,y,z} \in \{\frac{1}{2}, 1, 2, 4\}$ for individual axes and $\delta_m \in \{\frac{1}{4}, \frac{1}{2}, 1, 2\}$ for magnitude; scaling factors accommodate harmonics by scaling the computed cadence). The best of all these combinations for each dataset was determined by having the lowest mean squared ER by comparing to the FSR golden standard. The scaling factors scale the calculated cadence; for example if the user walks at 1.5 Hz and the time-based algorithm calculates a cadence of 0.75 Hz (*i.e.*, detects either left or right steps), a scaling factor of 2 would fix this ($0.75 * 2 = 1.5$).

5.2.1. Analysis of time-based algorithm

We found it was not possible to train the time-based algorithm to work on all LOPs and for all subjects with an Error Ratio below 5%; the minimum attained was $ER = 74\%$. The time-based algorithm for all LOP on one subject reached $ER = 18\%$, but this was only for the best case scenario.

Tuning the time-based algorithm for one best-case LOP on all subjects was more feasible: this achieved $ER = 12\%$ for *bag*. If we tune the algorithm for one LOP of each subject we may even get a lower *ER*; when tuned for Subject10's *arm*, the algorithm reached $ER = 7.8\%$ (Table 8). In the next section, we will show that these results are not nearly as good as the performance of RRACE with $ER = 5.8$ for the 8 s window variant.

5.2.2. Comparisons with frequency-based algorithm

Fig. 7 shows box plots of Error Ratio of all the RRACE and time-based algorithm variants ordered by the median of Error Ratio (ER). We divided them into five categories as identified in the figure's caption, differing by algorithm and breadth of training set, where a more specific (but unrealistic) training set generally leads to better performance in this test.

Because it is unproductive to compare each of these algorithms with the rest, we have chosen the best of each category in addition to the worst-case RRACE variant (one-second window) which are marked by blue ticks and blue dashed-line box plots on Fig. 7. This is a highly conservative comparison which tends to favor the time-based algorithm. We used the same data for verification of each time-based algorithm that was used for their training and secondly, the Error Ratio of all versions of the frequency-based algorithm is measured across all LOPs of all subjects. RRACE was not trained or tuned in this comparison.

Thus, the single "fair" comparison is between either version of RRACE (green in Fig. 7), and the time-based algorithm trained on *all* subjects and *all* LOPs (red). Table 8 summarizes these comparisons, but in order of mean rather than median; thus subject10's *arm* comes after the 1-s RRACE in the figure but before it in this table.

The best of time-based categories—Subject10's *arm*, *bag* (across all subjects), Subject10 (all body locations), and all subjects' body locations—and the 8-s and 1-s variants of RRACE appear on the first column of Table 8. Their respective Error Ratios are listed in the second column. It is statistically incorrect to compare these values without testing the statistical significance of their difference. Therefore, we used unpaired Z-tests (with Bonferroni correction for multiple comparisons) to (a) test the statistical significance of the difference between each two algorithms (one from the first column vs. another one from the third to seventh column of the second row), and (b) measure the maximum difference while maintaining statistical significance which does not apply to pairs that are not significantly different such as *bag* vs. 1-s RRACE; this also applies to Tables 3–6.

In particular, the difference between the best variant of RRACE and the bests of all categories of time-based algorithm, 1.2, 5.5, 10.2, and 67.5 presented on rows 4, 6, 7, and 8 (row of Subject10's *arm*, row of *bag*, row of Subject10, and row of all subjects' body locations) and 3rd column (column of 8-s RRACE) are important to us; these values show that RRACE has a much lower Error Ratio than any of the time-based algorithms and this difference is statistically significant.

6. Discussion

The goal of this research was to develop a cadence measurement algorithm for accelerometer-equipped mobile phones equipped with accelerometers. We required this algorithm to be robust and work out-of-the-box with an Error Ratio of 5% or less (comparable to Yang et al.'s *waist-mounted* cadence measurement device [27] and MPTrain of Oliver and Flores-Mangas [8]). First, we will review the nature of RRACE's error, its performance on different LOPs and robustness to subject differences, and compare it with the time-based algorithm. Then we will examine its main weakness, and finally we will discuss the best choice for window size.

