Examining Approaches for Mobility Detection Through Smartphone Sensors

Thomas Tregel[1(✉)], Andreas Gilbert[2], Robert Konrad[1], Petra Schäfer[2], and Stefan Göbel[1]

[1] TU Darmstadt, Darmstadt, Germany
{thomas.tregel, robert.konrad, stefan.goebel}@kom.tu-darmstadt.de
[2] Frankfurt University of Applied Sciences, Frankfurt, Germany
{andreas.gilbert, petra.schaefer}@fbl.fra-uas.de

Abstract. The ubiquity of smartphones with integrated positioning systems, and multiple sensors for movement detection made it possible to develop context-sensitive applications for both productivity and entertainment. Location-based games like Ingress or Pokémon Go have demonstrated the public interest in this genre of mobile-only games – games that are exclusively available for mobile devices due to their sensor integration. For these games mobility is a key component, which defines and influences the game's flow directly.

In this paper we compare different approaches and available frameworks for mobility detection and examine the frameworks' performances in a scenario-based evaluation.

Based on our finding we present our own approach to differentiate between different modes of public transport and other common modes of movement like walking, running or riding a bicycle. Our approach already reaches an accuracy of 87% with a small sample size.

Keywords: Mobility · Mobility detection · Machine learning

1 Introduction

The purpose of recognized types of movement is diverse. Depending on the application, different levels of detail are required in the recognition. If the user is to be animated solely for more exercise in the fresh air and thereby improve his health, the recognition of different types of mobility such as walking or running, cycling and skating is needed. Distinctions regarding different types of vehicles are not necessary. If, however, it is possible to differentiate between different emission levels, additional vehicles must be distinguished from each other. So there are in addition to the usual cars with combustion engine electric cars or cars with hybrid drive. There are also motorbikes and public transport such as trams and buses. In order for the respective level of pollutant emissions to be approximately assigned, a precise distinction of the vehicles is required. Although there are two implementations for mobility detection for Android in the form of an API and a framework, the distinction with these

S. Göbel et al. (Eds.): JCSG 2018, LNCS 11243, pp. 217–228, 2018.
https://doi.org/10.1007/978-3-030-02762-9_22

implementations is not sufficiently detailed possible. These allow at most a distinction between non-existent mobility, walk, run, drive and ride a bicycle.

Our goal in this paper is to show possibilities and limitations of mobility detection with Android smartphones in both literature and open available frameworks. To this end, existing approaches are first examined and compared, and relevant components from comparable approaches are analysed. Subsequently, existing implementations for Android are examined, implemented and compared with each other. In addition to the accuracy of the recognition, particular attention should be paid to resource consumption.

From the results obtained, a concept is created and implemented that can more quickly differentiate more modes of transport. This implementation is checked for accuracy in detail. The goal is to be able to distinguish vehicles from each other and to operate with greater accuracy than in existing implementations.

In the context of Location-based games or mobile-only games this system enables the use personalization and adaptation methods with respect to the user's mobility behaviour.

2 Mobility Detection Using Smartphones

The following approaches have been developed in the current decade. At the beginning of the decade, Nokia was the most popular operating system for smartphones with Symbian operating system versions, but Android and iOS also became more relevant in the marketplace than at the end of the last decade [1]. The following approaches are in chronological order.

2.1 "Movement Recognition Using the Accelerometer in Smartphones"

The approach of Sian Lun Lau and K. David studied below was designed in 2010 [2]. As hardware the Nokia N95 8 GB with the operating system Symbian S60 3rd edition FP1 was used. Built into this is an accelerometer that measures the acceleration on three different axes in the x, y and z directions.

