Real-time Motion Detection for Android Smartphones

Cassiano Andrade, Ismael Silva, Glívia Barbosa, Flávio Coutinho
Department of Computing
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, Brazil
cassiano.b.andrade@gmail.com, ismaelsantana@cefetmg.br, gliviabarbosa@decom.cefetmg.br, fegemo@cefetmg.br

***Abstract*—Real-time motion detection can be used to monitor a wide range of physical activities, such as those typically leveraged by exergames, but the literature present few algorithms. Motivated by this demand, we aimed at evaluating and proposing motion detection algorithms in Android smartphones in order to comply with three requirements: accuracy, precision, and recall above 95%. We evaluated the existing algorithms, Google and Step Detector. The results of this evaluation showed the Google algorithm as the most promising, but it did not meet all the requirements. Therefore, a new algorithm, based on it, was proposed and evaluated. Thus, this paper introduces the Castor algorithm and a tool for analyzing motion detection algorithms. In our evaluation, Castor was able to overcome Google in most scenarios and could adapt to lower-speed movements, when Google's algorithm could not. As contributions, this paper presents Castor as an algorithm to detect movements and also a tool which can be extended for implementing and evaluating new algorithms.**

***Keywords*-motion detection; accelerometer; Android; exergames;**

I. Introduction

Smartphones provide sensors such as accelerometers and gyroscopes which can be used by algorithms for indoors motion detection. A wide range of applications can leverage the ability to identify the event of a movement in real-time, such as for physical activities monitoring or exergames.

Although there are mobile apps which can detect movement from different physical activities in Google's and Apple's app stores[1], their algorithms are rarely made public. In the literature, there is a lack of motion detection algorithms with reasonable accuracy, precision and recall that works in real-time. Therefore, we aimed to evaluate and propose a motion detection algorithm that can detect movements in real-time with precision, accuracy and recall above 95% for the Android platform.

To achieve our goal, we followed the steps of: (a) developing a tool for proposing and analyzing motion detection algorithms, (b) implementing and evaluating the algorithms Google [4] and Step Detector, (c) developing our motion detection algorithm called Castor and (d) comparatively evaluating Castor and Google

We developed Castor as an improvement over Google's algorithm [4] and submitted both to four experiments, which evaluated their performance on different smartphone position on the body, through different types of exercises, their robustness for detecting movements in different speeds and various smartphone devices.

Castor overcame some detection adversities which happened in Google's algorithm, and it had average accuracy, precision, and recall results above 95%. Google could not satisfy such requirements in 14.29% of the test cases, against 9.52% by Castor. In these last cases, Castor's performance was hampered by noises due to its adaptive detection method, which we describe in the next sections.

As a collateral result, we developed and made public an extensible tool for authoring and evaluating real-time motion detection algorithms.

Our contributions comprise the Castor algorithm, which can be used for detecting movements by Android applications, but can also be implemented in other environments that provide three-dimensional accelerometers; and the tool which can be extended for implementing and evaluating other motion detection algorithms.

Therefore, Castor can empower applications that aim to motivate users to practice physical exercises. This approach can reach a vast amount of people since even the most basic smartphones have accelerometer sensors that can detect acceleration in three dimensions [6].

The remainder of the paper is organized as follows: Section II provides a brief overview of the literature regarding motion detection algorithms. Section III explains the experiments made with Google and Step Detector algorithms, while Section IV describes Google's algorithmic structure, which served as the foundation for the proposed algorithm. Section V explains the Castor algorithm, and in Section VI we describe the evaluation experiments made with Google and Castor. Finally, Section VII analyzes the limitations and future lines of research stemming from this work.

II. Background - Related Works

Hassan et al. [1] aimed to record different human activities by (1) obtaining data from the accelerometer and gyroscope sensors in order to calculate mean, median and autoregressive coefficients, (2) using Kernel Principal Component Analysis (KPCA) and Linear Discriminant Analysis (LDA) to make these metrics more robust and (3) Deep Belief

[1](a) Strava Running and Cycling, (b) Runkeeper and (c) Pacer

Network (DBN) application to correctly classify their human activity. They compare their approach to Support Vector Machine (SVM) and Artificial Neural Network (ANN) techniques. The main disadvantage of this approach is the necessity of high processing time due to the Deep Learning technique.

Strohrmann et al. [2] studied the movements performed by a runner's arm attaching a smartphone to its biceps. The authors' algorithm tries to detect the overpassing of the arm through the central line of the body. This approach showed low accuracy since the best accuracy value was 80.73%.

San et al. [3] use the Hidden Markov Chains (HMC) statistical model for motion detection, and they compare its results with those obtained using Fixed-point Arithmetic (FA). This statistical model requires a large amount of data to correctly model the human movement types that one wishes to classify.

Google algorithm [4] uses the three-dimensional acceleration vectors obtained by the accelerometers in order to estimate the current speed and, thus, evaluate if a movement occurred. The algorithm counts a movement when the estimated speed exceeds, in a growing state, the threshold value, a predefined constant whose value is 15.

