2017 IEEE 37th International Conference on Distributed Computing Systems

PTrack: Enhancing the Applicability of Pedestrian Tracking with Wearables

Yonghang Jiang, Zhenjiang Li, Jianping Wang
Department of Computer Science, City University of Hong Kong
yhjiang4-c@my.cityu.edu.hk, zhenjiang.li@cityu.edu.hk, jianwang@cityu.edu.hk

***Abstract*—The ability to accurately track pedestrians is valuable for variant application designs. Although pedestrian tracking has been investigated excessively and owned a well-suited sensing platform, the proposed solutions are far from being mature yet. Pedestrian tracking contains step counting and stride estimation two components. Step counting already has commercial products, but the performance is still unreliable and less trustworthy in practice. Stride estimation even stays in the research stage without ready solutions released on the market. Such a non-negligible gap between long-term research investigation and technique's actual usage exists due to a series of crucial applicability issues unsolved, including design vulnerability to interfering activities, extracting purely body's movement from additive sensor signals, and parameter training without user's intervention. In this paper, we deeply analyze human's gait cycles and obtain inspiring observations to address these issues. We incorporate our techniques into existing pedestrian tracking designs and implement a prototype, PTrack, using LG smartwatch. We find PTrack effectively enhances the system applicability and achieves promising performance under very practical settings.**

I. Introduction

Motivation. Pedestrian tracking applies two primary operations, namely *step counting* and *stride estimation* [1], to calculate the moving distance of a user. The ability of accurately tracking pedestrians is valuable for plenty of applications, *e.g.,* healthcare to achieve a quantitative awareness of daily fitness statuses to reduce occupational disease risks [1], new customer assessments beyond traditional medical records by insurance companies [2], location-based service designs [3]–[5] using dead-reckoning to improve the energy efficiency by accessing energy-consuming sensors less, *e.g.,* GPS and WiFi, and also boost system robustness when localization becomes unreliable.

To leverage above benefits, pedestrian tracking is originally developed on smartphones, having extensive research efforts made so far [1], [6]–[8]. Nevertheless, they suffer a primary limitation for long that smartphones can be easily out of the user's body. Recently, wearable devices, *e.g.,* smartwatch, wrist-bands, etc., bring an emerging opportunity, as they are attached to user's body more tightly and durably every day, and are also equipped with motion sensors [9]. Although wearables already help overcome the platform limitation, excessive user's feedback unveils that step counting products on the market are highly inaccurate [10]–[13]. Their results may not truthfully reflect users' activity levels, leading to inappropriate fitness indications. For stride estimation, to our best knowledge, it primarily stays in the research stage yet. Given these facts, it is natural to wonder what fundamental issues are still unsolved and challenge the pedestrian tracking's applicability in practice?

Challenge. To answer this question, we first understand how existing methods achieve pedestrian tracking. The device is attached to user's body, so that accelerometers could generate a sequence of distinguishable signals, *e.g.,* peaks alternating with valleys, as the user moves [7]. Steps then can be counted by peak detection [1], [3], [4] or its variants [7]. To further estimate stride length, the body's vertical displacement within each step is first derived from accelerometer readings. This per-step displacement is then passed to biomechanical models to estimate the length for each stride [14].

Pedestrian tracking oughts to be mature with above excessively investigated theories and the well-suited wearable sensing platform. However, we find existing efforts overlook a set of crucial unsolved issues, which still impair and dominate the usability of pedestrian tracking in practice.

1) User's arms are involved in plenty of daily activities, *e.g.,* eating, playing games, taking photos, swinging, watching time, etc., which can also generate similar alternating motion sensor signals. If such external interference is not carefully excluded in existing solutions, it is inevitable to frequently mis-trigger step counting and overestimate the result (this issue exists on smartphones as well when they are held by the user). More seriously, due to the same reason, step counting can also be easily compromised or cheated by spoofing devices [15], making its results highly untrustworthy.

2) Even above interfering activities can be excluded so that precise step counting becomes feasible, wearable sensors still measure the additive effect from both the arm and body as the user walks, but stride is purely determined by the body's movement [7], [14]. It is thus non-trivial to solely extract the desired body's motion information from the aggregated signal to accurately estimate the stride length. In addition, the stride estimation usually requires user-specific parameters as input, *e.g.,* user's arm and leg lengths. Measurement errors made by inexperienced users could further lead to continuous performance deterioration.

Contribution. To make pedestrian tracking practically usable, we make the following contributions atop existing pedestrian tracking designs to dramatically enhance its applicability.

1) To reliably count steps, our key observation is the composition difference of various activities: interfering activities are mainly generated by *rigid* objects, *e.g.,* user's arm or

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.111

hand, while walking is an *aggregated* activity from both the arm and body, whose movements are concurrent but relatively independent. To disclose such a difference, we find that if we project accelerometer readings to two perpendicular directions, the signals along each direction reach a series of critical points, *e.g.,* turning or crossing points, nearly at the same time for interfering activities due to the rigid property. In contrast, such a critical point matching becomes highly asynchronous during walking. By leveraging this unique feature, we can thus exclude interfering activities to achieve accurate step counting.

