Open source step counter algorithm for wearable devices

ANNA BRONDIN*, MARCUS NORDSTRÖM*, CARL MAGNUS OLSSON*, and DARIO SALVI*, Malmö University

Commercial wearable devices and fitness trackers are commonly sold as black boxes of which little is known about their accuracy. This poses serious issues especially in health-related contexts such as clinical research, where transparency about accuracy and reliability are paramount.

We present a validated algorithm for computing step counting that is optimised for use in constrained computing environments. Released as open source, the algorithm is based on the windowed peak detection approach, which has previously shown high accuracy on smartphones. The algorithm is optimised to run on a programmable smartwatch (Pine Time) and tested on 10 subjects in 8 scenarios, with varying varying positions of the wearable and walking paces.

Our approach achieves a 89% average accuracy, with the highest average accuracy when walking outdoor (98%) and the lowest in a slow-walk scenario (77%). This result can be compared with the built-in step counter of the smartwatch (Bosch BMA421), which yielded a 94% average accuracy for the same use cases. Our work thus shows that an open-source approach for extracting physical activity data from wearable devices is possible and achieves an accuracy comparable to the one produced by proprietary embedded algorithms.

CCS Concepts: • **Applied computing** → **Consumer health**; • **General and reference** → *Measurement*; • **Computer systems organization** → Sensors and actuators.

Additional Key Words and Phrases: Step counting, Wearable devices, Open source

ACM Reference Format:
Anna Brondin, Marcus Nordström, Carl Magnus Olsson, and Dario Salvi. 2020. Open source step counter algorithm for wearable devices. In *IOT-HSA-2020: Workshop on Internet of Things based Health Services and Applications, October 7–9, 2020, Malmö, Sweden.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Tracking one's health using a wearable device is commonplace today, with e.g. more than one third of US citizens having largely positive experiences from it [11]. Such wearable devices quantify a mix of physical activity indicators (e.g. steps, calories burned, and distance walked) and physiological indicators (e.g. heart rate, heart rate variability and blood oxygen saturation) [10]. Although studies about accuracy of commercial devices are being conducted [1, 5, 8], the products themselves are sold as *black box* solutions sharing no details about the algorithms they employ or the accuracy they achieve in defined use cases. This is problematic in contexts like clinical research, where information about the accuracy of the measurements and comparison among subjects are paramount.

To address the lack of transparency, this paper reports from the development of an open-source algorithm suitable for fitness trackers and smart-watches. Existing open-source solutions for

*Authors contributed equally to this research.

IOT-HSA-2020, October 7–9, 2020, Malmö, Sweden

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

wearable devices often lack well-tested algorithms for extracting data from the raw sensor data (e.g. from accelerometers or photoplethysmography) [3]. The aim of our study is therefore to develop an algorithm for computing steps from raw accelerometry, optimised for embedded devices. We do so based on a previously validated approach [13], but adapt the algorithm for running in a more constrained environment. After parameter optimisation to suit the hardware of our smartwatch, our algorithm is then assessed in terms of accuracy and compared with the dedicated step counter of the smartwatch.

2 RELATED WORKS

Earlier step-counting approaches used mechanical spring-levered pedometers with coupled digital counters [2], but, since the wide adoption of smartphones in large populations, a number of algorithms have been proposed for that platform. Several of these algorithms have been compared in one of the most cited research articles related to step counting and walk detection [4]. This work compared ten algorithms using 130 walk traces from 27 people, and were collected using smartphone accelerometers in six positions. The results showed windowed peak detection as a particularly suitable approach given a median error of 1.3% despite the simplicity of the algorithm. To validate the results, users were asked to walk at varying speeds, while video recording the activity to allow the researchers to retrieve a ground truth step count.

While [4] is one of the most comprehensive in literature, the implementation of their algorithms is not available as open-source. This motivated the work of [13], who further optimized the windowed peak detection algorithm in order to make it work on smartphones and released the source code with an open-source license. Their implementation addressed Android smartphones and is therefore programmed in Java. The need to run a virtual machine nonetheless adds a computational cost that is not recommendable in constrained environments like smartwatches.

Instead, when applied to wearable devices, step counting algorithms usually take the form of peak detection, zero-crossings detection, auto-correlation methods and frequency-based methods [12]. Of these, the FFT analysis showed a high intraclass correlation coefficient in most tested scenarios, except when subjects walked at varying speeds where peak detection and auto-correlation offer a less accurate, but more reliable solution [12].