The Android algorithm [5], called Step Detector, uses a native software sensor labeled step detector to detect motion. This sensor uses other sensors, based either on software or hardware, to accomplish its detection logic. As of 2019, 95.3% of current Android smartphones have the step detector sensor [5].

The motion detection techniques proposed in the three first works mentioned above cannot detect motion in real time because they have, respectively, high processing time [1], low accuracy [2] and the necessity for a large volume of training data [3].

Google [4] and Android [5] algorithms can be used in Android smartphones to detect motion in real-time. The following section describes the experiments we made with both algorithms to verify if they accomplished the precision, accuracy and recall requirements.

III. Experiments with Google and Step Detector

Google and Step Detector algorithms use, respectively, the accelerometers and the step detector sensors. In order to test them simultaneously, we developed a tool as an Android app which grants access to multiple sensors, enabling the implementation and analysis of motion detection algorithms that can use different strategies for acquiring input data.

Fig. 1 exposes the package diagram that shows the tool packages divided into logical groupings showing the dependencies between each one of them. The controller of each motion detection algorithm is responsible for processing the sensing information and supplying the processing results to the other components.

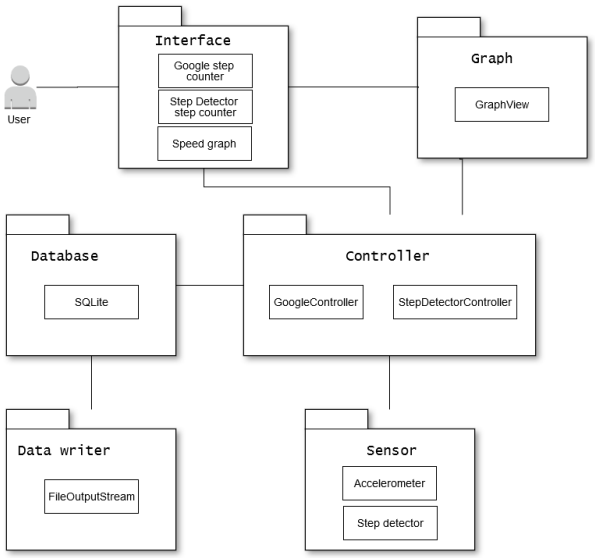

Figure 1. Package diagram of the motion detection algorithm analysis tool.

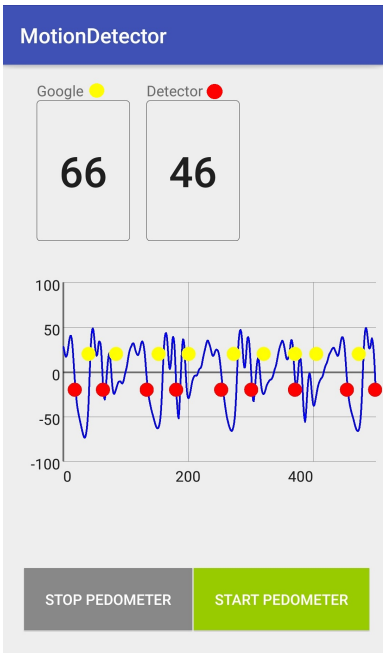

Figure 2. Screenshot of the tool's graphical interface.

The tool for analyzing motion detection algorithms[2] had as requirements (1) to work in Android smartphones, (2) to use different sensors simultaneously, (3) to export the sensors data for analysis during and after the detection period and (4) to be extensible for different motion detection algorithms, allowing their implementation with low effort.

The sensors data export occurred in two ways: (1) presen-

[2]https://bitbucket.org/move2playteam/motiondetectors-android-application/

tation of a speed graph over time in the tool interface and (2) data recording in the smartphone's local storage. This data consisted of (a) acceleration data, (b) speed data and (c) timestamp information of each motion detection.

Fig. 2 presents the tool single graphical interface, illustrating a 30 seconds execution of a person walking with the smartphone on their arm. The upper part displays the Google and Step Detector movement counters and, in the central part, there is a speed over time graph with detection points for each algorithm. The detection points values in the y-axis were statically defined so they have no physical meaning.

Once the tool was developed, and Google and Step Detector algorithms implemented, we performed two experiments: (1) experiment with different people and (2) experiment with different smartphones. The following sections describe these experiments.

A. Experiment with different people

In the experiment with different people, we selected a heterogeneous set of seven people who attend the gym to do running machine and bicycle activities. Table I characterizes the experiment participants. We sought a significantly heterogeneous sample of people to capture different usage realities.

TABLE I. EXPERIMENT PARTICIPANTS CHARACTERIZATION.