As we grab the essential difference between the interfering and pedestrian activities, our solution hence could be *training-free* and more *generic*, *e.g.,* the design requires no labeled data set for training, and the performance thus will be not limited to the predefined interfering activities set.

2) We deeply analyze people's gait cycles for stride estimation and decompose each cycle into subtle phases. By utilizing the geometrical relationship of the user's arm cross different phases, we propose a new solution to derive the sole movement of the body through wearables that is required by the stride estimation model. The stride length for each step then can be estimated. Based on the geometrical relationship, we further propose a self-training mechanism to automatically explore required user profiles used in the estimation, *e.g.,* arm and leg lengths, without user's explicit measure and input.

3) As this paper is positioned to enhance pedestrian tracking's applicability, instead of reinventing the entire system stack, we thus aim to provide clear interfaces for our techniques to the pedestrian tracking modules from existing solutions. We instantiate a concrete design, namely PTrack, by integrating with the state-of-the-art approach [6] and implement a prototype using LG smartwatch Urbane [16]. Comprehensive evaluations lasting more than one month show that steps can be accurately counted by PTrack, achieving an error rate as low as 0.02 with extensive interfering activities, and the average per-step stride estimation error is also small, *e.g.,* $5.3cm$ merely. To further understand PTrack's benefit to upper-layer applications, we evaluate it in an indoor navigation case study with dead-reckoning. Along a $141.5m$ navigation route, the derived walking distance from PTrack is $136.4m$, very close to each other.

Roadmap. We conduct literature review and detail our design motivation in §II. We introduce PTrack design in §III and evaluate its performance in §IV before concluding in §V.

II. Problem Statement and Motivation

Literature review. When a device is attached to user's body, on-board motion sensors could capture acceleration changes, which repeat and alternate from peaks to valleys [17]. Existing designs [1], [3], [4], [6], [8] thus apply peak detection or its variants to count steps. However, due to variant interfering activities, these designs therefore can be highly inaccurate in the daily use [10]–[12], [12]. More seriously, the vulnerability to external interferences could also trivially spoof a step counter [15], making its result highly untrustworthy. Machine learning based designs, like [18], could adopt labeled data for training to exclude certain predefined activities. However, their performance is mainly limited to the training data set.

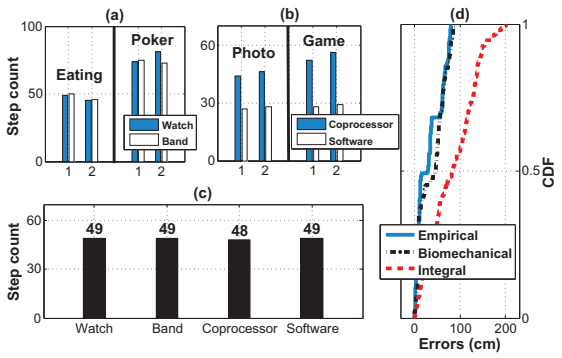

Fig. 1. Miscounted steps on (**a**) wearables and (**b**) mobiles interfered by different activities, where "1" and "2" represent the user is standing and seated. (**c**) Counting can be spoofed intentionally for all devices. (**d**) Stride estimation errors achieved using existing models and the integral on wearables directly.

For stride estimation, the key is to obtain the parameters that describe body's movement within each step [14], *e.g.,* vertical displacement, min/max accelerations, etc. Stride estimation models [6], [19]–[22] are then applied to estimate strides for each step (they usually do not require gait types to be consistent). However, the major hurdle to apply these methods is the requirement of a precise measurement of sole body's movement, since wearable sensor readings are normally aggregated from both user's arm and body motions. To the best of our knowledge, how to preserve purely the body's movement through aggregated signals is still lacking.

Overestimated step counts. Daily interfering activities could frequently mis-trigger the step counter, leading to overestimated results. We perform two typical activities to understand this issue: 1) eating with knife and fork and 2) playing porker cards. Each activity is examined in two rounds when the user stands and is seated, respectively. Fig. 1(a) depicts the results using a LG smartwatch [16] and Mi Band [23]. Both devices have the built-in step counter. Ideally, the counter should keep silent, but it is mis-triggered from 40 to 80 times in 2 min. As such interfering activities can frequently occur in practice, Fig. 1(a) clearly indicates that existing step counter on wearables fails to truthfully reflect user's fitness level.

