Urban Routing for Bicycles

Luis Miguel da Cunha e Silva Martins Costa
Urban Routing for Bicycles

Luis Miguel da Cunha e Silva Martins Costa

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Teresa Andrade
External Examiner: Prof. Adriano Moreira
Supervisor: Prof. Ana Aguiar

July 29, 2013
Abstract

The bicycle, as an affordable and environmentally friendly vehicle, is becoming, in recent years, an alternative means of transportation to motorized vehicles.

Recently, people are starting to leave their motorized vehicles at home and, instead, are starting to use their bikes for commuting, e.g., from their homes to their jobs. This tendency is happening due to several factors, like the economic crisis, people being more concerned about the environment or their health, or even without any specific justification.

Currently, several tools already exist to improve the experience of the bicycle rider. However, they are limited, in a bicycle riding perspective, since they lack ways to include specific road properties that are mostly important when riding a bicycle. These parameters vary from the road roughness to the presence of a cycleway in the road, and they should always be taken into account when finding a cyclist’s optimal route in terms of safety and efficiency.

In the context of this dissertation it is proposed a routing service for bicycles only, in order to empower people who want to use their bikes as a means of transportation with a tool to find optimal paths from their current location to their destination, like with any other vehicle. This routing service aims to help people traveling in an urban environment. Unlike other routing services, it considers a set of road parameters which are important to bicycles and which a road should have in more or less quantity, in order to improve the quality of the route.

In the implemented service the user can define a set of properties, to which he can assign weights, in order to inform the service which are the parameters that he values the most, ordered by their importance. Therefore, it is possible to differentiate roads by characteristics like slope, speed limit, number of directions, presence of a cycleway and width.

To gather the different parameters used to find optimal routes it was created a strong support system that feeds the service. This was possible by making use of a set of existing tools and using the database provided by the SenseMyCity application, coupled with the collaborative project OpenStreetMap.
Resumo

A bicicleta, como meio de transporte de baixo custo e amigo do ambiente, tem-se vindo a mostrar recentemente, como um meio de transporte alternativo aos veículos motorizados.

Recentemente, as pessoas estão a começar a deixar os seus veículos motorizados em casa e, em vez disso, a usar as suas bicicletas para se movimentarem na cidade, como por exemplo, para irem de casa ao trabalho. Esta tendência tem-se vindo a verificar devido a vários fatores, como a crise económica, as pessoas estarem mais preocupados com o meio ambiente ou a sua saúde, ou mesmo sem uma justificação específica.

Atualmente, já existem várias ferramentas para melhorar a experiência do ciclista. No entanto, estas são limitadas, numa perspectiva de andar de bicicleta, uma vez que não oferecem maneiras de incluir propriedades rodoviários específicas que são importantes para quem anda de bicicleta. Estes parâmetros variam desde as condições da estrada até à presença de uma ciclovia, e devem ser sempre tidos em consideração quando se procura pela rota ideal de um ciclista, em termos de segurança e eficiência.

No contexto desta dissertação é proposto um serviço de roteamento apenas para bicicletas, de modo a capacitar as pessoas que desejam utilizar as suas bicicletas como meio de transporte com uma ferramenta para encontrar caminhos ideais de seu local atual para o seu destino, como com qualquer outro veículo. Este serviço de roteamento tem como objetivo ajudar as pessoas que viajam em um ambiente urbano. Ao contrário de outros serviços que encontram caminhos, este considera um conjunto de parâmetros de estrada, que são importantes para quando se viaja de bicicleta, e que uma estrada deve ter mais ou menos em quantidade, de modo a melhorar a qualidade do percurso.

No serviço implementado o usuário pode definir um conjunto de propriedades, para os quais ele pode atribuir pesos, a fim de informar o serviço quais são os parâmetros que ele valoriza mais, ordenados por importância. Portanto, é possível diferenciar estradas por características como o declive, o limite de velocidade, o número de direcções, a presença de uma ciclovia e a largura.

Para reunir os diferentes parâmetros utilizados para encontrar as melhores rotas foi criado um forte sistema de apoio que alimenta o serviço. Isto foi possível através da utilização de um conjunto de ferramentas existentes e utilizando a base de dados fornecida pela aplicação SenseMyCity, juntamente com o OpenStreetMap.
Acknowledgements

The completion of this dissertation marks the beginning of my life as an engineer. As such I want to thank all who contributed to the beginning of a new cycle.