Person ID	Characterization			
	Gender	*Age*	*Height*	*Weight*
P1	Female	23 years old	172 cm	57 kg
P2	Male	22 years old	175 cm	59 kg
P3	Male	53 years old	162 cm	62 kg
P4	Female	70 years old	165 cm	63 kg
P5	Female	30 years old	159 cm	52 kg
P6	Male	40 years old	171 cm	81 kg
P7	Female	29 years old	160 cm	66 kg

The evaluation assessed the algorithms for their *accuracy* for each person and each activity, according to (1)

$$A = \left(1 - \frac{|M_d - M_p|}{M_p}\right).100\% \quad (1)$$

in which M_d is the quantity of movements detected and M_p is the amount of movements performed.

Table II lists the activities to which each participant has been submitted. They were defined based on the walking and pedaling contexts for two speed intervals each.

TABLE II. EXPERIMENT PROPOSED ACTIVITIES.

Activity ID	Activity description
Running machine 1	Walking from 2 km/h to 4 km/h
Running machine 2	Walking from 4 km/h to 6 km/h
Bicycle 1	Pedaling a bicycle from 50 rpm to 60 rpm
Bicycle 2	Pedaling a bicycle from 60 rpm to 80 rpm

Each one of the seven participants performed the four activities. In each case, the smartphone was located in an armlet and this armlet was affixed on the person's biceps region. The smartphone used was one of brand Asus and model ZE551ML. Table III presents all the 28 experiment results.

TABLE III. RESULTS OF THE EXPERIMENT WITH DIFFERENT PEOPLE.

Activity ID	Person ID	Accuracy	
		Step detector	*Google*
Running machine 1	P1	83,67%	83,67%
	P2	100%	82,22%
	P3	100%	100%
	P4	94,74%	94,74%
	P5	95,24%	88,1%
	P6	95%	96,67%
	P7	98,21%	100%
Running machine 2	P1	100%	91,38%
	P2	94,64%	100%
	P3	64,44%	82,22%
	P4	98,18%	98,18%
	P5	100%	100%
	P6	75,95%	93,67%
	P7	97,47%	93,67%
Bicycle 1	P1	18,52%	96,3%
	P2	81,36%	98,31%
	P3	17,78%	53,33%
	P4	58,18%	98,18%
	P5	92,06%	98,41%
	P6	96,15%	94,23%
	P7	73,33%	96,67%
Bicycle 2	P1	97,22%	97,22%
	P2	97,3%	97,3%
	P3	3,33%	96,67%
	P4	94,74%	98,25%
	P5	98,55%	98,55%
	P6	85,19%	100%
	P7	97,5%	97,5%

B. Experiment with different smartphones

The experiment with different smartphones employed six distinct smartphones in an exercise in which one participant went from walking to running. It evaluated the algorithms in terms of *precision* (2), where TP stands for true positive (motion detected successfully) and FP indicates false positive (mistakenly detected motion), and *recall* (3), also known as sensitivity, where FN means false negative (motions that occurred but were not detected):

$$P = \frac{TP}{TP + FP}.100\%, \quad (2)$$

$$R = \frac{TP}{TP + FN}.100\%. \quad (3)$$

This experiment objective was to evaluate the algorithms assertiveness and sensitivity in different hardware contexts using the metrics precision and recall. The six smartphones used were: (a) Asus - ZE520KL, (b) Asus - ZE551ML, (c) Asus - ZE552KL, (d) Motorola - Moto G3, (e) Motorola - Moto Z2 Play and (f) Samsung - Galaxy J5.

The experiment took place through a single activity in which the tester attached the armlet to his ankle and made

TABLE IV. RESULTS OF THE EXPERIMENT WITH DIFFERENT SMARTPHONES.

Smartphone brand and model	Step Detector					Google				
	TP	*FP*	*FN*	*P*	*R*	*TP*	*FP*	*FN*	*P*	*R*
Asus - ZE520KL	60	0	4	100%	93,75%	64	1	0	98,46%	100%
Asus - ZE551ML	62	2	5	96,87%	92,54%	64	0	3	100%	95,52%
Asus - ZE552KL	58	1	8	98,30%	87,88%	55	1	11	98,21%	83,33%
Motorola - Moto G3	-	-	-	-	-	60	0	2	100%	96,77%
Motorola - Moto Z2 Play	57	0	7	100%	89,06%	48	0	16	100%	75%
Samsung - Galaxy J5	-	-	-	-	-	59	1	1	98,33%	98,33%

the transition from walking to running within a thirty-second period. Table IV presents the experiment results.

C. Discussion

The experiments have shown that Step Detector can detect walking and pedaling movements, especially at higher speeds. However, the native software sensor was absent in some smartphones making it impossible for the algorithm to work.

In the experiment with different people, Google got the best results in most cases. However, in the experiment with different smartphones, Google obtained the worst recall (75%) of the overall test in the Motorola - Moto Z2 Play case.

Analyzing this case's speed over time graph shown by Fig. 3, a phenomenon was observed in which the speed signal moved to below the abscissa axis, which caused several false negatives by Google algorithm since the speed value did not exceed the algorithm's predefined threshold value of 15.

