INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Deteção do modo de transporte para o cuidado de idosos

NUNO FILIPE DE OLIVEIRA CARDOSO
Outubro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Transport Mode Detection for Elder Care Management

Nuno Filipe Oliveira Cardoso

Dissertation for obtaining a Master's Degree in Computer Engineering, Specialization in Computer systems

Advisor: Nuno Pereira

Porto, October 2016

Resumo

Com o aumento da esperança média de vida, é percetível o envelhecimento da população e, consequentemente, uma diminuição das condições sociais e económicas para o cuidado diário de idosos. Tendo isto em conta, mais atenção está a ser dada aos sistemas de cuidados remotos para ajudar os pacientes a cuidar de si mesmos, diminuindo a necessidade de cuidados de saúde convencional. A atividade física humana é uma característica importante de alguns dos novos sistemas de prevenção e monitorização, portanto parâmetros relacionados com esta temática têm recebido um interesse crescente por parte dos profissionais de saúde, como, por exemplo, fisioterapeutas, cuidadores de idosos e nutricionistas.

Este projeto propõe-se a detetar vários modos de transporte, usando para isso informação proveniente dos vários sensores disponíveis num smartphone comum, adicionando ainda métodos que se aproveitam das informações circundantes, tais como pontos de acesso Wi-Fi de forma a melhorar a precisão global do sistema.

Ao monitorizar os modos mais comuns de transporte utilizados pela população sénior, como caminhar, autocarro, metro, comboio e carro, será possível auxiliar o trabalho dos profissionais de saúde, dando-lhes conhecimento sobre os hábitos de atividade física do utilizador, permitindo ainda perceber se o idoso tem uma vida sedentária, se existem mudanças nos padrões de movimento diário, fornecer indicação sobre a falta de atividades sociais, se é suficientemente independente na sua vida quotidiana, ou se está a fazer exercício suficiente.

Esta tese descreve o modo como vários classificadores referidos no estado da arte para a deteção do modo de transporte foram treinados e avaliados. O classificador, Decision Tree, demonstrou melhor performance na deteção do modo de transporte e foi implementado de forma a poder ser utilizado por outras aplicações Android. Para isso foi implementado um sistema como prova de conceito para demonstrar o funcionamento do classificador de transporte. O classificador foi incorporado numa aplicação para o cuidado de idosos, que pode ser usado por profissionais de saúde e afins para monitorizar os padrões de atividade diária das pessoas idosas.

O classificador foi criado usando mais de 24 horas de dados de transporte de um grupo de 15 indivíduos e pode, teoricamente, atingir mais de 96,1% de precisão. No entanto, uma validação do mundo real do sistema implementado obteve 88,97% de precisão.

Em alinhamento com a investigação feita nesta tese, um artigo foi submetido e aceite para conferência. O anexo C apresenta o trabalho aceite na octogésima Conferência Internacional IEEE Healthcom, realizada em Munique, em setembro de 2016.

Palavras-chave: Reconhecimento de atividades humanas, Processamento de sinal, Extração de características, Machine Learning, Cuidado de idosos, Deteção de transportes

Abstract

With an increase in average life expectancy, the aging of the population is discernible and consequently a reduction in social and economic conditions for elderly daily care. Taking this into account, more attention is being given to remote care systems to assist the patients and help them take care of themselves, lowering the conventional health care necessity. Human physical activity monitoring is an important feature of such systems, and has received an increasing interest from health-related professionals, such as physiotherapists, elders' caregivers and nutritionists.

This project's main purposes are the detection of transport modes (walking, bus, car, metro, ...) using data from different sensors available on a common smartphone, and the addition of methods that take advantage of the surrounding environment, such as Wi-Fi Access Points, in order to improve the global accuracy of the system. By monitoring the most common modes of transport used by this population it is possible to support the work of caregivers, giving them knowledge about the user's activity habits, perceive whether the elder is having a sedentary life, whether there are changes in the daily movement patterns, and more.

This thesis describes how various classifiers referred to in the state of the art for the detection of the transport mode have been trained and assessed. The classifier, Decision Tree, showed better performance in the transport mode detection and it was implemented so that it could be used by other Android applications. In order to do it, a system was implemented as proof of concept to demonstrate the operation of the transport classifier. The classifier was incorporated in an application for elderly care, which can be used by health professionals and alike to monitor the daily activity patterns of the elderly.

The classifier was created using more than 24 hours of transportation data from a group of 15 individuals and Weka classification shows it can achieve over 96.1% overall accuracy. However, a real world validation of the implemented system obtained 88.97% accuracy.

In alignment with the investigation done in this thesis, a paper was submitted and accepted for conference publication at the IEEE HealthCom. The Appendix C presents the accepted paper.

Keywords: Human Activity Recognition, Signal Processing, Feature Extraction, Machine Learning, Elderly care, Transport Detection

Agradecimentos

Gostaria de agradecer à *Associação Fraunhofer Portugal Research* pela oportunidade de me ajudar a crescer como profissional ao longo dos últimos meses. Um agradecimento especial ao supervisor João Madureira por me apoiar sempre que necessário. Obrigado pela excelente orientação, conselhos e constante troca de conhecimento.

Queria também agradecer ao professor Nuno Pereira pela disponibilidade demonstrada ao longo do projeto. Ajudou-me na construção do artigo que foi aceite para conferência. Agradeço também a sua dedicação na revisão e correção do relatório. Assim como os sucessivos votos de confiança na minha capacidade para a realização deste projeto.

Por último, mas não menos importante, quero agradecer à minha família pelo apoio e suporte que me deram ao longo do projeto.

Index

List of Figures

List of Tables

Acronyms and symbols

AHP	Analytic hierarchy process
AP	Access Point
HAR	Human Activity Recognition
HMM	Hidden Markov Model
IoT	Internet of Things
MCDA	Multi-criteria decision analysis
MEMS	Micro-Electro-Mechanical-Systems
SoC	System-on-a-Chip

1 Introduction

The main motivation behind this project is the implementation of a system capable of Human Activity Recognition (HAR) allowing the caregiver, a kin or a medic, to assist elderly people by checking their daily activities and learning the person's activity habits. Detecting the transport mode is an important HAR tool, as it allows the caregivers to perceive whether the elder is having a sedentary life, whether there are changes in the daily movement patterns, provide indication about lack of engagement in social activities, and whether they are sufficiently independent in their everyday life, or doing enough exercise.

This dissertation concerns human physical activity classification using motion sensors embedded in smartphones. Due to their popularity, processing and sensing capabilities, smartphones have the potential to become an “electronic stethoscope”, helping health professionals achieve better diagnosis and treatment.

A platform like a smartphone has several sensors that might be useful, GPS signal, as well as Wi-Fi information. This particular work intends to exploit the accelerometer sensor signals as the main source of information to detect the user's transport mode as it is the one that delivers the most

useful data related to activity recognition and it is also the most optimized sensor regarding battery usage.

This implementation will also take advantage of the surroundings, given the abundance of Access Points (AP) that provide relevant information. The GPS will not be used for this end, because it is often unavailable (indoors, such as inside buildings, at home, underground or doctor's office), making it impossible to get the user location. As the application is intended to be used in Porto city, it is possible to take advantage of the APs equipped in several public transports, such as buses, trains and in the near future, metros, which will also have APs installed in their carriages. These APs can be easily recognizable and thus, we can imply that, with a certain degree of certainty, the user is probably inside the correspondent public transport.

The project is being developed for Fraunhofer Portugal for Assistive Information and Communication Solutions (Fraunhofer AICOS). Fraunhofer Portugal AICOS focuses its activity in the area of assistive information and communication solutions.

1.1 Problem

The problem to be solved is the detection of the user's transportation mode by using the accelerometer sensor signals as the main data source.

Detecting the transport mode is a complex and hard problem to solve. In addition, the device used to obtain the transport mode can't be obstructive and big, as it would bring demotivation to its usage, rendering the system useless. So, it is necessary that this detection is performed by a small, practical and convenient device with capabilities to send, in real time, the information obtained to the caregivers.

1.2 Approach

The purpose of this work is to design a solution that allows the detection of the transportation mode being used, with inertial sensors, namely the acceleration sensor, as the main data source.

The approach followed these steps:

1. Collect the state of the art - on methods for detecting the transportation mode, including different sensors and classification algorithms;
2. Evaluate classification algorithms identified in the state of the art - classification algorithms must be evaluated and compared to determine the most suitable and effective classifier;
3. Detect the user's transportation mode - Stand Still/Sit, Walk, Car, Metro, Train and Bus;
4. Improve system accuracy - by taking advantage of the information on nearby Wi-Fi Access Points;
5. Implement a library for transport mode detection - Define an interface whose methods implemented can be used as a library by other developers;
6. Implement a prototype system - which will demonstrate the functionality of the transport mode detection API implementation by registering the transport-related activities of an elder, therefore allowing the caregiver to learn about the person's activity habits;
7. Evaluate the solution – by comparing against existing solutions and against other studies.

1.3 Objectives

The main goal of this thesis is to develop a transport detection algorithm using smartphones sensors. To consider this goal accomplished, several sub-goals were outlined:

- **State of the art and existing solutions improvement** – the algorithm developed should achieve better results than the results obtained in similar studies and existing solutions;
- **Obtain at least 85% accuracy** – the algorithm should detect the transport mode and obtain at least 85% accuracy;
- **Usage of smartphone features common even in low end devices -** modern smartphones have several features that can be useful to detect the user transport such as inertial sensors, GPS, Wi-Fi, GSM, and Bluetooth. These features are found even in low-end devices;
- **Create a transport API –** the classifier should be developed in a way that any developer can easily add context awareness to their application by detecting the transport mode of the user;

- **Demonstrate the functionality of the transport API** - through a prototype system by registering the transport-related activities of an elder and therefore allowing the caregiver to learn about the person's activity habits.

1.4 Achieved Results

Relying on the power and ubiquity of modern smartphones for HAR purposes is a very attractive proposition that previous researchers have explored. In this work, it is presented the development of a classifier for transport mode detection, which uses the smartphone inertial sensors and wireless communication capabilities.

The classifier can identify the following transportation modes: walking, in bus, in car, in train or in metro. Additionally, if the phone is still for a period of time, it detects an individual as being inactive. The classifier was created using over 24 hours of transportation data from a group of 15 individuals and Weka classification shows it can achieve over 96.1% overall accuracy. Nevertheless, a real world validation of the implemented system obtained 88.97% overall accuracy.

The classifier is embedded in an Android application for elderly care, which can be used by caregivers and alike to learn about the daily activity patterns of elderly people.

1.4.1 Paper Publication

In alignment with the investigation done in this thesis, a paper was submitted and accepted for conference for publication at the IEEE HealthCom. The published article titled, Smartphone-based Transport Mode Detection for Elderly Care, was presented at the 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (IEEE Healthcom), held in Munich in September 2016.

1.5 Document Structure

This document is divided in eight chapters, respectively, Introduction, State of the Art, Background on Machine Learning, Machine Learning for Transport Detection, A Framework for Transport

Detection, Implementation, Evaluation of the Solution and Conclusion. The following paragraphs detail the contents of each chapter, and maps them with the expected thesis outcomes, as described by the course syllabus.

Section 1 - "Introduction" details the context and problem in which this work fits, states the planned approach to the development of this work along with the achieved results (**Outcome 1**).

Section 2 – "State of the Art" points out the conclusions of some studies on elderly population which indicate the most used transports modes. This chapter also presents some previous studies in the area of the activity monitoring, some existing APIs, possible additional methods to improve an activity monitoring system accuracy and the platforms which may be needed on this work (**Outcome 2**).

Section 3 – "Background on Machine Learning" provides the theoretical concepts that contextualize the reader with the main principles considered in this work, detailing each step of a classic pattern recognition methodology of a learning algorithm.

Section 4 – "Machine Learning for Transport Detection" reviews the classification algorithms referred in the state of the art and details how each classifier was evaluated. The possibility of integrating existing Activity Recognition APIs in this work is assessed (**Outcome 3**).

Section 5 – "A Framework for Transport Detection" presents the design of the Transport API along with a description of the architecture of the system, the details of existing processes (**Outcome 4**).

Section 6 – "Implementation" describes the methodology and approach followed in the classification process, step by step. The code to interact with the API is stated followed by a demonstration of the UI of the proof of concept system. (**Outcome 5**).

Section 7 – "Evaluation of the Solution" explains the steps taken to evaluate the finalized solution (**Outcome 6**).

Section 8 – "Conclusion" reviews the objectives of this work, reveals the achievements and future work.

Appendix A – presents more confusion matrices obtained during testing and evaluation.

Appendix B - presents a value analyses along with a business model, Canvas.

Appendix C - presents the design of proof of concept system along with a description of the architecture of the system, the details of existing processes and the structure of the database

Appendix D – presents the submitted and accepted article at 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (IEEE Healthcom).

2 State of the Art

This chapter points out the conclusions of some studies on elderly population which indicate the most used transports modes. This chapter also presents some previously studies in the area of the activity monitoring, some existing APIs, possible additional methods to improve an activity monitoring system accuracy and the platforms which may be needed on this work. At the end of the chapter are presented the conclusions of the state of the art.

2.1 Transport Mode Detection

The modes of transport most used by elders as indicated in several studies are public transportations and car. A study (Viana 2010) done on elderly population in Porto interviewed several elders and demonstrates that the most used public transportation by this population, is the bus, followed by metro and train. A similar study (CEDRU & BCG 2008) reveals that to reach their destination, Portuguese elders mainly go on cars, on public transports or on foot. Other study (Carneiro et al. 2012) also points that the most used transportation modes are buses and cars, which includes taxis, family members' cars or own cars. Transport modes such as biking, running and motorcycle are not mentioned in any of these studies.

The following sections present previous studies in the area of activity monitoring and transport monitoring. The related work is classified by the resources of the smartphone used. Starting with the works that employ GPS signals, then accelerometer only, and then a combination of sensors.

2.1.1 Detection Using GPS

Some current systems use mainly the GPS signal to perform this task and can only distinguish between general modes (walking, running, car). Within areas where this signal is weak or non-existent, these systems do not work properly. The inertial sensors, mainly the accelerometer, can be used to overcome these problems.

A method of detection of the transport mode was proposed (Widhalm et al. 2012), which generates the features for classification from GPS readings and accelerometer data. This approach also includes position data from cell networks to prevent the system to malfunction when there are GPS signal losses. When the trajectory cannot be accurately reconstructed, it relies on accelerometer features. To differentiate the transport modes (Bus, Car, Bike, Tram, Train, Subway, Walk and Motorcycle) it was used a combination of classifier ensemble with Hidden Markov Model and achieved an average accuracy of 75.8%.

Similarly, (Anjum & Ilyas 2013) also used GPS and accelerometer data but additionally used gyroscope data. Features were extracted from these signals in windows of 5 seconds and supplied data to the classifier. Several classifiers were tested with these features and it was determined that Decision Tree outperforms the other classifiers with a 94.4% precision and 94.2% recall rate (accuracy). This model detected 7 different physical activities, including walking, running, climbing stairs, and descending stairs, cycling, driving and remaining inactive.

Reddy (Reddy et al. 2010) combine GPS and accelerometer to recognize between stationary, walking, running, biking and motorized transportation, achieving over 93% accuracy. Classification is performed with a hybrid classifier consisting of a decision tree and a first order discrete HMM classifier.

2.1.2 Detection Using Only Accelerometer Sensor

Recent studies present systems which only use accelerometer signals for HAR. Yang (Yang 2009) presented a framework that performs physical motion recognition using smartphone accelerometer. The features were extracted from accelerometer signals and then used as input for the Decision Tree classifier, obtaining 87.6% accuracy for the detection of physical activities, such as sitting, standing, walking, running, driving and bicycling. Another paper, (Kwapisz et al. 2011) describes and evaluates a system using smartphone on-board accelerometer sensors for HAR. The data were collected in a frequency of 20Hz (50ms), divided in 10-second segments and six features were extracted from it. This study considered six activities, including walking, jogging, ascending stairs, descending stairs, sitting and standing, and it accomplished 90% accuracy using a Decision Tree classifier.

Fraunhofer (Aguiar, Rocha, et al. 2014) proposed a solution to reliably detect falls merely using smartphone integrated accelerometer sensor. When detecting a fall, the system tracks the user location and sends SMS and email notifications to a set of contacts. To optimize battery consumption, this solution sets the sampling frequency to 4Hz (250ms) when users is motionless, otherwise sets the frequency to 67Hz (15ms). After testing three machine learning classifiers using several features as input, the classifier with overall better performance and sensitivity was the Decision Tree with 97% overall accuracy.

Another solution proposed by Fraunhofer Portugal (Aguiar, Silva, et al. 2014) detects physical activities, such as sitting, standing, walking, running and tilting. To optimize battery consumption, when the user is motionless the sample frequency is reduced from 33.33Hz to 4Hz. A Decision Tree classifier was trained after extracting twelve features from accelerometer signals. This solution achieved an average accuracy of 99.5% for the pocket usage and 99.4% when the phone was used on the belt.

An Accelerometer-Based Transportation Mode Detection on Smartphones (Hemminki et al. 2013) preprocessed and transformed the sensor values and with it, constructed gravity horizontal and vertical representations of the accelerometer measurements using a sliding window with 50% overlap and a duration of 1.2 seconds. Then the features were extracted from both vertical and

horizontal representations and the classification relied on a three-stage hierarchical classification framework (kinematic, stationary and motorized classifier) for transportation mode detection. Each of the three classifiers considers a variant of AdaBoost as the instance-based classifier.

2.1.3 Detection Using a Combination of Sensors

In an attempt to achieve better results, another solution (Figueira et al. 2016) combines two sensors, accelerometer and barometer. The signals were obtained in windows of 5 seconds and with a frequency of 30Hz for the accelerometer and 5Hz to the barometer. Using a Decision Tree classifier to detect daily activities (walking, running, sitting, standing, upstairs and downstairs) this framework obtained 94.53 ± 6.82% accuracy.

2.2 Activity Recognition APIs

Existing designs for detection of mode of transport and activity recognition, such as the "Google Activity Recognition API" or the Intel's "Activity Recognition API" may be incorporated to an application to add context-aware capabilities. The performance of these APIs will be studied in the next chapters to assess the possibility of integrating these solutions with my implementation.

2.2.1 Google Activity Recognition API

The Google Play services APK contains the individual Google services and runs as a background service in the Android OS. The developer interacts with the background service through the client library and the service carries out the actions (Figure 1). Through Google Play Services the developer can access Google features such as Activity Recognition and Locations APIs.