3 METHODS

We adopted the algorithm used in [13] for three reasons: it has shown to be computationally efficient, its accuracy has been validated in different walking scenarios, and because of the availability of its implementation as open-source. The algorithm consists of the following stages:

- **Pre-processing**: combines the three orthogonal acceleration samples (x_i, y_i, z_i) into one magnitude value according to $m_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$. If the sampling method presents jitter, the magnitude values can be also linearly interpolated.
- **Filtering**: implements a low-pass FIR-filter to cut off the noise from frequencies above 3Hz.
- **Scoring**: amplifies the peaks of the signal to make their detection easier. It uses the mean difference computed as $p_i = \frac{\sum_{k=-N, k\neq i}^{N} (m_i - m_{i+k})}{2N}$, where m is the magnitude signal.
- **Detection**: determines the peaks in the signal by comparing the magnitude m_i with a running mean μ and standard deviation σ of the signal. A peak is detected if $m_i - \mu > \sigma \cdot th$.
- **Post-processing**: handles false positives from the previous stage by having a sliding window of fixed size t_{window} and selecting the higher peak within the window.

The original implementation presented in [13] was ported from Java to C and adapted to support running in constrained execution environments. The source implementation used parallel threads

for each stage of the algorithm, therefore we had to serialize the computation of each stage, using statically allocated buffers between them (implemented in [9]). Furthermore, to allow the computation on devices without dedicated floating point unit, we implemented all the computations using integers, particularly we used the coordinate rotation digital computer (CORDIC) [15] implementation of the square root [17] and split the comparison in the detection stage into two parts according to: $m_i - \mu > \sigma \cdot th_{whole} + \frac{\sigma}{th_{frac}}$.

In order to test how these changes affect the accuracy of the algorithm, we applied it to the same dataset previously collected in [13]. The signals in this dataset were acquired on a mobile phone using the API offered by Android. The API provides the acceleration as a multiple of gravity, therefore we converted that number to an integer by multiplying it by a constant.

After ensuring that the algorithm's accuracy was comparable to its Android counterpart, we moved on to testing the algorithm on a wearable device - in our case the Pine Time programmable smartwatch. To do so, we adapted an existing open-source firmware based on the Zephyr Real Time Operating System [14] that easily allowed running the algorithm for each acceleration samples gathered at 100 Hz. The samples were also streamed via Bluetooth to a smartphone app which was used to collect data for off-line analysis. The Pine Time smartwatch includes the Bosch BMA421 accelerometer, which has built-in intelligence for step counting. This allowed us to compare our results with the ones produced by the dedicated hardware.

The accuracy of the algorithm depends on how some parameters (i.e. N in the scoring stage, the threshold in the detection stage and t_{window} in the post-processing stage) are optimised in relation to the characteristics of the signal. While mobile phones offer normalised signals expressed as multiples of the gravity, the same does not apply to signals acquired directly from the hardware. In order to optimise the parameters, we therefore acquired data in shorter walks of 50-100 steps, where steps were manually counted as a reference. We then used these data to optimise the algorithm though a grid search of the parameters space.

Finally, the optimised implementation was tested with 10 participants. Each participant performed 500 m walks at varied speeds on a treadmill according to scenarios similar to those proposed in [6] and described in Table 1. These scenarios allow for controlled speed while representing a variety of circumstances that a device is likely to be exposed to. To include a more realistic type of terrain, we also tested the "arms swinging" scenario on gravel and asphalt outdoor.

As ground truth data, we used the average between what was manually counted by an observer and the output of a Polar stride sensor placed on a participants' shoe. The stride sensor provides the cadence every second over Bluetooth. Given that cadence is the number of times per minute the foot with the sensor hits the ground, the total number of steps was therefore computed by multiplying the total duration of the walk by 2 times average cadence. To validate this approach, one participant performed four walking and three jogging tests outdoor on gravel and asphalt while an observer manually counted the steps. This led to an estimated 98% accuracy.

While conducting the tests we observed that the constrained hardware resources of the smartwatch did not make it possible to keep all its functionalities (step counting algorithm, Bluetooth streaming, embedded step counter and screen interaction) running simultaneously. Therefore, we decided to place two watches on each participant, one running our algorithm and streaming the data over Bluetooth, and the other using the embedded step counter (figure 1). This allowed us to still collect all the data simultaneously.

Arms	Speed	Terrain
Hands in pockets	normal walk @4.5kmh	treadmill
Arms half way extended and stationary, one hand holding a phone	normal walk @4.5kmh	treadmill
Folded arms, arms crossed on chest	normal walk @4.5kmh	treadmill
Arms swinging	slow walk @2.5kmh	treadmill
Arms swinging	normal walk @4.5kmh	treadmill
Arms swinging	fast walk @6.5kmh	treadmill
Arms swinging	running @8.5kmh	treadmill
Arms swinging	normal walk	mixed outdoor: gravel and asphalt

Table 1. Walk scenarios employed during tests.