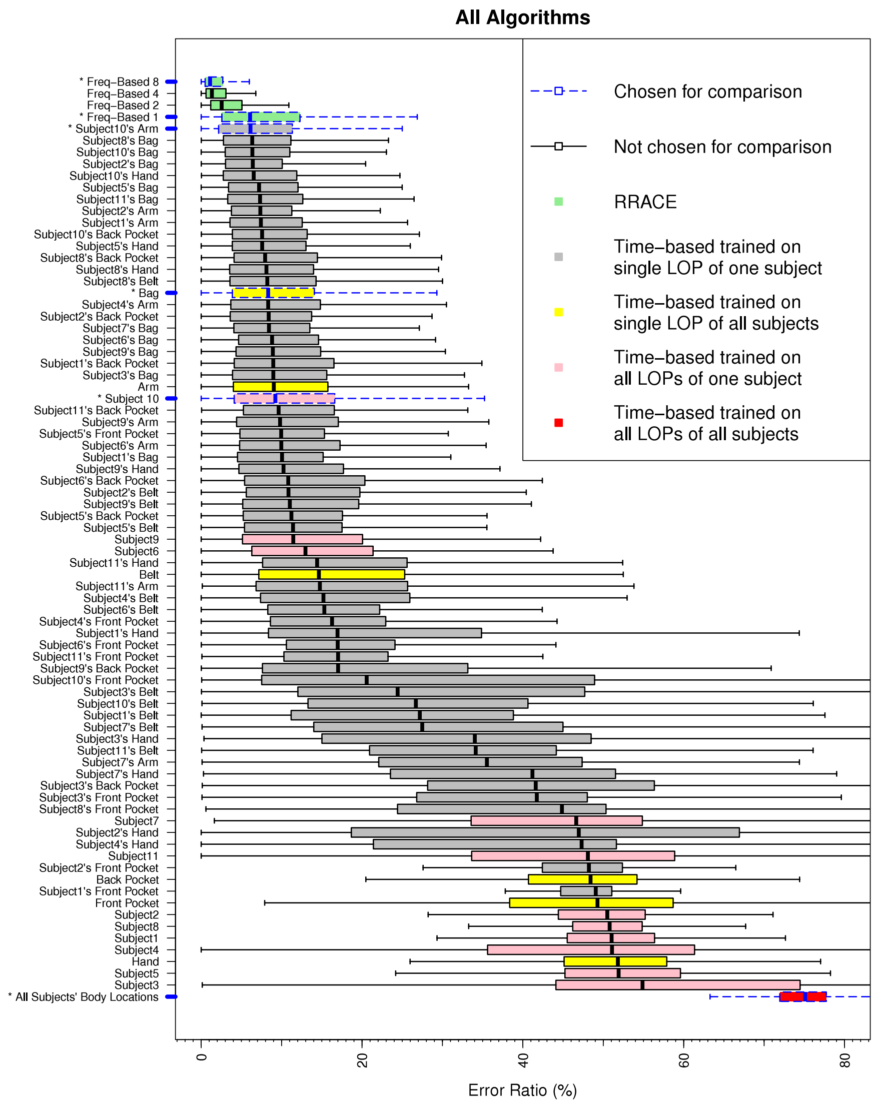

Fig. 7. Error Ratio compared for all algorithm variants and ordered by median. (a: GREEN) 4 window sizes of RRACE (first four); time-based algorithm trained on: (b: RED) all subjects' LOPs (last one), (c: PINK) all LOPs of each single subject, (d: YELLOW) one LOP of all subjects, and (e: GREY) single LOP of one subject. Algorithms chosen for quantitative comparison are marked by blue ticks and blue dashed-line box plots.

Table 8
Unpaired Z-test comparison of error ratios of the best and the worst versions of the frequency-based algorithm and the best of each category of time-based algorithm. Algorithm variants are ordered by increasing *ER* mean. See Table 3 for more information.