The Google algorithm generally showed the best accuracy results and high precision values, as the worst precision value was 98.21%, obtained in the Asus - ZE552KL case. Besides, this algorithm can be used by any Android smartphone while Step Detector is supported by 95.3% of current Android smartphones [5].

Some of the possible causes of the phenomenon observed in the figure could be (1) problems with the smartphone's accelerometer and (2) integration simplification error.

Due to the best results and the greater robustness obtained by Google algorithm, we observed the opportunity to improve this algorithm in order to satisfy the variations between smartphones and to achieve better accuracy results. Based on this, the Google algorithm was adapted and gave rise to the Castor algorithm. Therefore, Section IV describes the Google algorithm in depth, and Section V presents the adaptations made on it originating the Castor algorithm.

IV. Background - Google Algorithm

Google algorithm leverages the accelerometers to perform motion detection [4]. The algorithm uses the three-dimensional acceleration vectors obtained from the sensor to estimate the current speed and, thus, evaluate if a movement occurred. From these vectors, the average acceleration $\vec{a}_{avg(t_{current})}$ at any time is calculated as follows:

$$\vec{a}_{avg(t_{current})} = \frac{\sum_{x=t_{initial}}^{t_{current}} \vec{a}_{(x)}}{t_{current} - t_{initial}}, \quad (4)$$

where $t_{current}$ is the current time and $\vec{a}_{(t_{current})}$ is the three-dimensional vector of the most recent acceleration

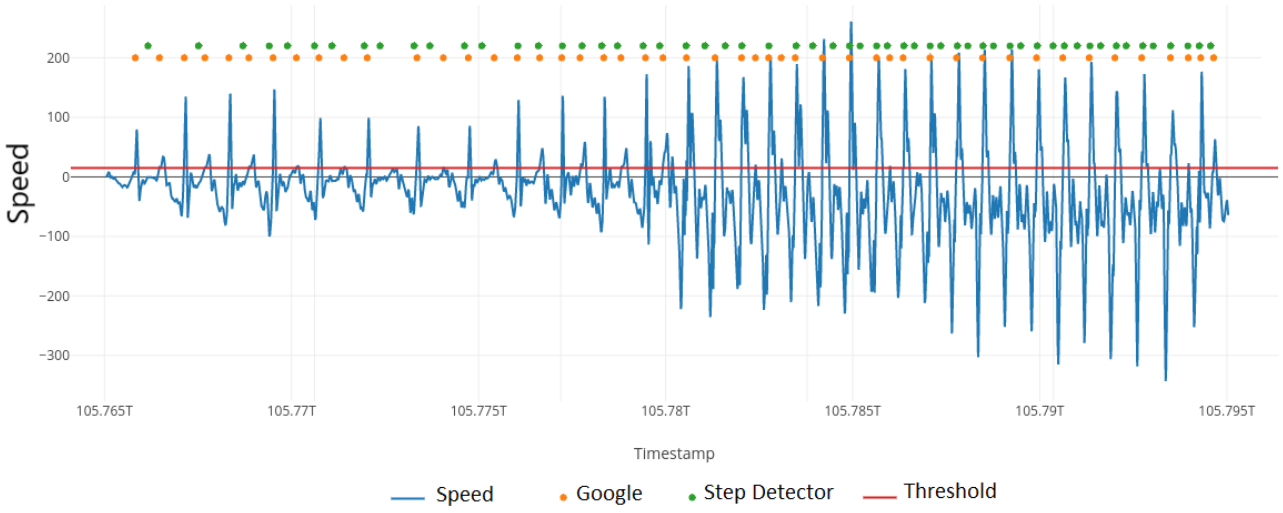

Figure 3. Motorola - Moto Z2 Play smartphone case in the experiment with different smartphones.

provided by the accelerometer. To perform this calculation, the algorithm keeps a sliding window of the last fifty acceleration vectors obtained by the sensor, that is, $t_{current} - t_{initial} \leq 50$.

t is known that the movement acceleration signal acting on the smartphone behaves like a sine wave, that is, its mean approximately equals zero. Therefore, the average acceleration $\vec{a}_{avg(t_{current})}$ roughly represents the vector of gravity $\vec{g}_{(t_{current})}$ that acts on the smartphone, that is,

$$\vec{g}_{(t_{current})} \approx \vec{a}_{avg(t_{current})} = (g_x, g_y, g_z). \quad (5)$$

Thus, it estimates the gravity component acting on the smartphone by normalizing the mean acceleration, which is given by

$$|\vec{g}_{(t_{current})}| = \sqrt{g_x^2 + g_y^2 + g_z^2}. \quad (6)$$

In order to obtain the smartphone acceleration value free of the gravity influence, $\vec{a}_{(t_{current})}$ must be projected on $\vec{g}_{(t_{current})}$ so the gravity vector influence can be excluded. The axis parallel to the gravity vector was called the Z axis, that is, $a_{z(t_{current})}$ is the acceleration component of same direction as the gravity vector's, so that

$$a_{z(t_{current})} = \frac{\vec{g}_{(t_{current})}}{|\vec{g}_{(t_{current})}|} \cdot \vec{a}_{(t_{current})} - |\vec{g}_{(t_{current})}|. \quad (7)$$

This way, the current speed $s_{(t_{current})}$ was calculated as a consequence of the sum of the fifty most recent Z-axis acceleration components as follows:

$$s_{(t_{current})} = \sum_{x=t_{initial}}^{t_{current}} a_{z(x)}. \quad (8)$$

Once the speed scalar is obtained, the Google algorithm counts a motion when the speed $s_{(t_{current})}$ exceeds, in a growing state, the threshold value, a predefined constant in the algorithm whose value is 15.