To my family for being my safe harbor through my whole life and to whom I can never thank enough.

To my brother João for being the person who always made me look at all situations with clarity and "beating" me up when I did things the wrong way.

To my friends Rui Fontinha, Helder Sousa and João Correia for the loyalty, the friendship and for the help they gave me to grow as a better version of myself.

Finally, to Prof. Ana Aguiar for the support in making important decisions, for the good advices and for the level of commitment that always demanded as my supervisor throughout this dissertation.

Luis Miguel da Cunha e Silva Martins Costa
“Life is like riding a bicycle - in order to keep your balance, you must keep moving.”

Albert Einstein
## Contents

1 Introduction 1  
1.1 Context and Motivation .............................................. 1  
1.2 Objectives ................................................................... 1  
1.3 Dissertation Structure ................................................. 2

2 Related Work 3  
2.1 Solutions to Improve the Bicycle Riders Experience ............ 3  
    2.1.1 Logging ................................................................. 3  
    2.1.2 Sharing ................................................................. 4  
    2.1.3 Geowikis ............................................................... 5  
    2.1.4 Routing ................................................................. 5  
2.2 Relevant Parameters for Cyclists .................................... 6  
    2.2.1 Road Roughness .................................................... 6  
    2.2.2 Road Gradient ....................................................... 7  
    2.2.3 Other Parameters .................................................. 11  
2.3 Collaborative Data Projects ........................................... 11  
    2.3.1 SenseMyCity ......................................................... 11  
    2.3.2 OpenStreetMap ..................................................... 12  
2.4 Summary ...................................................................... 12

3 Bicycle Navigation in Urban Environments 13  
3.1 Problem Description .................................................... 13  
3.2 Urban Routing Service for Bicycles ................................ 13  
3.3 System Architecture .................................................... 14  
    3.3.1 Parameters Module ............................................... 14  
    3.3.2 Optimal Path Module ............................................ 16  
    3.3.3 Android Application Module ................................. 17  
3.4 Relevant Parameters for Cyclists ................................... 17  
3.5 Metric Equations and Algorithms ................................ 18  
    3.5.1 Cycleway Existence .............................................. 19  
    3.5.2 Speed Limit Equation ........................................... 19  
    3.5.3 One Way vs Two Ways ........................................ 20  
    3.5.4 Road’s Slope Equation ......................................... 20  
    3.5.5 Road Width Equation ......................................... 21

4 Implementation 23  
4.1 Technologies .............................................................. 23  
    4.1.1 OpenStreetMap API ........................................... 23
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Classification of road roughness by ISO.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Slope Rule</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>OpenStreetMap’s Web Page</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>System Architecture UML</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Parameters Module</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Optimal Path Module</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Cycleway Partial Metric Equation</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Maxspeed Partial Metric Equation</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>Oneway Partial Metric Equation</td>
<td>21</td>
</tr>
<tr>
<td>3.7</td>
<td>Slope Partial Metric Equation</td>
<td>21</td>
</tr>
<tr>
<td>3.8</td>
<td>Width Partial Metric Equation</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Sequence Diagram for the Find Road Module</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Sequence Diagram of the Gather of all OSM Parameters</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>Sequence Diagram of the Estimation of the Slope</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Sequence Diagram of the Calculation of the Partial Metrics</td>
<td>33</td>
</tr>
<tr>
<td>4.5</td>
<td>Sequence Diagram of the Android Handler</td>
<td>34</td>
</tr>
<tr>
<td>4.6</td>
<td>Sequence Diagram of the Bicycle Metric</td>
<td>34</td>
</tr>
<tr>
<td>4.7</td>
<td>Sequence Diagram of the Route Finder</td>
<td>35</td>
</tr>
<tr>
<td>4.8</td>
<td>Sequence Diagram of the Mobile Application</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>Route of Bicycle Metric for the First Test</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>Route of Shortest Route Metric for the First Test</td>
<td>41</td>
</tr>
<tr>
<td>5.3</td>
<td>Route of Bicycle Metric for the Second Test</td>
<td>42</td>
</tr>
<tr>
<td>5.4</td>
<td>Route of Shortest Route Metric for the Second Test</td>
<td>42</td>
</tr>
<tr>
<td>5.5</td>
<td>Route of Bicycle Metric for the Third Test</td>
<td>43</td>
</tr>
<tr>
<td>5.6</td>
<td>Route of Shortest Route Metric for the Third Test</td>
<td>43</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Road Gradient Methods Restrictions .................................. 9
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GPX</td>
<td>Global Positioning System eXchange Format</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HyperText Transfer Protocol Secure</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MPH</td>
<td>Miles per hour</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>OSM</td>
<td>OpenStreetMap</td>
</tr>
<tr>
<td>SMC</td>
<td>SenseMyCity</td>
</tr>
<tr>
<td>SRTM</td>
<td>Shuttle Radar Topography Mission</td>
</tr>
<tr>
<td>TS</td>
<td>Traveling Salesman</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>URFB</td>
<td>Urban Routing for Bicycles</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Context and Motivation