Through the experiment, we further observe that even on the original platform of existing approaches, *i.e.,* smartphones, the similar problem still exists. To unveil this issue, we install two top-ranked pedometer APPs (from AppStore) on IPhone 6, with and without using Apple's built-in coprocessor. Fig. 1(b) demonstrates that the step counter can be mis-triggered from 27 to 56 times in 2 minutes, when two common activities are performed, *e.g.,* 1) taking photos and 2) playing phone games. Because the vulnerability to the external interference, Fig. 1(c) finally shows that these existing designs on both wearable and mobile devices can be easily spoofed by a device[1] like Fig. 7,

[1]Spoofing devices can generate an alternating motion pattern repeatedly to cheat existing step counter designs for easily accumulating counted steps.

where the counters tick 48 times in only $40s$.

The vulnerability shown in Fig. 1(a) to (c) is the key hurdle prohibits existing step counting approaches directly adopted in insurance- and finance-related businesses for broader social usage, *e.g.,* customer assessment based on their fitness records.

Stride estimation with mixed signals. Wearable sensors reflect the aggregated activities from both the arm and the body, but stride is purely determined by the body's movement. Since, to the best of our knowledge, a dedicated stride estimation design for wearable devices is still lacking, we thus apply existing stride estimation solutions on wearables directly to understand the achieved performance. According to [14], we implement both biomechanical and empirical models on the smartwatch and summarize the result in Fig. 1(d), where we indeed obtain highly inaccurate performance.

One natural question to stride estimation is: why not double-integrate horizontal accelerations to calculate the stride length? Within each step, the horizontal velocity of the body consists of two parts: a relatively stable baseline part v_0 and a time-varying part v_t. The stride length can be estimated by $v_0 \times t$, where t is duration of one step, but the integral of body's horizontal accelerations leads to v_t. For wearables, the horizontal accelerations are further mixed with arm's movements. Hence, the double-integral on horizontal accelerations is infeasible to directly calculate the stride length, as Fig. 1(d) shows.

Summary. Fig. 1 unveils the unsatisfied performance achieved by existing pedestrian tracking designs in practice. An effective countermeasure to enhance their usability is thus urgently desired. We elaborate our attempt in PTrack in the next section.

III. System Design

A. Design Overview

PTrack components own clear interfaces to the pedestrian tracking modules from existing systems for enhancing their applicability. Fig. 2 instantiates a concrete incorporation with the state-of-the-art approach [6] to illustrate the design.

PTrack step counter. Raw accelerometer readings are first passed to existing step counting modules, including low-pass filter, peak detection and acceleration segmentation, grayed in Fig. 2. The output accelerometer signals temporarily act as gait cycle candidates. They will be further projected to vertical and anterior two perpendicular directions (§III-B2), based on which real gait cycles can be recognized by the gait type identification module, so that the step counter can be correctly updated (§III-B1).

PTrack stride estimator. This component estimates the user's per-step stride length. It takes vertical accelerations and user's arm length as input to extract purely user's body movement during walking (named as *bounce*), based on a body bounce model we propose for wearable devices (§III-C1). Then body bounce is passed to the biomechanical model [6] to derive per-step stride length. In PTrack, the stride estimator is further extended with a user profile self-training design to automatically explore the required user profiles in the estimation (§III-C2).

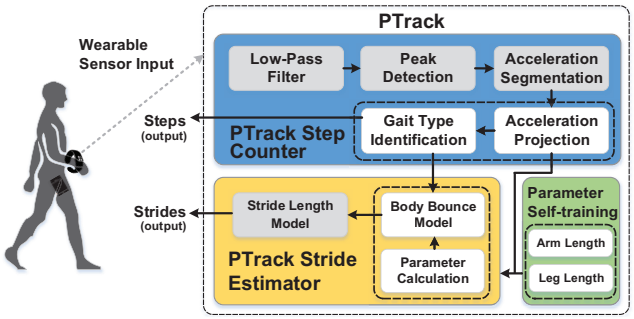

Fig. 2. Operation flow in PTrack. PTrack provides clear interfaces to incorporate with existing step counting and stride estimation modules (grayed).

B. Design of PTrack step counter

The output signals from existing step counting modules are temporarily viewed as gait candidates. This can help eliminate very ineligible activities, *e.g.,* mouse moving or keystroking without significant vertical motions. To further identify the real walking activities from the candidates, accelerations need to be projected in a prior. In the rest of this section, we focus on the identification first, and postpone the projection to §III-B2.

1) Gait type identification: To identify the walking activities from gait candidates, we have the following observation:

Key observation. Interfering activities are mainly generated by *rigid* objects, *e.g.,* user's arm/hand[2], while walking is an *aggregated* activity from both the arm and the body, and these two motion sources are *concurrent* but relatively *independent*. Their composition thus differs. To unveil this difference, we find if the movement of an interfering activity is projected to two perpendicular directions, due to the rigid nature, accelerations along these two directions are *synchronous*, *e.g.,* 1) signals reach peak and/or valley points (denoted as *turning* points) simultaneously or 2) signal along one direction reaches a turning point and signal on the perpendicular direction equals to zero (denoted as a crossing point). In contrast, such coincidence is impaired when the user walks. Their superposition will distort acceleration profile's matching along two projected directions, which motivates the following design.