V. Castor Algorithm

Castor, abbreviation for **Cas**siano Detec**tor**, is an adaptive motion detection algorithm developed in this paper. Based on the detection approach proposed by the Google algorithm, Castor similarly counts a movement when the current speed exceeds the threshold value. However, in Castor, the threshold is not a constant but a variable that adapts to the speed signal over time.

This new approach aims to maintain Google's detection performance, observed in Section III, with the hypothesis of making it adaptable to speed signal displacement around the abscissa axis.

The algorithm recalculates the threshold value at each detection cycle based on the following values: (1) mean maximum speed, (2) mean amplitude and (3) standard deviation, given the $(t_{current} - t_{initial})$ last runs, called time window.

The average maximum speed is the arithmetic mean of the sum of all the maximum speed values obtained in the time window, that is,

$$avgMaxSpeed = \frac{\sum_{x=t_{initial}}^{t_{current}} maxSpeed_{(x)}}{t_{current} - t_{initial}}. \quad (9)$$

Analogously, the mean speed amplitude is the arithmetic mean of the sum of all amplitudes obtained in the time window, which is given by

$$avgAmp = \frac{\sum_{x=t_{initial}}^{t_{current}} amp_{(x)}}{t_{current} - t_{initial}}. \quad (10)$$

Then, the variance of the speed amplitude in the time window and the respective standard deviation was calculated from the following relations:

$$variance = \frac{\sum_{x=t_{initial}}^{t_{current}} (amp_{(x)} - avgAmp)^2}{t_{current} - t_{initial} - 1}, \quad (11)$$

$$\sigma = \sqrt{variance}. \quad (12)$$

The goal of updating the threshold value is to place it at the speed signal center so that all movements can be detected. However, there are some types of physical activities, such as elliptical activities, in which two consecutive speed signal periods have a rather significant amplitude difference.

Based on this motivation, the purpose of calculating the threshold is to make it slightly offset above the speed signal center. In such a way, the subtraction of the maximum speed mean by half amplitude mean is responsible for centering the threshold on the speed signal, that is,

$$threshold = avgMaxSpeed - \frac{avgAmp}{2}. \quad (13)$$

To obtain the slight upward displacement effect, Castor adds half of the standard deviation to the threshold value as follows:

$$threshold = avgMaxSpeed - \frac{avgAmp}{2} + \frac{\boldsymbol{\sigma}}{\mathbf{2}}. \quad (14)$$

This adaptive threshold approach is a candidate solution to the speed signal displacement problem around the abscissa axis. To validate this hypothesis, we evaluated Castor and Google. The following Section presents experiments done in a gym environment.

VI. Experiments with Castor and Google

The evaluation of Castor and Google algorithms comprised four parts, each comparing both algorithms in terms of accuracy, precision, and recall values. The **first part** evaluated the performance of the algorithms in distinct armlet positions in running machine and elliptical exercises, in order to define the smartphone best positioning between the arm and leg spots. The **second part** compared the algorithms during bicycle and abdominal exercises in order to analyze their detection range for various exercises. Later, the **third part** evaluated different speed values in the running machine to define the impact of speed variation on the performance of the algorithms. Finally, in the **fourth part**, experiments were performed on four distinct smartphones to evaluate the robustness of the algorithms on different Android hardware.

A. Part 1: Experiments for armlet position

This first set of experiments evaluated the impact of the armlet position that held the smartphone on the algorithms' performance. Thereof, the positions defined were the arm, next to the biceps region, and the leg, next to the calf region. The equipment in which this armlet position changing is possible are running machine and elliptical. In this way, we carried out four experiments: (A1) running machine with armlet on the arm, (A2) running machine with armlet on the leg, (A3) elliptical with armlet on the arm and (A4) elliptical with armlet on the leg. The smartphone used was the Asus - ZE551ML, each experiment lasted 35 seconds, and the same person performed all of them. Table V shows the results.

TABLE V. ALGORITHMS PERFORMANCE ON ARMLET POSITION EXPERIMENTS.