In recent years, more and more people are starting to leave their car at home for health and environmental concerns and trying to find alternative ways of moving from a point to another. Bicycles as an affordable, environmentally friendly alternative transportation mode to motorized travel \[\text{[RSD}^{+}\text{10]}\], are becoming a popular urban means of transportation.

Therefore, there is a need to help residents and tourists which chose the bicycle as their means of transportation in finding the best route between two points, taking into account certain parameters which they consider important to measure the utility of traveling. According to a study by the Charlottesville Bike Route Planner \[\text{[TCB}^{+}\text{10]}\], bicycle riders differentiate the safety of the roads by speed limit, the presence of the cycleway and the width.

The main motivation of this dissertation is empowering people with a tool that encourages them to use their bicycles as a means of transportation more often. The tool is an automatic way of finding a route between two points. It uses an innovative way to compute the utility to travel each road: on one hand, defines normalized values for parameters relevant for bicycles; on the other hand, it uses weights provided by the user to personalize the relevance of each parameter, to report which parameters are the most important to have in the route.

This project is not meant to help people in their recreational or workout projects but to help cyclists traveling in an urban environment and improving their routes in terms of efficiency or safety, according to their specific needs.

1.2 Objectives

The objectives for this dissertation are:
Introduction

- Develop a routing service to suggest optimal routes for cyclists based on a set of parameters [TCB+10] that measure the route in terms of efficiency or safety from the point of view of the cyclist;

- The users must be able to personalize the routes according to their own needs by assigning weights to the different parameters;

- Merge external collaborative projects with a platform designed to gather the default importance of the parameters for each road;

- Develop an Android application for the final user to access the service anywhere in a city.

The project on this dissertation is a prototype of this routing service. It will be restricted to a specific area in the chosen city: Porto, Portugal. This area has the following boundaries: minimum latitude of 41.1441 (bottom); minimum longitude of -8.6223 (left); maximum latitude of 41.1793 (top); maximum longitude -8.5935 (right). This is limited because the automatic gather of the data for all streets in the city would consume too much time. It is estimated that gathering the information about this area will take approximatively two weeks to conclude.

1.3 Dissertation Structure

In this dissertation, in addition to the Introduction - Chapter 1 - there are 5 more chapters. In the Chapter 2 the related work is reviewed. Chapter 3 covers the problem and the solution of this dissertation. Chapter 4 has the technologies used as well as the implementation steps. Chapter 5 presents the tests and results. Finally, Chapter 6 draws the final conclusions and discusses future work.
Chapter 2

Related Work

This chapter reviews previous work in the areas of solutions to improve the bicycle riders’ experience, methods to calculate and/or gather important road parameters, and collaborative data projects.

First, it reviews the available solutions to improve the bicycles riders experience divided by three types. It starts with the applications for tracking and logging the route that the user is traveling, followed by the applications for sharing routes in a community, followed by the geowikis and then finally the applications to calculate an optimal route between two points.

Second, a review is presented of methods/algorithms to obtain/calculate parameters about roads which can be used to compute the utility of each road to find the optimal path between two points.

Thirdly, a review about existing collaborative data projects which are relevant to use in this project is presented.

Lastly, at the final section, there is a summary about what was reviewed in this chapter.

2.1 Solutions to Improve the Bicycle Riders Experience

The bicycle is an affordable means of transportation that is environmentally friendly. People that use bicycles either to travel from one point to another, to exercise or for recreational purposes, often dedicate some time to find optimal routes, in order to improve their experiences according to each one’s specific needs. This section reviews four different types of applications that exist to improve the bicycle riders’ experiences.