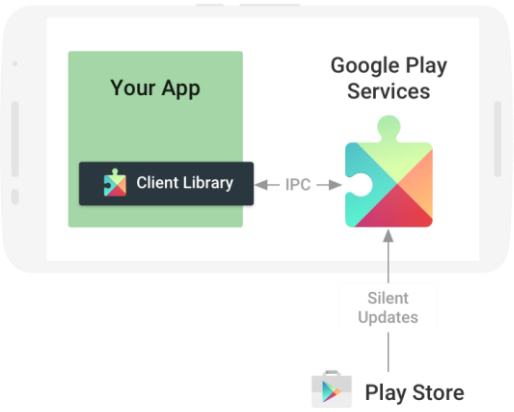


Figure 1 - The Google Play services APK on user devices receives regular updates for new APIs, features, and bug fixes (Google 2016c)

The activities are detected by periodically waking up the device and reading short bursts of sensor data. It only makes use of low power sensors in order to keep the power usage to a minimum.

Google's API returns the detected activity of the device with an associated confidence, such as IN_VEHICLE, ON_BICYCLE, ON_FOOT, RUNNING, STILL, TILTING, UNKNOWN, WALKING. Being that when it returns ON_FOOT the user is either walking or running, both sub-activities of ON_FOOT, and when it returns UNKNOWN means that the API was unable to detect the current activity. (Google 2016b)

When the developer requests activity updates to the API, he sets the detection interval value which will determine the update frequency of the activities reports. Activities may be received more frequently if another application has also requested activity updates at a faster rate. The opposite may also occur if the activity detection service requires more samples to make a more accurate prediction. (Google 2016a)

2.2.2 Intel Activity Recognition API

Intel Context Sensing SDK is a library available for Android and Windows that helps the developers incorporate context-aware capabilities and services in their applications. The SDK is flexible, offering several methods to use the services, either independently or in combination. The SDK

includes Context APIs, which are useful to create context-aware applications by taking advantage of many built-in context type providers. Several context states can be sensed using this SDK, such as Activity Recognition, Audio classification, Beacons, Terminal context, Pedometer, Location, Installed applications, Running applications, Battery, Calendar, Call, Contacts, Device information, Music, Network, Device Position, User-Defined Gesture, Ear Touch Event Detection, Nearby Restaurants, Place, Weather, etc. The Activity Recognition was the context API with more extensive study related to this work.

The Intel's "Activity Recognition API" returns all modes of transport that person may be using at the time and the probability of each one, including SEDENTARY, WALKING, RUNNING, BIKING, INCAR, INTRAIN, RANDOM and NONE (Figure 2). The INTRAIN activity is not yet implemented, the activity RANDOM is reported if user is doing some activity, but not recognized and the activity NONE is reported if the activity inferred is not in the filter list. The developer has the option to filter the activities to be sensed and the option to set the frequency of activity reports. (Intel 2016)

Figure 2 – Intel's API Recognized activities (*Intel* 2014)

2.3 Platforms and Sensing Technologies

The Micro-Electro-Mechanical-Systems (MEMS) technology available has largely evolved in the recent years, especially with the arising and evolution of the smartphones. These systems are low cost, low power consumption and small chips with at least one sensor that detects changes in the environment.

The platforms explained here are potential candidates to be used in this project and the SensorTag is an example of a small device with motion sensing capabilities.

2.3.1 Smartphone

In the past few years, smartphones have seen a great increase of their capabilities with many new applications and hardware evolutions to support it. Every year smartphones are becoming more and more common and they are used as an imperative tool in everyday activities. As smartphones have MEMS technology embedded, it makes them ideal for HAR. The wide range of micro sensors included in them allows to analyse the surrounding ambient, the movements and the position (Grankin et al. 2012).

A new technology is now being added in the new smartphones called Sensor Hubs, this chip is useful to perform some low-level computation allowing the System-on-a-Chip (SoC) to be in a suspend mode. Having such technology in the smartphone greatly reduces the power consumption, opening new possibilities to HAR systems, environmental monitoring and health care. Previous studies show that Sensor Hubs can achieve up to 61% and 80% power savings (Balasubramanian et al. 2013).

2.3.2 SensorTag

The SensorTag is a device of the Internet of Things (IoT) manufactured by Texas Instruments capable of communicating through three standards in this area, being them: Bluetooth Smart, 6LoWPAN and ZigBee. (Texas Instruments 2016)

For its capabilities this device is small (5 x 6.7 x1.4 cm), the SensorTag (Figure 3) has incorporated the following sensors:

- Ambient temperature
- Infrared temperature
- Humidity
- Barometer
- Motion (accelerometer, magnetometer and gyroscope)
- Luxometer
- Key press state.

Even though Texas Instruments claims that the device has 1 year of battery lifetime, it is only possible with a frequency of 1 second. As it is needed a much bigger frequency for this project, the battery, realistically, only lasts for less than a day, since it starts giving connection errors.

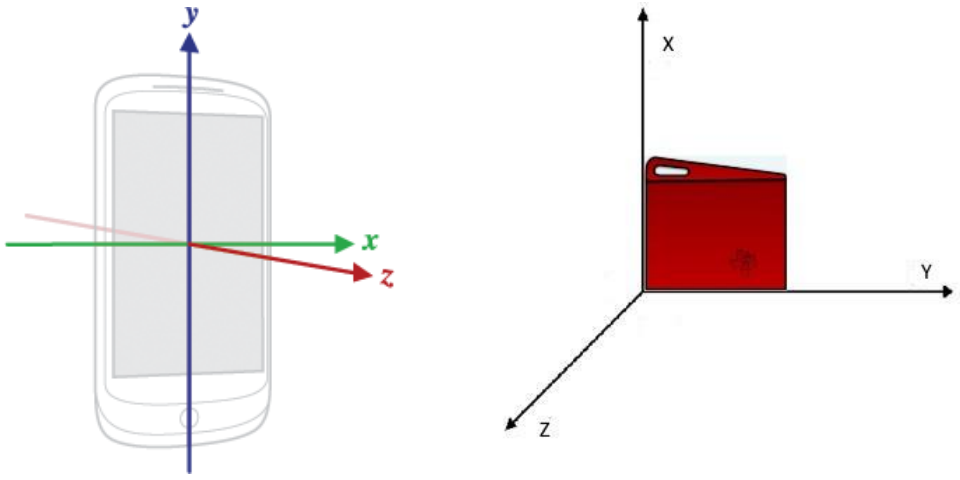
Figure 3 – SensorTag (Texas Instruments 2016)

2.3.3 Sensors

Of all the smartphones sensors and SensorTag sensors incorporate, the only relevant sensors are the motion sensors, which will now be described.

2.3.3.1 Accelerometer

An acceleration sensor measures the acceleration applied to the device, including the force of gravity on the three axis x, y and z (Figure 4). Acceleration changes can be caused by change of direction or by change of velocity. The same representation of a smartphone coordinate system axis is used for acceleration sensors, gyroscope, etc.



(a) (b)

Figure 4 – (a) Representation of a smartphone coordinate system axis (Developer.android.com 2015) and (b) SensorTag axis representation

> *"The basic mechanism underlying acceleration measurement is often described in terms of a mass–spring system, which operates under the principles of Hooke's law (F = kx), and Newton's 2nd law of motion (F =ma). When a mass–spring system is submitted to a compression or stretching force due to movement, the spring will generate a restoring force proportional to the amount of compression or stretch. Given that mass, and the stiffness of the spring can be controlled, the resultant acceleration of the mass element can be determined from characteristics of its displacement."* (Kavanagh & Menz 2008)

2.3.3.2 Gyroscope

The gyroscope measures the rate of rotation in rad/s around a device's x, y, and z axis. Further, unlike accelerometers, gyroscopes are not affected by errors related to external environmental factors such as gravitational and magnetic fields.

"A vibrating gyroscope consists of a resonating member whose vibration pattern undergoes a deformation whenever the sensor rotates, because of the onset of the Coriolis force." (Antonello & Oboe 2012)

2.3.3.3 Magnetic Sensor

Besides gyroscope, another way to calculate the orientation of an object is by using magnetic sensors, commonly referred to as compasses detect magnetic fields and measure their absolute position relative to Earth's magnetic north and nearby magnetic materials. Unlike a gyroscope, a magnetometer gets an absolute direction (relative to the earth coordinate system), instead of relative direction which obeys to the smartphone coordinate system.

2.3.3.4 Barometer

This sensor measures the environmental pressure which can be used to track changes in user's altitude. A solution (Figueira 2015) combined accelerometer and barometer to track elevation gained and thus better detect if the user is going upstairs or downstairs.

2.4 Conclusion

This section presents the conclusions of the state of the art.

2.4.1 Transport Mode Detection

All these works mention several classifiers that were evaluated, including Naïve Bayes, Decision Tree and Support Vector Machine (SVM). Nevertheless, most of them came to the same conclusion that Decision Tree is the most accurate classifier to solve these types of problems. To achieve better results, several statistical features need to be extracted from the accelerometer signals. The most stated features on these works are mean, median, standard deviation, entropy, minimum and maximum.

The accuracy of each study, presented in Table 1, demonstrates an overall accuracy between 75% and 94%. Other solutions like (Kwapisz et al. 2011) and (Figueira et al. 2016) do not detect transport modes but detect instead activities such as sitting, standing, walking, running, tilting, going upstairs and downstairs, managing to achieve better results this way compared to the other studies.

The most identical study (Widhalm et al. 2012), regarding the transports detected, achieves 75.8% accuracy.

Table 1 – Results obtained by each study

Sensors Used	Study	Accuraccy
GPS + Accelerometer	(Widhalm et al. 2012)	75.8%
	(Anjum & Ilyas 2013)	94.2%
	(Reddy et al. 2010)	93.6%
Accelerometer	(Yang 2009)	87.6%
	(Kwapisz et al. 2011)	90%
	(Aguiar, Silva, et al. 2014)	99.5%
	(Hemminki et al. 2013)	80.1%
Combination	(Figueira et al. 2016)	94.53 ± 6.82%

2.4.2 Activity Recognition APIs

These APIs have the potential of adding context awareness to an application, allowing to detect several activities and transports. Along with a preliminary test of these solutions, the possibility of integrating these solutions with my implementation is assessed in Section "4.4 Activity Recognition APIs".

2.4.3 Platforms

Using a IoT sensor like SensorTag forces the users to acquire and use an additional device on their everyday living, possibility decreasing the usage of the system. Because smartphones are so common and are used as an imperative tool in everyday activities, makes them the ideal tool for HAR.

The accelerometer is the most used sensor for HAR since it is the most optimized sensor regarding battery usage and it is also the one that delivers the most useful data related to HAR. Changes in direction and velocity cause variations in the measured accelerations and thus reflecting the intensity and direction of one's movements. When the subject moves, the measured acceleration is a combination of one's orientation and velocity, allowing the distinction of different activities. (Figueira 2015)

Both, gyroscope and magnetometer, allow to calculate the orientation of the device, detecting changes orientation and direction. The barometer sensor measures the altitude of the device, helping with the detection of activities such as walking downstairs and upstairs.

3 Background on Machine Learning

The activity monitoring process begins with gathering the motion data. After obtaining the motion data, it is required to generate and select the necessary signal features. These features are then classified using Machine Learning techniques and algorithms. Finally, the classifiers need to have its performance evaluated.

Machine learning is a branch of computer science concerned with induction problems where an underlying model for predictive or descriptive purposes has to be discovered, based on known properties learned from a training set.

Machine learning algorithms can be divided into different categories, depending on the learning process:

- **Supervised learning** uses labelled data to train an algorithm, which then becomes able to classify unlabeled data. Supervised learning methods significantly outperform the unsupervised ones if the test data doesn't contain unlabeled data. Decision trees, SVM and Naïve Bayes are some examples of supervised learning algorithms (Wilde 2010; Laskov et al. 2005).

- **Unsupervised learning** uses unlabeled data method to directly build recognition models. Such an approach uses density estimation methods to discover groups of similar examples in order to create learning models (Wilde 2010).
- **Semi-supervised learning** typically uses a large amount of unlabeled data, together with the labeled data to train the algorithm. This approach falls between supervised learning and unsupervised learning (Lopes et al. 2012).

In Figure 5 it is presented a classic pattern recognition methodology of a learning algorithm, where each step will be introduced in more detail in the following chapters. Given a sample value, the classifier assigns it to a class (Y) by a learning process from its characteristics. In a supervised learning, for each sample there is a respective known label, Y. In this research the label is the respective transport mode.

Figure 5 – Classification architecture scheme

3.1 Data Acquisition

In order to train a new classifier, the first step is to obtain the raw sensors signals and record then into a dataset. Different signal components will be derived from this input in the following steps. In order to standardize the signals, it should be used a fixed sample frequency, which defines the number of samples per second.

To proceed with the classification after the development of the classifier, the signals should use the same frequency used in the training process.

3.2 Signal Processing

After dataset collection, the raw data needs to be processed to be used in Machine Learning algorithms. There are several segmentation methods that can be applied, like sliding windows, to enhance relevant signal properties and obtain useful information from a continuous stream of data. The signal is divided in equal windows. In this case, each feature extracted for each window is then used as an input for the recognition algorithms that will associate it with the respective transportation mode.

It is very important to find a balanced window size. On one hand, decreasing the window size reduces battery consumption and allows to detect simple activities but any blunt movement easily "spoils" the classification. On the other hand, with a bigger window size it is possible to detect more complex activities but if the window is too big every classification will have similar values and thus harder to detect the differences. (Banos et al. 2014)

3.3 Feature Extraction and Selection

Feature extraction consists in transforming a large quantity of “raw” data into a set of useful features, providing relevant information that will have an essential role in the classification process.

Classification problems require the selection of a subset of features to represent the patterns to be classified. The feature selection impacts, among other things, the accuracy of the classification, the cost and time needed for learning a classifier and performing the classification from the learned classifier. If the selected features fail to describe the patterns due to lack of available information, hence regardless of the learning algorithm used, the accuracy of the classifier will be reduced.

After features extraction, it is possible that many of the features are either redundant or irrelevant, and can be removed without incurring much loss of information and allowing to reduce this time consuming and heavy task. So, keeping the dimensionality of the feature data as small as possible is very important, being obtained by a feature selection process. (Yang & Honavar 1998)

3.4 Classification

After features extraction and selection, the next step is to apply Machine Learning techniques in order to construct a classifier. The choice of the algorithm depends on the type of dataset available and the objectives of the investigation. In this work, only supervised learning methods were used. Some of these algorithms are referred to in the state of the art, including Naïve Bayes, Decision Tree and Support Vector Machine will be now described.

After training these classifiers with a labelled supervised training method, the best performing classifier can be implemented (details in the next subchapter). The implemented classifier will classify the object using the provided features.

3.4.1 Decision Tree

A decision tree is composed by nodes and leaves. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that node can assume. Instances are classified starting at the root and sorted based on their feature values. A small decision tree example is represented in Figure 6. In order to classify the objects “N” and “P”, we start at the root of the tree, evaluate the features value, and take the appropriate branch. This process continues until the object is asserted.

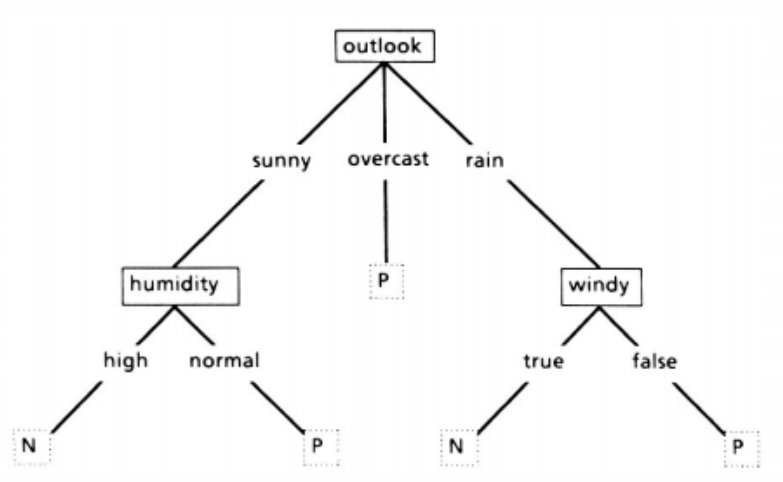


Figure 6 – A simple decision tree (Quinlan 1986)

One approach to help avoid overfitting training data is by pruning the induced decision tree. If the two trees employ the same kind of tests and have the same prediction accuracy, the one with fewer leaves is usually preferred. (Kotsiantis et al. 2007)

According to (Breiman et al. 1984) the tree complexity has a crucial effect on its accuracy, and is usually measured by one of the following metrics (Rokach & Maimom 2014): the total number of nodes, total number of leaves, tree depth and number of attributes used. So trading accuracy for simplicity is a trade-off that should be considered (Bohanec & Bratko 1994). Typically, the goal is to find the optimal Decision Tree by minimizing the prediction error as well as minimizing the number of nodes.

3.4.2 Support Vector Machines

SVM is a rather recent supervised machine learning technique. SVM works by maximizing the hyperplane margin that separates two data classes. Data points that lie on the margin are known as support vector points and the solution is represented as a liner combination of only these points (see Figure 7). Other data points are ignored. Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data. (Kotsiantis et al. 2007)

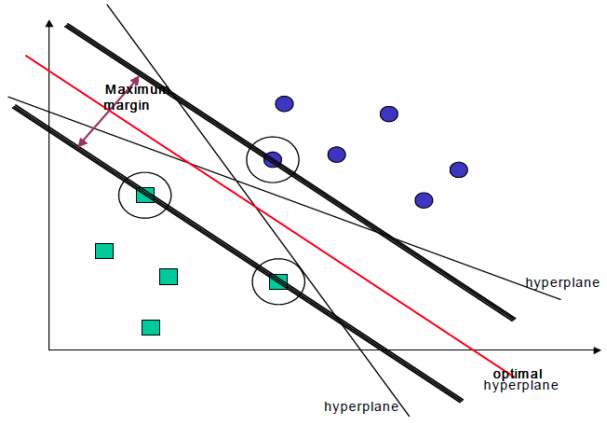


Figure 7 – SVM Maximum Margin (Kotsiantis et al. 2007)

3.4.3 Naïve Bayes

Naïve Bayes classifier is the simplest of the Bayesian classifiers, in that it assumes that all features are independent of each other. Once the independence assumption, the parameters for each attribute can be learned separately, simplifying the learning process, specially when the number of attributes is large. The major advantage of the Naïve Bayes classifier is its short computational time for training. (McCallum & Nigam 1998; Kotsiantis et al. 2007)

3.5 Validation

No single learning algorithm can uniformly outperform other algorithms over all use cases. The simplest approach is to estimate the accuracy of the candidate algorithms on the problem and select the one that appears to be most accurate. (Kotsiantis et al. 2007)

If a classifier learns and tests on the same data, a problem of overfitting is assured to occur. Therefore, the constructed model would exaggerate in minor fluctuations of data, leading to a perfect score on the test results. But testing new data would provide a poor predictive performance. One way to avoid overfitting, transversely on all learning algorithms, is by using a k-fold Cross-Validation method.