ID	Algor.	TP	FP	FN	Accuracy	Precision	Recall
A1	Castor	49	1	0	97,96%	98%	100%
	Google	49	0	0	100%	100%	100%
A2	Castor	47	0	1	97,92%	100%	97,92%
	Google	46	0	2	95,83%	100%	95,83%
A3	Castor	77	0	1	98,72%	100%	98,72%
	Google	77	0	1	98,72%	100%	98,72%
A4	Castor	46	10	1	80,85%	82,14%	97,87%
	Google	44	5	3	95,74%	89,8%	93,62%

B. Part 2: Experiments for exercises using bicycle and abdominal

The second part of the evaluation contemplated two exercises: (B1) bicycle, with the armlet located on the leg, and (B2) abdominal, with the armlet located on the arm. Thus, this part conducted two experiments for each type of exercise. The smartphone used was the Asus - ZE551ML, each experiment lasted 35 seconds, and the same person performed both experiments. Table VI exhibits the achieved results.

TABLE VI. ALGORITHMS PERFORMANCE ON BICYCLE AND ABDOMINAL EXPERIMENTS.

ID	Algor.	TP	FP	FN	Accuracy	Precision	Recall
B1	Castor	51	1	0	98,04%	98,08%	100%
	Google	50	1	1	100%	98,04%	98,04%
B2	Castor	49	2	1	98%	96,08%	98%
	Google	0	0	50	0%	-	0%

C. Part 3: Experiments with different speeds

These experiments evaluated the algorithms performance in four running machine speed variations: (C1) slow speed (2.5 km/h), (C2) medium (5 km/h), (C3) fast (7.5 km/h) and (C4) transient speed from slow to fast (2.5 km/h to 7.5 km/h). Thus, the same person performed four experiments. The device used was Asus - ZE551ML and each experiment lasted 35 seconds. Table VII shows the results.

TABLE VII. ALGORITHMS PERFORMANCE FOR DIFFERENT SPEED EXPERIMENTS.

ID	Algor.	TP	FP	FN	Accuracy	Precision	Recall
C1	Castor	44	0	1	97,78%	100%	97,78%
	Google	45	0	0	100%	100%	100%
C2	Castor	63	0	0	100%	100%	100%
	Google	63	0	0	100%	100%	100%
C3	Castor	94	0	1	98,95%	100%	98,95%
	Google	94	0	1	98,95%	100%	98,95%
C4	Castor	75	2	0	97,33%	97,4%	100%
	Google	75	0	0	100%	100%	100%

D. Part 4: Experiments with different smartphones

The last experiments evaluated the algorithms' robustness on different mobile devices. The smartphones used were (D1) Asus - ZE551ML, (D2) Asus - ZE520KL, (D3) Motorola - Moto Z2 Play and (D4) Samsung - Galaxy J5. For each one, a single exercise experiment lasting 35 seconds was performed, and it comprised transitioning from walking to running on flat ground, that is, in an open environment equipment-free. The same person performed all experiments with the armlet on the arm. Table VIII exhibits the achieved results.

TABLE VIII. ALGORITHMS PERFORMANCE ON DIFFERENT SMARTPHONES EXPERIMENTS.

ID	Algor.	TP	FP	FN	Accuracy	Precision	Recall
D1	Castor	79	0	0	100%	100%	100%
	Google	79	0	0	100%	100%	100%
D2	Castor	77	5	1	94,87%	93,9%	98,72%
	Google	78	4	0	94,87%	95,12%	100%
D3	Castor	77	2	0	97,4%	97,47%	100%
	Google	76	1	1	100%	98,7%	98,7%
D4	Castor	76	0	0	100%	100%	100%
	Google	76	0	0	100%	100%	100%

E. Discussion

This section presented fourteen experiments, of which four aimed at supporting the decision of the best armlet

position in running machine and elliptical exercises, two involved bicycle and abdominal exercises, four evaluated different running machine speeds and the last four focused on assessing the performance of the algorithms on different smartphones.

First, in terms of applicability, we could observe that only Castor algorithm complied with the proposed detection of all experiments since Google proved to be inadequate to slow speed exercises, as noted in the abdominal exercise case (B2).

As discussed previously, the motion detection algorithm must meet requirements of accuracy, precision and recall above 95%. Thus, we performed 14 exercises and, for each exercise, those 3 metrics were calculated, so Castor and Google algorithms were evaluated in 42 cases each as a whole.

Google did not meet the requirements in six cases (14.29%): (a) accuracy of 89.8% and (b) recall of 93.62% in the elliptical experiment with the armlet located on the leg (A4) (c) accuracy, (d) precision and (e) recall either null or invalid on the abdominal experiment (B2), and (f) 94.87% accuracy in the Asus-ZE520KL smartphone case (D2).

Castor, on the other hand, did not meet the requirements in four cases (9.52%): (a) accuracy of 80.85% and (b) precision of 82.14% in the elliptical experiment with the armlet located on the leg (A4) (c) accuracy of 94.87% and (d) precision of 93.9% in the Asus-ZE520KL smartphone case (D2).

In terms of accuracy, Castor and Google algorithms obtained valid results in all fourteen experiments (100%). Fig. 4 presents the box diagram of the valid accuracies obtained by the algorithms in the evaluation.