2.1.1 Logging

This kind of application aims at sharing people’s experiences in a community. It does not allow people to just share/draw their routes in the community. The users have to actually ride the route to share it because it logs the route the user travels.
Related Work

Generally, the user only starts the application on his mobile phone and it does the rest. The application tracks the route that the user travels and records all kinds of information about the route using GPS and phone sensors like accelerometer and microphone gathering parameters like, road noise, pavement roughness, distance traveled, speed and time. Good examples of logging applications are Biketastic [RSD+10], BikeNet1, EveryTrail2, and MapMyRide3. From these, the one that most stands out is Biketastic4 [RSD+10].

The main difference between Biketastic [RSD+10] and other ride loggers is that its objective is to log commuter routes (urban routes that every means of transportation use) and not only routes used for exercising or recreational trips. The mobile application records all sort of data while a user rides through a new or familiar route. Through the GPS the application can calculate the spatial and temporal extent, the length of the route and the speed. Using the accelerometer and the microphone it can record the road roughness, the pavement conditions and the noise level along the route. Biketastic also allows the users to take pictures or videos along the way. The route and all its information can later be visualized on the online map. According to the information it its website this application runs on iPhone and Android [RSD+10].

BikeNet is an application for exercise and recreational purposes. Its main objective is measuring the user experience taking into account parameters of two kinds: cyclist performance and environmental health. The cyclist performance is measured by current and average speeds, distance traveled and calories burned. The environmental health is measured by levels of pollution, allergen and noise and roughness of the terrain. All route information can be accessed on an on-line repository [EML+07].

EveryTrail5 is an application adapted mostly to road and mountain routes. The user can take pictures along the trip. Its main purpose is sharing the users’ trips and outdoor activities with each other, connecting people with similar interests. The platform is the combination of website with a mobile application that tracks the user’s route.

MapMyRide6 is a popular ride logger focused on exercise. Its main objective is to make the user workout. It tracks the users’ exercises and the user can later share some of his exercises achievements with his friends in order to challenge them in a healthy way. The system is composed by a website and a mobile application.

2.1.2 Sharing

Generally, these are tools which are built in websites made on top of mapping APIs to enable the users to draw and share their familiar routes with a community. Two popular websites for sharing routes are Bikely and Veloroutes.

---

1 http://www.bikenet.com/
2 http://www.everytrail.com/
3 http://www.mapmyride.com/
4 http://www.biketastic.com/
5 http://www.everytrail.com/
6 http://www.mapmyride.com/
Related Work

Bikely\(^7\) is a bicycle route mapping and sharing website. It allows the users to upload their routes based on GPX\(^8\) which is the GPS exchange format or draw a familiar route in its website. In the website the user can also search for other peoples routes and download them.

Veloroutes\(^9\) is a mapping website for bicycle routes. Its main objective is to allow the users to save, share and find familiar routes. It also allows users to create routes and calculate the average steepness between two points on the world map. It measures the quality of the bicycle routes in two levels, good and bad.

2.1.3 Geowikis

Geowikis are crowd source projects whose the main objective is to enrich the information available on collaborative maps like OpenStreetMap\(^10\). Generally, the geowiki will be a graphical layer that overrides the aspect of the map. An example of geowiki is a layer where the user can see on the map the topography of the land for a region. Users have to manually insert the data that they want to add or edit [FMS\(^+\)12].

Usually, geowikis for bicycles provide information about the road conditions as well as places like bicycle shops, hospitals, police departments, recreational and forest parks which are of interest for cyclists.

OpenCycleMap\(^11\) is a geowiki for bicycles based on data from the OpenStreetMap. At lower zoom levels it shows an overview of national cycling networks and at higher zoom levels its purpose is to help bicycle riders to choose the optimal path to cycle on and to show places like where to park their bicycles.

Cyclopath\(^12\) is another geowiki for bicycles. Its main objective is the sharing of information between cyclists. It has a companion Android application to help cyclists to access the information about the route while they ride.

2.1.4 Routing

Routing tools are services which estimate the best route between two points taking into account rules that evaluate the route. Popular rules which are generally found in this services are the shortest and the fastest route between two points. From the services of this category stand out three which can estimate paths for bicycles: RideMyCity, OpenTripPlanner and Google Maps.

