# Cycle Our City goes Mobile

Rodrigo José Madeira Lourenço
Instituto Superior Técnico
rodrigo.m.lourenco@tecnico.ulisboa.pt

***Abstract*—For urban mobility, cycling presents itself as a cleaner, cheaper, and healthier alternative to motorized transportation. To ease the adoption of the bicycle as viable means of transportation, CycleOurCity was developed. The platform recommends appropriate cycling routes given the characteristics of the city's road network, as classified by the participants. However, due to the efforts required to contribute, the platform presented low adoption levels.**

**This thesis proposes the design, implementation and evaluation of a mobile crowdsensing-based system, capable of leveraging the sensors found in smartphones wielded by a diverse community of participants. The purpose of this solution is to extend CycleOurCity, to allow the route classification tasks to be performed transparently and effortlessly by the participants' smartphones. By shifting the source of knowledge to a crowd equipped with a wide range of sensors, the acquired information should become more precise and inexpensive.**

**To understand the viability of our solution, as an alternative to CycleOurCity's human-based classification, we developed and evaluated Scout - an Android prototype. The prototype was evaluated with regard to its ability to accurately classify the slope and pavement type of the roads traveled by the participants. Our results show that the proposed solution is capable of classifying a street's characteristics with equivalent precision to human-based classifications. In particular, Scout is capable of measuring a street's slope degree with a 79% precision, and classifying the pavement type with 88.2% precision. Furthermore, because Scout is oblivious to the participant's bias, our solution's precision surpasses the human-based when classifying slope.**

## I. Introduction

One of the major challenges for the 21st century will be to mitigate the negative effects of motorized transportation. The flexible and convenient nature of the bicycle makes it the ideal alternative for short journeys, in an urban environment [1]. It is also a cleaner, cheaper and healthier solution than the car [2].

Cities should be concerned with making cycling efficient, safe and comfortable. The existence and quality of adequate cycling infrastructures plays a crucial role in achieving these requirements, ultimately leading to a higher bicycle commuter share [3]. So what happens in cities with weak cycling infrastructures?

In cities with weak cycling infrastructures, cyclists must attempt to find the safest routes to reach their destinations. This, however, is not always a trivial task. A route's safety is mainly characterized by its traffic density and the pavement condition. A street with high traffic density is generally dangerous as most drivers disregard safety rules designed to protect the cyclists, for instance, maintaining a minimum 1.5 m distance when passing a bicycle. The pavement condition is also a very important criteria. While it is easy for a cyclist to go around a pothole or bump, if the overall street pavement is in bad condition, the route becomes uncomfortable and dangerous.

To support a city's bicycle commuters, there are services that allow users to automatically obtain route recommendations for their daily commutes. And even though currently there are many available alternatives most of them exhibit fundamental limitations. Many, like the popular Google Maps[1], have high maintenance costs, which lead to incomplete, outdated or inaccurate information, regarding a city's infrastructures. This lack of information often leads to either unsatisfactory routes or even, in many cities, the absence of such services [4]. Furthermore, because they generally only take into account the distance necessary to reach the final destination, the safety of its users is often disregarded. This ultimately leads to routes that put the users in awkward and even dangerous situations.

In order to tackle this problem, *CycleOurCity* [4], a collaborative route planner, was designed. The system allows users to find the best routes for their daily commutes, while taking into account an extensive knowledge of a city's routes and their characteristics, as well as the user's preferences on different criteria, such as slope, length, and safety.

In order to do so, the system takes on a crowdsourcing approach, where participants collaborate by classifying routes within a city. As participants contribute, the system's knowledge grows and becomes more accurate. This allows CycleOurCity to recommend increasingly better routes.

While a crowdsourcing approach allows the system to scale to any city, with no additional costs, it also presents some challenges. The quality of the recommended routes depends on the existence of many different points of view and their correctness. This means that the success of CycleOurCity depends on the existence of a great number of participants, a challenge that is common to most crowdsourcing platforms. More specifically, CycleOurCity relies on a vast number of participants spread around the urban centre and actively participating by contributing with classifications.

In his study, Nunes [4] identified several areas of improvement. Currently, classifying a route is an arduous task where the participants must individually classify every segment of that route. So, there is a need to lessen the classification task's burden. A challenge this thesis will address by employing an approach based on a novel paradigm of mobile applications,

[1] https://maps.google.com

referred to as *Mobile Crowdsensing* [5]. By taking advantage of the intelligence and mobility of a community of participants equipped with multi-modal sensing mobile devices, it is possible to infer the characteristics of a city's cycling infrastructures. These inferences are performed with no need of user interaction, thus allowing classification tasks to be performed in an effortless and unobtrusive manner.

Smartphones present themselves as an ideal tool for a great number of reasons. When compared to traditional mote-class sensors, smartphones come equipped with greater computational and communication capacities. They are also outfitted with several multi-modal sensors. And finally, there are currently millions of these devices active in the field, making their potential tremendous.

Traditionally, mobile crowdsensing systems encompass two core components. An application running on the mobile device, which is responsible for leveraging the sensors, in order to gather data that pertains to the phenomena being studied. And a back-end where the intelligence *per se* is located. This structure allows CycleOurCity to transition smoothly from crowdsourcing to crowdsensing since the current system already implements most of the back-end component.

To better understand how a mobile crowdsensing-based approach facilitates the classification task lets consider the following scenario. When a participant initiates a ride, he mounts his smartphone on the bicycle and initiates the mobile crowdsensing application. While the participant is riding, the application is leveraging the smartphone's sensing capabilities to measure and classify the characteristics of the route, just as if it was being manually classified by the participant. Meanwhile, the classifications inferred by the application are being sent to a back-end server. The latter allows the information gathered by the participant to be aggregated and made available to the CycleOurCity community, thus improving CycleOurCity's knowledge base. When the participant reaches his destination, he turns off Scout and goes about his daily routine. Hence, the participant may contribute to the CycleOurCity in an effortless and unobtrusive manner.