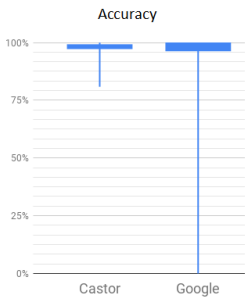

Figure 4. Box diagram of valid accuracies obtained by the algorithms.

The box diagram was generated based on four values: (1) maximum value, represented by the vertical line highest value, (2) minimum value, represented by the vertical line lowest value, (3) third quartile, represented by the box upper value and (4) first quartile, represented by the box lower value. The first quartile is a delimiter number that indicates that 25% of the values from the dataset lie below it, while in the third quartile they are 75%.

Regarding accuracy, Castor obtained valid values in the fourteen experiments (100%) and Google in thirteen (92.86%), since in the abdominal experiment (B2) it did not detect any movement. Fig. 5 shows the box diagram of the valid precisions obtained by the algorithms in the evaluation.

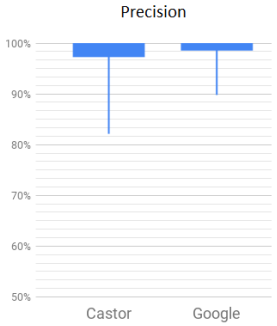

Figure 5. Box diagram of valid precisions obtained by the algorithms.

In terms of recall, Castor and Google obtained fourteen valid values (100%). Fig. 6 shows the box diagram of the valid recall values obtained by the algorithms in the evaluation.

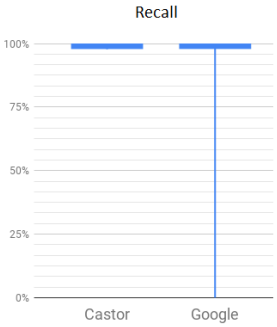

Figure 6. Box diagram of valid recalls obtained by the algorithms.

From the box diagrams of Fig. 4, 5 and 6, we see, at first, the high discrepancy between the minimum and first quartile values of accuracy and recall obtained by Google algorithm, that is, for these cases, the algorithm had a high performance dispersion between the experiments.

For Google, this dispersion was mostly because of its performance in the abdominal experiment (B2), obtaining 0% of accuracy and recall since its threshold value is fixed at 15. Therefore motions with maximum speed below this value are not detected.

Castor, on the other hand, obtained differences between the first quartile and minimum values of accuracy and recall smaller than Google. Thus, Castor exhibits greater stability than Google in terms of (1) proportion of movements detected on movements performed and (2) false negative rate. In terms of accuracy, Google was more stable due to its low false positive rate.

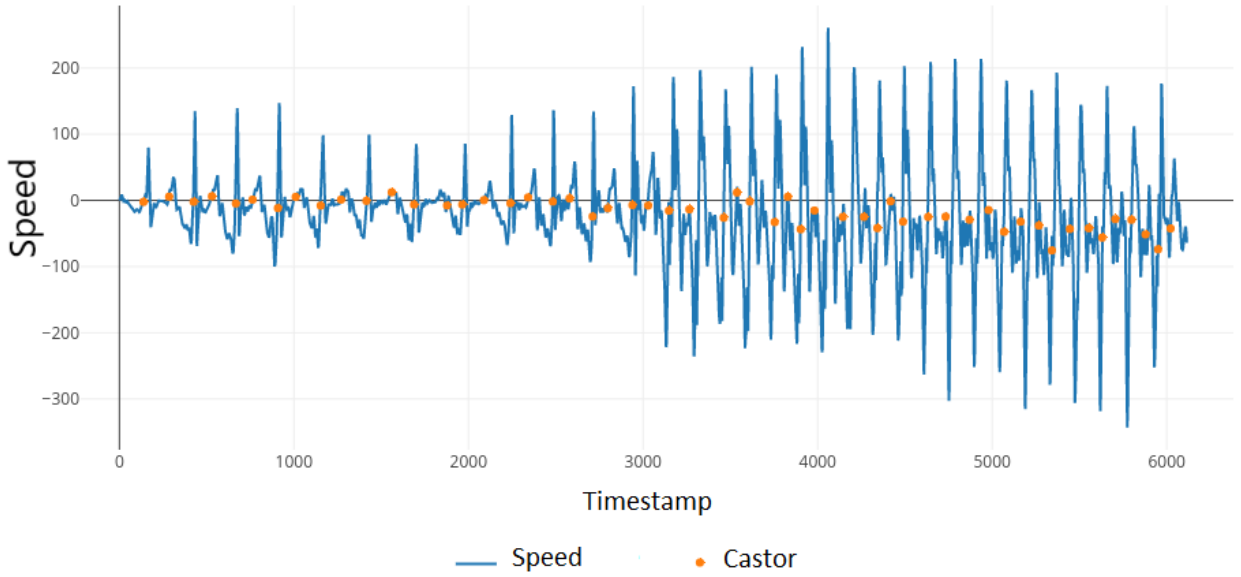

Figure 7. Simulation of the Motorola - Moto Z2 Play smartphone experiment shown in Section III-C using Castor.