Ride the City\(^13\) is an automatic route finder which estimates paths on urban environments. It has three modes of estimating the route: safer route, safe route and direct route. Ride the City also has routing restrictions like avoid motorways because bicycles can’t travel in that kind of road. It is available on multiple cities of five countries: USA, Canada, Australia, France and Spain.

---

\(^7\)http://www.bikely.com/
\(^8\)http://www.topografix.com/gpx.asp
\(^9\)http://veloroutes.org/
\(^10\)http://www.openstreetmap.org/
\(^11\)http://www.opencyclemap.org/
\(^12\)http://cyclopath.org/
\(^13\)http://www.ridethecity.com/
Related Work

OpenTripPlanner is a collaborative effort from the public transportation agency serving Portland on the state of Oregon USA, called TriMet\(^{14}\) and the developers of OneBusWay\(^{15}\) and of Graphserver\(^{16}\). This project main purpose is to find a route which uses multiple means of transportation, e.g., taking a train to some point and then get on the bicycle and go to another point. The paths can also be estimated to take just one means of transportation.

Google\(^{17}\) have a routing service for bicycles in Google Maps\(^{18}\) only for American cities. It allows users to choose if they want to avoid motorways and/or tolls and it gives the shortest route between the start and the end points.

2.2 Relevant Parameters for Cyclists

When traveling on a bicycle people often consider parameters about the road that they do not have to take into consideration when traveling on a motorized vehicle. Therefore, in order to distinguish a good from a bad path to travel between two points with a bicycle there are parameters about the road, e.g. the slope or the speed limit that are important to consider.

Most of the routing systems available do not take into consideration the safety parameters for bicycles like: the road’s width, the presence of the cycleway on the road, the road intersections on the route, the speed limit, the road conditions and the traffic density [TCB\(^{+}\) 10]. This section approaches some known ways of gathering, finding and calculating those parameters aiming to use them later on this project in order to calculate the utility to travel each road.

2.2.1 Road Roughness

Reference [GOLC08] proposes a method to estimate the road roughness from accelerometer data.

2.2.1.1 Estimation through Accelerometer data

To classify the roughness of the road the International Standards Organization (ISO\(^{19}\) 8608:1995) proposes a method that uses Fourier analysis to calculate the power spectral density (PSD) function of the surface. It classifies the profile into ‘A’ (very good), ‘B’ (good), ‘C’ (average), ‘D’ (poor) and ‘E’ (very poor) roughness indices, as shown in Figure 2.1.

\(^{14}\)http://trimet.org/
\(^{15}\)http://onebusaway.org/
\(^{16}\)http://bmander.github.io/graphserver/
\(^{17}\)https://www.google.com
\(^{18}\)https://maps.google.com/
\(^{19}\)http://www.iso.org/
Related Work

Figure 2.1: Classification of road roughness by ISO.

Vibrations and road profile can be related on the transform function in the next equation:

\[ H(\Omega) = \frac{PSD_{acc}(\Omega)}{PSD_{road}(\Omega)} \]  \hspace{1cm} (2.1)

where \( PSD_{acc} \) and \( PSD_{road} \) are the PSD for a frequency \( \Omega \) due to the vehicle accelerations and road profile, respectively. The PSD of a road can be estimated based on the PSD acceleration measured over the road profile [GOLC08].

2.2.2 Road Gradient

The road gradient also called slope refers to the inclination of the road to the horizontal. This section reviews two methods to estimate it.

Estimation using CAN bus data

CAN (or Control Area Network) is a serial communications protocol. In automotive electronics, engine control units, sensors, anti-skid-systems, etc. are connected using CAN [Bos91].

This method [MWW03] starts by creating a longitudinal acceleration vehicle model using the CAN bus data in the next equation:
Related Work

\[
a = \frac{1}{M} \left\{ \left( \frac{(T_e \times \mu) \times i_t \times i_d}{r} \right) - (b_p \times k_h) - (f \times M \times g \times \cos(\theta)) \right. \\
- \left. (0.5 \times \sigma \times c_w \times A \times v^2) - (M \times g \times \sin(\theta)) \right\} - \eta(i_t)
\] (2.2)

The \( \theta \) in this equation is the longitudinal road gradient that this method is estimating.