A 10-fold cross-validation is a method which divides the dataset into 10 parts (known as folds), holds out each part in turn, and averages the results. This way each data point in the dataset is used once for testing and 9 times for training. The results obtained for each fold can then be averaged to produce a single estimation.

The "Detailed Accuracy By Class" provides information about how the classifier performance varies according to the class predicted. The following values are detailed for each class:

- **TP Rate**: rate of true positives (instances correctly classified as a given class), equivalent to Recall
- **FP Rate**: rate of false positives (instances falsely classified as a given class)
- **Precision**: proportion of instances that are truly of a class divided by the total instances classified as that class

- **Recall**: proportion of instances classified as a given class divided by the actual total in that class, equivalent to TP rate
- **F-Measure**: A combined measure for precision and recall
- **ROC area**: illustrates the performance of the classifier, an "optimal" classifier will have ROC area values approaching 1.

The "Confusion Matrix" allows visualization of the performance of an algorithm, each column of the matrix represents the instances predicted by the model while each row represents the "real" instances, which were provided to the model.

Statistical tests

Hypothesis testing never constitutes a proof that our claim or hypothesis is valid, but It supports the belief that our observations/claims did not occur by chance. To validate the claim that one classifier outperforms another a test with paired groups should be employed to ensure that are being tested neck a neck.

The usual process of hypothesis testing consists of four steps (Gomes 2015):

1. **Formulate the null hypothesis H0 (or claim) -** for example, classifier x has more precision than classifier Y;
2. **Identify a test statistic** - that can be used to assess the truth of the null hypothesis. First a Shapiro-Wilk test should be done to accept or reject that the data came from a normally distributed population. If the data came from a normally distributed population, then a parametric test should be used, otherwise it should be used a non-parametric test;
3. **Compute the p-value -** The p-value is defined as the probability, under the assumption of hypothesis H0, of obtaining a result equal to or more extreme than what was actually observed;
4. **Compare the p-value against an acceptable significance value α** - if p-value ≤ α, the null hypothesis is ruled out, and the alternative hypothesis is considered valid.

Types of Statistical tests:

- **Parametric:** (Ex. T-test) strong assumptions about the distribution of the underlying data. Thus, it is more likely to detect a significant effect when one truly exists.
- **Non-parametric:** (Ex. Wilcoxon Signed Rank Test) weaker assumptions about the data, but are also typically less powerful (less apt at rejecting the null hypothesis when it is false) than their parametric counterparts. If the sample size is to small its often difficult, if not impossible, to verify that all the assumptions hold.

3.6 Data Mining Tool

The evaluation of each of the classifiers can be made using Waikato Environment for Knowledge Analysis (Weka) software. Weka is a collection of machine learning algorithms for data mining tasks, the algorithms can be applied directly to a dataset and it contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. This tool provides a classification algorithm based on all the classifiers described in the previous section, the algorithms names are presented in Table 2.

Table 2 – Weka classifier algorithms naming

Classifier	**Weka algorithm**
Decision Tree	J48
SVM	SMO
Naïve Bayes	Naïve Bayes

3.7 Conclusion

This thesis project follows the same steps of a classic pattern recognition methodology of a learning algorithm described here. The next chapter presents the experiments and evaluations of the studied classifiers. It describes in detail the validation step, reporting the conducted tests that

where done to determine the “best” classifier. The implementation of the “best” classifier follows the same steps of the training classifier with the difference on the classification step. Instead of training the classifier by mapping the provided label with the respective features, it will use what learned to infer the correspondent label to the calculated features.

4 Machine Learning for Transport Detection

The classification algorithms referred to in Chapter 2 State of the Art, from the area of Machine Learning, including Naïve Bayes, Decision Tree and Support Vector Machine classifiers were evaluated. The classifiers were tested and trained using a 10-fold cross-validation.

The focus of this chapter is on the validation step of a classic pattern recognition methodology of a learning algorithm (Chapter 3). The other steps are common to the final implementation of the classifier, so the description related to those steps are described in Chapter 6, “Implementation”.

The possibility of integrating existing Activity Recognition APIs in this work is assessed after some preliminary tests.

4.1 Preparation of Experiments

To start with the evaluation of existing solutions and approaches, first it’s needed to record a dataset with all the possible data from the accelerometer axes. This dataset was used to test, train and evaluate several classification algorithms. It was created by a simple application which requires the tester to annotate on the app the current transportation mode, which must be changed by the tester on the app every time he changes between modes. The transport mode must be provided to test and train the classifiers.

Although there is only one stream of accelerometer events, different signal components can be derived from this input. The signals used in these experiments are detailed in Section “6.1 A Classifier for Transport Mode Detection”.

Data acquisition

In order to perform the evaluation of each classifier a dataset needs to be constructed. For the construction of the dataset a group of 15 volunteers proceeded with their routine while using a smartphone in the pocket. Over 24 hours of mobility data were recorded for the six transport modes. Each transport mode had over 4 hours of data from several users.

4.2 Classifier Tests

As mentioned above, the candidate classifiers tested here are the ones most referred to in the State of the Art, such as Naïve Bayes, Decision Tree and SVM.

The evaluation of each of the classifiers was made using Waikato Environment for Knowledge Analysis (Weka) software. The classification results in Weka shows a "Detailed Accuracy By Class" and a "Confusion Matrix". The information provided by these results are detailed in Section 3.5.

The summary results for our activity recognition experiments are presented in Table 3, providing information about the classifiers overall performance.

The Naïve Bayes classifier treats all features as independent and is the simplest of these classifiers. We present its performance in Table 3 to provide a baseline, but will omit further results from this classifier as its performance is significantly worse.

Table 3 - Weighted Average Accuracy by Classifier

Algorithm	TP Rate	FP Rate	Precision	Recall	F-Measure	Roc Area
Tree J.48	0.961	0.008	0.961	0.961	0.961	0.990
SMO	0.927	0.015	0.928	0.927	0.926	0.977
NaiveBayes	0.827	0.035	0.831	0.827	0.826	0.945

Both, the J48 and the SMO, have yielded a good performance, with all the metrics above 92% (with the exception of FP rate). Even so, SMO is still outperformed by the J48 Decision Tree classifier on all performance metrics, which has all the metrics 3% higher than SMO.

As explained in more detail in “0

Classification”, trading accuracy for simplicity is a trade-off that should be considered. Typically, the goal is to find the optimal Decision Tree by minimizing the prediction error as well as minimizing the number of nodes. Bearing the trade-off in mind I managed to reduce the tree size by 35.46% and only reduce the accuracy by 0.33%, by tuning the Weka J48 classifier.

Table 4 demonstrates the Detailed Accuracy by Class for J48 Classifier after reducing the tree size. "Still/Sit" is the class with higher TP, 99.0% followed by the "Walk" and “Car” class with 97.8% TP. The corresponding confusion matrix is illustrated in Figure 8 with 17502 Total Number of Instances. The diagonal has the higher value, i.e., for each activity the majority prediction is for the correspondent class.

Table 4 - Detailed Accuracy by Class for J48 Classifier

TP rate	FP Rate	Precision	Recall	F-Measure	Roc Area	Class
0.978	0.006	0.970	0.978	0.974	0.994	Car
0.927	0.013	0.936	0.927	0.931	0.982	Metro
0.964	0.008	0.961	0.964	0.962	0.990	Train
0.932	0.014	0.931	0.932	0.931	0.981	Bus
0.978	0.003	0.984	0.978	0.981	0.995	Walk
0.990	0.003	0.985	0.990	0.987	0.997	Inactive

```
    a    b    c    d    e    f   <-- classified as
 2862   18   16   25    0    5 |    a = CAR
   13 2717   44  117   14   26 |    b = METRO
   23   46 2775   24    7    5 |    c = TRAIN
   45   88   38 2711   23    5 |    d = BUS
    3   17   16   26 2859    3 |    e = WALK
    3   17    0    8    2 2901 |    f = INACTIVE
```

Figure 8 – Confusion Matrix for J48 classifier

Table 5 demonstrates the Detailed Accuracy by Class for SMO Classifier and just like the J48 classifier the classes with higher TP Rate are “Still/Sit” and “Walk”. In the corresponding confusion matrix (Figure 9) it’s possible to verify that the classifier wrongly predicts a “Metro” when the real class is “Bus”.

Table 5 – Detailed Accuracy by Class for SMO Classifier

TP Rate	FP Rate	Precision	Recall	F-Measure	Roc Area	Class
0.961	0.017	0.917	0.961	0.939	0.987	Car
0.906	0.031	0.856	0.906	0.880	0.967	Metro
0.910	0.017	0.912	0.910	0.911	0.983	Train
0.829	0.011	0.936	0.829	0.879	0.934	Bus
0.977	0.002	0.989	0.977	0.983	0.998	Walk
0.976	0.009	0.955	0.976	0.965	0.994	Inactive

```
   a    b    c    d    e    f   <-- classified as
2811    9   55   46    0    5 |    a = CAR
  25 2656   93  101    5   51 |    b = METRO
  43  134 2621    8   12   62 |    c = TRAIN
 182  249   43 2413    9   14 |    d = BUS
   0   31   30    3 2857    3 |    e = WALK
   3   23   32    7    6 2860 |    f = INACTIVE
```

Figure 9 – Confusion Matrix of SMO classifier

4.3 Statistical test

To compare the performance of the two classifiers with best results it was used the Weka tool. The test consists of using a paired 10-fold cross-validation (Paired T-test) for both classifiers, thereby obtaining for each fold the accuracy of each classifier. The obtained data (Figure 10) is the necessary input to proceed with the statistical tests.

The annotation *v* or * indicates that a specific result is statistically better (*v*) or worse (*) than the baseline scheme (in this case, Tree J48) at the significance level specified (0.05). The results of SMO are statistically worse than the baseline established by J48. At the bottom of each column after the first column is a count (xx/ yy/ zz) of the number of times that the scheme was better than (xx), the

same as (yy), or worse than (zz), the baseline scheme on the datasets used in the experiment. For this measure SMO is worse than J48.

Dataset	(1) trees.J4	(2) funct
1	(10) 95.03	92.97 *
2	(10) 94.62	92.68 *
3	(10) 94.67	92.29 *
4	(10) 94.44	92.97 *
5	(10) 94.47	92.66 *
6	(10) 94.29	92.48 *
7	(10) 94.45	92.57 *
8	(10) 94.87	92.79 *
9	(10) 94.58	92.71 *
10	(10) 94.39	92.75 *
	(v/ /*)	(0/0/10)

Figure 10 – Paired T-Test. J48 classifier identified as (1) and SMO classifier as (2)

4.3.1 Shapiro-Wilk Test

The Shapiro–Wilk test utilizes the null hypothesis principle to check whether a sample came from a normally distributed population. The test, illustrated in Figure 11, shows that, for alpha= 0.05, the null hypothesis that the data came from a normally distributed population cannot be rejected. So to compare these classifiers it should be used a t-test.

Shapiro-Wilk Test		
	trees	*SVM*
W	0,9282078	0,9558788
p-value	0,4304975	0,7380557
alpha	0,05	0,05
normal	yes	yes

Figure 11 - Shapiro-Wilk Test

4.3.2 T Test: Two Paired Samples

The T-test is a parametric test, such as Wilcoxon Signed-Rank Test for Paired Samples for non-parametric test, allows to test which classifier is better, for a given alpha. The obtained mean accuracies on the experiments are 94.58% for the J48 classifier and 92.69% for the SMO classifier. This test is done to confirm that J48 is better for the accuracy measure and it was not only a matter of luck. For alpha = 0.05 and having:

- H0: mean_tree = mean_SMO (Is the null-hypothesis of this test)
- H1: mean_tree > mean_SMO (Intend to demonstrate)

The results of this statistical test, illustrated in Figure 12, show that p=8.983E-10 < 0.05 for the one tail (H1: mean_tree > mean_SMO), then we can accept that, for this alpha, the J48 algorithm is more accurate than the SMO algorithm.

T Test: Two Paired Samples								
SUMMARY			Alpha	0,05		Hyp Mean Diff	0	
Groups	*Count*	*Mean*	*Std Dev*	*Std Err*	*t*	*df*	*Cohen d*	*Effect r*
trees	10	94,581	0,227325					
SVM	10	92,687	0,2082226					
Difference	10	1,894	0,2493636	0,0788557	24,018554	9	7,5953336	0,9922897
T TEST								
	p-value	*t-crit*	*lower*	*upper*	*sig*			
One Tail	8,983E-10	1,8331129			yes			
Two Tail	1,797E-09	2,2621572	1,715616	2,072384	yes			

Figure 12 - T Test: Two Paired Samples

4.4 Activity Recognition APIs

The Activity recognition APIs described in Chapter 2 might be a useful tool to integrate in the thesis project to improve performance. The possibility of integrating these APIs is assessed in this section.

After some preliminary testing it became clear that both of the studied Activity Recognition APIs, Google's and Intel's, are not very accurate and thus both return inconsistent values (for example,

indicate that the person is in a vehicle when it is running). From the activities intended to be detected for this work, only still, walking and car are predicted by these APIs.

With an overall accuracy of 80.33%, Google's API performs really well when compared with the Intel's API, which only has 48.74% accuracy. The Intel's accuracy for Train and Metro transports is not shown since their API does not support those transports yet. The Intel's API accuracy for Walk activity is only 2.1% due to the fact that most of the times it classifies walking as running. Google's API ranks amongst the worst performers, when compared to other state of the art approaches in Table 6.

Table 6 – APIS accuracies (%)

	Google Activity Recognition API	**Intel Activity Recognition API**
Car	63.18	45.61
Bus	86.86	48.14
Train	72.75	N/A
Metro	62.65	N/A
Walk	97.22	2.10
Inactive	99.31	99.12
Overall	80.33	**48.74**

Google's API despite having a detection interval which can be defined by the developer, has an inconsistent frequency update due to several factors such as battery saving or in an attempt to get a more accurate prediction the API may delay the update indefinitely until it has a prediction with "enough" accuracy (Google 2016a). With the exception of the "still" activity, this API also tends to predict all activities as a tilting activity once it corresponds to a change of the device angle relative to the gravity.

Intel's API proved to be more responsive than Google's and with a more consistent update frequency. Even though this API has the benefit of returning the probability for each activity that the person may be doing at the time, it has really low accuracy.

4.5 Conclusion

The classifier tests show that J48 classifier achieves better performance over SMO. And the statistic tests support the belief that this observation did not occur by chance. Validating the claim that J48 classifier outperforms SMO for the accuracy measure and it was not only a matter of luck.

Including one of the Activity Recognition APIs in my implementation would bring more complexity to the project, add inconsistent values and more battery consumption. Later these APIs will be evaluated in a more precise way by testing both APIs with my implementation and comparing each implementation's accuracy as explained in more detail in “7.2 Comparing Against Activity Recognition APIs”.

5 A Framework for Transport Detection

This section pertains to the technical design and implementation of the API. The various types of diagrams (use cases, sequence, deployment, etc.) included in this section each provide a different view of the application, including its interfaces, roles and responsibilities, business rules, processes as well as the information collected and maintained.

The solution is composed by two main structures, the Transport Detection API and the proof of concept system. The API is the "core" and main focus of this work, so any details of the proof of concept system can be read in Appendix C.

5.1 Requirements

This section enumerates the functional and nonfunctional requirements of the project as a whole with a brief description of each requirement. Later in this chapter, the features of the API will be specified with the use of UML. The features of the proof of concept will be specified in Appendix C.

5.1.1 Functional Requirements

Table 7 – Functional Requirements

Functional requirement	Description
Start/Stop the transport detection	The elderly shall be able to start and stop the transport detection API as he pleases.
Display daily activity and transport patterns	Through the Website and the mobile app, the caregiver should be able to analyse the daily activity patterns of the elderly and therefore perceive whether the elder is having a sedentary life, if there are changes in the daily movement patterns, provide indication about lack of engagement in social activities, if they are sufficiently independent in their everyday life, or doing enough exercise.
Update data to server	The transport data should be updated to server to allow the caregivers to analyse the daily activity patterns of the elderly.

5.1.2 Non Functional Requirements

Table 8 – Non Functional Requirements

Non Functional requirement	Description
Update data to server in real time	The transport data should be updated in real time as soon as the classification is available, allowing the caregivers to analyse the daily activity patterns of the elderly.
Obtain 85% accuracy	The classifier should obtain at least 85% accuracy in real use cases scenarios. Allowing the detection of the transport of the elderly with assurance.

Integration with Fraunhofer Portugal libraries	This implementation should be integrated with Moverlib and Smartcompanionlib. Accelerometer data and feature calculation shall be withdrawn from MoverLib. And the app should use SmartcompanionLib UI elements to have similar UI elements of other Fraunhofer's apps. Both libraries and its integration will be more detailed further in this chapter.
Employ CouchDb and Android	In order to facilitate the inclusion of this work in Fraunhofer Portugal for which it is being developed, some technologies such as CouchDB and Android application development will be employed.

5.2 System Glossary

Table 9 – Transport API Glossary

Term	Description
MoverLib	A library developed by Fraunhofer Portugal which obtains accelerometer data and calculates the features from the obtained signal
TransportLib	The developed library in this project to classify the transport modes
SmarTrans	The API name of Transport Detection API as a whole. Includes the TransportLib and MoverLib
SmartCompanionLib	The library used by Fraunhofer to maintain a similar UI across all of their apps

5.3 System Overview

SmarTrans features an API for other applications to access its services. The SmarTrans API provides a facade onto the proof of concept system, providing operations to start the API, request/remove transport updates to a specific class and stop the API. To facilitate this integration, the API provides a callback interface, to where the transport classifications will be sent upon subscription.

The classification process follows an Event-Driven Architecture (EDA). The processing middleware accepts the accelerometer events, processes these events and then informs interested consumers via subscription.

SmartTrans integrates TransportLib which relies on MoverLib to obtain accelerometer data, feature calculation and distinguish between sitting and standing stances.

5.4 Use Cases

The use cases of the transport API are listed in Table 10, the use case diagram of the proof of concept system is presented in Appendix C.