However, given the nature of mobile crowdsensing systems, the success of CycleOurCity, as a mobile crowdsensing system, depends on how certain challenges are addressed. We focus on addressing two key challenges:

- Without proper mechanisms capable of appropriately interpreting raw sensor data, the gathered data is worthless [6]. In order to enable the inference of relevant information, mobile crowdsensing systems must employ data-mining algorithms, as well as statistical analysis techniques.
- Human behavior and context are also key challenges in the process of sensor data interpretation [6]. There is a need for mechanisms capable of increasing the system's robustness, by enabling the system to deal with the user's context.

This document proposes the design, implementation and evaluation of a client-server architecture based on mobile crowdsensing, which enables CycleOurCity to perform its route classification tasks in an effortless and unobtrusive manner. The proposed solution takes advantage of the sensing capabilities of smartphones wielded by a community of participants, to classify the features that characterize a route.

Our thesis's main contribution is the extension of the CycleOurCity platform to a mobile crowdsensing-based architecture, with the introduction of *Scout* — the mobile crowdsensing application. The latter will focus on measuring and classifying the slope and pavement condition of the streets traveled by the participants while using the application. In particular:

1) A street's slope is inferred and classified using the pressure and location sensors, which enable the classifications to be performed regardless of the device's position and orientation.
2) A street's pavement type and its overall condition are inferred and classified using a supervised learning-based approach, more specifically, a `J48` decision tree is employed. Furthermore, we employ data normalization and calibration, which assure the classification's accuracy, regardless of the device or its orientation.

Supporting cycling through mobile crowdsensing is not a novel approach. One of the earliest examples is Bikenet [7], which due to the rapid technological advancements in smartphone technologies, quickly became outdated. Biketastic [8] and Tripzoom [9] are the most current examples of mobile crowdsensing systems designed to support or promote cycling. Tripzoom was designed to recognize the activities and transportation modalities adopted by the users, this information was then used to guide the users towards more sustainable modes of transportation. While this is a relevant system, it is neither formally evaluated nor focuses of classifying the characteristics of the streets traveled by the participants. Biketastic [8], on the other hand, was designed to classify the pavement condition and the traffic density of the roads traveled by its users. However, to classify those characteristics, Biketastic employs crude and limited techniques, when compared to state-of-the-art techniques. Furthermore, to the best of our knowledge Biketastic lacks rigorous evaluation of their effectiveness and precision.

To the best of our knowledge, the proposed solution is the first to classify a street's pavement and its overall condition as a whole. Most of the studied solutions [10], [11], [8] focus on quantifying the condition of the road as the existence of punctual anomalies found in the road, for instance, potholes and bumps. For dynamic means of transportations such as the bicycle, which can easily avoid these anomalies, the existent body of work is not an appropriate solution. By mapping the pavement conditions of a city's road network as a whole, it becomes possible to provide a better understanding of the road network's state.

This report will start by presenting an overall study on mobile crowdsensing in Section II. Then in Section III, we describe the proposed architecture, designed to facility the effortless and unobtrusive classification of a street's slope and pavement. Section IV presents a comprehensive evaluation that

demonstrates that Scout is a reliable and viable alternative to user-based classification. Finally, in Sections V and VI we draw our final conclusions and present possible future work, respectively.

## II. Related Work

*Mobile crowdsensing* is a particular type of mobile sensing systems, which relies on a large community of participants equipped with mobile devices with sensing capabilities. This approach presents itself as the next logic step for crowdsourcing platforms. Mobile crowdsensing allows crowdsourcing platforms to shift their source of knowledge from the user to the mobile device. Due to the user's sway-able nature, contributed data may be influenced by their bias. By shifting the source of knowledge, platforms are able to gather information of higher quality and accuracy. In their study, Lane et al. [6] identified the set of factors and technological advancements that ultimately led mobile sensing applications to grow and gain momentum. First and possibly one of the most important factors is the appearance of low-cost mobile devices, such as smartphones, fitted with embedded sensors. Second, the existence of open *Software Development Kits*, as well as appropriately defined APIs, lowered the barrier of entrance for developers, allowing more applications to be developed. Third, the appearance of application markets allowed developers to distribute their applications to a larger number of users, scattered across the globe. And finally, technological advancements allowed developers to take advantage of the resources found in back-end servers, which typically far exceed the computational capabilities of mobile devices.

Most mobile crowdsensing systems resort to a client-server architecture [5], [6]. This architectural design encompasses a mobile client application primarily responsible for capturing raw sensor data. And a back-end server that is responsible for sensor data aggregation. From the moment the data is captured until its interpretation, raw sensor data goes through several filters and transformations. This data processing flow is represented in Figure 1, and is ensued as follows. First, data mediation is employed to ensure the sensor data's quality, by discarding or amending low-quality samples. Second and depending on the sensors employed, calibration is performed. The filtered sensor data is adjusted according to the device's orientation and position, in order to increase the systems robustness [12]. Third, raw sensor data is transformed and converted into features, which represent specific characteristics of the sensor data according to different domains. Relevant features are key in enabling the interpretation processes to unveil useful information [13]. Finally, the extracted features are analyzed and interpreted to elicit relevant information, typically through data-mining algorithms and statistical analysis tools.

### A. Road Condition Monitoring

*a) $P^2$:* was designed by Eriksson et al. [10] in order to monitor the conditions of a road by detecting anomalies, such as potholes or bumps, while traveling by car. The proposed system leverages both the accelerometer and GPS sensors. The first is used to measure the road anomalies, while the latter is used to attest the location of the road anomalies and to measure the user's speed. Feature extraction and road anomaly inference are performed by a back-end server. The core extracted features are the accelerometer readings' *variance* and *peaks*. A sequence of filters is then applied to the features to distinguish between road anomalies, such as potholes, and road characteristics, such as train tracks. The filter used to identify road anomalies is referred to as *z-peak*. The filter analyses peaks measured along the accelerometer's vertical axis, *i.e.* the z-axis. Despite its potential, the solution's evaluation revealed a high susceptibility to both false positives and negatives.