Identifying user's walking. Fig. 3 illustrates the decomposition of a user's walking (and also its variants like jogging, running, etc.), in Fig. 3(a), as two concurrent activities: swinging in Fig. 3(b) and stepping in Fig. 3(c). For each motion, we highlight their accelerations within one gait cycle and mark all the critical points, *i.e.,* turning and crossing points. We indeed observe that when a rigid object moves, *e.g.,* arm in Fig. 3(b) and body in Fig. 3(c), the accelerations along two different directions are tightly synchronized[3]. Because these two activities reach (most) critical points at different times

[2]Interfering activities to be eliminated for step counting in PTrack mainly involve the arm/hand motions (wearables on user's wrist) when the body is relatively stable, since otherwise they should be counted into walking as well.

[3]The obvious offset at several points, *e.g.,* point 5 in Fig. 3(b) and points 5 and 9 in Fig. 3(c), is because the cushioning effect from elbows and knees. It slightly impairs the rigidity of arm and body, and thus increases the offset.

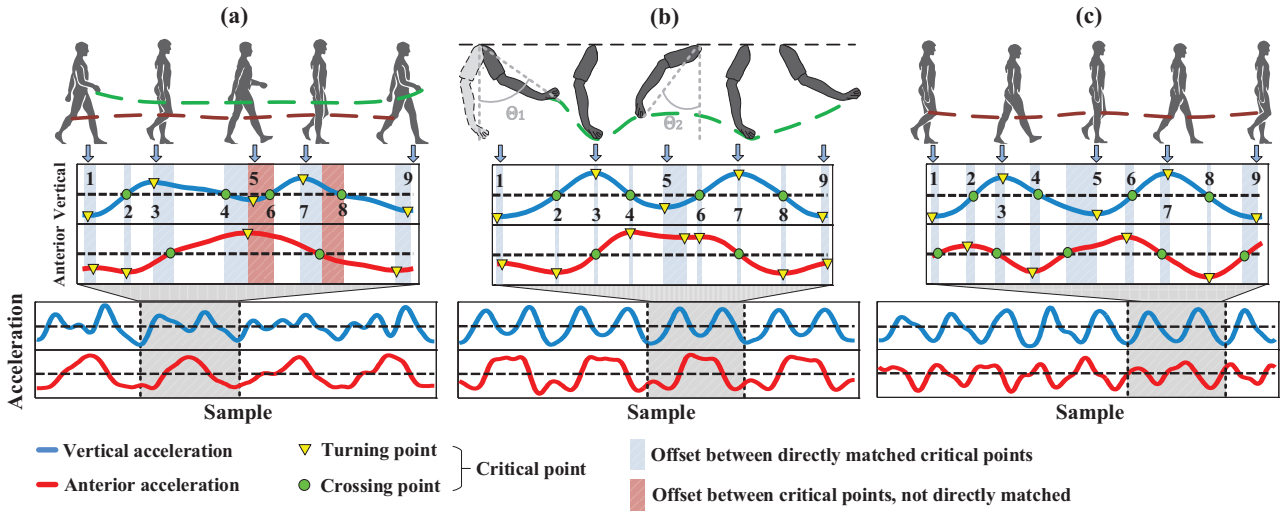

Fig. 3. Projected accelerations along vertical and anterior two perpendicular directions for 3 types of motions: (**a**) walking, (**b**) swinging and (**c**) stepping. For each type, we highlight accelerations within one gait cycle and mark all critical points, *e.g.,* turning and crossing points, used for the step counter design.

in one cycle, when the user performs them concurrently in Fig. 3(a), the critical points from the combined signals cross two directions thus exhibit evident offsets. For certain critical points, like points 6 and 8 in Fig. 3(a), their corresponding matching points on the anterior direction even disappear.

To quantify the synchronized level between two points, for any critical point n_v on the vertical direction, we define offset:

$$\delta(n_v) \quad = \quad w(n_v) \cdot \frac{|n_v - c(n_v)|}{n}, \qquad (1)$$

where $w(n_v)$ is a weighting factor, $c(n_v)$ is the closest critical point to n_v on the anterior direction, $|n_v - c(n_v)|$ indicates the number of samples between points n_v and $c(n_v)$, and n is the total amount of acceleration points in a gait cycle. In Eq. (1), $\frac{|n_v - c(n_v)|}{n}$ represents the normalized offset between points n_v and $c(n_v)$, which is further weighted by $w(n_v)$ to improve the offset reliability. In our current design, $w(n_v)$ is computed as the normalized sample number between n_v and the previous critical point along the same direction as n_v.

Therefore, our first step of the gait type identification design is to utilize an offset threshold δ to identify the user's walking, if the offset is greater than δ; Otherwise, the current activity is temporarily classified as an interference, since tiny offsets are likely produced by the movement from rigid objects. Extensive evaluation in §IV shows accurate and reliable performance, when δ is empirically set as 0.0325 in current implementation.