On average, Castor obtained the highest accuracy and the lowest false negative rate in the experiments. On the other hand, Google had the lowest false positives rate. The fact that Castor has a false positive rate greater than Google's is consistent with its algorithmic structure because its threshold adapts over time. Consequently, it is more susceptible to noise over the entire speed signal, while Google is only susceptible around its threshold value.

To assess if Castor's adaptive threshold value would be robust enough to overcome situations such as the one we experienced with the Motorola Moto Z2 Play device (described in Section III-C), we conducted one more test simulating how Castor would detect movements if it received the same input from that experiment. Fig. 7 presents the simulation results.

We observed that Castor was adaptable to the phenomenon once its threshold followed the speed signal. Out of the 64 movements performed, Castor successfully detected 54 and did not detect 10. Therefore, Castor obtained accuracy of 100% and recall of 84.37%, that is, accuracy equal to Google's and recall 9.37% *p.p. higher*.

Thus, Castor's hypothesis of a solution to make Google adaptable to different speed waveforms was met, since, in the abdominal experiment (B2), Castor made its threshold feasible so it could detect such exercise's typical movements and, in the simulation above, it presented a smaller number of false negatives than Google.

Therefore, we point that Castor can successfully detect slow, fast and accelerated movements, whereas Google successfully detects only the fast ones but with results as good as those obtained by Castor. Such flexibility from Castor allows the detection in more diverse exercises, which are those that require practitioners to perform slower movement speed, and it allows applicability in other initiatives, such as climbing activities.

Finally, the tool for motion detection algorithms analysis proved to be versatile and practical for developing and evaluating algorithms, since it enabled the execution of all experiments carried out on this paper.

Section VII displays our final remarks, summarizing the results and presenting some suggestions of future work.

VII. Conclusions and Future Work

This paper aimed at evaluating and proposing motion detection algorithms in Android smartphones that could detect indoor movements such as those performed during physical activities. Our methodology consisted of (a) the development of an analysis and evaluation tool for motion detection algorithms, (b) the implementation of Google and Step Detector algorithms in the tool, followed by their evaluation, (c) the development of the motion detection algorithm Castor and (d) the comparative evaluation of the algorithms Castor and Google.

Thus, this project generated a tool for motion detection algorithms analysis and the Castor algorithm. This algorithm represents an alternative solution to the motion detection problem, and it provides an accessible algorithm for various purposes. Moreover, the developed tool contributes to the academic environment by providing an extensible instrument for motion detection algorithms implementation and analysis.

This tool was a facilitator in terms of development, reproduction, and testing of motion detection algorithms. It allowed the reproduction of Google and Step Detector algo-

rithms, the subsequent implementation of Castor algorithm and the evaluation of Google and Castor simultaneously in an effective and versatile way.

The experiments performed on Castor algorithm presented promising results. The algorithm proved to be adaptable to detection adversities and had average accuracy, precision and recall results above 95%. Also, Castor exhibits better results when the armlet is positioned on the arm instead of the leg in walking and elliptical exercises, it can be implemented on all Android smartphones [5], and it can also detect low-speed motion.

However, the evaluation showed that noise could impact Castor performance. In this way, Castor can be improved with a filter that prevents motion detection in really close timestamps. Therefore, as future works, it is suggested to add to Castor (1) a filter that performs this function based on the sensor sampling period and (2) the concept of variable sliding window, in which instead of always using the 50 vectors of acceleration to estimate speed, to make this number adaptable so it represents complete wave periods more accurately.

REFERENCES

[1] M. Hassan, M. Uddin, A. Mohamed and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning", Future Generation Computer Systems, vol. 81, pp. 307-313, 2018. ISSN 0167-739X.

[2] C. Strohrmann, J. Seiter, Y. Llorca, and G. Trster, "Can smartphones help with running technique?", Procedia Computer Science, vol. 19, pp. 902-907, 2013. ISSN 1877-0509.

[3] R. San-Segundo, J. Lorenzo-Trueba, B. Martnez-Gonzlez, J. Pardo, "Segmenting human activities based on hmms using smartphone inertial sensors", Pervasive and Mobile Computing, vol. 30, pp. 84-96, 2016. ISSN 1574-1192.

[4] Google, "Simple Pedometer", *GitHub*, 2019. [Online]. Available: https://github.com/google/simple-pedometer. [Accessed: 22-Mar-2019].

[5] Android, "Monitoring Sensor Events", *Android*, 2019. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors. [Accessed: 22-Mar-2019].

[6] T. Costa, I. Silva, G. Barbosa, F. Coutinho, "An Architecture for Using Smartphones as Interfaces for Computer Games". Proceedings of SBGames 2018, pp. 611-614, 2018. ISSN 2179-225.