*b) Nericell:* extends the work of Eriksson et al. [10] by providing an additional filter — *z-sus*, which is applied when the car is moving at lower speeds, more specifically bellow 25 Kmh. Even though at low speeds the *z-sus* detector outperforms the *z-peaks* filter, the proposed solution is still susceptible to high rates of both false positives and negatives.

It is important to note that both solutions were specifically designed for scenarios where the accelerometer is located inside a moving car. Additionally, the proposed solutions focus on measuring a road's condition as the existence of isolated anomalies, namely potholes and bumps. The solution we propose, however, addresses the road condition as a whole. Because the system we propose was designed to run while the participants ride their bicycles, and due to the bicycle's dynamic nature, in many cases potholes and bumps may be avoided by the rider. Hence, the *z-peak* and *z-sus* filters cannot be appropriately employed in our scenario. Instead, Scout classifies a road's condition as one of four pavement classes, defined according to the pavement type and its condition.

### B. Mobile Crowdsensing Systems for Cycling Support

*a) Biketastic:* is a mobile crowdsensing-based platform developed by Reddy et al. [8], in order to allow participants to share their cycling routes knowledge and experience, with ease and efficiency. Biketastic employs a client-server architecture that comprises a mobile application and back-end server, and focus on inferring the traffic density and condition of the roads traveled by the participants. Several sensors are leveraged to infer the different attributes that characterize a route, namely the GPS, accelerometer and microphone. To assure the solution's robustness both the accelerometer and microphone readings are subject to calibration. All sensor data is processed and interpreted in the server. Noise, which is employed as a traffic indicator, is directly inferred through time-domain-based extracted features. Road conditions are also inferred directly from the accelerometer's high peak feature. Although the system was evaluated by 12 volunteers, during a two week period, the precision of Biketastic's inferences were never formally evaluated.

*b) Tripzoom:* is a project developed by Broll et al. [9]. It was designed as a mobile crowdsensing application that

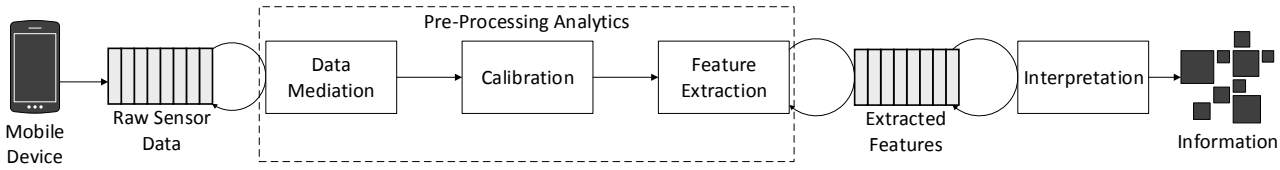


Fig. 1: Overview of sensor data processing flow in mobile sensing systems

combines mobility data and patterns, obtained through smartphones with sensing capabilities, in order to promote more sustainable mobility behaviors. Tripzoom is based on client-server architecture.

The mobile application leverages the smartphones' sensing capabilities to measure the participant's mobility behaviours. More specifically, the GPS, GSM and WiFi are employed. The mobile application's architecture is divided into several modules, with different responsibilities. The proposed system focus mainly on detecting the participants transport modality, and the activity being performed. Because the authors do not provide a comprehensive system evaluation, it is impossible to fully assess the Tripzoom application's effectiveness.

## III. Proposed Solution

This project proposes the extension of the existing CycleOurCity platform – *CycleOurCity v0*, to a mobile crowdsensing-based system. The system resorts to a client-server architecture, which encompasses a mobile client application — *Scout*, and a back-end server — *CycleOurCity v1*. The former is a mobile crowdsensing-based application responsible for leveraging the sensors found in modern smartphones, namely the following sensors are employed: *i*) *location*; *ii*) *linear acceleration*; *iii*) *pressure*; and *iv*) *rotation vector*. The data generated by the sensors is then used to infer the slope and pavement type of the routes traveled by the participants, while using the Scout application. Once these characteristics are inferred by Scout, they are transmitted to the *CycleOurCity v1* back-end server. *CycleOurCity v1* is responsible for aggregating the information inferred by the mobile application, and relaying it to the CycleOurCity platform. *CycleOurCity v1* encompasses the native CycleOurCity platform, as represented in Figure 2, which had already been designed and developed by Nunes [4].

*CycleOurCity Scout*, is a mobile crowdsensing-based application, which was designed to run on Android smartphone devices. CycleOurCity Scout's architecture is depicted in Figure 2. The back-end server is partially implemented by the existing CycleOurCity platform — *CycleOurCity v0*. It provides persistent storage and allows its users to search and classify routes, through the *web portal* interface. *CycleOurCity v1*'s architecture is represented in Figure 2.

Because the purpose of the proposed solution is to allow the characteristics of the roads traveled by the participants to be monitored, the system presents no real-time constraints. Therefore, our solution is a fully delay-tolerant system.

### A. Mobile Crowdsensing Architecture

As the stream sensor data is produced by the leveraged sensors, the *Sensor Manager* dispatches the sensor stream onto the *Sensing Engine*. The latter splits the stream into frames that are processed by a specific pipeline: one designed to monitor the route's slope, and another for road condition monitoring. At the end of each pipeline the sensor sample frames are converted into classifications that specify the route's slope and pavement type.

To minimize the energy costs, inherent from data transmission, the results are stored on a buffer. Mobile networks, like 3G, are susceptible to energy wastage caused by intermittent data transfers [14]. Because the system is delay-tolerant, this energy wastage may be minimized by performing bulk transfers instead. Additionally, this mechanism also allows our solution to postpone data transmission operations until a more efficient communication channel is present. It is the responsibility of the *CycleOurCity Client* to manage this buffer. Only when the buffer is nearly full or under certain conditions, will the classifications be transmitted to the back-end server.