Algorithm	*ER* (%)	Difference with				
		8-s RRACE	Subject 10's Arm	1-s RRACE	Bag	Subject10
8-s Window RRACE (a)	5.8	–	–	–	–	–
Subject10's Arm (e)	7.8	1.2	–	–	–	–
1-s Window RRACE (a)	11.5	5.4	2.9	–	–	–
Bag (d)	11.9	5.5	3.2	Not sig	–	–
Subject10 (c)	17.9	10.2	8.1	4.6	4.0	–
All subjects' body locations (b)	73.5	67.5	65.0	1.8	60.9	53.7

6.1. *The nature of RRACE's error*

A small number of outliers are responsible for some of the error in RRACE's readings. These are of two types: (a) random readings as a result of irregularities in the signal, and (b) harmonic readings which happen when the main frequency component gets smaller than its harmonics. These outliers may be avoided by filtering the outcome of RRACE. The rest of the error is caused by hardware measurement error and delay from the 4 s window.

6.2. *RRACE meets criteria for* 4/6 *of tested locations; time-based for* 0/6

Movement at four LOPs (*arm*, *bag*, *belt*, and *front pocket*) contain sufficient consistent information for RRACE to make accurate estimates and RRACE does not need to be calibrated in order to work there. They each achieve a 3%–5% *ER*, satisfying the criteria laid out above.

In contrast, the time-based algorithm was highly sensitive to LOP. It was almost impossible to tune the time-based algorithm for three of the LOPs, *front pocket* among them. The LOP that fit the time-based algorithm the best was *bag* with almost double the *ER* of the 8-s and 4-s window RRACE.

6.3. *RRACE is robust to subject differences*

The time-based algorithm was very sensitive to subject differences. It could not be trained to work on all LOPs of all subjects, and when trained on single LOPs, only 12% was achieved, in only one location (*bag*). Because it was calibrated by subject, its present tuning would not work on subjects outside our experiment with no adjustment. However, RRACE achieved much lower *ER* for all subjects with no prior tuning to compensate for subject differences.

6.4. *RRACE is sensitive to very slow speeds*

Our outdoor validation results showed that, like other pedometers, RRACE is sensitive to speed. The highest Error Ratio belongs to the slowest speed with $ER = 6.3\%$. We attribute this worsened performance to two possible causes:

(a) at lower speeds, walking cycles take longer and fewer cycles are captured in a fixed window size. As anticipated, this weakens RRACE. Mitigation requires use of a larger window size, *e.g.*, by dynamically changing the window size to fit the speed.
(b) walking becomes less autonomous and more irregular when subjects are asked to walk at very low speeds, especially because users can easily choose to walk as slowly and irregular as they want, while at high speeds step interval is bounded by subject's physique.

The time-based algorithm is less affected by walking speed because it just detects single steps, no matter how irregular or distant from each other they are. Thus one practical approach might be to shift to a time-based algorithm at low speeds.

6.5. *RRACE window length of 4 s is best*

Our results showed that the highest accuracy (lower *ER*) is achieved at larger window sizes. The difference in Error Ratio is substantial for 1 vs. 2-s windows, and for 2 vs. 4-s windows, but not for 4 vs. 8 s. A 4-s window size seems the ideal length among our candidates as a compromise between responsiveness and accuracy.

7. Conclusion and future work

In this paper, we introduced a new algorithm for measuring cadence through a frequency-domain analysis of accelerometer data from smart phones, called RRACE. This algorithm's advantages are strong robustness to location on body, orientation, and to individual physiological parameters, resulting in exceptional usability and suitability for a broad range of consumer-type applications.

We also presented an experiment design to verify our and other algorithms. Our user-based validation showed that RRACE performs well under different speed conditions, providing 5% or lower error for four of the six common LOPs examined: *front pocket*, *bag*, *arm* and *belt*, consistent with the previous work in a single location [27], and producing 8% and 11% for the other two: *back pocket* and *hand*. RRACE's primary weakness is a drop in performance for slow and irregular walking, a flaw which can be mitigated by dynamically adjusting the window size to maximize accuracy at the cost of more latency, and/or switching to a time-based algorithm at slow speeds.

We compared RRACE with a state-of-the-art published time-based algorithm which we tuned in every way possible; our highly conservative comparisons show that RRACE is substantially more accurate than the time-based algorithm tuned for any subset of the data. Our results show that RRACE is also superior to the time-based algorithm in terms of independence from LOP and robustness to user differences. The exception is for very low and/or irregular speeds, situations which many applications of a cadence detection method might classify as a different gait and analyze using a different algorithm. We also plan to extend the comparisons to include autocorrelation of time-based techniques, which may share some of the advantages of a frequency based approach.