Identifying user's stepping. However, with this offset-based design, user's stepping will also be eliminated. In fact, stepping is another common gait pattern, *e.g.,* the user walks while carrying a handbag, answering the phone, thrusting hands into pockets, etc., which needs to be included. Nevertheless, stepping exhibits quite similar acceleration matching patterns as those from arm/hand gestures, which will thus stay mixed with the interference after the offset-based identification. To address this issue, we further leverage two observations:

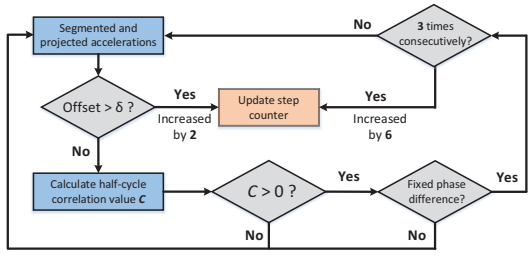

Fig. 4. Operation flow to update the number of steps in PTrack step counter.

- On the anterior direction, stepping produces an always-ahead movement, leading to repeated (co)sine-like acceleration patterns. However, constrained by the arm length, arm/hand gestures cannot produce such movements. Their movements are back and forth in general, which will inevitably incur sudden acceleration phase changes.
- On the other hand, existing studies, like [22], find that vertical and anterior accelerations of purely the user's body exhibit a fixed phase difference (*i.e.,* one quarter of a period). In the stepping case, wearable accelerations can unveil such a phase difference, but interfering activities are not ensured to have this feature.

Based on these two observations, we can further separate user's stepping from interfering activities. In particular, in one gait cycle, the user steps twice for the left and right legs, respectively. As a result, similar (co)sine-like acceleration patterns also appear twice in one gait cycle. When we perform auto-correlation [24] for the anterior accelerations within each gait cycle, it generates a large and positive correlation value, denoted as C, at the half-cycle point. On the contrary, the auto-correlation for interfering activities is not ensured to be positive at the half-cycle point due to the arm length limit. When arm changes its moving direction, the sine-like wave

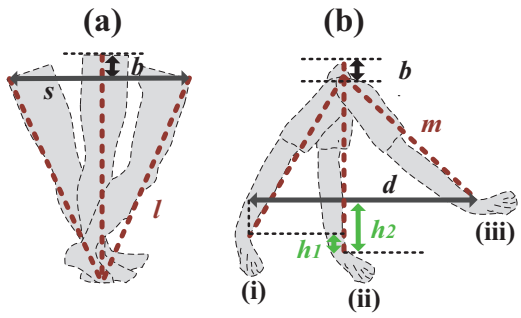

Fig. 5. Illustration of the stride estimation. (**a**) Existing stride estimation model. The key is to calculate the body's vertical displacement, *i.e.,* bounce b. (**b**) The geometric relation to derive b through wearable sensors in PTrack.

becomes a cosine-like wave, leading to a negative C.

Putting them together. As interfering activities do not always lead to positive C values, we thus confirm the stepping case when C is positive and two-direction accelerations also exhibit the fixed phase difference multiple times consecutively. Fig. 4 summarizes the entire gait cycle identification procedure.

2) Acceleration projection: To unveil the offset difference in Fig. 3, accelerations need to be projected to vertical and anterior two perpendicular directions. The vertical accelerations can be directly acquired from motion sensor APIs on both Android and iOS platforms [25]. For anterior accelerations, we notice when a user walks, its arms move back and forth along the anterior direction repeatedly. Hence, the shape of accelerations projected to the horizontal plane already indicates the moving direction using the least square technique.

C. Design of PTrack stride estimator

This component estimates user's stride length and Fig. 5(a) illustrates the principle well studied in the literature, where stride s can be derived using body's vertical displacement (also known as bounce b) and user's leg length l from the equation $l^2 = (s/2)^2 + (l-b)^2$. According to [1], [6], [14], this principle can be rephrased as:

$$s \quad = \quad k \cdot \sqrt{l^2 - (l-b)^2}, \tag{2}$$

where k is a calibration factor trained for each user during the initialization phase. Using Eq. (2), we can derive stride s through a per-step estimation of bounce b. Existing stride estimation approaches thus assume motion sensor firmly attached to the body, so that b can be derived from the vertical accelerations. As wearable sensors measure the additive movements from both the arm and body, existing stride estimation principles thus cannot be applied for wearables directly.

1) PTrack parameter estimator: To derive bounce b from wearable sensors, we look at three key moments of a gait cycle shown in Fig. 5(b). They indicate user's hand reaching the backmost, vertical and foremost three critical positions, respectively. The motion from (i) to (ii) corresponds to the motion from points 5 to 7 in Fig. 3(a). Similarly, the motion from (ii) to (iii) corresponds to points from 7 to 9 in Fig. 3(a).