Once the route classifications reach the back-end server, it is the *OTP Translator*'s responsibility to associate the results with edges from the *CycleOurCity v0*'s directed route graph. By doing so, the results may be inserted directly into CycleOurCity's knowledge database, just as if the classifications had been manually performed by a human participant.

*1) SensorManager:* is the component responsible for managing and dispatching the sensor samples to different components, according to their sensor type. In particular it manages the following sensors: *i*) *Location*, *ii*) *Linear Accelerometer*, *iii*) *Rotation Vector*, and *iv*) *Pressure* sensors.

Once the samples are produced, this component dispatches them. If the samples are produced by either the *location* or *rotation vector* sensors, they are sent to the *Active Geo Tagger*. However, if they are produced by the *linear accelerometer* or *pressure* sensors, they are tagged with the *Active Geo Tagger*'s current *geoTag*[2], and only then dispatched to the

[2]The *geoTag* contains the most current location and rotation matrix.

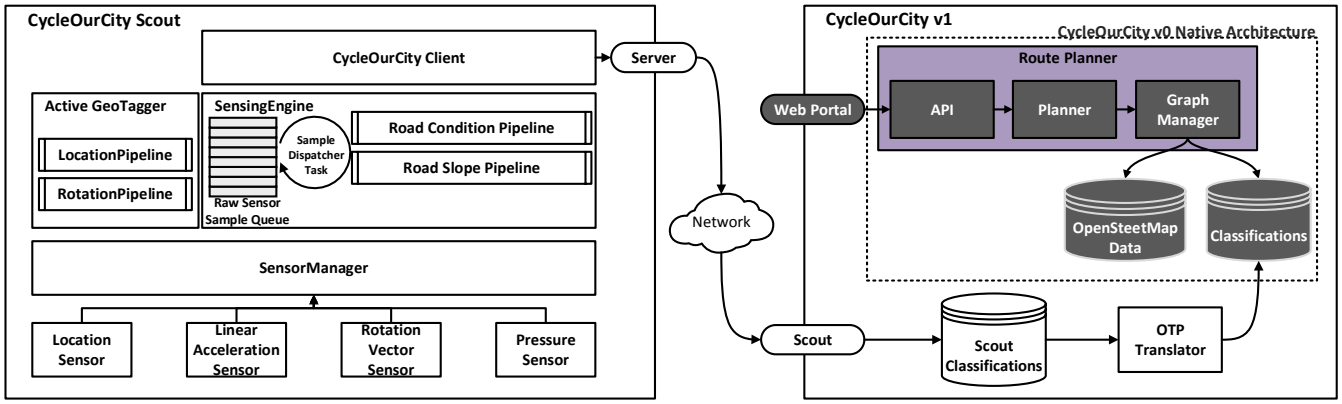


Fig. 2: *CycleOurCity goes Mobile*'s architectural design

*Sensing Engine* component.

*2) Sensing Engine:* is our solution's core component, and can ultimately used by both the Scout mobile application and the *CycleOurCity v1* back-end server. This component is responsible for managing the pipelines that classify the slope and pavement type of the street's traveled by the participants.

Before presenting the *Sensing Engine* in detail, it is important to understand the architectural requirements that drove its design.

Sensor data processing responsibilities are typically distributed between the mobile application and the back-end server. However, sharing these responsibilities and finding a balanced solution is not always trivial. To ease the development of mobile crowdsensing-based systems, an important architectural quality requirement is portability. To assure this, the *Sensing Engine* was developed as an external library that may ultimately be employed in the mobile application and the back-end server. Additionally and to achieve portability, all mobile operative system specific APIs had to be avoided. In the proposed solution, all sensor processing and interpretation logistics are performed by the Scout mobile application. However, the *Sensing Engine*'s library allows the server to be easily endowed with the capability to perform sensor processing and interpretation. This approach allows developers to focus on designing the sensor processing and interpretation logistics, regardless of where they may be performed. Once the logistics have been implemented, different configurations may be evaluated, to determine the one that preserves the user's mobile experience without compromising the server's scalability.

Finally, for simplicity and performance sake it was also decided that each sensor processing pipeline should not depend on the results of other pipelines. By making each pipeline independent, it becomes possible to improve performance by enabling concurrent pipeline execution.

With a better understanding of the main architectural requirements that guided our solution's design. We proceed by describing the *Sensing Engine*'s architecture. The *Sensing Engine* is characterized by two pipelines: *i*) *Road Condition Pipeline*; and *ii*) *Road Slope Monitoring Pipeline*. A *Sample Dispatcher Task* manages a sensor sample queue, and at fixed rate intervals distributes the samples onto each specific pipeline. The pipelines are then executed and the samples processed and interpreted. Because each pipeline is independent, they may be executed concurrently.

*a) Road Slope Pipeline:* is responsible for processing the pressure sensor samples to determine the slope of the roads travelled by the participants. Given a window of pressure samples it starts by discarding those that are not coupled with the participant's location or that are considered stationary. Then the samples are merged into a single value, which corresponds to the median of all pressure values. Using the *density altitude* formula represented in Equation 1 the participant's elevation is computed. In this formula, $p$ is the pressure measured by the sensor, the remaining variables are constants that are simplified in the right-hand side of the equation.

$$elevation(p, p_0) = \frac{T_0}{L}\left(1 - (\frac{p}{p_0})^{\frac{RL}{gM}}\right) = 44330 \times \left(1 - \frac{p}{1013.25}^{\frac{1}{5.225}}\right) \quad (1)$$

Given the derived elevation and the one computed in the previous pipeline iteration, as well as the distance traveled between the two iterations, the road's slope degree is derived using the formula represented in Equation 2.

$$slope = \frac{\Delta elevation}{distance} \quad (2)$$

Finally, the pipeline converts the road's slope degree into a qualitative classification according to the mappings in Table I, which are specified according to CycleOurCity's slope classification's [4].