Next, this method computes the actual vehicle acceleration, measured by differentiating the wheel speeds. If the two accelerations are different from each other then the difference between them is called the error.

The next step is computing the longitudinal acceleration vehicle model using the error that is the difference between the two accelerations as the new \( \theta \) until the vehicle models acceleration converges with the actual acceleration. When they do converge, the last \( \theta \) used is the road grade.

The data needed to estimate the slope through this project is hard to obtain. Not only the CAN bus is not available in all vehicles (CAN bus it’s only mandatory in Europe since 2004\(^{20}\)) but also the equipment needed to retrieve this data has to be bought separately from the car and this brings additional unwanted costs to the project.

**Estimation using GPS**

GPS (Global Positioning System) is nowadays available in almost all smart-phones. The estimation of the road grade through the GPS can be performed by using the ratio between the vertical velocity \( (v_d) \) and the forward velocity \( (v_f) \) \([BRG01]\) through the next equation:

\[
\alpha = \tan^{-1}\left( \frac{-v_d}{v_f} \right)
\] (2.3)

The GPS available on smart-phones only gathers a part of this equation: the forward velocity \( (v_f) \). So, now, in order to estimate the road gradient there are two options available. On one hand, estimating the vertical velocity using the altitude \( (z) \) in different time instants \( (t) \) as follows:

\[
-v_{d,z} = -\frac{z(t) - z(t - 1)}{\Delta t}
\] (2.4)

and then using the equation of the ratio between the two velocities shown above on Equation 2.3.

Moreover, taking advantage of other parameters gathered by the GPS, the latitude, the longitude and the altitude, the estimation of the road gradient can be estimated by calculating the variation of the altitude at multiple points and the distance between those points. Therefore, the equation to calculate the slope through this last method is as follows \([AE95]\):

\[
\text{gradient} = \frac{\text{maxaltitude} - \text{minaltitude}}{\text{distance}}
\] (2.5)

\(^{20}\)http://www.obddiag.net/adapter.html
Table 2.1: Road Gradient Methods Restrictions

<table>
<thead>
<tr>
<th>Algorithm/Method</th>
<th>Data Source</th>
<th>Algorithm/Method Needs</th>
<th>Project Has Access to</th>
<th>Is it Possible to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal Road Gradient Estimation Using Vehicle CAN Bus Data [MWW03]</td>
<td>CAN bus</td>
<td>hydraulic break line pressure, braking coefficient, engine torque, gear ratio, tire radius, mass, rolling resistance, aerodynamic drag, climbing resistance, longitudinal acceleration, tires velocity along the way</td>
<td>forward velocity, distance, altitude, latitude, longitude</td>
<td>No</td>
</tr>
<tr>
<td>Road Grade and Vehicle Parameter Estimation for Longitudinal Control Using GPS [BRG01]</td>
<td>GPS</td>
<td>vertical and horizontal velocity</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Vehicle Sideslip and Roll Parameter Estimation using GPS [RRG02]</td>
<td>GPS</td>
<td>vertical and horizontal velocity</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Robust Sideslip Estimation Using GPS Road Grade Sensing to Replace a Pitch Rate Sensor [RBL09]</td>
<td>GPS</td>
<td>vertical and horizontal velocity</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Road Grade Estimation for On-Road Vehicle Emissions Modeling Using Light Detection and Ranging Data [ZF06]</td>
<td>GPS</td>
<td>vertical and horizontal velocity</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.1 shows a summary of the methods reviewed, the variables that each needs to estimate the road gradient and whether this project have access to them. As it can be seen in the table, none of the studied articles estimate the road gradient from the variables that this project have access to.