From (i) to (ii), the user's arm moves downward while the body moves upward, leading to the following equation:

$$h_1 \quad = \quad r_1 - b, \tag{3}$$

where b is the body's bounce, r_1 indicates purely the arm's vertical displacement, and h_1 is the aggregated displacement of the wearable device that can be directly calculated from its vertical accelerations. Parallel to Eq. (3), we have the following equation for the movement from (ii) to (iii):

$$h_2 \quad = \quad r_2 - b, \tag{4}$$

where r_2 and h_2 represent the vertical displacement of the arm and the device, respectively. Finally, the geometric relationship disclosed in Fig. 5(b) from (i) to (iii) leads to:

$$d \quad = \quad \sqrt{m^2 - (m-r_1)^2} + \sqrt{m^2 - (m-r_2)^2}, \tag{5}$$

where m is the user's arm length and d represents the arm's anterior moving distance within one single step, which can be directly calculated from anterior accelerations. By substituting Eqs. (3) and (4) into (5), we can derive the close-form expression of bounce b (omitted due to page limit), and then apply Eq. (2) for stride estimation. Note that above calculations will convert to compute bounce b directly in the stepping case.

The calculation of h_1, h_2 and d above needs the double-integral of accelerations. A direct integral is erroneous [6], but the accuracy can be improved to the millimeter level by using the mean-removal technique as in the recent work [26]. This technique requires acceleration signals to be divided into pieces, where each piece starts and ends when the object's velocity (incurred by this acceleration) equals to zero. We check that all three parameters satisfy this requirement. The mean-removal technique thus can be applied in the calculation.

2) User profile self-training: Pedestrian tracking requires highly user-specific profiles, *e.g.,* arm (m) and leg (l) lengths. Explicit measure and input certainly incur inconvenient user experience. Moreover, measurement errors made by inexperienced users could lead to continuous performance deterioration. To address this issue, we propose a self-training design to avoid the user's intervention, with two steps: **Step 1**: Search for the optimal arm length $\hat{m}$. Then based on Eqs. (3) to (5), per-step bounce b can be precisely obtained. **Step 2**: Search for the optimal leg length $\hat{l}$. Then based on Eq. (2) and b, per-step stride s can be accurately derived. The technical details are omitted due to the page limit.

IV. Evaluation

We implement PTrack with LG Urbane [16] to examine PTrack's performance. Initially, the step counting is directly ready to work, while the stride estimator needs user's profile information further, including the arm and leg lengths. For the comparison purposes, we also help each user manually measure this information. With this setup, PTrack is then tested in various environments, on different types of road surfaces and different walking styles. The experiments last for more than one month and we assist each user to record their entire testing processes to obtain the ground truth for the evaluation.

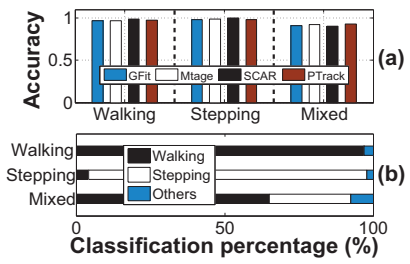

Fig. 6. Overall step counting performance. (**a**) Accuracy comparison among, GFit [16], MTage [6], SCAR [18] and PTrack. (**b**) Breakdown of the identified gait types in PTrack.

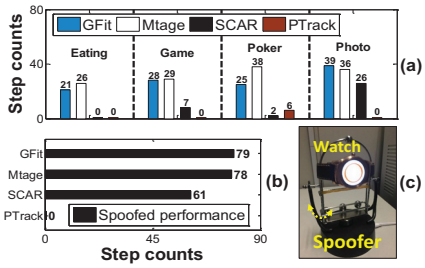

Fig. 7. Performance against interfering activities. We plot a tiny bar for "0". Mis-counted steps due to (**a**) interfering activities and (**b**) the spoofing technique. (**c**) The spoofer is used.

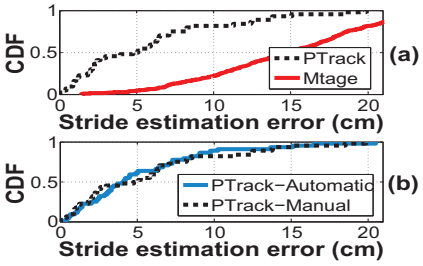

Fig. 8. Comparison of stride estimation performance between (**a**) PTrack and Montage [6], (**b**) different settings of PTrack.

A. Step counting performance

Approaches for comparison. For step counting, we compare the performance of PTrack with the following approaches:

- **Google Fit** (GFit): official built-in step counter developed by Google from industry. The device is on user's wrist.
- **Montage** (Mtage) [6]: state-of-the-art design from the research community, adopting the peak detection principle. As existing approaches [1], [3], [4], [8] follow a similar design principle , we do not compare PTrack with each of them individually.
- **SCAR** [18]: a machine-learning based approach, adopting the labeled data for training to mainly exclude predefined interfering activities. In this section, we collect data for both pedestrian activities, *e.g.,* walking, stepping and their mixture, and some typical interfering activities, *e.g.,* eating, gaming and playing poker, to form the training set.