*b) Road Condition Pipeline:* is responsible for processing and interpreting the linear acceleration sensor samples to determine the pavement type and its overall condition of the roads traveled by the participants. Similarly to the previous

| Class | CycleOurCity [4] | Scout |
|---|---|---|
| 1 | Steep Decline | $]-100\%, -24\%[$ |
| 2 | Gentle Decline | $[-24\%, -1\%]$ |
| 3 | Plane | $]-1\%, 1\%[$ |
| 4 | Effortless Climb | $[1\%, 13\%]$ |
| 5 | Arduous Climb | $]13\%, 20\%]$ |
| 6 | Impossible Climb | $]20\%, 100\%[$ |

TABLE I: *CycleOurCity*'s slope class mappings

| Class | Description |
|---|---|
| A | Cobblestone or gravel in good condition |
| B | Cobblestone or gravel in bad condition |
| C | Asphalt or tarmac in bad condition |
| D | Asphalt or tarmac in good condition |

TABLE III: *CycleOurCity*'s road pavement classes

pipeline, the *Road Condition Pipeline* starts by discarding any samples that are neither coupled with the participant's location, inverted rotation matrix, nor the accelerometer's hardware offsets. To assure the precision and the robustness of the classifications inferred by the pipeline, it starts by normalizing and projecting the samples. Normalization is employed to reduce the negative effects caused by hardware inherent noise, which varies from device to device. These offsets are obtained during a one-time calibration process. The normalization stage is based on the one performed by Lu et al. in Jigsaw [15]. Projection, on the other hand, is employed to deal with the effects that different device's orientations have on the sensors' readings. By multiplying the linear accelerometer's readings with the inverted rotation matrix, it becomes possible to project the sensor values onto an absolute coordinate system [16]. In addition to being susceptible to the device's orientation, the linear acceleration sensor is also susceptible to the device's position. This challenge is dealt by adopting a non-technical approach, where it is assumed that the device is fixed to the bicycle's handle-bar at all times. Also known as a *fixed-position* solution, this approach has also been employed in other projects [17], [18]. With the quality of the sensor data assured, the pipeline's next step is to extract the feature vector. During this stage a *feature vector* that comprises the 9 time-domain features represented in Table II, is extracted. In addition to enabling and improving the interpretation process, the feature extraction stage also allows the sensor data to be summarized. Finally, the generated feature vector is fed onto a `J48` decision tree classifier, which implements the interpretation logistics. Depending of the different feature's values, the `J48` decision tree classifies the feature vector as one of the pavement classes represented in Table III.

| Time-domain Features | |
|---|---|
| Variance | Standard Deviation |
| Minimum | Maximum |
| Range | Root Mean Squares |
| Zero-Crossings | Mean-Crossings |
| Median-Crossings | |

TABLE II: *Road Condition Pipeline*'s features vector

*3) Active GeoTagger:* Because both *Sensing Engine*'s pipelines rely on the participant's locations, and to facilitate inter-pipeline independence the *Active GeoTagger* component was created. Similarly to the *Sensing Engine*, this component manages two pipelines. However, unlike the latter, the samples are processed actively so that they may be immediately consumed by the *Sensing Engine*'s pipelines. The *Active GeoTagger* manages two pipelines: *i*) *Location Processing Pipeline*; and *ii*) *Rotation Processing Pipeline*.

*a) Location Processing Pipeline:* deals with the location sensor's inaccuracies. There is a need to employ a admission control process, so that outliers may be removed. By doing so, the inferences made by the *Sensing Engine* can be associated to more accurate locations.

The pipeline starts by cleaning the sensor samples by removing irrelevant fields. Then an heuristic-based admission control process is employed, which discard samples considered to be outliers. This is a fundamental step to assure the accuracy of the routes tracked. Finally, the pipeline checks if the location sample is or not a *stationary sample*. A location is said to be stationary if the measured speed or traveled distance are zero.

*b) Rotation Vector Probe:* processes rotation vector samples. This pipeline starts by validating the rotation vector samples, discarding those with low accuracy. Then the samples are merged into one by computing the median of all values. Finally, the *rotation matrix* and corresponding *inverted rotation matrix* are generated. The latter is required by the *Road Condition Pipeline* in order to project the accelerometer readings onto an absolute coordinate system.

*4) CycleOurCity Client:* is primarily responsible for managing the data transmission processes.

The *CycleOurCity Client* component was designed to operate in one of three modes: *i*) *local-only*; *ii*) *eager*, which automatically transmits the classifications as soon as they are inferred; and *iii*) *buffered*, which only uploads the results once a certain amount data has been generated. Of the three modes of operation the latter is the most relevant one. By taking advantage of the system's delay-tolerant nature, and postponing the data transmission operations, it becomes possible to minimize the data transmission costs. This is particularly advantageous in mobile networks, such as 3G. For the buffer size, we chose a 100 KB buffer. This buffer size was chosen given evaluations performed by Balasubramanian et al. [19].

### B. CycleOurCity v1 — The back-end server

*CycleOurCity v1*'s main purpose is to allow the information inferred by the participating Scout applications, to be aggregated and made available to the CycleOurCity community.

Participating Scout applications send their classifications to the back-end server, where they are stored on the *Scout's*

*classification* database. Because the proposed system is fully delay-tolerant, there is no urgency in processing these results. Therefore, the back-end server may take advantage of periods of inactivity or lower traffic to process the results without compromising its performance.

Periodically, the *OTP Translator* component fetches the classifications from the *Scout's classifications* database and translates them according to the CycleOurCity's data model. More specifically, it converts the classification's geo-locations into edges belonging to CycleOurCity's directed route graph. Once this translation has been performed, the classifications are stored on *CycleOurCity's classification* database. This database is then used by the *Route Planner* in conjunction with OpenStreetMap's information, to generate an informed route graph. This which is used to recommend the best routes.

## IV. Evaluation

The purpose of this evaluation is to better understand if the proposed solution is a viable alternative to CycleOurCity's user-based classifications. Scout's correctness and viability are be evaluated both in terms of the inferred road slope and pavement type classifications. While the slope is traditionally a formally defined characteristic of a city's road network, the pavement type and its overall condition is more subjective. Therefore different evaluation methods are employed for the two criteria.