As well, our algorithm provides general guidelines for window size and robust spectral analysis. This information can be used to inform solutions to more complex realtime gait analysis problems, such as activity detection for fitness or rehabilitation applications, or individual gait identification for mobile security.

We are continuing to improve our algorithm. Some avenues likely to further increase its performance are to reduce the estimation outliers by using smart filters and adjust window size based on current cadence. We will also look into reducing the power consumption of our algorithm by reducing sampling and CPU usage when subject is in low activity mode. Finally, we look forward to deploying RRACE in the real world: we are engaged in employing cadence to measure other useful information about gait such as stride length and type of gait, and exploring deployment in a variety of real applications [29].

Acknowledgments

This work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the GRAND NCE. User data was collected under the University of British Columbia's Research Ethics Board approval H01-80470.

References

[1] E.L. Melanson, J.R. Knoll, M.L. Bell, W.T. Donahoo, J.O. Hill, L.J. Nysse, L. Lanningham-Foster, J.C. Peters, J.a. Levine, Commercially available pedometers: considerations for accurate step counting, Preventive Medicine 39 (2004) 361–368.
[2] R.C. Foster, L.M. Lanningham-Foster, C. Manohar, S.K. McCrady, L.J. Nysse, K.R. Kaufman, D.J. Padgett, J.a. Levine, Precision and accuracy of an ankle-worn accelerometer-based pedometer in step counting and energy expenditure, Preventive Medicine 41 (2005) 778–783.
[3] Y. Fujiki, K. Kazakos, C. Puri, J. Levine, S.M. Hospital, M. Clinic, I. Pavlidis, J. Starren, NEAT-o-Games: ubiquitous activity-based gaming, Methodology (2007) 2369–2374.
[4] J. Lester, B. Hannaford, G. Borriello, Are you with me? Using accelerometers to determine if two devices are carried by the same person, IEEE Pervasive Computing (2004) 33–50.
[5] X. Zhao, S. Saeedi, N. El-Sheimy, Z. Syed, C. Goodall, Towards arbitrary placement of multi-sensors assisted mobile navigation system, in: Proceedings of the 23rd International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2010, pp. 556–564.
[6] M.L. Wolf, Thomas Jefferson Abraham Lincoln Louis Brandeis and the Mystery of the Universe, Journal of Science & Technology Law-Boston University 1 (1995) 10.
[7] Y. Fujiki, iPhone as a physical activity measurement platform, in: Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems—CHI EA'10, ACM Press, New York, USA, 2010, p. 4315.
[8] N. Oliver, F. Flores-Mangas, MPTrain: a mobile, music and physiology-based personal trainer, in: Proceedings of the 8th Conference on Human–Computer Interaction with Mobile Devices and Services, ACM, 2006, pp. 21–28.
[9] C. Tudor-Locke, Taking steps toward increased physical activity: using pedometers to measure and motivate, President's Council on Physical Fitness and Sports Research Digest 3 (2002) 10.
[10] J.J. Kavanagh, H.B. Menz, Accelerometry: a technique for quantifying movement patterns during walking, Gait & Posture 28 (2008) 1–15.
[11] S. Consolvo, P. Klasnja, D.W. McDonald, D. Avrahami, J. Froehlich, L. LeGrand, R. Libby, K. Mosher, J.A. Landay, Flowers or a robot army? Encouraging awareness & activity with personal, mobile displays, in: Proceedings of the 10th International Conference on Ubiquitous Computing—UbiComp'08, ACM Press, New York, USA, 2008, pp. 54–63.
[12] Nike+iPod, 2012. http://www.apple.com/ipod/nike.
[13] Endomondo, 2013. http://www.endomondo.com.
[14] RuntasticPedometer, 2013. http://www.runtastic.com/en/apps/pedometer.
[15] R. de Oliveira, N. Oliver, TripleBeat, in: Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services—MobileHCI'08, ACM Press, New York, USA, 2008, pp. 255–264.
[16] Garmin, 2012. http://sites.garmin.com/forerunner910xt.