The data was first collected using a Python script for the appropriate operating system. Another script combined the collected data to determine the mode of travel. The chosen modes of movement were running, standing, sitting and running down and running up a staircase. As a condition to maintain the Accelerometer data, it was determined that the smartphone is in your pocket. Two different situations were examined. The first situation (S1) stated that the smartphone is in a fixed and predetermined position in the right pocket, while in the second situation (S2) the smartphone did not need to have a fixed position in any trouser pocket. Sampling rates of 60 Hz to 70 Hz have been achieved with the Nokia N95 8 GB by obtaining the accelerometer data. However, in this approach 32 Hz was considered sufficient for mobility detection, so higher sampling rates were not investigated. Sampling rates of 5 Hz, 10 Hz, 20 Hz and 40 Hz were compared with an accuracy of more than 90% already achieved for 5 Hz to 20 Hz. The extracted features were the average $\bar{x}$ and the standard deviation σ of both the accelerometer raw data and the Fast Fourier Transform (FFT) components

of the data. There were three combinations of features for this. The first combination (C1) included the average and standard deviation of the individual and aggregate axes of the accelerometer. The second combination (C2) included the average and standard deviation of the FFT components of each axis and the combination of axes. The third combination (C3) contained all features for each axis and for the combination of all axes.

For extracting the features, the sliding window approach was used. Compared window sizes were 5, 10, 20, 40, and 80 samples per window, so along with the sampling rates, these window sizes are equivalent to 0.5, 1, 2, and 4 s between getting new data. The windows also had an overlap of 50%. The classification was based on the rule-based learner JRip and the Supervised classifiers Decision Tree, Bayesian Network, Naive Bayes, K Nearest Neighbor (k-NN) and Support Vector Machine (SVM). Sequential Minimal Optimization (SMO) was used for the comparison. It was evaluated with 10-fold Cross Validation. Weka was used for the classification. The accuracy of the results was determined by the percentage of correctly classified mobility types on the total number of types of mobility occurring.

With the combinations of the classifiers C1 and C3 the best results were achieved with 99.27% for S1 and 96.59% for S2. In addition, higher accuracy was achieved if all axes were combined rather than considered individually. In addition, the classifier KNN was the most accurate. The best combination of samples and window sizes was a sampling rate of 20 Hz with 80 samples per window with the classifiers KNN, Bayesian Networks, Naive Bayes and JRip, as well as 10 Hz with 40 samples and 40 Hz with 80 samples. The highest accuracy reached was 99.27%. In addition, the results were tested against a new data set with S1. Here an accuracy of 91.95% was achieved, with levels achieved a lower accuracy than the other modes of mobility. Other modes of transport had an accuracy of almost 100%. Here the combination of C1 with KNN and Bayesian Networks with 10 Hz and 20 Hz as sampling rate and a window length of four seconds gave the highest accuracy.

2.2 "Activity Detection on Smartphones"

The following examines a Wirtl and Nickel approach from 2011, with which various activities can be distinguished directly on the smartphone [3].

Two smartphones with Android operating system were used. These include the HTC G1 from 2008 with Android 1.6 and the Motorola Defy from 2010 with Android 2.1. The HTC G1 averaged a sampling rate of 43 Hz, and the Motorola Defy one of 137 Hz. The accelerometer was used for detection. The devices should be horizontally attached to the belt in this approach, while the sensor data used were generated on a flat and straight path.

There were six classifying features, which were also considered separately from the sensor axis. To address the problem that an exact sample rate cannot be specified with Android, the data was interpolated before the features were extracted. The interpolation was achieved on the one hand through the assumption of the nearer value and on the other hand through a linear interpolation. The six characteristics were the time between extreme values TBP, the mean AVG, the standard deviation STD, the average absolute difference, the average resulting acceleration AAD and the class division BIN. The time

between the extreme values is calculated as the absolute difference between the minima i and maxima A, which in turn are recorded by a change from slopes to positive or negative values.

The average difference is calculated using $aad = \frac{1}{n}\sum_{i=1}^{n} |a_i - \bar{a}|$ and the resulting average acceleration using $ara = \frac{1}{n}\sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2 + z_i^2}$.