### A. Evaluation Methodology

All the evaluations were performed using a Nexus 5 device, running the Android L operating system, which comes equipped with a Qualcomm Snapdragon™ 800 2.26GHz CPU and 2GB of RAM.

Regarding the mobile crowdsensing data acquisition and evaluation, a standard bicycle with no suspension was used. The smartphone was then attached to the bicycle's handlebar using a low-cost bicycle phone mount.

To evaluate both the slope and pavement classification inferred by Scout, different evaluation streets were selected. Our initial plan was select these streets based on the number of classifications found in CycleOurCity's classification database[3]. However, after a preliminary analysis, it was identified a lack of user classifications' diversity in the CycleOurCity's classification database. To address this challenge we chose the evaluation streets based on two criteria: *i*) the number of classifications in CycleOurCity; *ii*) street's slope or pavement type; and *iii*) ease of test repetition. . The selected evaluation streets are enumerated in Table IV.

Then, to mitigate the lack of human-based classifications a user survey was performed, where we were able to acquire a total of 25 responses. These participants where inquired with regard to their perceptions of the selected street's slope and pavement type [4].

[3]All classifications found in the http://cycleourcity.org/ database by the 26th of August 2016.

[4]Some of the responses were discarded due to the participant's inexperience of lack of knowledge regarding the selected streets.

| **Pavement Type Evaluation** | |
|---|---|
| *Pavement Class* | Evaluation Street |
| A | Ribeira das Naus, bicycle path |
| B | Ribeira das Naus |
| C | Av. da Liberdade, bicycle path |
| D | Rua Pascoal de Melo |

| **Slope Evaluation** | |
|---|---|
| *Slope Class* | Evaluation Street |
| Plain | Av. Duque de Ávila |
| Effortless Climb | R. Pascoal de Melo |
| Accentuated Climb | Largo do Leão |

TABLE IV: Evaluation Streets

### B. Pavement Type and Overall Condition

During an initial preliminary stage, the Scout application was employed to gather accelerometer data, describing how the physical characteristics of the streets influence the device's accelerometer readings. The purpose of this stage was to gather an extensive data set, which could then be applied as a supervised-learning classifier's training set. Four different areas, in the city of Lisbon, were selected to represent each of the pavement classes described in Table III. These areas were selected based on their pavement characteristics and to the ease the acquisition of a large training set. A total of 1978 samples were collected, amounting to a total 1.65 hours of data. Each sample was characterized by a feature vector encompassing an extended version of the feature vector described in Table II. Finally, using the Weka platform [5], several supervised-learning classification algorithms were studied, namely: *i*) `J48` Decision Tree; *ii*) `MultilayerPerceptron` Neural Network; *iii*) `libSVM` Support Vector Machine; and *iv*) Naive Bayes. The different classifier algorithms were analyzed according to two metrics, precision and recall. These preliminary results represented in Table V, show that the best option is the `J48` decision tree classifier. This classifier presents both the highest precision and recall out of all evaluated classifiers.

Despite its high precision, to reach a 95% precision the `J48` classifier requires a tree with 79 nodes. Given the classifier's large size, we decided to study which of the features could be discarded, as to reduce the classifier's size. It is important to mention that the generated classifier had already been pruned by the Weka platform. This analysis lead us to two different configurations of the original `J48` classifier: *i*) *MediumTree*, which discards the *mean*, *median* and *range-crossings* features; and

*ii*) *SmallTree*, which in addition discards the *zero-crossings*, *mean-crossings* and *median-crossings* features. These two configurations are compared on different levels in Table VI. The *MediumTree* allows for a smaller and more manageable classifier. However and despite its increased size, the *MediumTree*'s performance overhead is negligible, as its average execution time is only 0.1065 ms. The precisions were measured by

[5]http://www.cs.waikato.ac.nz/ml/weka/

| | Classifiers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Decision Tree* | | *Neural Network* | | *SVM* | | *Naive Bayes* | |
| **Class** | Precision | Recall | Precision | Recall | Precision | Recall | Precision | Recall |
| A | 91.5% | 97.1% | 85.9% | 88.5% | 42.1% | 97.6% | 85.8% | 85% |
| B | 97.8% | 96% | 95.7% | 88.3% | 92.9% | 10.4% | 93.5% | 84.3% |
| C | 91.6% | 83.2% | 74.7% | 74.3% | 66.7% | 7.3% | 68.2% | 77% |
| D | 99.3% | 100% | 97.6% | 99.5% | 99.8% | 76.5% | 98.3% | 97.3% |

TABLE V: Classifier's precision and recall comparison
**SVM** — Support Vector Machines

| | Classifiers | | | | | |
|---|---|---|---|---|---|---|
| | *SmallTree* | | | *MediumTree* | | |
| **Class** | TP | FP | FN | TP | FP | FN |
| A | 226 | 76 | 9 | 216 | 30 | 19 |
| B | 144 | 19 | 31 | 144 | 30 | 31 |
| C | 54 | 32 | 83 | 89 | 42 | 48 |
| D | 364 | 5 | 9 | 362 | 7 | 11 |
| **Precision** | 85.7% | | | 88.2% | | |
| **Feature Vector Size** | 6 | | | 9 | | |
| **Tree Size** | 27 | | | 55 | | |
| **Execution Time** | 0.1032 ms | | | 0.1065 ms | | |

TABLE VI: *MediumTree* and *SmallTree* analysis
FP - False Positive and FN - False Negative