Table 10 – Use cases of transport API

Use case	Name
1	Start API
2	Stop API
3	Classify Accelerometer Signals

5.4.1 UC1: Start API

Goal of Use Case:	**Start receiving transport classifications**
Preconditions:	none
Success Post Conditions:	1. The API starts classifying the transport mode 2. The observer is receiving classification updates
Failed Post Conditions:	The elder is returned an error stating that it was not possible to start the API
Actors:	Elder, Smartphone system
Triggers:	This process is started by the Elder (human interaction)

5.4.1.1 Main Success Path

Step	Actor	Description	Branches	
			Condition	Location
1.	Elder	The elder tries to start the API		
2.	Smartphone System	Create Mover System. The Mover System can only be created if the required sensors are available.	Sensor not available	ALT1
3.	Smartphone System	Create Transport System to start classifying the transport mode		
4.	Smartphone System	Request Transport API for Updates		
5.	Smartphone System	Transport API registers the Observer that will receive the updates		
6.	Smartphone System	The use case ends		

5.4.1.2 Alt 1: Sensor not available

Step	Actor	Description	Branches	
			Condition	Location
1.	Smartphone System	The Smartphone System returns an error to the Elder, informing that it was not possible to start the API		
2.	Smartphone System	The use case ends		

5.4.1.3 Activity Diagram

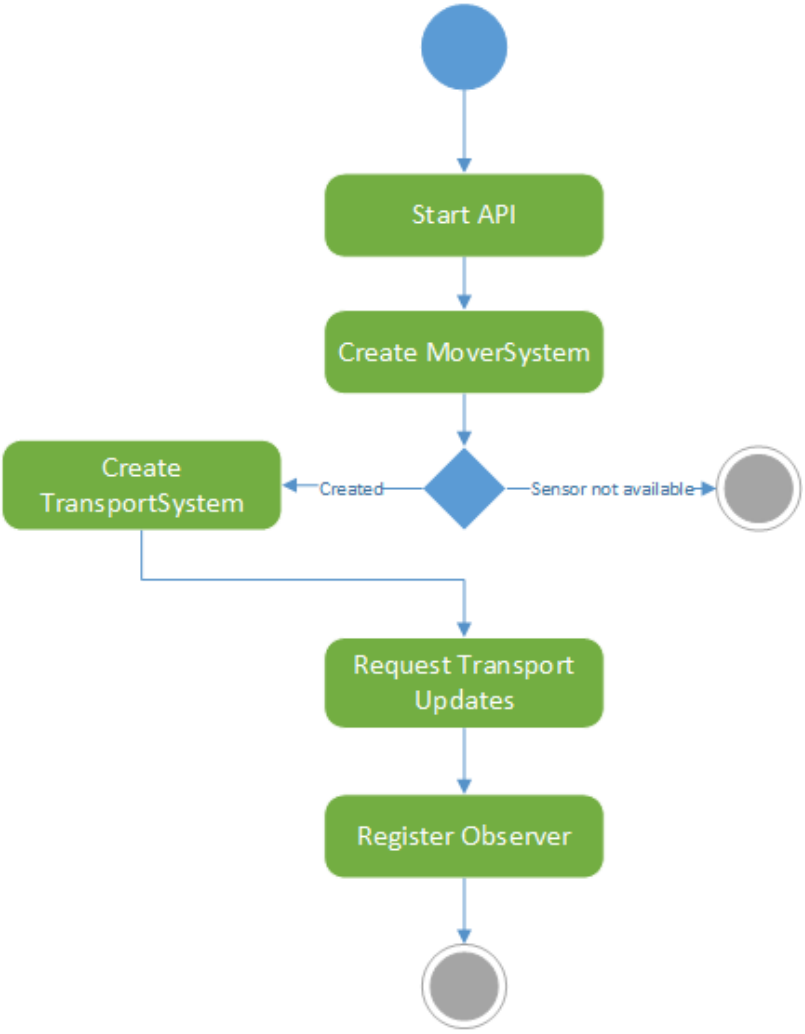


Figure 13 - Activity diagram for Use Case 1

5.4.1.4 Non-functional requirements and Assumptions

None.

5.4.2 UC2: Stop API

Goal of Use Case:	Stop receiving transport classifications
Preconditions:	The API is running
Success Post Conditions:	The API stops running
Failed Post Conditions:	The API is incorrectly stopped
Actors:	Elder, Smartphone System
Triggers:	This process is started by the Elder (human interaction)

5.4.2.1 Main Success Path

Step	Actor	Description	Branches	
			Condition	Location
1.	Elder	The elder tries to stop the API		
2.	Smartphone System	Transport API registers the Observer that will receive the updates		
3.	Smartphone System	Destroy Mover System		
4.	Smartphone System	Destroy Transport System. Stops every observer and provider.		
5.	Smartphone System	The use case ends		

5.4.2.2 Activity Diagram

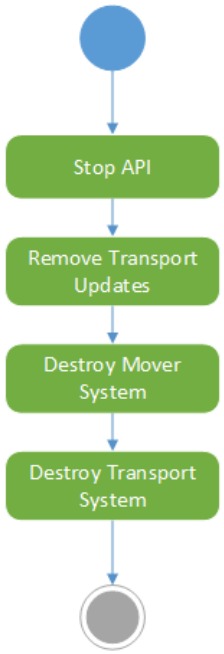


Figure 14 - Activity diagram for Use Case 2

5.4.2.3 Non-functional requirements and Assumptions

None.

5.4.3 UC3: Classify Accelerometer Signals

Goal of Use Case:	**Classify the transport mode using the accelerometer signals**
Preconditions:	Transport API is recording accelerometer signals
Success Post Conditions:	The classification is received
Failed Post Conditions:	none
Actors:	Smartphone system
Triggers:	This process is triggered when the received accelerometer events fill a time window interval

5.4.3.1 Main Success Path

Step	Actor	Description	Branches	
			Condition	Location
1.	Smartphone System	The accelerometer signals are processed, obtaining the features necessary to classify the transport mode		
2.	Smartphone System	The transport mode is classified		
3.	Smartphone System	The obtained classification is sent to the registered observers		
4.	Smartphone System	The use case ends		

5.4.3.2 Activity Diagram

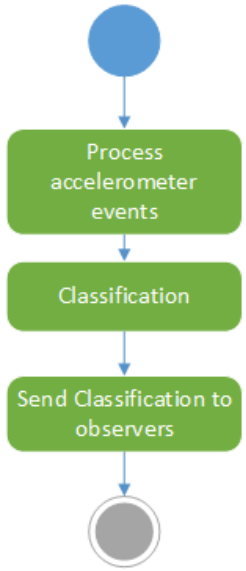


Figure 15 - Activity diagram for Use Case 3

5.4.3.3 Non-functional requirements and Assumptions

None.

5.5 Deployment Diagram

Figure 16 shows a conceptual view of an implementation of SmarTrans API.

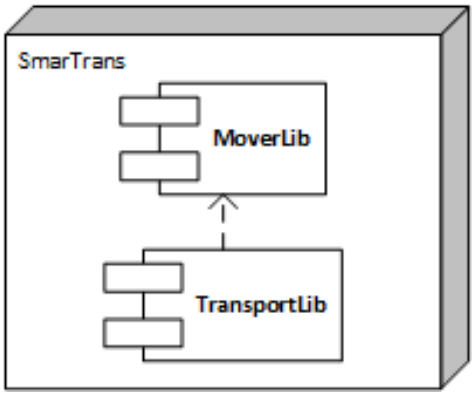


Figure 16 – Transport API Deployment Diagram

5.6 Sequence Diagrams

The sequence diagrams below describe how the groups of objects interact with each other.

5.6.1 UC1: Start API

Use case 1 maps to the interactions between the APP, the TransportSystem and MoverSystem.

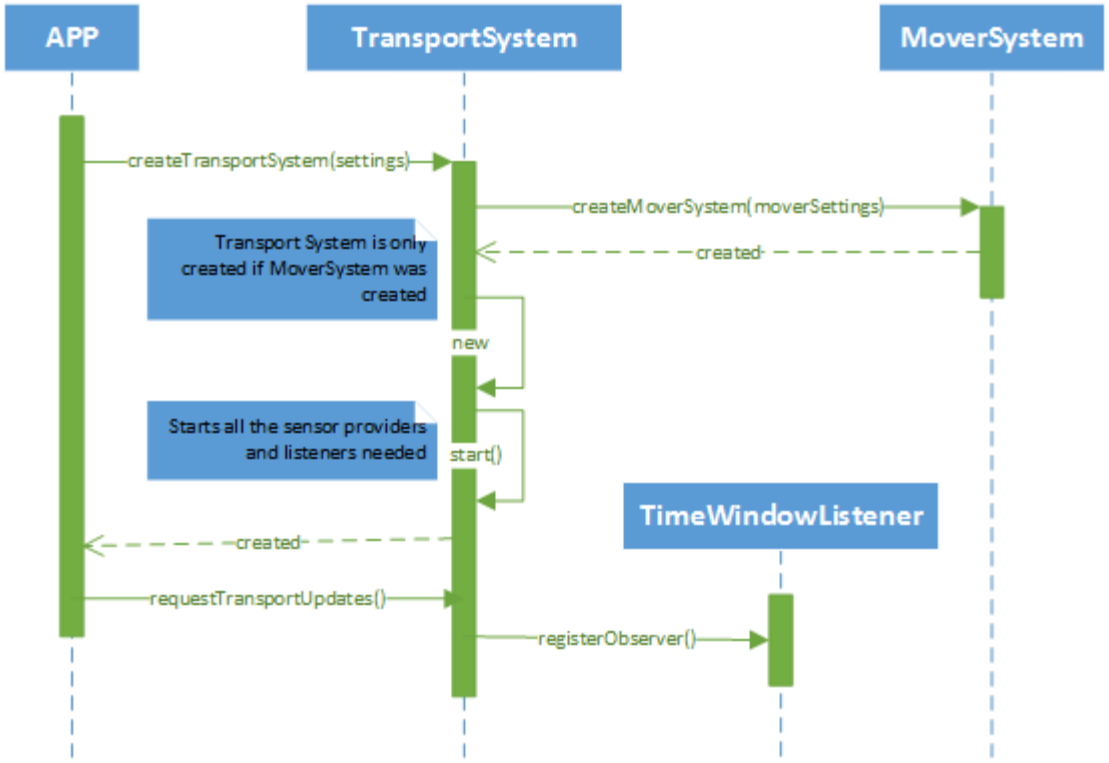


Figure 17 - Sequence Diagram for Use Case 1

5.6.2 UC2: Stop API

Use case 2 maps to the interactions between the APP, the TransportSystem, TimeWindowListener and MoverSystem.

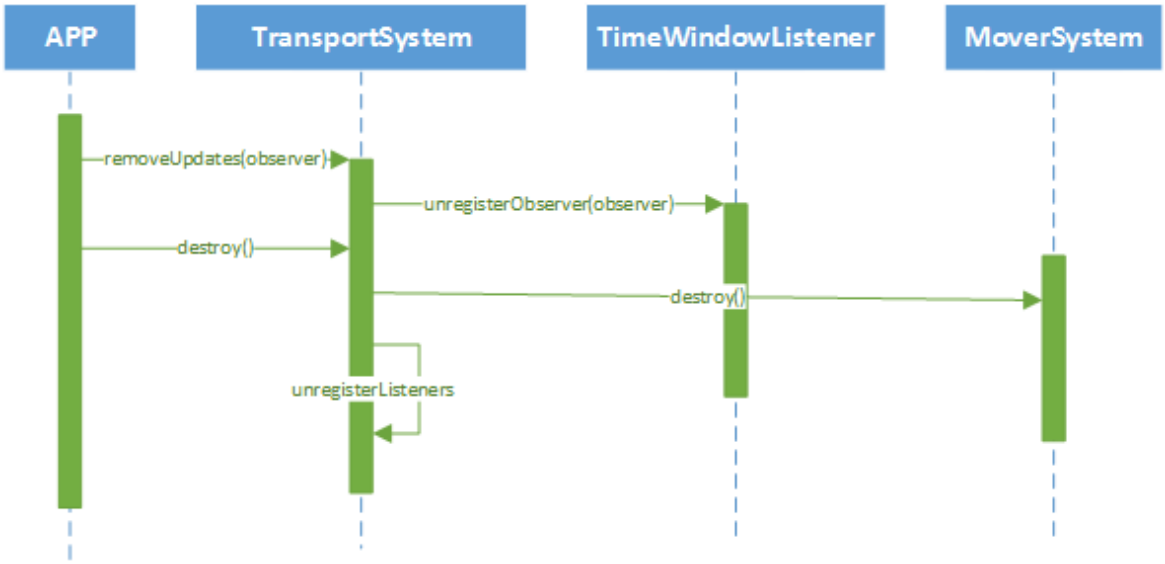


Figure 18 - Sequence Diagram for Use Case 2

5.6.3 UC3: Classify Accelerometer Signals

Use case 3 maps to the interactions between the TimeWindowSensorProvider, TimeWindowListener, FeatureContainer, ITransportClassifier and ITransportObserver. The interaction follows an Event-Driven Architecture (EDA) Architecture. The TimeWindowSensorProvider has no knowledge of the accelerometer subsequent processing, or the interested parties. The TimeWindowListener acts as the middleware, it accepts the events, processes them and then informs the interested consumers via subscription.

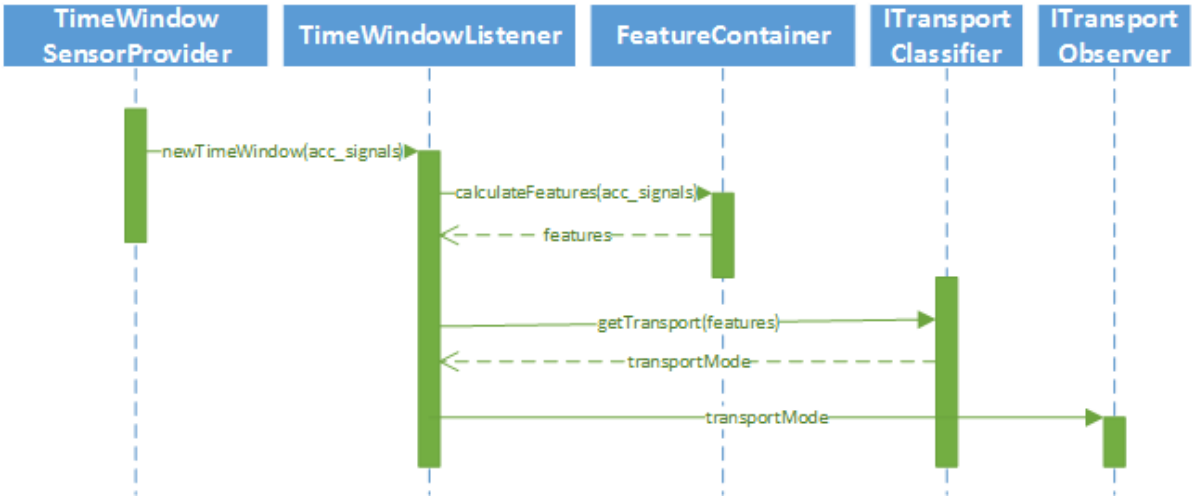


Figure 19 - Sequence Diagram for Use Case 3

6 Implementation

6.1 A Classifier for Transport Mode Detection

The validated classifier in Chapter 4 with the most accuracy was the Decision Tree classifier. This classifier was then implemented following the classic pattern recognition methodology.

6.1.1 Data Acquisition

All the necessary data is obtained from the accelerometer signals with a fixed sample frequency of 33.33Hz (a sample is taken every 30ms). An accelerometer sensor is a low power consumption sensor found even in low-end devices and is also the sensor that delivers the most useful data.

6.1.2 Signal Processing

The features calculated through the output provided by the accelerometer are divided into segments with a duration of 5 seconds (time windows). The size of the window is an important decision because if it is too big, every transport mode will have similar values and thus harder to detect the differences. If the window is too small, any blunt movement will "spoil" the classification.

Although there is only one stream of accelerometer events, different signal components can be derived from this input. At a first instance the x, y, z and magnitude values of acceleration are extracted and used to rise all the signals components. For each of these signals we derive two additional signal components: the gravity and the linear acceleration. Furthermore, the absolute value is calculated from each of these axis signals components.

The angles between the acceleration vector and each of the phone axes are also computed, resulting in signals angleX, angleY and angleZ. An additional signal is calculated by applying low pass filtering to the magnitude.

Thus, a total of 21 different signal components are used in the analysis.

6.1.3 Feature Extraction and Selection

In order to enhance the classification algorithms performance, several features are extracted from the accelerometer signals. The calculated features for each signal component are mean, median, maximum, minimum, root mean square, standard deviation, median deviation, interquartile range, minimum average, maximum average, peak height, average peak height, mean cross count, entropy, energy, skewness and kurtosis. Thus, we compute 17 features for each of the 21 different signal components, totalling 357 features for each time window.

After the feature extraction phase, usually it is possible to verify that many of them are either redundant or irrelevant, and can be removed without incurring much loss of information. With the help of the Waikato Environment for Knowledge Analysis (Weka) (Witten et al. 2011) tool to select the most relevant attributes, it was possible to reduce the tree size by 11.95% (from 421 to 371), to decrease the number of features from 357 to 40 and even to increase accuracy by 1.58%. SMO on the other hand decreased accuracy almost 10%. As explained in Section 0 the model complexity of an SVM is unaffected by the number of features encountered in the training data, therefore reducing the number of features will only reduce the accuracy.

6.1.4 Classification

In this work, supervised learning techniques were used to discriminate between 6 means of transportation (inactive, walking, car, bus, metro and train). Considering the gathered dataset and the respective features extracted, a classification algorithm based on decision trees (J48) was trained and implemented. The implemented Decision Trees classifier evaluates the features values, takes the appropriate branches until it asserts the final branch and outputs the obtained classification.

6.1.5 Post-Processing

This solution goes beyond the normal classic pattern recognition methodology of a learning algorithm, adding an additional step. The idea of the post-processing module is to reduce false classifications and therefore improve transport mode detection. This process starts by recording the classifier predictions and whether any transport Access point was detected.

Although the accelerometer sensor is the main method of transport mode detection, additional methods can be implemented to improve the system precision and accuracy. This implementation will take advantage of the surroundings, given the abundance of AP that provides relevant information. The fact that the application is intended to be used in Porto city, can also take advantage that several buses and trains have AP which can be easily identifiable.

Detect Public Transport's Wi-Fi

In order to attempt increasing the system performance I tried to understand if there is a way to determine if a Wi-Fi connection belongs to a STCP bus. To accomplish this, a simple Android application was developed that gathers data from nearby Wi-Fi connections. After saving the connection details of several STCP bus it was possible to verify that STCP hotspots can be detected through this characteristics:

- SSID : "STCP | PortoDigital"
- BSSID: "d4:ca:6d:__:__:__"
- Frequency: 2437

- Capacities: [ESS].

Even though STCP connections can be reliably detected, it doesn't ensure that the user is inside the bus because the user may be close to the bus in traffic, so some false positives may occur. The same applies if the connection is not detected it doesn't mean that the user is not in a bus since the bus may not have a Wi-Fi connection.

Later in the development, the same test was done to verify if it is also possible to distinguish the Wi-Fi connections available in train carriages. These connections can be detected using the same characteristics used to differentiate STCP hotspots.

As soon as "Metro do Porto" installs the infrastructure to support Wi-Fi connections in their carriages, the AP will be easy to detect and thus improving the system overall performance.

Implementation

To avoid miss classifications, the post-processing is performed with a delay of 30 sec of the classification result, allowing a better evaluation of the classification results having into account the past and "future" classifications.