[17] J. Lin, L. Mamykina, S. Lindtner, G. Delajoux, H. Strub, P. Dourish, A. Friday, Fish'n'Steps: encouraging physical activity with an interactive computer game, in: P. Dourish, A. Friday (Eds.), 8th International Conference on Ubiquitous Computing (Ubicomp 2006), in: Lecture Notes in Computer Science, vol. 4206, Springer, Berlin, Heidelberg, 2006, pp. 261–278.
[18] Y. Fujiki, K. Kazakos, C. Puri, P. Buddharaju, I. Pavlidis, J. Levine, NEAT-o-Games: blending physical activity and fun in the daily routine, ACM Computers in Entertainment 6 (2008) Article 21.
[19] Nike+Fuelband, 2012. http://nikeplus.nike.com/plus/products/fuelband.
[20] FitBit, 2012. http://fitbit.com.
[21] C. Matthews, Y. Ketema, D. Gebre-Egziabher, M. Schwartz, In-situ step size estimation using a kinetic model of human gait, in: Proceedings of the 23rd International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2010, pp. 511–524.
[22] N. Kern, B. Schiele, Context-aware notification for wearable computing, in: 2003. Proceedings Seventh IEEE International Symposium on Wearable Computers, 2003, pp. 223–230.
[23] J. Ho, Using context-aware computing to reduce the perceived burden of interruptions from mobile devices, in: The SIGCHI Conference on Human Factors in Computing, pp. 909–918.
[24] Y. Kawahara, H. Kurasawa, Recognizing user context using mobile handsets with acceleration sensors, in: IEEE International Conference on Portable Information Devices, Figure 1, pp. 1–5.
[25] K. Altun, B. Barshan, Pedestrian dead reckoning employing simultaneous activity recognition cues, Measurement Science and Technology 23 (2012) 025103.
[26] A.M. Monitors, 2012. http://apdm.com/products/movement-monitors/.
[27] C.C. Yang, Y.L. Hsu, K.S. Shih, J.M. Lu, Real-time gait cycle parameter recognition using a wearable accelerometry system, Sensors (Basel, Switzerland) 11 (2011) 7314–7326.
[28] W. Zijlstra, A.L. Hof, Assessment of spatio-temporal gait parameters from trunk accelerations during human walking, Gait & Posture 18 (2003) 1–10.
[29] O.S. Schneider, K.E. MacLean, K. Altun, I. Karuei, M.M. Wu, Real-time gait classification for persuasive smartphone apps: structuring the literature and pushing the limits, in: Proceedings of IUI'13, the 2013 International Conference on Intelligent User Interfaces, Santa Monica, CA, USA, pp. 161–172.
[30] J.D. Scargle, Studies in astronomical time series analysis. II—statistical aspects of spectral analysis of unevenly spaced data, The Astrophysical Journal 263 (1982) 835–853.
[31] S. Das, M. Murphy, F.W. Olin, S.A. Perrig, Detecting user activities using the accelerometer on android smartphones, Technical Report, 2010.
[32] N.R. Lomb, Least-squares frequency analysis of unequally spaced data, Astrophysics and Space Science 39 (1976) 447–462.
[33] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, second ed., Cambridge University Press, 1992.
[34] R. Knoblauch, M. Pietrucha, M. Nitzburg, Field studies of pedestrian walking speed and start-up time, Transportation Research Record 1538 (1996) 27–38.
[35] InterlinkElectronics, 2011. http://www.interlinkelectronics.com/.
[36] Arduino, 2012. http://www.arduino.cc/.
[37] Y. Cui, J. Chipchase, F. Ichikawa, A cross culture study on phone carrying and physical personalization, in: Proceedings of the 2nd International Conference on Usability and Internationalization, Springer-Verlag, 2007, pp. 483–492.
[38] S. Hirokawa, Normal gait characteristics under temporal and distance constraints, Journal of Biomedical Engineering 11 (1989) 449–456.
[39] M. Gordon, L. Zhang, B. Tiwana, R. Dick, PowerTutor, http://powertutor.org/documentation.html, 2011.
[40] E. Cuervo, A. Balasubramanian, D.k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 49–62.
[41] L.S. Brakmo, D.A. Wallach, M.A. Viredaz, μ Sleep: a technique for reducing energy consumption in handheld devices, in: Proc. Int. Conf. Mobile Systems, Applications, and Services, pp. 12–22.