Fig. 1. Test equipment

We finally released the code on Github with the MIT open source license and is available at https://github.com/Oxford-step-counter/C-Step-Counter.

4 RESULTS AND DISCUSSION

When running the algorithm on the dataset provided in [13], we obtained a slightly lower accuracy compared to the original Java implementation for smartphones (table 2). As all optimisation parameters for the algorithm at this point were left as the ones in the original work, this difference may be explained by the fact that the CORDIC implementation of the square root introduced an approximation that affected the overall accuracy.

We then optimised the parameters of the algorithm using data collected with the smartwatch. This optimisation process led to the following parameters being chosen: N in the scoring stage was set to 40, th_{whole} and th_{frac} in the detection stage to 2 both and t_{window} in the post-processing

Position	C version accuracy (%)		Java version accuracy (%)	
	Mean/SD	Min/Max	Mean/SD	Min/Max
Hand	**89** / 7	80 / 99	**92** / 8	67 / 99
Armband	**89** / 5	82 / 97	**95** / 3	91 /99
Back pocket	**92** / 8	82 / 100	**94** / 5	87 / 99
Front pocket	**96** / 4	88 / 100	**94** / 7	85 / 100
Neck pouch	**94** / 8	83 / 99	**95** / 7	86 / 100
Purse	**81** / 10	62 / 90	**81** / 15	59 / 97
All	**90** / 8	62 / 100	**92** / 9	59 / 100

Table 2. Accuracy statistics (mean, standard deviation, min and max) of the two versions of the step counter algorithm (C and Java) using offline testing of the dataset in [13].

Scenario	Our algorithm accuracy (%)		BMA421 accuracy (%)	
	Mean/SD	Min/Max	Mean/SD	Min/Max
Hands in pocket	**88** / 8	77 / 98	**98** / 3	91 / 100
Grabbed phone	**83** / 14	53 / 99	**98** / 4	87 / 100
Folded arms	**90** / 10	70 / 100	**99** / 1	97 / 100
Slow walking	**77** / 20	41/96	**63** / 28	22 / 97
Normal walking	**94** / 5	80/99	**98** / 2	93 / 99
Fast walking	**91** / 5	83 / 97	**98** / 3	93/100
Normal running	**95** / 5	84 / 100	**97** / 3	93 / 100
Walking outdoor	**98** / 2	95 / 99	**98** / 2	95 / 100
All	**89** / 12	41/100	**94** / 15	22 / 100

Table 3. Accuracy statistics (mean, standard deviation, min and max) of the step counting algorithm from testing with users at 50Hz. Comparison of our algorithm with the dedicated embedded step counter.

stage was set to 200ms. In addition, we also increased the sensitivity of the accelerometer from the default $2G$ to $16G$ and decreased the resolution, from the default 12 bit to 10 bit by removing the last 2 bits using bit shifting. On the data set used for optimisation, this led to an accuracy of 99.6% when sampling at 100Hz and 99.2% when sampling at 50Hz.

The final implementation was then tested with ten participants, six females and four males, aged between 21 and 60 and with a median age of 25 years. All participants were wearing two smartwatches and the stride sensor in the eight scenarios described in table 1. As accuracy was not affected by sampling at lower rates, all tests were conducted at 50Hz. The results of the tests are summarised in 3. Overall, they show that our algorithm performs slightly worse compared with the embedded algorithm in the BMA421 accelerometer, except for the slow walk scenario, where it still achieves a somewhat low accuracy but with a higher mean than the BMA421. The challenges with slow walking are not new to step counting (cf. [7, 16]) and were expected.

These results suggest that, when available, an embedded step counter may be more accurate than our algorithm and may therefore be preferable. This is also motivated by the embedded step counter having no need for using the CPU, thus reducing power consumption. However, when dedicated hardware is not present, our algorithm provides a relatively accurate alternative. As it also holds the advantage of being open source, it is modifiable and can be optimised to support different walking styles, contrary to the closed source proprietary alternatives.

5 CONCLUSIONS AND FUTURE WORK

In this study, we report from porting and optimising a step counting algorithm for hardware-constrained devices like smartwatches and other wearables. Our approach obtained an accuracy that was slightly inferior to a proprietary algorithm embedded in an hardware accelerometer. However, it remains suitable for use in devices where no such hardware step counter exist, or when openness, modifiability, or known accuracy are required - as is often the case in clinical research. The full implementation is available under an MIT open source license. With an average of 89% accuracy on different walking styles and arms positions, it is very promising when it is not possible to employ a dedicated step-counting hardware.