Classification is the division of the span between the maximum and minimum into ten equally sized classes and the subsequent counting of the values that can be assigned to an axis by a class. First, a one-way test was performed on the computer with one-third of the data. This related to the classification algorithm, the features, the interpolated sample rate, the duration of the data acquisition and the type of interpolation. In this case, raw data was also compared to preprocessed data in order to be surer of the effect of preprocessing. The 18 algorithms available at the time of this approach in Weka were run with all feature combinations and sample rates of 50 Hz, 100 Hz and 150 Hz with a data acquisition duration of 2 s, 5 s, 10 s and 15 s and the two interpolation data. In addition to classifying up-to-date data from the Accelerometer, the training of the Classifier has been implemented on the smartphone to optionally use newly acquired data to improve the classifier.

The activities were classified with Weka, maintaining the default parameters. There were 151 records, of which 51 were intended for fast walking, 49 for normal walking and 51 for slow walking. The data was collected from 51 subjects who were walking on level ground with the HTC G1 for about 17 s without any turns. The REPTree algorithm achieved the best result when using the test data with a data collection time of 10 s. Thus, this combination was used for implementation on smartphones.

The classification took 0.29 s with the Motorola Defy and 0.79 s with the HTC G1. The former detected 80% of the correct mode of mobility in a test by 20 subjects and 75% with the latter.

2.3 "Accelerometer-Based Transportation Mode Detection on Smartphones"

In the approach of Hemminki et al. [4] investigates techniques for discriminating modes of transport travel using smart phone sensors on Android devices.

The GPS receiver is not used in this approach, as it is efficient with the existing signal, but consumes a lot of battery and is dependent on uninterrupted view to the satellites. This can be a problem not only in buildings, but also on a route through subway stations, or even in vehicles where the user sits too far away from a window. Another problem with using the GPS receiver is only mediocre accuracy, should many different vehicle rides be distinguished. Instead, the accelerometer is mainly used for this approach because it does not have the aforementioned problems. Because it is consumed with this less battery, there is no dependence on external information sources such as satellites and it was a particularly fast and detailed query possible. For more accurate results, gravity is estimated. As a result, the horizontal speed can be determined more accurately, which is to be used to distinguish start-up and braking sequences. For the estimation an own algorithm was presented, which in contrast to the simple formation of the average is more robust against sudden changes of the

orientation of the smartphone. These sequences are different from means of mobility to means of transportation. In addition, new features called 'peak features' are introduced, which are intended to model patterns of acceleration and deceleration and to increase the performance of recognition. The process of mobility detection is divided into four different classifiers in this approach. First of all, the 'Kinematic Motion' Classifier attempts to detect the general mode of mobility. Depending on the outcome of this classifier, it will be passed to one of the three more specific classifiers for 'Walking', 'Stationary' or 'Motorized'. With "Stationary" is to be distinguished whether the user is in a vehicle or in a fixed location. If the former is the case, it is forwarded to the classifier "Motorized", with which a distinction from the means of transportation bus, train, tram, metro and car is provided. Between modes of travel in vehicles, a segment is assumed on foot, in which the user runs to the next means of transportation. To increase the accuracy, no frames are used for the classification of the raw data, but segments, which are formed depending on the types of mobility. The raw data is first revised with a low-pass filter to reduce noise. Then the data is aggregated with a sliding window approach with 50% overlap and a length of 1.2 s. The length was chosen to be able to react quickly to a change of mobility types and to distinguish them. Afterwards the calculated gravity is calculated with the accelerometer data and the result uses the integral as the current speed. The algorithm is an improvement to the simpler approach of Mizell et al. thought.

Preprocessing is followed by extracting features. This approach extracts frame-based features, peak-based features, and segment-based features, each applied to the appropriate situations. Frame-based features are designed to differentiate types of mobility on the basis of higher frequencies of the data, as they occur when walking on foot. Here, 27 features were calculated in the vertical and horizontal directions by, for example, average $\bar{x}$, variance, double integral, entropy, and sum of FFT coefficients. Peak-based features should characterize lower frequencies for mobility detection by extracting regions with strong changes through a stream-based algorithm. Segment-based features are used to characterize patterns of acceleration and deceleration. It identifies the frequency of accelerations and brakes, the frequency and duration of intermediate pauses in mobility, and the variance of individual peak-based features. Subsequently, the data is used for classification. AdaBoost is used to improve the learning algorithms. Decision trees with a depth of two were used for the Kinematic Motion classifier. Also in the Motorized classifier AdaBoost is used along with decision trees with a depth of two.