Overall accuracy. We first evaluate the performance of four approaches without intended interfering activities. To report a detailed experimental result, we classify walking activities to three categories according to the recorded ground truth: walking only, stepping only and a mix of these two gait types. We then investigate the achieved accuracies in each category by different approaches. Fig. 6(a) shows that as long as users purely walk or step, all four approaches are accurate, *e.g.,* the average accuracies of "GFit/Mtage/SCAR/PTrack" are as high as "0.97/0.97/0.99/0.98", "0.98/0.99/1.0/0.98" and "0.91/0.92/0.90/0.93" in the three scenarios, respectively. "Mixed" has a slightly lower accuracy due to the transition between different gait types.

As we can access raw sensor data in PTrack, in Fig. 6(b), we further plot the breakdown of the gait type identification results to deeply understand PTrack's performance. We observe that only 2.3%, 1.7% and 7.4% cases are mis-identified as interfering activities, *e.g.,* "Others", which explains the good step counting performance PTrack achieves in Fig. 6(a).

Robustness to interference. In addition to walking, users unavoidably perform many other activities in the daily life. Throughout the experiment, users perform plenty of uncontrolled activities to challenge each step counting approach. For instance, Fig. 7 depicts the performance when users eat with knife and fork, play poker cards, take photos and play phone games. Ideally, the counter should keep silent. However, Fig. 7(a) indicates that "GFit" and "Mtage" could be mis-triggered from 20 to 39 times in merely 60s.

The performance of machine-learning based SCAR highly depends on the training data set. For the evaluation purpose, we intentionally do not train SCAR for "Photo". Fig. 7(a) shows that it performs well for the first three activities but degrades dramatically for "Photo". As the design of PTrack grabs the essential difference between the interfering and pedestrian activities, PTrack is thus very robust to various external interferences, not limited to the predefined data set.

In Fig. 7(b), we further compare the vulnerability of four approaches to spoofing, *e.g.,* device in Fig. 7(c). The result shows that both "GFit/Mtage/SCAR" tick "79/78/61" times in 60s, while PTrack is robust against spoofing, making its step counting result highly trustworthy.

Figs. 6 and 7 together indicate that PTrack can achieve comparable and accurate step counting performance as existing approaches for the common pedestrian activities. Meanwhile, PTrack can also exclude various interfering activities without explicit training. Existing approaches lack such a capability.

B. Stride estimation performance

So far as we know, a dedicated stride estimation design for wearable devices is still lacking. The most relevant approach is Montage [6], which could derive user's moving distance using smart phones when phones are firmly attached to user's body and reflect purely the body's motion. On the other hand, we are not aware of any commercial stride estimation products. We thus focus on the comparison between Montage (Mtage) and PTrack, and also PTrack's performance of the self-training mechanism design.

Overall accuracy. To evaluate the stride estimation accuracy, users go through plenty of indoor and outdoor trajectories with measured trajectory lengths. In Fig. 8(a), we first compare PTrack with Mtage. In the experiment, we find that to well satisfy the assumption made by Mltage, the device needs to tightly attach to user's body or the user needs to firmly hold

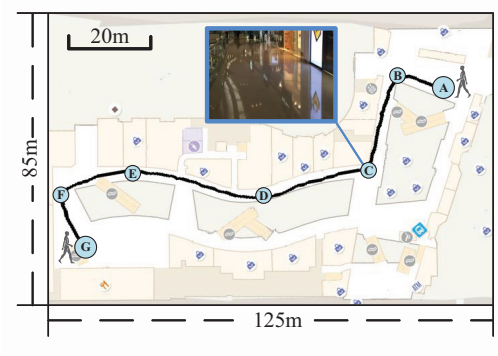

Fig. 9. Incorporating PTrack into indoor navigation design with dead-reckoning. The route is 141.5m and PTrack's measure is 136.4m.

the device (without relative motion *w.r.t.* the body), so that the vertical bounce can be directly measured. However, wearable devices measure additive motions (not merely body's) and the performance of Mltage unavoidably deteriorates. In contrast, as PTrack could derive bounce from mixed motion readings, the error is thus small, *e.g.,* about 5cm per-step on average.

We further examine the efficacy of PTrack's self-training design for user's arm and leg lengths. To this end, we manually measure these profiles for each user as a benchmark. Fig. 8(b) depicts the CDF of the per-step stride estimation error. Both "PTrack-Automatic" and "PTrack-Manual" are accurate, *e.g.,* their average errors are 5.3cm and 5.7cm, respectively. The reason that "PTrack-Automatic" can even slightly outperform "PTrack-Manual" is because we may not always find precise starting or ending points in manual measurements, but the self-training can avoid this issue.