A state machine is used to avoid incoherent transitions as well as invalid transitions between certain transports and inactive state. As we can see in Figure 20 the post-processing will invalidate incoherent transitions. In the case of transitioning between a transport and inactive this process will keep outputting the last transport to avoid classifications of inactive while the user is standing still or sitting in an immobile transport. Metro was the most transport being predicted as inactive because it is the one which is stopped in a station and in practice being inactive inside a stopped metro or at home is the same.

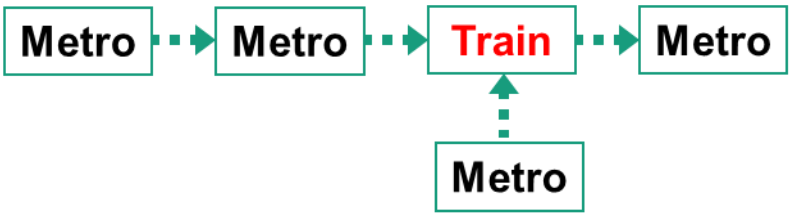


Figure 20 – Post-processing incoherent transitions

A weight is given to both the classified transport and to the identification of public transport Wi-Fi Access points in order to better determine the transport mode. Having in mind that, even though connections can be reliably detected, it doesn't guarantee that the user is inside the detected transport, so more weight is given to the classification. And if no connections are detected, no regards are given to it once the transport may not have an access point installed or the user may have the Wi-Fi disabled in his device.

6.2 API Integration

The API was developed to make its integration in any project as easy as possible.

So to initialize and start the API is as easy as we can see in Code 1. The "TransportSettings" class is implemented by the developer who wants to integrate this API. The class must implement the interface "*ITransportSettings*", and allows the developers to change some of the settings available, such as if he wants to save training logs or enable Wi-Fi. The developer can easily set to receive the classification results wherever he wants by requesting the updates to any class that implements the interface "*ITransportObserver"*. In the example code the results would be sent to the requested activity directly, and the class would need to implement the interface.

```
TransportRealSystem.createTransportSystem(this, new TransportSettings());
TransportRealSystem.requestUpdates(this);
```

Code 1 – API connection and initialization code

Likewise, to stop requesting the API more updates is as easy (Code 2).

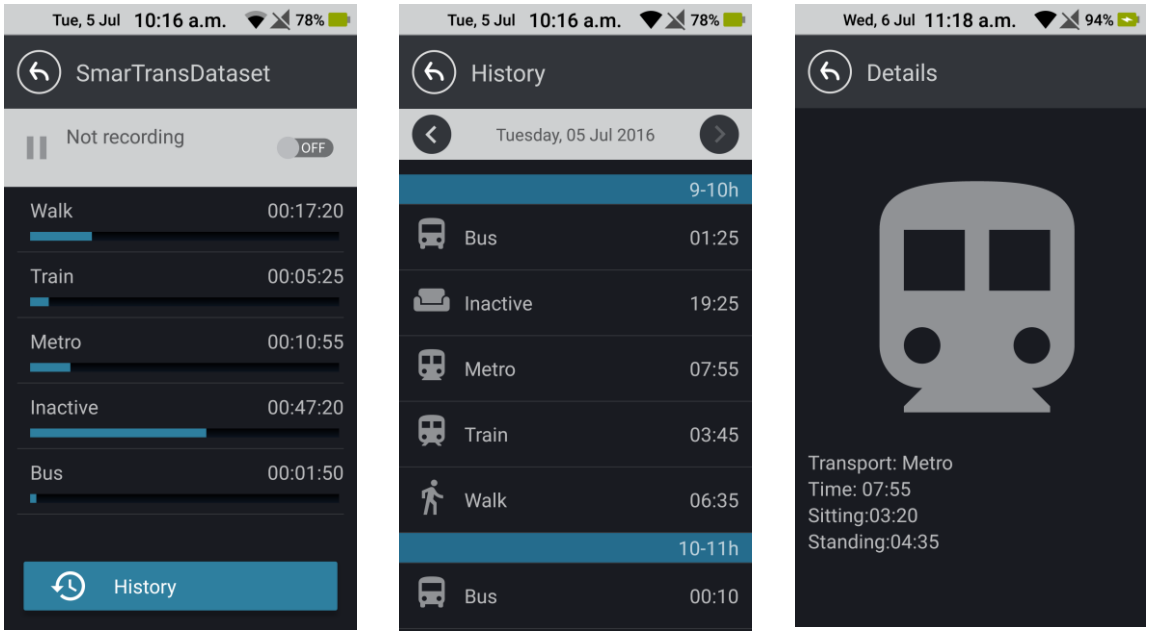
```
TransportRealSystem.removeUpdates(this);
TransportSystem.getInstance().destroy();
```

Code 2 – API stop code

6.3 User Interface

The interface of the Android application uses SmartCompanion UI elements. This library was developed by Fraunhofer Portugal AICOS and is used to maintain a similar UI across the different applications, providing all the UI elements used in this proof of concept system.

Figure 21 shows all the layouts existing in the proof of concept Android app. Figure 21a allows the user to check the transports total time recorded per transportation mode. Figure 21b lets the user view more detailed information per day and hour of day. For the selected transport, Figure 21c details, the time the user was sitting and standing.



(a) (b) (c)

Figure 21 – Android app UI. (a) Overall transports recorded, (b) Daily history, (c) Transport Details

Figure 22 represents the layout of the website. In the website the caregiver can check the elderly/patient activity in some graphics. As well as some details of the person demographic.

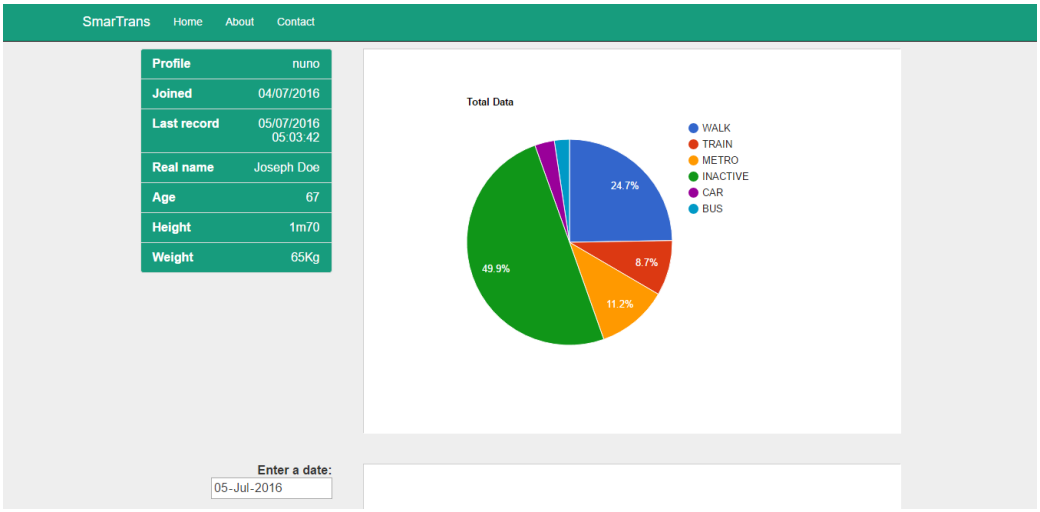


Figure 22 – Web page UI

7 Evaluation of the Solution

In this chapter will be described the procedures that will be taken to evaluate the final solution of this project.

7.1 Validation

To test the performance of the developed API in a real world scenario, a group of 6 volunteers proceeded with their routine while using a smartphone in the pocket and a smartwatch in their wrist.

The evaluation of the solution was accomplished by using the final solution application with some addition onto it. This evaluation version has the implementation of the transport mode detection, and also saves the results returned by the "Google Activity Recognition API" and by the "Intel Activity Recognition API".

Just like for the creation of the datasets for training, the user needs to select the transport mode he is currently using. For more precision in tracking the transport mode transitions, the user selects the transitions between transport modes in a smartwatch. Using this method, the user does not need to grab the phone to select the current transport mode, thus eliminating movements which would invalidate some seconds of data before and after the transition.

Using the users annotated transports, this real world validation showed that the implemented API has and accuracy of 89%.

7.2 Comparing Against Activity Recognition APIs

The comparison of my implementation against Google's and Intel's APIs were done using the same data from the validation, as described in the previews subchapter.

The obtained values of each API were saved in a file. This file had in each data point the following details:

- **Annotated transport mode** – selected by the user in the smartwatch, corresponds to the actual transport mode of the user
- **Google's API transport mode prediction** – corresponds to the detected transport mode by "Google Activity Recognition API"
- **Intel's API transport mode prediction** – corresponds to the detected transport mode by "Intel Activity Recognition API"

Using the annotated transport modes, in the smartwatch, by the user and comparing them with the results obtained by each API, the accuracy of each one can be easily calculated and compared with each other.

Table 11 – Evaluation of the solution Accuracy (%) by API

Transport	SmarTrans	Google's API	Intel's API
Walk	98.48	96.44	1.35
Bus	98.67	87.61	45.57
Car	80.01	81.22	43.18
Metro	95.36	38.37	N/A
Inactive	100.0	98.82	88.24
Train	61.34	74.08	N/A
Overall	88.97	78.76	44.59

In Table 11 it is possible to compare the accuracy of each API, the results correspond to the percentage of correct classifications per transport. The results obtained by my implementation,

SmarTrans, are overall positive. The bus prediction manages to obtain such positive result once the implementation has into account the Wi-Fi access points, enabling to distinguish better between car and bus. On the other hand, the train prediction may have fallen to overfitting since only two people volunteered to do datasets of the train transport. These datasets were also done in Lisbon urban trains which may have different accelerations/decelerations and velocities when compared to the validation tests done in Porto.

The results obtained in this test, by Google's and Intel's APIs, confirm the results obtained in the preliminary test of these solutions. Comparing to these APIs, we may consider that the developed work is broader, allowing to discriminate more transports.

Of the two APIs evaluated, Google's API obtains a reasonable result with 78.76% overall accuracy. In this evaluation test, Google's API achieved better results with the classification of Car and Train transport modes. But we are considering that when Google's API classifies as "IN_VEHICLE" it is correct for any of the discriminated transports in Table 11, therefore there is no distinction between Car, Bus, Metro and Train.

The big reason why Intel accuracy is so low is because the classifier distinguishes between running and walking, and in the test it is almost always classified as running. If we were to consider the classification "RUNNING" as correct, Intel would obtain 81.76% accuracy for the walking activity.

7.3 Comparing Against Other Studies

Comparing to the state of the art, despite that (Aguiar, Silva, et al. 2014; Figueira et al. 2016) achieved better accuracy results, both of these studies classify activities such as running, tilting, standing and walking. The developed work discriminates different activities which are more similar to each other.

The most identical study (Widhalm et al. 2012), regarding the transports detected, achieves 75.87% accuracy whereas the solution presented here obtains 96.1% accuracy. This solution combines GPS signals with accelerometer signals and additionally detects biking and motorcycling. When the GPS signal is lost, a rough location estimate is derived from the cellular network. Additionally, by using

the GPS as a source of data, misclassifications between car and train and between motorcycle and bus occur depending on driving style, route and traffic state.

Another similar study (Reddy et al. 2010), additionally detects biking and running but doesn't distinguish between the different motorized transports. Even so, the accuracy obtained was about 2.5% lower than the achieved in this work.

8 Conclusion

Human physical activity monitoring has received an increasing interest from health related professionals, such as nutritionists, physiotherapists and elders' caregivers as well as sport oriented and safety areas.

Relying on the power and ubiquity of modern smartphones for HAR purposes is a very attractive proposition that previous researchers have explored. In this work, it is presented the development of a classifier for transport mode detection that uses the smartphone inertial sensors and wireless communication capabilities.

Due to an increase in average life expectancy, the aging of the population is noticeable and consequently there is a reduction in social and economic conditions for elderly daily care. With this in mind, more attention is being given to remote care systems to assist the patients and help them take care of themselves, lowering the conventional health care necessity.

Concerning elderly care, this implementation monitors the most requested modes of transport by this population, such as walking, standing still/sitting, buses, metro/train and car.

The training dataset was created using over 24 hours of transportation data from 15 different users. Using this dataset, the classification algorithms referred in the state of the art were evaluated with the resort of statistical tests. The chosen classification algorithm was then developed.

8.1 Achievements

- **State of the art improvement** – The API solution was put to the test by being compared with existing solutions and studies, obtaining as much as 10% and 40% overall accuracy above Google's and Intel's APIs, respectively. The most identical study (Widhalm et al. 2012), regarding the transports detected, achieves 75.87% accuracy whereas the solution presented here obtains 96.1% accuracy.
- **Obtain at least 85% accuracy** – the algorithm detects the transport modes and Weka classification shows it can achieve over 96.1% overall accuracy. Nevertheless, a real world validation of the implemented system obtained 88.97% overall accuracy;
- **Usage of smartphone features common even in low end devices –** the detection of the transport mode is obtained with the use of accelerometer sensors and Wi-Fi. Such features are found even in low-end devices;
- **Create a transport API -** The transport mode detection is implemented as a library to be used by Fraunhofer Portugal for their projects. It hides the complexity behind the API providing a simple interface to the client to access its functionalities;
- **Demonstrate the functionality of the transport API -** To demonstrate the functionality of the transport mode detection API a proof of concept system has been developed with the responsibility of registering the transport-related activities of an elder, therefore allowing the caregiver to learn about the person's activity habits.

8.2 Future Work

The presented work leaves some topics that require additional research and can be explored in the future, that are described in the following paragraphs:

- **Obtain additional data** - More data means an improved classification for these kinds of systems. More datasets from different users should be added to the classifier, enabling the study of the system performance for large populations and reducing possible “overfitting” problems.
- **Detect initial acceleration** - Detecting the initial acceleration of each transport mode can help improve overall performance by giving more confidence to the classification.

References

(Aguiar, Silva, et al. 2014)	Aguiar, B. et al., 2014. Monitoring physical activity and energy expenditure with smartphones. IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), pp.664–667. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6864451.
(Aguiar, Rocha, et al. 2014)	Aguiar, B., Rocha, T. & Silva, J., 2014. Accelerometer Based Fall Detection for Smartphones. IEEE International Symposium on Medical Measurements and Applications (MeMeA).
(Anjum & Ilyas 2013)	Anjum, A. & Ilyas, M.U., 2013. Activity recognition using smartphone sensors. 2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013, pp.914–919. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84875973665&partnerID=40&md5=fca0baa3cb0519c323e39180c838e26e.
(Antonello & Oboe 2012)	Antonello, R. & Oboe, R., 2012. Exploring the potential of MEMS gyroscopes: Successfully using sensors in typical industrial motion control applications. IEEE Industrial Electronics Magazine, 6(1), pp.14–24.
(Balasubramanian et al. 2013)	Balasubramanian, A., LaMarca, A. & Wetherrall, D., 2013. Efficiently Running Continuous Monitoring Applications on Mobile Devices using Sensor Hubs. University of Washington Technical Report, (1). Available at: http://mobilehub.cs.washington.edu/papers/sensorhub.pdf.
(Banos et al. 2014)	Banos, O. et al., 2014. Window size impact in human activity recognition. Sensors (Basel, Switzerland), 14(4), pp.6474–99. Available at: http://www.mdpi.com/1424-8220/14/4/6474/htm.
(Bohanec & Bratko 1994)	Bohanec, M. & Bratko, I., 1994. Trading Accuracy for Simplicity in Decision Trees. Machine Learning, 15(3), pp.223–250.
(Breiman et al. 1984)	Breiman, L. et al., 1984. Classification and Regression Trees
(Carneiro et al. 2012)	Carneiro, R. et al., 2012. O Envelhecimento da População: Dependência, Ativação e Qualidade,
(CEDRU & BCG 2008)	CEDRU & BCG, 2008. Estudo de Avaliação das Necessidades dos Seniores em Portugal,
(Developer.android.com 2015)	Developer.android.com, 2015. Sensors Overview | Android Developers. Available at: https://developer.android.com/guide/topics/sensors/sensors_overview.html [Accessed December 20, 2015].
(Figueira 2015)	Figueira, C.R., 2015. Body Location Independent Activity Monitoring.
(Figueira et al. 2016)	Figueira, C.R., Matias, R. & Hugo, G., 2016. Body Location Independent Activity Monitoring. 9th International Joint Conference on Biomedical Engineering Systems and Technologies, pp.190–197.
(Gomes 2015)	Gomes, E., 2015. Experimentação e Avaliação. Available at: https://moodle.isep.ipp.pt/pluginfile.php/89587/mod_resource/content/8/Experimenta%C3%A7%C3%A3o e Avalia%C3%A7%C3%A3o.pdf [Accessed October 18, 2016].
(Google 2016a)	Google, 2016. ActivityRecognitionApi. Available at: https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi [Accessed June 7, 2016].

(Google 2016b)	Google, 2016. DetectedActivity | Google APIs for Android | Google Developers. Available at: https://developers.google.com/android/reference/com/google/android/gms/location/DetectedActivity [Accessed December 17, 2015].
(Google 2016c)	Google, 2016. Overview of Google Play Services | Google APIs for Android | Google Developers. Available at: https://developers.google.com/android/guides/overview [Accessed December 17, 2015].
(Grankin et al. 2012)	Grankin, M., Khavkina, E. & Ometov, A., 2012. Research of MEMS Accelerometers Features in Mobile Phone. 12th conference of Open Innovations Association FRUCT, pp.31–36.
(Hemminki et al. 2013)	Hemminki, S., Nurmi, P. & Tarkoma, S., 2013. Accelerometer-based transportation mode detection on smartphones. Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems - SenSys '13, pp.1–14. Available at: http://dl.acm.org/citation.cfm?doid=2517351.2517367.
(Intel 2016)	Intel, 2016. Context States Datasheet. Available at: https://software.intel.com/en-us/articles/sensing-context-states-datasheet [Accessed June 7, 2016].
(Kavanagh & Menz 2008)	Kavanagh, J.J. & Menz, H.B., 2008. Accelerometry: A technique for quantifying movement patterns during walking. Gait and Posture, 28(1), pp.1–15.
(Kotsiantis et al. 2007)	Kotsiantis, S., Zaharakis, I. & Pintelas, P., 2007. Supervised machine learning: A review of classification techniques. Available at: https://books.google.com/books?hl=en&lr=&id=vLiTXDHr_sYC&oi=fnd&pg=PA3&dq=machine+learning+classifiers&ots=CXrwsAYKjm&sig=CdxWD-yYrFm8yB9ZINbBNhHhbhg [Accessed September 27, 2016].
(Kwapisz et al. 2011)	Kwapisz, J., Weiss, G. & Moore, S., 2011. Activity recognition using cell phone accelerometers. ACM SIGKDD Explorations …, 12(2), pp.74–82. Available at: http://dl.acm.org/citation.cfm?id=1964918.
(Laskov et al. 2005)	Laskov, P. et al., 2005. Learning Intrusion Detection: Supervised or Unsupervised? Image Analysis and Processing–ICIAP, pp.50–57.
(Lopes et al. 2012)	Lopes, A., Moreira, J. & Gama, J., 2012. Semi-supervised learning: predicting activities in Android environment. Workshop on Ubiquitous Data Mining, p.38.
(McCallum & Nigam 1998)	McCallum, A. & Nigam, K., 1998. A comparison of event models for naive bayes text classification. AAAI-98 workshop on learning for text. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.9324&rep=rep1&type=pdf [Accessed September 28, 2016].
(Quinlan 1986)	Quinlan, J.R., 1986. Induction of Decision Trees. Machine Learning, 1(1), pp.81–106. Available at: http://link.springer.com/10.1023/A:1022643204877 [Accessed September 27, 2016].
(Reddy et al. 2010)	Reddy, S. et al., 2010. Using mobile phones to determine transportation modes. ACM Transactions on Sensor Networks, 6(2), pp.1–27.
(Rokach & Maimom 2014)	Rokach, L. & Maimom, O., 2014. Data mining with decision trees: theory and applications
(Skok 2013)	Skok, M., 2013. 4 Steps To Building A Compelling Value Proposition. Available at: http://www.forbes.com/sites/michaelskok/2013/06/14/4-steps-to-building-a-compelling-value-proposition/#62c277e91f2c [Accessed February 2, 2016].
(Texas Instruments 2016)	Texas Instruments, 2016. Simplelink SensorTag - TI.com. Available at: http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/tearDown.html [Accessed November 25, 2015].