2.4 "Activity Recognition Using Smartphone Sensors"

The approach described below was developed in 2013 by Anjum and Ilyas [5].

It was implemented on an Android smartphone, which in this case was a Samsung Galaxy Y with Android 2.3.3. No exact orientation of the smartphone was given.

The activities 'walking', 'running', 'climbing stairs', 'going down stairs', 'driving', 'cycling' and 'being inactive' should be differentiated. For detection, a classifier with the best found configurations is developed. This classifier was implemented in an Android application that performs real-time activity detection using this classifier. Initially, a dataset with 510 activity shots was created, for which an Android

application was written that enabled data collection. In this application, additional user information such as size, age, gender and the most used smartphone position were queried. The recorded sensor data was divided by a comma into a file specially written for this activity. The written data consisted of the three axes for the accelerometer and for the gyroscope, and the GPS data based on longitude, latitude and speed. In the Android version used in this approach, the sampling rates in 'Normal' with 5 Hz, 'UI' with 15 Hz, 'Game' with 50 Hz and 'Fastest' with platform-dependent 50 Hz to 100 Hz were selected. In this case, the 'UI' mode was used because a Nyquist rate of 15–16 Hz would allow a maximum signal frequency of 8 Hz to distinguish it from human activity. To compensate for the dependence on the orientation of the smartphone, the data was rotated to selected axes d1, d2 and d3.

The data was collected by 10 different people aged 12 to 25 years. The data from four of these people were used to train the classifier while the remaining data was used for testing.

Existing data was divided into ten different datasets using 10-fold Cross Validation. Different classifiers were trained, with the most performant, judged by iterating over all ten sets of data with the test set, used for the implementation. The examined classifiers used the algorithms C4.5, Naive Bayes, KNN and SVM. The classifier with method C4.5 had the best performance with a true positive rate of 95.2%, a false positive rate of 1.1%, a precision of 94.4% and a recall rate of 94.2%.

2.5 Comparison

In the following assessment of the approaches, a comparison of the existing approaches as well as relevant aspects of other approaches will be made. Based on this comparison, the approaches for their usability are evaluated for our own approach (Table 1).

The above table compares existing approaches in their significant points. Significant for the assessment were the hardware used, the recognizable mobility types, the sampling rates, the window sizes in seconds, and the overlapping of the windows in which Weka was used, as well as the algorithms examined for the classification. The highlighted entries indicate favoritism by the authors of the approaches, which were assessed by evaluations.

In the table it can be seen that there is no mode of mobility, which is recognizable from all examined approaches. In addition, specifications for data entry are made in all approaches, which are differentially restrictive depending on the approach. The approach described in [2] was to carry the smartphone in your pocket. In addition, two different situations were examined, which specified the orientation and the side of the smartphone in one case and, in the other case, allowed any orientation and side. As a result of these restrictive requirements, an accuracy of almost 100% was achieved, in particular with an exact specification of the position of the smartphones in the trouser pocket. Even with a rotation of the device or a change of the side of the trouser pocket, the accuracy dropped by about three percent from 99.27% to 96.59%. In the approach in [3] the devices were attached vertically to the belt and a flat and straight path was chosen, while in [4] the subjects were given whether the smartphone should be in the bag, trouser pocket or jacket pocket. In the approach described in [5], the data was collected with an indication of how the user transports the smartphone in most cases.

Table 1. Comparison of existing approaches in related work.