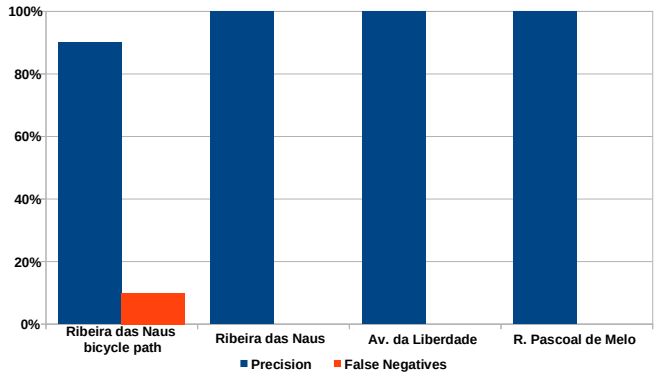


Fig. 3: *MediumTree* classifier's precision relative to the user-survey responses

implementing the two configurations in Scout, and evaluating the areas previously used to train the classifiers. The purpose of this approach, was to understand how the dynamic nature of a real-world setting would affect the classifier's performance. As is observable in Table VI, despite a noticeable precision decrease both classifiers still present high precision levels. Even though the *SmallTree* classifier presents an overall precision of 85.5%, we can see in its confusion matrix that it present a high number of false negatives when classifying streets with pavement class C, *i.e.* asphalt or tarmac in bad condition. Therefore, we decided to adopt *MediumTree* classifier.

Then to understand if the adopted classifier was a viable alternative to CycleOurCity's user-based classification, we compared Scout's and the user-survey responses with regard to the selected evaluation streets. Each of the four representative streets was then traversed up to 10 times using our smartphone equipped bicycle and classified using the Scout application. Scout's classifications were obtained by selecting the mode classification out of all classifications generated for each run. Then by adopting the user-survey responses as our ground-truth, we compare Scout's precision with the user-based classifications. This analysis is represented in Figure 3.

For all four evaluation streets, we can see that in most cases both Scout's classifiers inferences coincide with the classifications provided by the user-survey participants. Because for all evaluation streets, Scout's *MediumTree*'s inferred classifications coincide with the ones provided by the user-survey participants, we consider this solution to be a viable alternative to user-based pavement class classifications.

### C. Slope Evaluation

The purpose of this evaluation it to determine whether the *Road Slope Pipeline* is capable of accurately computing a street's slope, regardless of the participant's bias. To do so, we compare the results obtained by the Scout, with a ground-truth and the user-survey responses. The ground-truth selected for this study is based on Rosa Félix Lisbon's slope cartogram [20].

Using the Scout application, each of the slope evaluation streets was traversed and evaluated for a total of 10 runs. For simplicity's sake, we only accounted for positive slopes, *i.e.* climbs. Before evaluating the solution's viability, we first evaluated the correctness of the measured slope degree's with regard to the adopted ground-truth. The detailed results of this evaluation are represented in Figure 4. This evaluation revealed that the *Road Slope Pipeline* is capable of correctly measuring a street's slope degree with a 79% precision. While this precision is acceptable, we believe it to be higher. The nature of the Largo do Leão street hindered the acquisition of accurate locations, which ultimately lead to deviations in the measured slope degrees.

Then to determine the viability of the *Road Slope Pipeline*, as an alternative to human-based classification with regard to a street's slope, we compared Scout's slope classifications with the user-survey responses. This comparison is represented in Figure 5. To understand which of the two alternatives is capable of providing more precise classifications, we compare

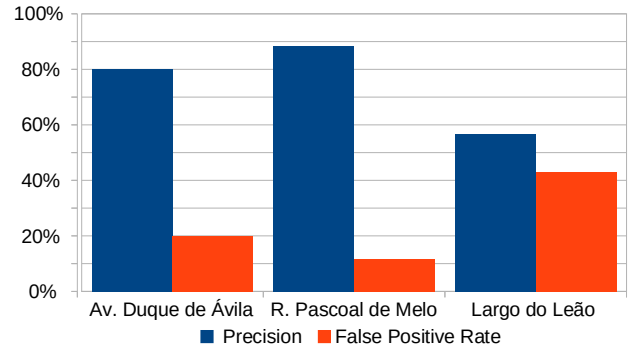


Fig. 4: Slope degree measurement's precision and false positive rate

both against the selected ground-truth.

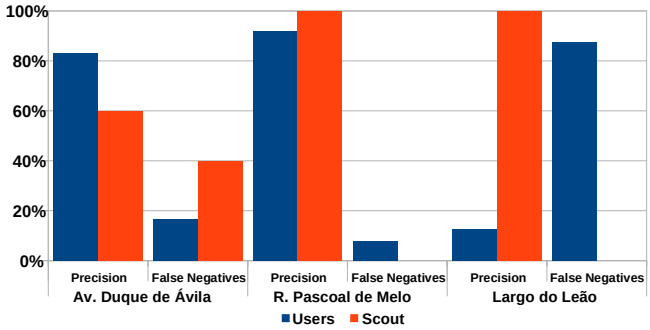


Fig. 5: Comparison between the user-based and Scout's slope classification with regard to the ground-truth.

In Av. Duque de Ávila, we can see that the users were more precise than Scout. There is, however, a reason for this precision decrease. Av. Duque de Ávila is not really a plain street, as there is actually a slight slope degree along its length (0-3%). So, even though Scout did not classify Av. Duque de Ávila with the same precision as the users, this should not be perceived as a misclassification, but as a side-effect of its increased precision.

In the case of the R. Pascoal de Melo, while most users classified the street correctly, Scout was able to do so with 100% precision.

Finally, in Largo do Leão we can see Scout's true potential. While Largo do Leão's climb does require more effort, than for instance R. Pascoal de Melo, its slope cannot be considered to be accentuated as it ranges between 5-8% [20]. This effort increase has a direct impact on the user's bias, which as we can see, most users classify this street it incorrectly. When compared to the user-based classifications, Scout is a better alternative. Because the application is oblivious to the user's bias, it can objectively classify a street's slope.

## V. Conclusion

Recent advancements in smartphone technology have lead to the appearance of a new paradigm of systems, the *mobile crowdsensing systems*. These have the potential to allow sensor-based studies to be performed at a larger scale and with reduced costs.