(Viana 2010) Viana, J., 2010. Porto: cidade amiga das pessoas idosas: um estudo centrado na perspectiva de idosos das freguesias de Miragaia e Vitória.

(Widhalm et al. 2012) Widhalm, P.I. of T., Nitsche, P.I. of T. & Brändle, N.I. of T., 2012. Transport Mode Detection with Realistic Smartphone Sensor Data. Icpr, (Icpr), pp.573–576.

(Wilde 2010) Wilde, A., 2010. An overview of human activity detection technologies for pervasive systems. Seminar paper, Department of Informatics, University of …. Available at: https://diuf.unifr.ch/main/pai/sites/diuf.unifr.ch.main.pai/files/education_seminar_activityrecognition_awilde.pdf.

(Witten et al. 2011) Witten, I.H., Frank, E. & Hall, M. a., 2011. Data Mining: Practical Machine Learning Tools and Techniques, Third Edition, Available at: http://www.cs.waikato.ac.nz/~ml/weka/book.html\nhttp://www.amazon.com/Data-Mining-Practical-Techniques-Management/dp/0123748569.

(Yang 2009) Yang, J., 2009. Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones. Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics, pp.1–9. Available at: http://doi.acm.org/10.1145/1631040.1631042.

(Yang & Honavar 1998) Yang, J. & Honavar, V., 1998. Feature Subset Selection Using a Genetic Algorithm. In Feature Extraction, Construction and Selection. Boston, MA: Springer US, pp. 117–136. Available at: http://link.springer.com/10.1007/978-1-4615-5725-8_8 [Accessed September 25, 2016].

A Performance Evaluation - Additional Confusion Matrix

In this appendix are presented additional confusion matrices obtained during testing and evaluation.

Using a 3 seconds time window, the J48 classifier obtained 95.8% precision.

```
    a    b    c    d    e    f   <-- classified as
 4792    9   45   70    1    5 |    a = CAR
   17 4602   95  204   16   38 |    b = METRO
   28  129 4633   35   31   13 |    c = TRAIN
   88  196   43 4531   30   19 |    d = BUS
    2   26   32   27 4909    3 |    e = WALK
    2   30    3   10    8 4881 |    f = INACTIVE
```

Figure A.1 – Confusion Matrix for J48 classifier with 3 second Window

Using a 7 seconds time window, the J48 classifier obtained 93.9% precision.

```
   a    b    c    d    e    f   <-- classified as
2005   11   11   39    0    5 |    a = CAR
  19 1848   61   97   13   21 |    b = METRO
  26   72 1863   32   12   16 |    c = TRAIN
  39  117   42 1838   16    6 |    d = BUS
   5   11   17   20 1977    2 |    e = WALK
   8   18    0   14    2 2031 |    f = INACTIVE
```

Figure A.2 - Confusion Matrix for J48 classifier with 7 second Window

```
   a    b    c    d    e    f   <-- classified as
2646   21  122  124    0   13 |    a = CAR
  15 2273  207  137    2  297 |    b = METRO
  97  327 2214   54    9  179 |    c = TRAIN
 207  357   76 1982   14  274 |    d = BUS
   0   30   29   26 2818   21 |    e = WALK
   0   47  345    9    3 2527 |    f = INACTIVE
```

Figure A.3 – Confusion Matrix for SVM classifier with feature selection

Without feature selection nor pruning the tree had 661 leafs.

```
   a    b    c    d    e    f   <-- classified as
2836   15   14   55    1    5 |    a = CAR
  13 2656   77  147   13   25 |    b = METRO
  23  104 2696   26   19   12 |    c = TRAIN
  55  126   32 2668   16   13 |    d = BUS
   2   13   25    9 2873    2 |    e = WALK
   7   17    6    7    2 2892 |    f = INACTIVE
```

Figure A.4 – Confusion Matrix for J48 classifier without tree pruning nor feature selection

Without feature selection, but with tree pruning the tree had 421 leafs.

```
   a    b    c    d    e    f   <-- classified as
2839   12   18   50    1    6 |    a = CAR
  16 2645   83  147   10   30 |    b = METRO
  28  103 2685   37   14   13 |    c = TRAIN
  65  136   35 2640   18   16 |    d = BUS
   2   16   25    9 2868    4 |    e = WALK
   8   18    6   13    3 2883 |    f = INACTIVE
```

Figure A.5 – Confusion Matrix for J48 classifier without feature selection

B Value Analysis

This work proposes to develop a platform for the detection of the transport mode, mainly using the inertial sensors. This platform is aimed to be used by Android developers with the intent of adding context awareness into their applications, allowing the developers to adapt the application behaviour based on information sensed from the physical environment. The product built upon this platform is targeted to the Health Care segment and has the intent of helping sustain the autonomy of the elders by assisting the caregivers, medics, nurses and kin, analyse the daily activities of the patients.

Value proposition is a “positioning statement that explains what benefit you provide for who and how you do it uniquely well.” (Skok 2013). Creating a value proposition is a part of business strategy that sets the strategy of the company to get new customers by defining the product, the target customers, the value provided to the customer and why the product is unique. The definition of value can be understood as being a balance between benefits (attributes or outcomes) and sacrifices (costs, time, effort, etc.).

The customer perceives the value by evaluating the benefits offered by the company (in form of products or services) and the cost (money, effort, time). The perceived value is the value that each customer has of the same product/service, the organizations involved can also have different perceptions of value.

Product Value

The product is a platform for detecting the transportation mode mainly using the accelerometer sensor. This product can be used by two different segments: health care and developers.

The benefits for the health care segments are the following:

- Help sustain the autonomy of the elders by assisting the caregivers, medics, nurses and kin, analyse the daily activities of the patients.
- Reduce elder's interaction with the computer since he doesn't need to send the daily activities by hand to the caregivers.

The sacrifice for the health care segment is the monthly subscription for caregivers and health care centres.

For the developers segment the benefits are the following:

- Add context awareness into their applications, allowing the developers to adapt the application behaviour based on information sensed from the physical environment.

The value sacrifice for the developers' segment is the subscription fee for its usage.

Negotiation Scenarios

A negotiation requires the existence of two or more participants, each of them with individual goals that may not be totally compatible but both parties share the purpose to reach an agreement. There is usually a process which involves several alternatives to be investigated.

A negotiation can end up with different outcomes:

- Win-Win: both parties end up benefited from the negotiation and within their target ranges (e.g., monetary value, business relationships, internal structures, human competence, environmental responsibility, social responsibility).
- Win-lose: One side of the negotiation profits at the other's expense. The agreement favors only one of the parties with no compensation to the other party.

- Lose-lose: Neither side achieves a desirable result. Failed negotiations are frequently considered a lose-lose situation and the parties are forced to seek alternative solutions. Alternatively, both parties can be too quick to make concessions, reaching a compromise that is fair, but detrimental to both sides.

Canvas

The business model defines how to combine the means to deliver value to the interested parts and capture value to the organization. The business model responds to questions as: "Who are the clients?", "What do they valorize?", and "How to reach the client?". A canvas model is a systematic and practical way to answer those questions.

Table B.1 - Business Model Canvas

<table>
<tr><td rowspan="2"><u>Key Partners</u></td><td><u>Key Activities</u></td><td rowspan="2"><u>Value Propositions</u></td><td rowspan="2"><u>Customer Relationships</u></td><td rowspan="2"><u>Customer Segments</u></td></tr>
<tr><td>Research And Development (R&D)

Android application development
Web application development

Accelerometer sensors dataset analysis</td></tr>
<tr><td rowspan="2">Health care infrastructures</td><td><u>Key Resources</u></td><td rowspan="2">A platform for detecting the transportation mode mainly using the accelerometer sensor

Help sustain the autonomy of the elders

Assist medics and nurses by analyzing the daily activities of the patients</td><td rowspan="2">Pre-sales service (Explain to health care infrastructures how to use the system)

3rd party APP</td><td rowspan="2">Elders

Elders' Caregivers (kin or medic)

Android Developers</td></tr>
<tr><td>Software

Mobile device

Server</td></tr>
<tr><td colspan="3"><u>Cost Structure</u></td><td colspan="2"><u>Revenue Streams</u></td></tr>
<tr><td colspan="3">Research And Development (R&D)

Server infrastructure costs</td><td colspan="2">Monthly subscriptions for caregivers and health care centers

Android API subscriptions fees</td></tr>
</table>

Analytical Methods of value creation

A Conceptual Model can be used to decompose the value for the customer. A Conceptual Model can be built on the combination of the following concepts: the concept of Forms of value and Value temporal positions; the concept of Value Network and on the network exchange of tangible and intangible deliverables among the network roles; and the Perceived Benefits (PB)/Perceived Sacrifices (PS).

To create value is necessary to take a series of measures, increase the benefits and/or decrease the sacrifices. The conceptual models and quantitative methods can help determine the best approach to take, given the conditions, which optimize the outcome (higher value). There are several methods which can aid analyzing and quantifying the creation of value, among them there is the Rough Set Theory, the Fuzzy Set Theory, the Game Theory, some Multi-criteria decision analysis (MCDA) such as Analytic hierarchy process (AHP), Measuring attractiveness by a categorical based evaluation technique (MACBETH) and Preference ranking organization method for enrichment evaluations (PROMETEE).

Game theory

Game Theory is a useful tool "that models situations where players (participants in a game) participate in a strategic situation (the game) in which they perform different actions attempting to maximize their profits, while at the same time minimize losses." (Schillings & Yang 2009).

AHP

AHP is a MCDA structured technique for dealing with complex decisions. AHP has three main phases starting by decomposing the decision problem into a hierarchy of sub problems. Then, the various criteria are compared to evaluate its relative importance. In the third phase, the AHP converts these evaluations to numerical values (weights or priorities), which are used to obtain the global priority for each alternative.

C Proof of Concept System Design

The "core" and main focus of the work is the API and this system is more a matter of proof of concept. Using the Transport Mode detection API, the Proof of concept system demonstrates the functionality of the implementation by registering the user's transport-related activities on the server and allows to check the activities history on the website.

Since it is a proof of concept system there is no user management. There are no passwords and if the entered username does not exist the system automatically creates one.

C.1 Use Cases

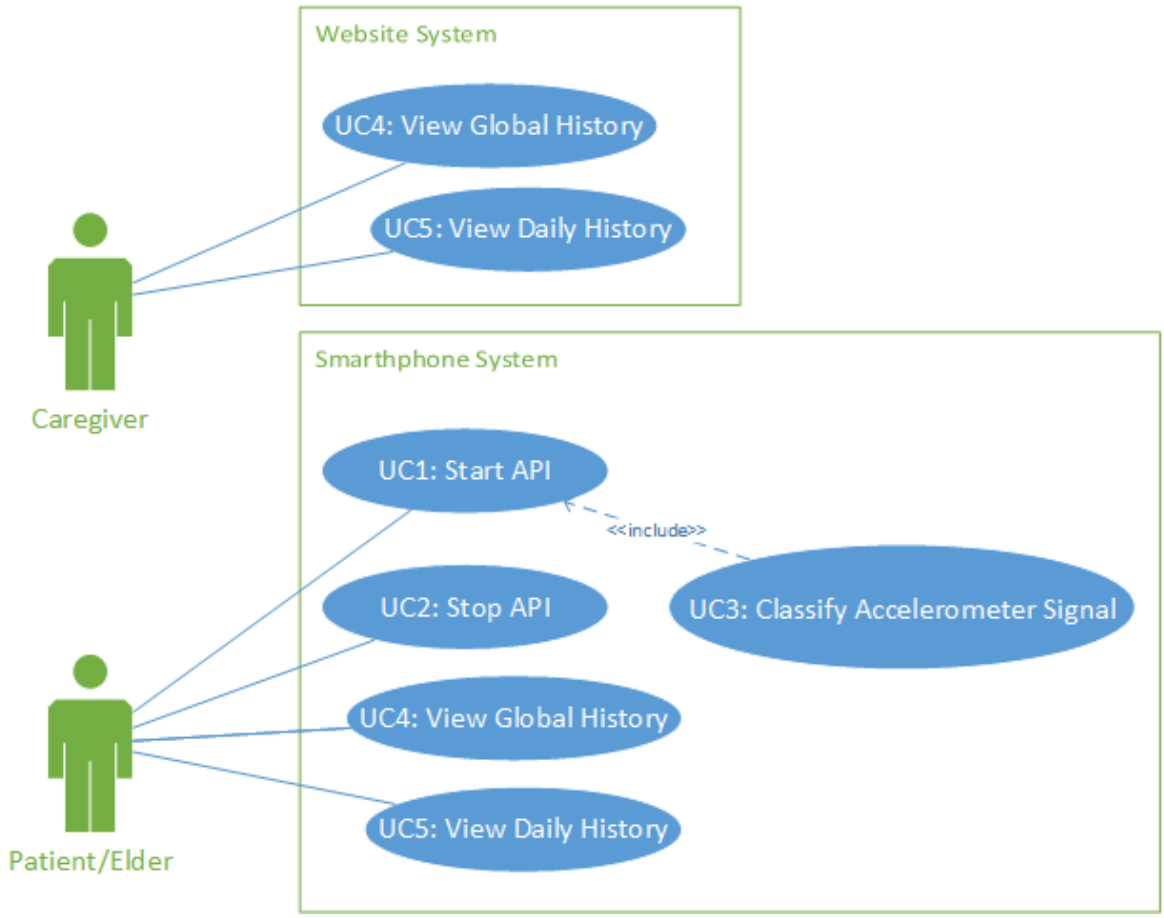


Figure C.1 - Use Case Diagram

C.1.1 UC4: View Global History

Goal of Use Case:	**Present the Elder/Caregiver the global history of the elder.**
Preconditions:	none
Success Post Conditions:	The actor is presented with the elder history
Failed Post Conditions:	none
Actors:	Elder, Caregiver, Smartphone System, Website System
Triggers:	This process is started by the Elder or the Caregiver (human interaction)

C.1.1.1 Main Success Path

Step	Actor	Description	Branches	
			Condition	**Location**
1.	Elder, Caregiver	The Elder/Caregiver navigates to global history page	Empty history	ALT1
2.	Smartphone System, Website System	Present the global history of the transport modes used by the elder		
3.	Smartphone System, Website System	The use case ends		

C.1.1.2 Alt 1: Empty History

Step	Actor	Description	Branches	
			Condition	**Location**
1.	Smartphone System, Website System	The System Informs the elder/caregiver that no information is available		
2.	Smartphone System, Website System	The use case ends		

C.1.1.3 Activity Diagram

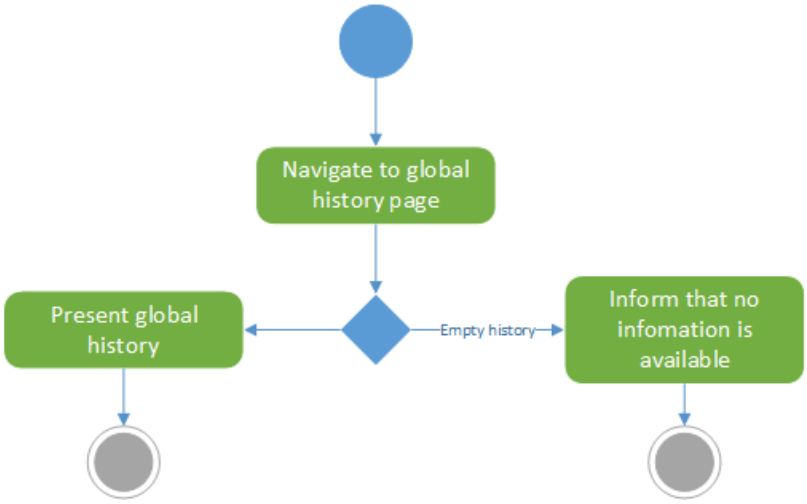


Figure C.2 – Activity diagram for Use Case 4

C.1.1.4 Non-Functional Requirements and Assumptions

None.