Case study. To further understand PTrack's benefit to upper-layer applications, we evaluate it using an indoor navigation case study in a large shopping center, where Fig. 9 depicts the floor map. After a user comes out from a store, the user is at position A and he wants to go to elevator at position G, following five intermediate markers from B to F in the figure. In the experiment, from spot B to C to D, the user intentionally crosses a 4-meter wide corridor twice. During the entire navigation, the tracked trajectory provided by PTrack closely follows the suggested route. Fig. 9 clearly indicates that pedestrian tracking by PTrack is highly accurate so that such subtle differences can be reflected. Overall, the average per-step error along the navigation route from A to G is 5.1cm. As not all the indoor places are likely to be well covered by the localization service, PTrack could significantly boost the accuracy and robustness of such location-based applications.

V. Conclusion

This paper presents our attempt to develop PTrack to enhance the usability of pedestrian tracking. PTrack addresses design vulnerability to interfering activities, preserving purely body's movement from additive sensor signals, and parameter training without the user's intervention three key applicability issues. We implement PTrack and evaluation its performance. Experiments show PTrack effectively enhances the design applicability and achieves promising performance under very practical settings. In the future, we plan to adaptively tune the threshold δ and also conduct more experiments to further understand PTrack's performance.

Acknowledgment

This work is supported in part by the ECS grant from Research Grants Council of Hong Kong (Project No. CityU 21203516), the Start-up grant from City University of Hong Kong (Project No. 7200480), and the CRF grant from Research Grants Council of Hong Kong (Project No. C7036-15G).

References

[1] M. Alzantot and M. Youssef, "Uptime: Ubiquitous pedestrian tracking using mobile phones," in *Proc. of IEEE WCNC*, 2012.
[2] "Forbes," http://www.forbes.com/.
[3] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-navi: Self-deployable indoor navigation system," in *Proc. of ACM MobiCom*, 2014.
[4] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: unsupervised indoor localization," in *Proc. of ACM MobiSys*, 2012.
[5] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system." in *Proc. of USENIX NSDI*, 2013.
[6] L. Zhang, K. Liu, Y. Jiang, X.-Y. Li, Y. Liu, and P. Yang, "Montage: Combine frames with movement continuity for realtime multi-user tracking," in *Proc. of IEEE INFOCOM*, 2014.
[7] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Communications Surveys & Tutorials*, 2013.
[8] N. Roy, H. Wang, and R. Roy Choudhury, "I am a smartphone and i can tell my user's walking direction," in *Proc. of ACM MobiSys*, 2014.
[9] L. Yang, Y. Guo, X. Ding, J. Han, Y. Liu, C. Wang, and C. Hu, "Unlocking smart phone through handwaving biometrics," *IEEE Transactions on Mobile Computing*, 2015.
[10] https://www.wsj.com/articles/SB10981274571634828 3.
[11] http://www.sonicboomwellness.com/pedometer-accuracy.
[12] https://www.verywell.com/most-common-pedometer-problems-3435757.
[13] https://help.fitbit.com/articles/en_US/Help_article/1136.
[14] J. Jahn, U. Batzer, J. Seitz, L. Patino-Studencka, and J. G. Boronat, "Comparison and evaluation of acceleration based step length estimators for handheld devices," in *Proc. of IEEE IPIN*, 2010.
[15] "UNFIT BITS," http://www.unfitbits.com/.
[16] "LG Watch Urbane," http://www.lg.com/us/smartwatch/urbane.
[17] Z. Yang, L. Shangguan, W. Gu, Z. Zhou, C. Wu, and Y. Liu, "Sherlock: Micro-environment sensing for smartphones," *IEEE Transactions on Parallel and Distributed Systems*, 2014.
[18] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook, "Simple and complex activity recognition through smart phones," in *Proc. of IEEE IE*, 2012.
[19] W. Zijlstra and A. L. Hof, "Displacement of the pelvis during human walking: experimental data and model predictions," *Elsevier Gait & posture*, 1997.
[20] D. Alvarez, R. C. González, A. López, and J. C. Alvarez, "Comparison of step length estimators from weareable accelerometer devices," in *Proc. of IEEE EMBS*, 2006.
[21] I. Bylemans, M. Weyn, and M. Klepal, "Mobile phone-based displacement estimation for opportunistic localisation systems," in *Proc. of IEEE UBICOMM*, 2009.
[22] J. W. Kim, H. J. Jang, D.-H. Hwang, and C. Park, "A step, stride and heading determination for the pedestrian navigation system," *Journal of Global Positioning System*, 2004.
[23] "Xiaomin Band," http://www.mi.com/en/miband/.
[24] J. Han, C. Qian, P. Yang, D. Ma, Z. Jiang, W. Xi, and J. Zhao, "Geneprint: generic and accurate physical-layer identification for uhf rfid tags," *IEEE/ACM Transactions on Networking*, 2016.
[25] P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in *Proc. of ACM MobiCom*, 2014.
[26] H. Wang, T.-T. Lai, and R. R. Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proc. of ACM MobiCom*, 2015.