The focus of this thesis was to understand if a mobile crowdsensing-based system could be a viable and improved alternative to CycleOurCity's human-based route classification. The effort required to classify a route has been one of CycleOurCity's major obstacles to success, ultimately undermining its adoption by end users. Through the extension of the CycleOurCity platform to a mobile crowdsensing-based approach, it becomes possible to perform the route classification tasks in a transparent and effortless manner. However, in order for this to be possible without compromising the quality of the routes recommended by CycleOurCity, the proposed solution must be capable of classifying the characteristics of a street with the same precision as human-based classifications.

To allow the route classification task to be performed effortlessly, we developed and evaluated *Scout*, a mobile crowdsensing-based application. By taking advantage of the modern smartphones' sensing capabilities, Scout is capable of correctly measuring the slope and pavement type of the routes traveled by the participants. As demonstrated in Section IV, Scout is a viable alternative to CycleOurCity's user-based classifications. In particular, our evaluation shows that Scout is capable of correctly classifying a street's slope with an 79% precision, and the pavement type with 88.2%. While we cannot claim Scout to be more precise than human-based pavement classifications, as this is a subjective criterion, we have demonstrated that Scout's pavement classifications consistently to coincide with the user-based ones. Regarding a street's slope, because the human-based classifications are susceptible to the participant's bias, the classifications might not always correspond to the street's actual slope degree[6]. Since Scout's slope classifications are unbiased, they are more precise than the user-based classifications.

While a street's slope and pavement can be perceived as static characteristics of a city's route network, by allowing the topography and pavement mappings to be performed by a large community of volunteers, it becomes possible to reduce the costs associated these mapping endeavours. The pavement type mappings are especially valuable, as these are not typically maintained by any organization and can be used to support urban planning initiatives.

## VI. Future Work

The next step for the *CycleOurCity* project is to launch Scout as a production-ready application, for instance, in Google's *Play Store*[7]. However, for this to be possible we must provide a user-friendly application with added value. To offer a more attractive application, therefore attracting more

[6]Defined by Rosa Félix's slope cartogram [20]
[7]https://play.google.com/

participants, we have envision Scout as a route guiding app that guides its users to a specific destination, and while on route, in the background, is classifying the physical characteristics of the route.

One of the biggest challenges for any city's bicycle commuter is to choose the safest routes, in particular the ones that avoid streets with high traffic density. Hence, it would be of great value to provide Scout with the ability to classify a street's traffic density. There are two possible approaches to this challenge. A simpler approach would be to ask the participants to mark the most dangerous streets on a summarized route track, once they reached the end of their commute. A less intrusive, although more complex approach would be to leverage the device's sensing capabilities, namely the microphone and camera, to enable Scout with the ability to infer the street's traffic density.

Because the information currently being captured by Scout is of great value, not only to the CycleOurCity community, it would be interesting to enable public access to this information. This would facilitate and enable more responsive urban planning initiatives. While the topography of a city is typically known and seldom changes, the pavement type and in particular its condition are more dynamic characteristics of a city's route network. By facilitating access to this information city halls, for instance, could employ more responsive road reconstruction initiatives.

Finally to maximize Scout success, there are many mobile crowdsensing specific challenges that should be addressed. Namely, it is of great importance to assure the participants' privacy and security. Additionally, and because the participants may not be willing to participate if the application compromises the device's battery lifetime, it is important to address the energy-consumption challenges. These challenges may be addressed at three different levels, by adjusting the sensor's sampling rates as to conserve energy, by avoiding computationally intensive tasks, and by improving the data transmission operations.

## REFERENCES


[1] European Cyclist's Federation, "Cycling in urban areas," 1993.

[2] Interface for Cycling Expertise (I-ce), "The Economic Significance of Cycling: A study to illustrate the costs and benefits of cycling policy," 2000.

[3] European Environmental Agency (EEA), "A closer look at urban transport," 2013.

[4] N. F. Nunes, "Planeador colaborativo de deslocações de bicicleta em meio urbano," Master's thesis, Instituto Superior Técnico Lisboa, 2013.

[5] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *Communications Magazine, IEEE*, vol. 49, no. 11, pp. 32–39, November 2011.

[6] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell, "A survey of mobile phone sensing," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 140–150, Sept 2010.

[7] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. a. Peterson, G.-S. Ahn, and a. T. Campbell, "The BikeNet mobile sensing system for cyclist experience mapping," *Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07*, p. 87, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1322263.1322273

[8] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: Sensing and mapping for better biking," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1817–1820. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753598

[9] G. Broll, H. Cao, P. Ebben, P. Holleis, K. Jacobs, J. Koolwaaij, M. Luther, and B. Souville, "Tripzoom: An app to improve your mobility behavior," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '12. New York, NY, USA: ACM, 2012, pp. 57:1–57:4. [Online]. Available: http://doi.acm.org/10.1145/2406367.2406436

[10] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '08. New York, NY, USA: ACM, 2008, pp. 29–39. [Online]. Available: http://doi.acm.org/10.1145/1378600.1378605

[11] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 323–336.

[12] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli, "A survey on smartphone-based systems for opportunistic user context recognition," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 27:1–27:51, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2480741.2480744

[13] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[14] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 280–293. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644927

[15] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 71–84. [Online]. Available: http://doi.acm.org/10.1145/1869983.1869992

[16] Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, "A robust dead-reckoning pedestrian tracking system with low cost sensors," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*. IEEE, 2011, pp. 222–230.

[17] S. Hoseinitabatabaei, A. Gluhak, and R. Tafazolli, "udirect: A novel approach for pervasive observation of user direction with mobile phones," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, March 2011, pp. 74–83.

[18] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 337–350. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460445

[19] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 280–293. [Online]. Available: http://dl.acm.org/citation.cfm?id=1644927

[20] R. Félix, "Gestão da mobilidade em bicicleta. necessidades, factores de preferência e ferramentas de suporte ao planeamento e gestão de redes. o caso de lisboa," Master's thesis, 2012.