Approach	Hardware	Movement types	Sampling rates	Window size	Overlap	Features	Classifier
(1)	Nokia N95 with Symbian S60	Walking, Standing, Sitting, Stairs up, Stairs down	5 Hz **10 Hz** **20 Hz** 40 Hz	0.5 s 1 s 2 s **4 s**	50%	$\bar{x}$, σ of raw data, FFT	Decision Tree, BN, NB, KNN, SVM + 10-Fold Cross Validation, JRip, SMO
(2)	HTC Android 1.6 and 2.1	Fast, normal, slow walking	50 Hz 100 Hz 150 Hz	2 s 5 s **10 s** 15 s	–	T_e, $\bar{x}$, σ, average absolute difference, average resulting acceleration	**REPTree** & all other Weka classifiers
(3)	Nexus S, Galaxy S2 & S3	Standing, Running, Car, Bus, Tram, Train, Metro	60 Hz, 100 Hz	1.2 s	50%	Lowpass 90% & own Frame-/Peak-/Segment-based features	AdaBoost
(4)	Samsung Galaxy Y with Android 2.3.3	Walking, Running, Stairs up, Stairs down, Bike, Vehicle, None	15 Hz	5 s	–	$\bar{x}$, σ, Acceleration-magnitude, FFT	**C4.5 Decision Tree**, NB, KNN, SVM + 10-Fold Cross Validation

Here, for example, a trouser pocket, shirt pocket, handbag and hand were distinguished. This approach also uses user data to include size, weight, and gender. In this approach, the accuracy was not nearly 100%, but decreased to 94% −95%.

The sampling rates are also very different. The choice was justified in the approach in [2] and the approach in [5]. In the former, the rates of 10 Hz and 20 Hz are based on the evaluation results, while in the latter; the 15 Hz is due to the Nyquist frequency, which allows a maximum recoverable signal frequency of 8 Hz. These 8 Hz were considered sufficient to detect human mobility. Also the window sizes vary depending on the approach. In the approach described in [3], a duration of ten seconds was considered best after evaluation, while in [2] four seconds were chosen as a result of evaluation. In half of the approaches, an overlap of the windows was mentioned, which in both cases had a size of 50%. Also half of the approaches used for coaching and using the classifier Weka. For this purpose, different features were extracted from approach to approach. The use of raw data and the use of Fourier transforms was also compared in [2]. The result of this comparison was that it was more accurately classified with raw data. Most commonly, the mean and standard deviation were used as features, with additional features defined in [3]. In [4], the noise in the data was reduced with a low-pass filter with 90% energy. As versatile as the selected features are the

classifiers. For this comparison, the following table with classifiers is also included in further comparable approaches.

3 Existing Frameworks

In comparison to the previously presented approaches the following two frameworks can be directly integrated and evaluated on mobile devices. In contrast to the previous approaches the decision-making process is not described. We aim to compare them regarding their prediction accuracy and the time required to identify and switch towards the real type of movement.

3.1 Awareness API

Google introduced the Awareness API at the Google I/O 2016 conference [6]. With this API the context recognition has to be implemented in a resource-saving way. There are seven different types of contexts to distinguish. These seven types are local time, location and location, headset state, weather, mobility, and nearby interesting places called beacons. The distinct modes of mobility are *vehicle*, *bi*cycle, standing, *running*, *walking*, *without movement* and *unknown*. A detailed distinction between vehicles does not take place.

The Awareness API consists of two different APIs. The Snapshot API can be used to make a request about the user's current context. The Fence API, on the other hand, defines so-called fences for describing a context to be observed. If the user enters such a fence, the app will be notified without having to open it. For continuous detection of mobility types, notification of a change makes the Fence API more suitable than the Snapshot API. However, if you only need to know the mode of travel at that particular time at certain times, the Snapshot API can also be used to actively run a query.