C.1.2 UC5: View Daily History

Goal of Use Case:	**Present the Elder/Caregiver the daily history of the elder.**
Preconditions:	none
Success Post Conditions:	The actor is presented with the elder's detailed history of the selected day
Failed Post Conditions:	none
Actors:	Elder, Caregiver, Smartphone System, Website System
Triggers:	This process is started by the Elder or the Caregiver (human interaction)

C.1.2.1 Main Success Path

Step	Actor	Description	Branches	
			Condition	**Location**
1.	Elder, Caregiver	The Elder/Caregiver navigates to daily history page		
2.	Elder, Caregiver	The Elder/Caregiver selects the desired date	No data available	ALT1

3.	Smartphone System, Website System	Present the history of the transport modes used by the elder of the selected day		
4.	Smartphone System, Website System	The use case ends		

C.1.2.2 Alt 1: No Data Available

Step	Actor	Description	Branches	
			Condition	**Location**
1.	Smartphone System, Website System	The System Informs the elder/caregiver that no information is available		
2.	Smartphone System, Website System	The use case ends		

C.1.2.3 Activity Diagram

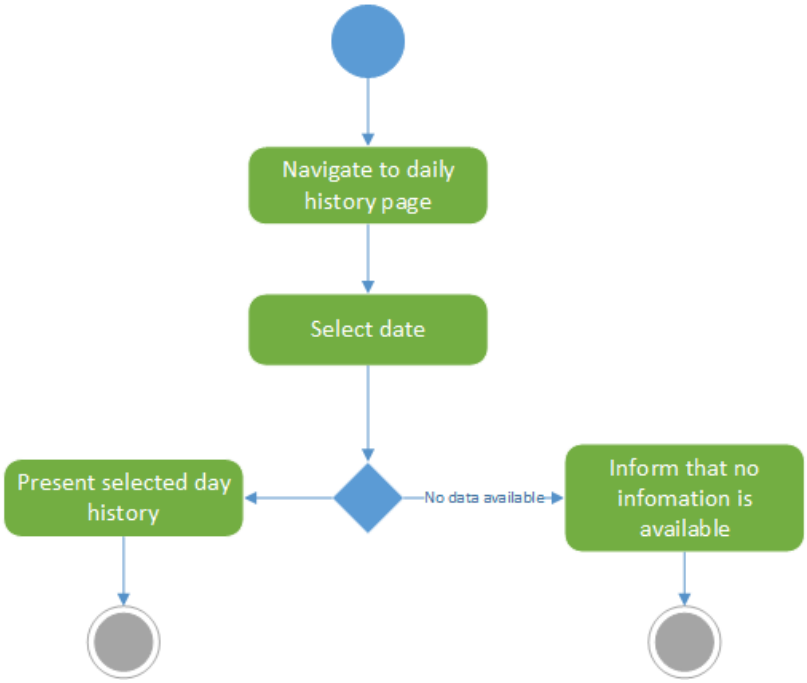


Figure C.3 – Activity diagram for Use Case 5

C.1.2.4 Non-Functional Requirements and Assumptions

None.

C.2 Deployment Diagram

Figure C.4 shows a conceptual view of the prototype system.

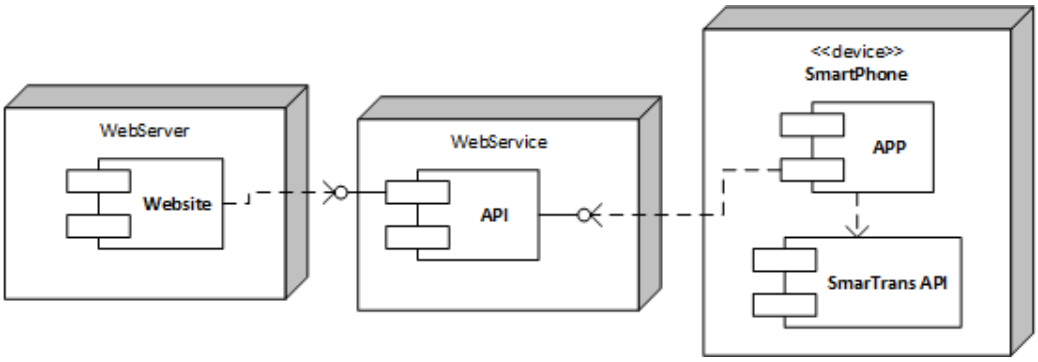


Figure C.4 - Prototype Deployment Diagram

C.3 Web Service

The following operations/message types are supported, and follow the synchronous request/reply scenario:

Table C.1 - Database Operations (name of operations are merely indicative)

Operation	Content-Type	HTTP Method
getGlobalData	application/json	GET
getDailyData	application/json	GET

C.3.1 GetGlobalData

Refer to UC (Use Case) 4. A call to this operation results in the Web Service sending over the list of the total utilization of each transport by the user.

The input parameters are sent in json format in the URI with the following fields:

Table C.2 – getGlobalData Request Parameters

Name	Description	Type
AccountId	The account identifier	String

Output parameters are sent in json format in the Response body with a list of the following fields:

Table C.3 – getGlobalData Response Parameters

Name	Description	Type
TransportIdentifier	A transport identifier	String
TimeRecorded	Time recorded in seconds	int

C.3.2 GetDailyData

Refer to UC (Use Case) 5. A call to this operation results in the Web Service sending over the list of the utilization of each transport by the user of a specified day.

The input parameters are sent in json format in the HTTP body with the following fields:

Table C.4 – getDailyData Request Parameters

Name	Description	Type
AccountId	The account identifier	String
Day	Is a day identifier with the format 'yyyymmdd'	int

Output parameters are sent in json format in the Response body with a list of the following fields:

Table C.5 – getDailyData Response Parameters

Name	Description	Type
TransportIdentifier	A transport identifier	String
TimeStarted	Timestamp of the beginning of the identification of this transport	long
TimeRecorded	Time recorded in seconds	int
TimeSitting	Time siting in seconds	int
TimeStanding	Time standing in seconds	int

C.4 Sequence Diagrams

The sequence diagrams below describe how the groups of objects interact with each other.

The interaction between these objects follows a Model View Presenter (MVP) Architecture. MVP is a derivation of the Model View Controller (MVC) architectural pattern. The presenter does all the logic leaving the view with no logic inside them, only UI logic. Unlike the typical MVC, the presenter also decides what happens with the view when the user interacts with it.

C.4.1 UC4: View Global History

Use case 4 maps to the interactions between the BalanceActivity, Presenter and Model.

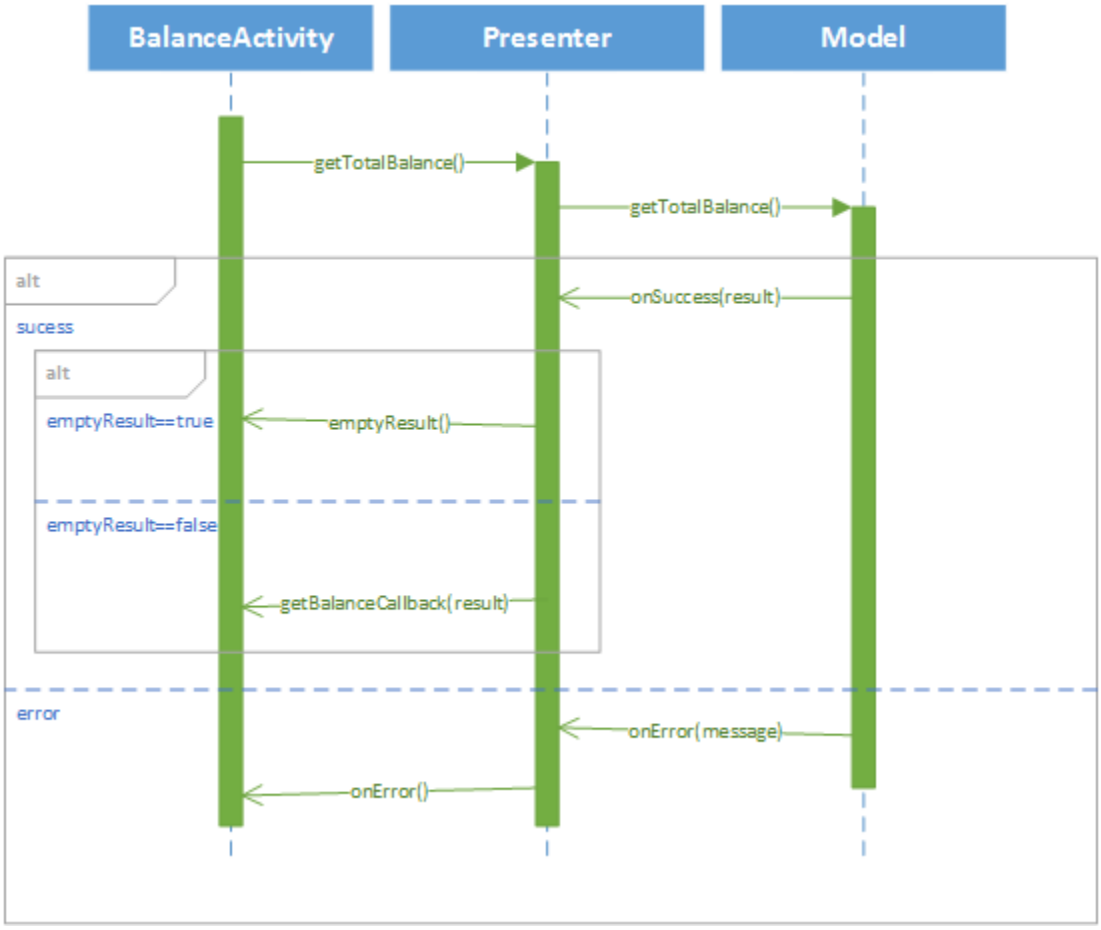


Figure C.5 - Sequence Diagram for Use Case 4

C.4.2 UC5: View Daily History

Use case 5 maps to the interactions between the HistoryActivity, Presenter and Model.

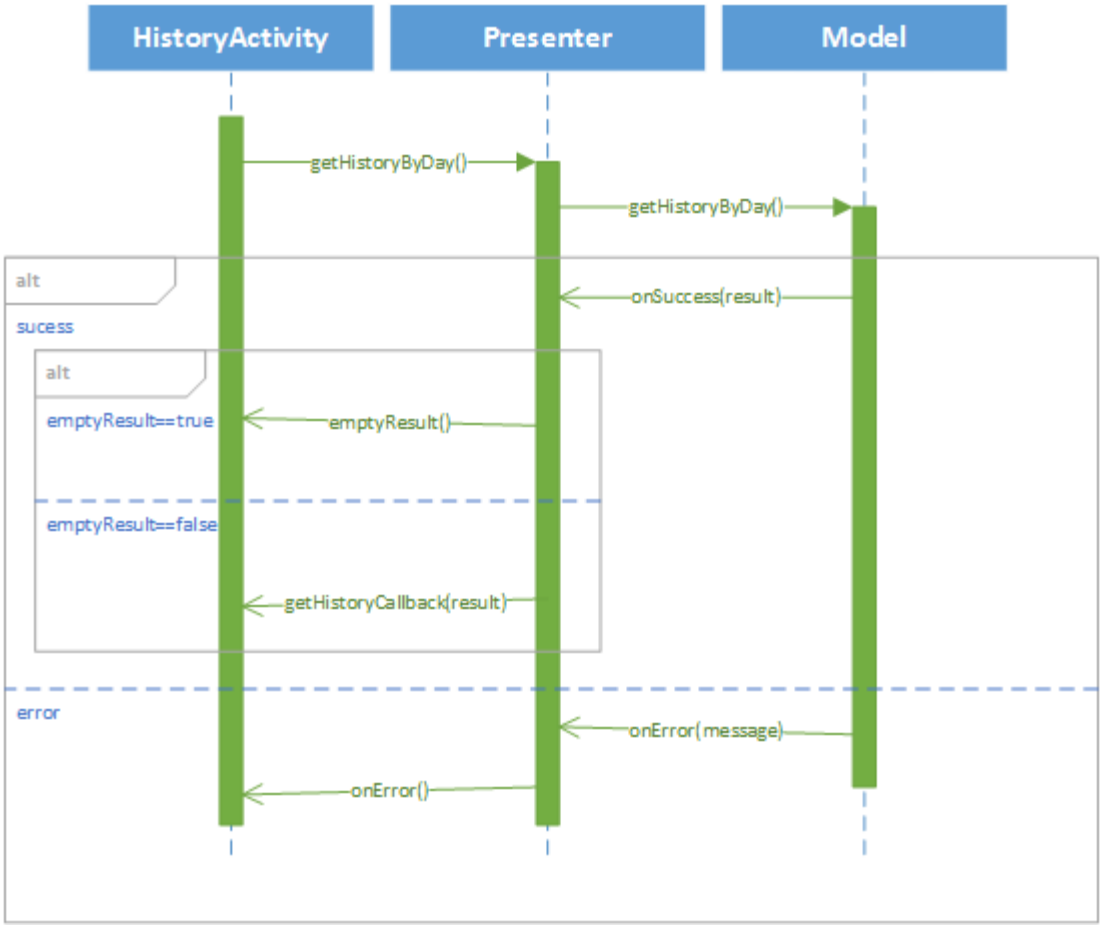


Figure C.6 - Sequence Diagram for Use Case 5

C.5 Database

Replication and synchronization capabilities of CouchDB make it ideal for using it in mobile devices, where network connection is not guaranteed, but the application must keep on working offline. CouchDB is well suited for applications with accumulating, occasionally changing data, on which pre-defined queries are to be run and where versioning is important.

CouchDB it is a document-oriented NoSQL database that uses JSON to store data, JavaScript as its query language using MapReduce, and HTTP for an API. And so it uses the HTTP methods POST,

GET, PUT and DELETE for the four basic CRUD (Create, Read, Update, Delete) operations on all resources.

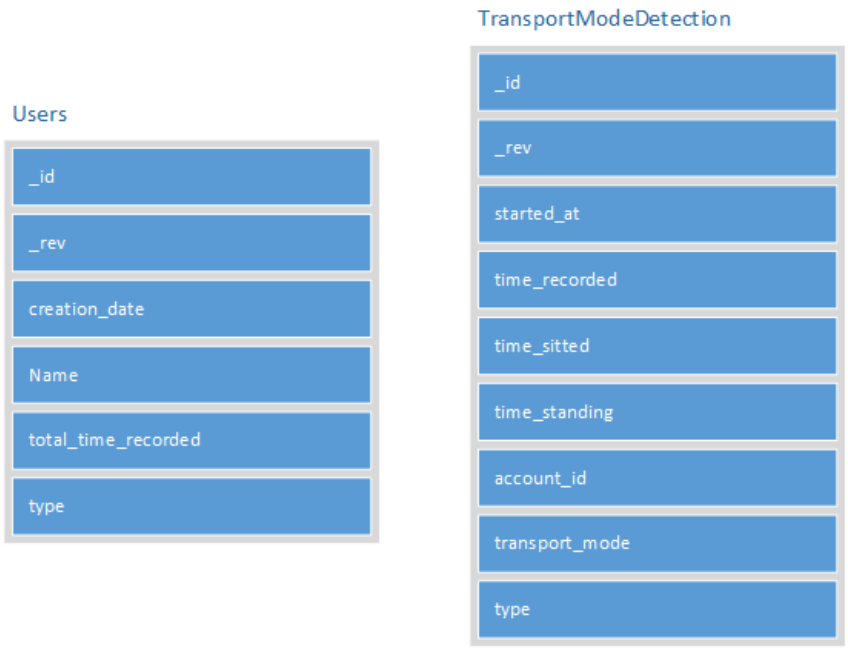


Figure C.7 - Entity Relationship Diagram

“The first two members (_id and _rev) are for CouchDB’s housekeeping and act as identification for a particular instance of a document.” (Couch DB Guide 2016)

The field "_id" is a unique identifier of the document. The field "_rev" (revision Id) describes the version of a document. Each change creates a new document version (that again is self-contained) and updates the _rev.

D Conference Paper

In this appendix is presented the publication, Smartphone-based Transport Mode Detection for Elderly Care, which demonstrates the work developed in this thesis. This article was submitted and accepted for publication at the 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (IEEE Healthcom), held in Munich in September 2016.

Smartphone-based Transport Mode Detection for Elderly Care

Nuno Cardoso[1,2], João Madureira[1]
[1]Fraunhofer Portugal AICOS
Porto, Portugal
1100644@isep.ipp.pt, joao.madureira@fraunhofer.pt

Nuno Pereira[2]
[2]Computer Engineering Department of the
School of Engineering of the Polytechnic of Porto
Porto, Portugal
nap@isep.ipp.pt

***Abstract*—Smartphones are everywhere, and they are a very attractive platform to perform unobtrusive monitoring of users. In this work, we use common features of modern smartphones to build a human activity recognition (HAR) system for elderly care. We have built a classifier that detects the transport mode of the user including whether an individual is inactive, walking, in bus, in car, in train or in metro. We evaluated our approach using over 24 hours of transportation data from a group of 15 individuals. Our tests show that our classifier can detect the transportation mode with over 90% accuracy.**

Keywords - Inertial Sensors, Accelerometer, Classification Algorithms, Transport Detection, Smartphones

I. INTRODUCTION

Due to their popularity, processing and sensing capabilities, smartphones have the potential to become an "electronic stethoscope", helping health professionals achieve better diagnosis and treatment. In this work, we use the capabilities of common smartphones to build a system capable of performing unobtrusive Human Activity Recognition (HAR) and allow the caregiver, a kin or a medic, to learn about the daily activity patterns of elderly people.

Specifically, we detect the transport mode of the smartphone carrier, which is an important tool to access, for example, if the person is having a sedentary life, if there are changes in the daily movement patterns, provide indication about lack of engagement in social activities, if they are sufficiently independent in their everyday life, or doing enough exercise.

While our target application is elderly healthcare, being able to collect information about how means of transportation are used is useful for a number of applications, such as monitoring/controlling disease [1], urban planning [2], or social interactions [3], just to name a few. It is important to consider that, by targeting a medical care scenario, where patients are monitored under their consent, we avoid important security and privacy implications.

Using the smartphone for HAR is not a new idea. A number of works have already addressed the issue of transport mode detection (walking, car, bus, metro) using GPS, inertial sensors, and barometric sensors available in smartphones (e.g. [4]–[6]). Recently, Google [7] and Intel [8] have released libraries for detection of the mode of transport and activity recognition, that may be incorporated in an application to add context-aware capabilities. We later present an evaluation of these libraries showing that their performance is still lacking.

In our work, we focus on using inertial sensors and take advantage of the surrounding wireless access points (WAP) to improve our estimates. We prefer this method instead of using GPS, which is often unavailable (indoors, such as inside buildings, at home, underground or doctor's office), making it impossible to get the user location. By identifying surrounding WAP, it is possible to obtain a rough location (between 10 and 50 meters accuracy) using available public services. Furthermore, given this rough location, the details about nearby stations and bus stops can be gathered, and used to improve our estimates about the user transport mode.

Finally, we also incorporate an additional indication of the transportation mode by identifying WAP from the vehicle. Nowadays, it is common to find WAP in buses, trains or underground, and identifying these is often trivial (as we will detail later in this work).

We have collected over 24 hours of mobility data from a group of 15 users to train and test different classifiers, with different inputs, and show that our prototype elderly care application can accurately detect the transportation mode with 95.6% accuracy. More specifically, as depicted in Fig. 1, the transportation modes identified by this solution include whether an individual is inactive, walking, in bus, in car, in train or in metro. The inactive mode is detected whenever the user is standing still, sitting or the phone is laying on a table.

The remainder of this document is organized as follows. Section II reports some of the works related with this study, while in Section III we look into available activity recognition libraries and briefly analyze their features and performance. Section IV details the design and main objectives of this work.

Fig. 1. Recognized activities

Section V describes the developed algorithms and the methodologies followed. Section VI contains the classifiers evaluation and performance results. Finally, Section VII, contains the discussion of future work along with the conclusion.

II. Related Work

We classify the related work by the resources of the smartphone used. We start by looking at works that employ GPS signals, then accelerometers only, and then a combination of sensors. This section is complete with a summary of the accuracy achieved by the relevant previous works.