Future work will be needed to further improve the accuracy of the algorithm and validate it in more challenging scenarios such as for slow or impaired walking styles, on different terrains, and with different types of users. Finally, as our work reported here focussed on accuracy and did not assess power consumption, future studies would be needed to understand trade-offs between accuracy and computational requirements or resources utilisation.

ACKNOWLEDGMENTS

This work is partially funded by the Knowledge Foundation through the Internet of Things and People research profile.

REFERENCES

[1] Parastoo Alinia, Chris Cain, Ramin Fallahzadeh, Armin Shahrokni, Diane Cook, and Hassan Ghasemzadeh. 2017. How Accurate Is Your Activity Tracker? A Comparative Study of Step Counts in Low-Intensity Physical Activities. *JMIR mHealth and uHealth* 5, 8 (Aug. 2017), e106. https://doi.org/10.2196/mhealth.6321

[2] David R Bassett, Lindsay P Toth, Samuel R LaMunion, and Scott E Crouter. 2017. Step counting: a review of measurement considerations and health-related applications. *Sports Medicine* 47, 7 (2017), 1303–1315.

[3] Ganapati Bhat, Ranadeep Deb, and Umit Y Ogras. 2019. OpenHealth: Open-source platform for wearable health monitoring. *IEEE Design & Test* 36, 5 (2019), 27–34.

[4] Agata Brajdic and Robert Harle. 2013. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 225–234.

[5] Meredith A Case, Holland A Burwick, Kevin G Volpp, and Mitesh S Patel. 2015. Accuracy of smartphone applications and wearable devices for tracking physical activity data. *Jama* 313, 6 (2015), 625–626.

[6] Yunhoon Cho, Hyuntae Cho, and Chong-Min Kyung. 2016. Design and implementation of practical step detection algorithm for wrist-worn devices. *IEEE Sensors Journal* 16, 21 (2016), 7720–7730.

[7] Yuanyuan Feng, Christopher K Wong, Vandana Janeja, Ravi Kuber, and Helena M Mentis. 2017. Comparison of tri-axial accelerometers step-count accuracy in slow walking conditions. *Gait & posture* 53 (2017), 11–16.

[8] Rahel Gilgen-Ammann, Theresa Schweizer, and Thomas Wyss. 2020. Accuracy of Distance Recordings in Eight Positioning-Enabled Sport Watches: Instrument Validation Study. *JMIR mHealth and uHealth* 8, 6 (2020), e17118.

[9] Anders Kalør. 2014. *Ring-Buffer*. https://github.com/AndersKaloer/Ring-Buffer

[10] Aida Kamišalić, Iztok Fister, Muhamed Turkanović, and Sašo Karakatič. 2018. Sensors and functionalities of non-invasive wrist-wearable devices: A review. *Sensors* 18, 6 (2018), 1714.

[11] Justin McCarthy. 2019. *One in Five U.S. Adults Use Health Apps, Wearable Trackers*. https://news.gallup.com/poll/269096/one-five-adults-health-apps-wearable-trackers.aspx

[12] Matthew B Rhudy and Joseph M Mahoney. 2018. A comprehensive comparison of simple step counting techniques using wrist-and ankle-mounted accelerometer and gyroscope signals. *Journal of Medical Engineering & Technology* 42, 3 (2018), 236–243.

[13] Dario Salvi, Carmelo Velardo, Jamieson Brynes, and Lionel Tarassenko. 2018. An optimised algorithm for accurate steps counting from smart-phone accelerometry. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 4423–4427.

[14] The Linux foundation projects. 2020. *Zephyr Project*. https://www.zephyrproject.org/

[15] Jack Volder. [n.d.]. The CORDIC computing technique. In *Papers presented at the the March 3-5, 1959, western joint computer conference on XX - IRE-AIEE-ACM '59 (Western)* (San Francisco, California, 1959). ACM Press, 257–261. https://doi.org/10.1145/1457838.1457886

[16] Christopher K Wong, Helena M Mentis, and Ravi Kuber. 2018. The bit doesn't fit: Evaluation of a commercial activity-tracker at slower walking speeds. *Gait & posture* 59 (2018), 177–181.
[17] Convict Épiscopal Luxembourg. 2002. *Square-root based on CORDIC*. https://www.convict.lu/Jeunes/Math/square_root_CORDIC.htm