3.2 Neura SDK

The NeuraLabs Neura SDK [7] provides comprehensive context recognition for both iOS and Android. The time of origin can be assumed according to the code on Github5 at the end of 2016. In addition other contextual information, the user's mode of travel may determine whether the user has just started or stopped walking, running, or driving a vehicle. Thus, it is the modes of travel, on foot 'and' drive vehicle 'distinguishable. In addition, after one hour and two hours, a notification can be received that the user has not moved during this period. These types of mobility can be used when including the Neura SDK named *userStartedRunning*, *userFinishedRunning*, *userStartedWalking*, *userFinishedWalking*, *userStartedTransitByWalking*, *userFinishedTransitByWalking*, *userIsIdleFor1Hour*, *userIsIdleFor2Hours*, *userStartedDriving* and *userFinished Driving*.

3.3 Comparison

In a 40-minute-long test scenario we used the presented frameworks in parallel and logged the identified mobility types. Figure 1 shows that mobility detection by no means works perfectly.

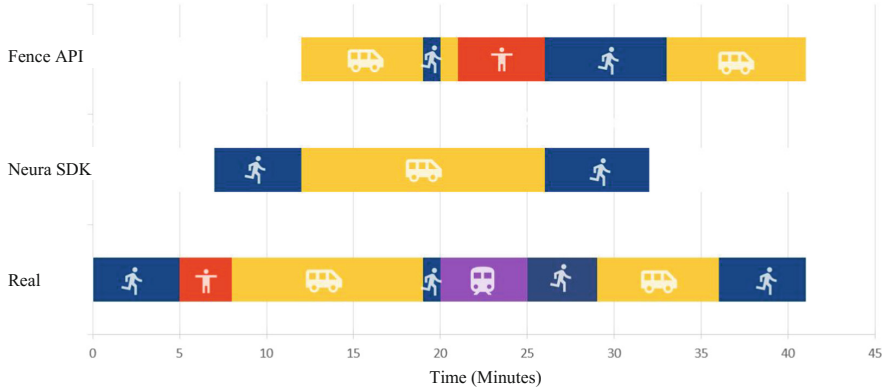

Fig. 1. Comparison of a test scenario over 45 min using two available Android frameworks.

Waiting times cannot be detected by the Neura SDK, so they are ignored. This is particularly evident in the longer journey in a vehicle, which is detected instead of the switch to the tram. The Fence API more reliably detects the changeover between modes of transport. The part of the tram ride in which the tram has been waiting was considered missing. In addition, the detection of a change of mobility takes place only after a time delay. Based on the test results, the Neura SDK has a minimum delay of five minutes and a maximum delay of seven minutes. For the Fence API, the minimum delay is one minute, while the maximum delay is four minutes.

In summary, it can be seen that the Awareness API for mobility detection offers more advantages due to more detailed distinctions, less extensive implementation, less impact on application design, lower resource consumption, and less delay in recognition than the Neura SDK. In addition, if only an active query of the current mode of transport is required, the Snapshot API would be preferred to the Fence API, in particular due to less implementation effort. But even the Awareness API does not provide a sufficient distinction for the mobility detection, since in particular no vehicles can be differentiated among themselves.

4 Consolidated Approach

For our own approach the goal is to be able to distinguish between the different modes of transportation like bike, car, bus or tram.

The identification of the modes of movement in this concept should refer exclusively to the data of these sensors. The use of GPS data was reported by (3) is

considered inappropriate because the collection of data places restrictive conditions on the environment. Thus, the user may not be in a subway or train station or other building, and should sit as close to the window as possible in a car for a clear view of the satellites delivering the GPS data.

In order to differentiate between these modes of transportation we choose similar features as these presented in the related work, with all of them being calculated in a parameterized time window with a default value of five seconds:

- Arithmetic mean: for each axis the mean values are calculated
- Standard deviation: for each axis the standard deviation is calculated
- Time between extreme values: the time between minima and maxima is determined separated for each axis.
- Standard deviation of time between extreme values: instead of using the average time between extreme values, the standard deviation of time differences is used. The extracted information represents how irregular the extreme values are.
- Number of extreme values: Analogue to the calculation of time between extreme values and their standard deviation the number of extreme values is extracted in the given time window.