Detection using GPS

Some current systems use mainly the GPS signal to perform this task and can only distinguish between general modes (walking, running, car). Within areas where this signal is weak or non-existent, these systems do not work properly. The inertial sensors, mainly the accelerometer, can be used to overcome these problems.

A method of detection of the transport mode was proposed in [4] which generates the features for classification from GPS readings and accelerometer data. This approach also includes position data from cell networks to prevent the system to malfunction when there are GPS signal losses. When the trajectory cannot be accurately reconstructed, it relies on accelerometer features. To differentiate the transport modes (Bus, Car, Bike, Tram, Train, Subway, Walk and Motorcycle), a combination of classifier ensemble with Hidden Markov Model was used and an average accuracy of 75.8% was achieved.

Similarly, [5] also used GPS and accelerometer data and additionally used gyroscope data. Features were extracted from these signals in windows of 5 seconds and supplied data to the classifier. Several classifiers were tested with these features and it was determined that Decision Tree outperforms the other classifiers with a 94.4% precision and 94.2% recall rate (accuracy). This model detected 7 different physical activities, including walking, running, climbing stairs, and descending stairs, cycling, driving and remaining inactive.

Reddy [9] combine GPS and accelerometer to recognize between stationary, walking, running, biking and motorized transportation, achieving over 93% accuracy. Classification is performed with a hybrid classifier consisting of a decision tree and a first order discrete HMM classifier.

Detection Using Only Accelerometer Sensors

Recent studies present systems which only use accelerometer signals for HAR. Yang [6] presented a framework that performs physical motion recognition using smartphone accelerometer. The features were extracted from accelerometer signals and then used as input for the Decision Tree classifier, obtaining 87.6% accuracy for the detection of physical activities, such as sitting, standing, walking, running, driving and cycling. Another paper, [10] describes and evaluates a system using smartphone onboard accelerometer sensors for HAR. The data was collected at a frequency of 20Hz (50ms), divided into 10-second segments and six features were extracted from it. This study considered six activities, including walking, jogging, ascending stairs, descending stairs, sitting and standing, and it accomplished 90% accuracy using a Decision Tree classifier.

Another solution, proposed by Aguiar [11] detects physical activities, such as sitting, standing, walking, running and tilting. To optimize battery consumption when the user is motionless, the sampling frequency is reduced from 33.33Hz to 4Hz. A Decision Tree classifier was trained after extracting twelve features from accelerometer signals. This solution achieved an average accuracy of 99.5% for the pocket usage and 99.4% when the phone was used in the belt.

Another study on Accelerometer-Based Transportation Mode Detection on Smartphones [12] preprocessed and transformed the sensor values and with it, constructed horizontal and vertical gravity representations of the accelerometer measurements using a sliding window with 50% overlap and a duration of 1.2 seconds. Then the features were extracted from both vertical and horizontal representations and the classification relied on a three-stage hierarchical classification framework (kinematic, stationary and motorized classifier) for transportation mode detection. Each of the three classifiers considers a variant of AdaBoost as the instance-based classifier.

Detection Using a Combination of Sensors

In an attempt to achieve better results, another solution [13] combines two sensors, accelerometer, and barometer. The signals were obtained in windows of 5 seconds and with a frequency of 30Hz for the accelerometer and 5Hz to the barometer. Using a Decision Tree classifier to detect daily activities (walking, running, sitting, standing, going upstairs and downstairs) this framework obtained 94.53 ± 6.82% accuracy.

Summary of Previous Results

The accuracy of each study, presented in TABLE *I*, demonstrates an overall accuracy between 75% and 94%. Other solutions like [10] and [13] do not detect transport modes but detect instead activities such as sitting, standing, walking, running, tilting, going upstairs and downstairs, managing to achieve better results this way compared to the other studies.

The most identical study [4], regarding the transports detected, achieves 75.8% accuracy.

TABLE I. RESULTS OBTAINED BY EACH STUDY

Sensors Used	Study	Accuracy
GPS + Accelerometer	[4]	75.8%
	[5]	94.2%
	[9]	93.6%
Accelerometer	[6]	87.6%
	[10]	90%
	[11]	99.5%
	[12]	80.1%
Combination	[13]	94.53 ± 6.82%

III. Off-the-Shelf Activity Recognition Libraries

The recently released Google Activity Recognition [7] and Intel Activity Recognition [8] APIs are a readily available option for mode of transport and activity recognition. These libraries were developed to facilitate the development of context-aware applications, and provide a range of related functionalities. Prior to developing our own classifiers, we have evaluated these state-of-the-art libraries in order to assess their usefulness for our elderly care application.

We developed an application that periodically queried the users for their current transport mode (car, bus, walking, other) and collected the current activity estimate from the libraries. The results for 1 hour of each transport mode, while using a smartphone in the pocket, are presented in TABLE *II*. With an overall accuracy of 80.33%, Google's API performs really well when compared with the Intel's API, which only has 48.74% accuracy. The Intel's accuracy for Train and Metro transports is not shown since their API does not support those transports yet. The Intel's API accuracy for Walk activity is only 2.1% due to the fact that most of the times it classifies walking as running. Google's API ranks amongst the worst performers, when compared to other state of the art approaches in TABLE *I*.

TABLE II. APIS ACCURACIES (%)

	Google Activity Recognition API	Intel Activity Recognition API
Car	63.18	45.61
Bus	86.86	48.14
Train	72.75	N/A
Metro	62.65	N/A
Walk	97.22	2.10
Inactive	99.31	99.12
Overall	**80.33**	**48.74**

Google's API despite having a detection interval which can be defined by the developer, has an inconsistent frequency update due to several factors such as battery saving or in an attempt to get a more accurate prediction the API may delay the update indefinitely until it has a prediction with "enough" accuracy [7]. With the exception of the "still" activity, this API also tends to predict all activities as a tilting activity once it corresponds to a change of the device angle relative to the gravity.

Intel's API proved to be more responsive than Google's and with a more consistent update frequency. Even though this API has the benefit of returning the probability for each activity that the person may be doing at the time, it has really low accuracy.

IV. Design

The main focus of this article is the design of the transport mode detection. The following subsections will detail its design.

Design Objectives

The main objective is to perform effective and unattended transport mode detection. We outline the following desirable features (DF):

- DF1) Usage of smartphone features common even in low-end devices;
- DF2) Working indoors and outdoors;
- DF3) Support for different users with no additional training;
- DF4) Detect the mode of transport with an accuracy of, at least, 90%.

Regarding DF1, modern smartphones have several features that can be useful to detect the user transport such as inertial sensors, GPS, Wifi, GSM, and Bluetooth. These features are found even in low-end devices.

Bluetooth can be useful for detecting user location, and user location can be useful to detect transportation mode. Indeed, many systems have been built to provide user location using Bluetooth (e.g. [14], [15]), however, these systems rely on the existence of Bluetooth beacons, which are not widely available outdoors (DF2). Interestingly, while apparently the same could be said about WiFi, the reality today is that public WiFi signals are abundant outdoors, due to their longer range, from WAP installed outdoors or, more interestingly, from WAP installed in vehicles, particularly public transportation vehicles such as buses or trains. We use WiFi information to help differentiate transportation modes that would be difficult to distinguish using only accelerometer-based features (such as vehicles moving at similar speeds). More discussion on this later in Section IV.B.

GPS signals are very useful to obtain information about location and speed outdoors, however GPS signals are generally unavailable indoors, and this is an important use-case of our application, as we expect users to be often indoors, and also to use transportation means that have stops/stations underground, or have significant portions of their route underground. For this reason, and for DF2, we discard the option of using GPS.

GSM can also provide useful inputs for transport mode detection, however previous research has shown no advantage of using GSM in conjunction with accelerometer data, when compared with WiFi [9].

This discussion leaves us with DF3 and DF4, which we will evaluate in Section VI and discuss in Section VII.

Using WAP Identity

WAPs can be identified to provide information about the user location. This information is used to improve the inertial sensor-based classification, and we use this information to Identify WAP in public transportation vehicles.

With this information, we can infer that it is highly possible that the user is using that transport. To do this, we rely on the fact that WAP installed in buses and trains are easily identifiable. In our case-study application, being implemented in the City of Porto, Portugal, we were able to easily identify the WAP installed in buses and trains through all the network identifiers which share a common prefix.

TABLE III. OVERALL ACCURACIES OF EACH CLASSIFIER

Classifier	TP Rate	FP Rate	Precision	Recall	Roc Area
Tree J48	95.6	0.9	95.6	95.6	98.7
SMO	92.4	1.5	92.6	92.4	97.7
Naïve Bayes	61.9	7.6	67.3	61.9	88.1

While it is a rather simple method, we expect that it would be easily extendable to other locations.

Inertial Sensor Features

In order to enhance the classification algorithms performance, several features are extracted from the accelerometer signals. More discussion on this in Section V.

V. APPROACH

Data Collection

In order to perform the evaluation of each classifier a dataset needs to be constructed. For the construction of the dataset a group of 15 users proceeded with their routine while using a smartphone in the pocket (this will allow to test DF3).

To train and test different classifiers over 24 hours of mobility data were recorded for the six transport modes. Each transport mode had over 4 hours of data from several users.

Input Features

1) Decompose acceleration

The features calculated through the output provided by the accelerometer are divided into segments with a duration of 5 seconds (time windows). The size of the window is an important decision because if it is too big, every transport mode will have similar values and thus harder to detect the differences. If the window is too small, any blunt movement will "spoil" the classification. In order to standardize the signals, a fixed sample frequency of 33.33 Hz is used (a sample is taken every 30ms).

Although there is only one stream of accelerometer events, different signal components can be derived from this input. At a first instance the *x, y, z* and *magnitude* values of acceleration are extracted and used to rise all the signals components. For each of these signals we derive two additional signal components: the gravity and the linear acceleration. Furthermore, the absolute value is calculated from each of these axis signals components.

The angles between the acceleration vector and each of the phone axes are also computed, resulting in signals *angleX*, *angleY* and *angleZ*. An additional signal is calculated by applying low pass filtering to the magnitude.

Thus, a total of 21 different signal components are used in our analysis.

2) Calculate Features

In order to enhance the classification algorithms performance, several features are extracted from the accelerometer signals. The calculated features for each signal component are mean, median, maximum, minimum, root mean square, standard deviation, median deviation, interquartile range, minimum average, maximum average, peak height, average peak height, mean cross count, entropy, energy, skewness and kurtosis. Thus, we compute 17 features for each of the 21 different signal components, totalling 357 features for each time window.

3) Feature selection

After features extraction, it is possible that many of them are either redundant or irrelevant, and can be removed without incurring much loss of information. With the help of the Waikato Environment for Knowledge Analysis (Weka) [16] tool to select the most relevant attributes, managing to reduce the tree size by 11.95%, decrease the number of features from 357 to 40 and even to increase accuracy by 1.58%.

4) Transport Classification

Considering all the features extracted from the accelerometer signals, the classification algorithms Naïve Bayes, Decision Tree and Support Vector Machine classifiers were trained and evaluated using Weka [16]. This tool provides a classification algorithm based on Decision Trees named J48. Weka also offers several SVM algorithms, such as the one used in our tests, SMO.

The classifiers were tested and trained using a 10-fold cross-validation, this method divides the dataset into 10 parts (known as folds), holds out each part in turn, and averages the results. This way each data point in the dataset is used once for testing and 9 times for training.

VI. RESULTS

In this section we report detailed results of the performance analysis we conducted for the three types of classifiers.

According to [17] the tree complexity has a crucial effect on its accuracy, and is usually measured by one of the following metrics [18]: the total number of nodes, total number of leaves, tree depth and number of attributes used.

So trading accuracy for simplicity is a tradeoff that should be considered [19]. Typically, the goal is to find the optimal Decision Tree by minimizing the prediction error as well as minimizing the number of nodes.

Bearing the tradeoff in mind we managed to reduce the tree size by 42.64% and only reduce the accuracy by 0.33%, by tuning the Weka J48 classifier.

The summary results for our activity recognition experiments are presented in TABLE III, providing information about the classifiers overall performance. For all classifiers, we report True Positives (TP) rate, False Positives (FP) rate, Precision, Recall rate and ROC curve area.

The Naïve Bayes classifier treats all features as independent and is the simplest of these classifiers. We present its performance in TABLE *III* to provide a baseline,

TABLE IV. CONFUSION MATRIX FOR J48 MODEL

		Predicted Class						Precision (%)
		Inactive	Metro	Train	Car	Bus	Walk	
Actual Class	Inactive	**2898**	14	2	5	5	7	98.3
	Metro	32	**2722**	47	11	113	6	92.7
	Train	4	50	**2749**	25	21	53	95.3
	Car	6	13	19	**2858**	28	2	96.8
	Bus	4	123	37	51	**2678**	17	93.5
	Walk	4	13	30	3	20	**2854**	97.1
	Recall (%)	98.9	92.9	94.7	97.7	92.0	97.6	**95.6**

TABLE V. CONFUSION MATRIX FOR SMO MODEL

		Predicted Class						Precision (%)
		Inactive	Metro	Train	Car	Bus	Walk	
Actual Class	Inactive	**2877**	18	26	1	7	2	93.7
	Metro	68	**2714**	62	4	98	2	85.2
	Train	78	156	**2603**	22	4	39	91.3
	Car	13	7	72	**2790**	44	0	93.2
	Bus	31	261	39	176	**2395**	8	93.9
	Walk	2	31	50	2	2	**2837**	98.2
	Recall (%)	98.2	92.1	89.7	95.4	82.3	97.0	**92.4**

but will omit further results from this classifier as its performance is significantly worst.

Both, the J48 and the SMO, have yielded a good performance, with all the metrics above 92% (with the exception of FP rate). Even so, SMO is still outperformed by the J48 Decision Tree classifier on all performance metrics, which has all the metrics 3% higher than SMO.

TABLE IV and TABLE V show the results of the classification for the J48 and SMO, respectively. Precision and Recall values were computed from the detailed statistical analysis of the results. The J48 classifier achieves an accuracy of 95.6% and the SMO classifier reaches 92.4% (the bottom right cell shows the computed accuracy). It is possible to notice that both classifiers have trouble making the distinction between Metro and Bus, with the most misclassifications. Due to the fact that all the means of transportation have short periods of immobility, there is also a small number of misclassifications with Inactivity.

VII. CONCLUSION

Relying on the power and ubiquity of modern smartphones for HAR purposes is a very attractive proposition that previous researchers have explored. In this work, we present the development of a classifier for transport mode detection that uses the smartphone inertial sensors and wireless communication capabilities. The final goal is to embed our classifier in an application for elderly care, that can be used by care givers and alike to learn about the daily activity patterns of elderly people.

Our classifier can identify the following transportation modes: walking, in bus, in car, in train or in metro. Additionally, it detects if an individual is inactive, if the phone is still for a period of time.

We have tested our classifier using data from 15 different users, to conclude da our classifier performs well for different users (DF3). The classifier also exhibited an accuracy above 90% in all tests performed (DF4).

As future work, we intend to study adding more indicators of the user transportation mode. For example, by detecting a transition from stagnant to movement near a stop/station, we can infer that it is highly likely that user is waiting for public transportation (underground/tram, train or bus). It is trivial to identify stops/stations using public landmark databases that provide online APIs, such as [20] or [21] in combination with [22] or [23].

REFERENCES

[1] S. Eubank, H. Guclu, V. S. Kumar, M. V Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang, "Modelling disease outbreaks in realistic urban social networks," *Nature*, vol. 429, no. 6988, pp. 180–184, 2004.

[2] M. W. Horner and M. E. O'Kelly, "Embedding economies of scale concepts for hub network design," *J. Transp. Geogr.*, vol. 9, no. 4, pp. 255–265, 2001.

[3] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, "Human mobility, social ties, and link prediction," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1100–1108.

[4] P. I. of T. Widhalm, P. I. of T. Nitsche, and N. I. of T. Brändle, "Transport Mode Detection with Realistic Smartphone Sensor Data," *Icpr*, no. Icpr, pp. 573–576, 2012.

[5] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," *2013 IEEE 10th Consum. Commun. Netw. Conf. CCNC 2013*, pp. 914–919, 2013.

[6] J. Yang, "Toward Physical Activity Diary : Motion Recognition Using Simple Acceleration Features with Mobile Phones," *Proc. 1st Int. Work. Interact. Multimed. Consum. Electron.*, pp. 1–9, 2009.

[7] Google, "ActivityRecognitionApi," 2016. [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi. [Accessed: 07-Jun-2016].

[8] Intel, "Context States Datasheet," 2016. [Online]. Available: https://software.intel.com/en-us/articles/sensing-context-states-datasheet. [Accessed: 07-Jun-2016].

[9] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Trans. Sens. Networks*, vol. 6, no. 2, pp. 1–27, 2010.

[10] J. Kwapisz, G. Weiss, and S. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. …*, vol. 12, no. 2, pp. 74–82, 2011.

[11] B. Aguiar, J. Silva, T. Rocha, S. Carneiro, and I. Sousa, "Monitoring physical activity and energy expenditure with smartphones," *IEEE-EMBS Int. Conf. Biomed. Heal. Informatics*, pp. 664–667, 2014.

[12] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," *Proc. 11th ACM Conf. Embed. Networked Sens. Syst. - SenSys '13*, pp. 1–14, 2013.

[13] C. R. Figueira, R. Matias, and G. Hugo, "Body Location Independent Activity Monitoring," *9th Int. Jt. Conf. Biomed. Eng. Syst. Technol.*, pp. 190–197, 2016.

[14] S. Bobek, O. Grodzki, and G. J. Nalepa, "Indoor microlocation with BLE beacons and incremental rule learning," in *Proceedings - 2015 IEEE 2nd International Conference on Cybernetics, CYBCONF 2015*, 2015, pp. 91–96.

[15] E. Munguia Tapia, S. S. Intille, and K. Larson, "Activity Recognition in the Home Using Simple and Ubiquitous Sensors," *Pervasive Comput.*, vol. 3001, pp. 158–175, 2004.

[16] I. H. Witten, E. Frank, and M. a. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*, vol. 54, no. 2. 2011.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, vol. 19. 1984.

[18] L. Rokach and O. Maimom, *Data mining with decision trees: theory and applications*. 2014.

[19] M. Bohanec and I. Bratko, "Trading Accuracy for Simplicity in Decision Trees," *Mach. Learn.*, vol. 15, no. 3, pp. 223–250, 1994.

[20] "One.Stop.Transport." [Online]. Available: https://www.ost.pt/. [Accessed: 07-Jun-2016].

[21] "Public transport API." [Online]. Available: http://transport.opendata.ch/. [Accessed: 07-Jun-2016].

[22] Combain Mobile AB, "Combain CPS API," 2016. [Online]. Available: https://combain.com/api/. [Accessed: 07-Jun-2016].

[23] Unwired Labs (India) Pvt. Ltd., "Reliable IoT Positioning," 2016. [Online]. Available: https://unwiredlabs.com/. [Accessed: 07-Jun-2016].