In a WEKA-based [8] evaluation using 722 collected samples spread across seven categories with the main focus on vehicular modes of transportation we achieved an accuracy of 87% using 10-fold cross validation on both a random forest and an IBk algorithm as shown in Table 2. While these algorithms have comparable accuracy the mean absolute error for the k-nearest neighbors-based algorithm with IBk is substantially smaller.

Table 2. Algorithm performance using 10-fold cross validation

Algorithm	Accuracy	Mean absolute error	Root mean squared error
JRip (rule-based)	79.499%	0.0713	0.217
J48 (decision-tree)	85.133%	0.0498	0.1994
IBk (k-NN)	87.167%	0.0392	0.1904
Random forest	86.698%	0.0735	0.1721

Going into detail it can be seen from the classifiers' confusion matrix in Table 3 that a differentiation, in particular between car and bus, is not exact. With 10-fold cross evaluation, about 8% of the car's records were classified as a bus, while about 22% of the bus's records were classified as cars. In the case of rail, the misclassification as a car or bus is less than about 6% less. A misclassification of the vehicles as a lack of mobility has also occurred, but more misclassifications were expected here as busses, trains or cars often stop at traffic lights or intersections. The reason for this may be that in this time the engine is on and the vibration of the vehicle can be detected by the sensors, so that a difference to the lack of mobility can be detected here. The only class without misclassification is the one to classify standing still.

Table 3. Confusion matrix for the best random forest classifier

	Bike	Car	Bus	Tram	On foot	Still	Unknown
Bike	121	2	1	0	0	0	0
Car	1	259	21	0	0	0	0
Bus	0	41	144	0	0	1	1
Tram	0	3	2	23	1	0	2
On foot	0	4	0	3	19	0	0
Still	0	0	0	0	0	66	0
Unknown	0	2	0	0	0	2	5

5 Conclusion

In this paper we examined and compared six approaches for mobility detection. All four scientific approaches present different features and parameters that promise good detection results. However, because the frameworks are not openly available for testing they cannot be evaluated in a comparative study. For the two openly available frameworks our evaluation has shown problems regarding live detection of mobility change due to minute-long delays in detection. Additionally, neither of the presented systems is able to distinguish between different vehicle-based modes of transportation like car, bus, tram or train.

We presented an approach designed to distinguish between these vehicle-based modes of transportation in addition to those covered by the existing approaches.

Our approach can be improved in accuracy by collecting more sensor data over a longer period of time. Another aspect that can be implemented with today's common resources is the comparison of the isolated classifier presented in this work with a combination with Markov chains or automata. This would allow the start and stop sequences, which are significant for buses and trams, to be included in the classification.

The integration into a mobility-based mobile game is the next step to follow in order to provide an appealing game experience and furthermore to collect additional data to improve prediction accuracy.

Acknowledgment. The research presented in this paper was partially funded by the LOEWE initiative (Hessen, Germany) within the research project "Infrastruktur – Design – Gesellschaft" as project mo.de.

References

1. Schonfeld, E.: Mobile OS 2009 market share (2017). https://techcrunch.com/2010/02/23/smartphone-iphone-sales-2009-gartner/. Accessed 15 June 2018
2. Lau, S.L., David, K.: Movement recognition using the accelerometer in smartphones. In: Future Network and Mobile Summit (2010)
3. Wirtl, T., Nickel, C.: Aktivitätserkennung auf Smartphones. In: International Conference of the Biometrics Special Interest Group (2011)
4. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (2013)
5. Anjum, A., Ilyas, M.U.: Activity recognition using smartphone sensors. In: IEEE 10th Consumer Communications and Networking Conference (2013)
6. Google: Google Awareness API (2016). https://developers.google.com/awareness/. Accessed 15 June 2018
7. Neura: Neura SDK (2017). https://dev.theneura.com/. Accessed 15 June 2018
8. Holmes, G., Donkin, A., Witten, I.H.: Weka: a machine learning workbench. In: Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems (1994)