So, in order to estimate this parameter this project chose among other alternatives to use the equation of the slope presented on the Book of Calculus - A Complete Course [AE95]. This equation consists on using the distance traveled and the altitude difference as shown on Figure 2.2.
At Figure 2.2 there are two variables that we have to estimate in order to get the slope. Thus, the method to calculate the slope follows the three steps by the next order:

1. The *Run* has to be estimated taking into account the distance between the two farthest pair of coordinates in a certain road. In order to calculate the distance between pairs of coordinates we need to follow three steps accordingly to the *MovableTypeScripts*\(^{21}\) website:

(a) First, estimate the difference in latitude and longitude between the latitudes and longitudes of the two pairs of coordinates:

i. 
\[
diff_{\text{latitude}} = \text{Radians}(\text{latitude}_1 - \text{latitude}_2) \tag{2.6}
\]

ii. 
\[
diff_{\text{longitude}} = \text{Radians}(\text{longitude}_1 - \text{longitude}_2) \tag{2.7}
\]

(b) Then, uses the Haversine formula to calculate the distance being \(R = 6,371 \text{km}\) the radius of Earth and \(d\) the distance in meters:

i. 
\[
a = \sin\left(\frac{\text{diff}_{\text{latitude}}}{2}\right) \times \sin\left(\frac{\text{diff}_{\text{latitude}}}{2}\right) + \cos(\text{latitude}_1 \times \frac{\pi}{180}) \times \cos(\text{latitude}_2 \times \frac{\pi}{180}) \times \sin\left(\frac{\text{diff}_{\text{longitude}}}{2}\right) \times \sin\left(\frac{\text{diff}_{\text{longitude}}}{2}\right) \tag{2.8}
\]

ii. 
\[
c = 2 \times \arctan\left(\frac{\sqrt{a}}{\sqrt{1 - a}}\right) \tag{2.9}
\]

iii. 
\[
d = R \times c \tag{2.10}
\]

\(^{21}\)http://www.movable-type.co.uk/scripts/latlong.html
Related Work

\( \text{Run} = d \) \hspace{1cm} (2.11)

2. The \textit{Rise} is the difference of altitude between the two points on the first step and is estimated from the following equation:

\[ \text{Rise} = |\text{altitude}_1 - \text{altitude}_2| \] \hspace{1cm} (2.12)

3. The slope of the road in degrees is the \textit{arctangent} ratio between the Rise and the Run:

\[ \text{Slope} = \arctan\left(\frac{\text{Rise}}{\text{Run}}\right) \] \hspace{1cm} (2.13)

Therefore, by estimating the distance between the two farthest points of a certain road and the difference of altitude between them this project will be able to estimate the gradient of the road.

2.2.3 Other Parameters

Through review of the study of the \textit{Charlottesville Bike Route Planner} \cite{TCB10} which reviews a set of parameters including the road width, the presence of the track for bicycles in the road and the speed limit, came the conclusion, that it would be advantageous for this dissertation to obtain those parameters from the collaborative project, OpenStreetMap\textsuperscript{22} (OSM). The method of how to get this parameters will be reviewed on the Implementation chapter, the Chapter 4.

2.3 Collaborative Data Projects

2.3.1 SenseMyCity

SenseMyCity is a new application for smart phones with Android operating system developed under the Future Cities Project\textsuperscript{23}. This is an application designed to aid research projects in collecting data. It gathers information about the everyday life of the users through a choice of the sensors available on the smart phone.

The data is gathered every second after the application is turned on. That way, the users can record, consciously and voluntarily the information about their daily routines and then later they can consult it on the website created for that purpose.

This dissertation will use the SenseMyCity’s database. This have already \textit{471 456 GPS points} which have a latitude, a longitude and an altitude that will be applied on the estimation of the road parameters. As all this points have an altitude associated to them then with this information the road gradient can be estimated as it can be reviewed on the Section 2.2.2.

\textsuperscript{22}http://www.openstreetmap.org/
\textsuperscript{23}http://futurecities.up.pt/site/
2.3.2 OpenStreetMap

The OpenStreetMap\(^{24}\) (OSM), is a collaborative project founded with the objective of making a geographic map that everyone could use to create and share free geographic info about the world. It makes available to everyone information that other geographic maps available on the Internet conceal behind legal and technical restrictions, i.e. a graph of the road network.

![OpenStreetMap's Web Page](image)

Figure 2.3: OpenStreetMap’s Web Page

2.4 Summary

Primarily, in this chapter, was reviewed what has been done so far to improve the bicycle riders experience. There is a gap on the tools to improve the cyclists’ experience. There are no known tools to find the best route to travel on a bicycle between two points considering what the user thinks that it is safe or efficient. This project will provide a tool to let users decide what is best among the parameters which were reviewed in this chapter. Some techniques to estimate parameters about the road were described and others can be obtained from collaborative projects over the Internet. Finally were reviewed some collaborative data projects which can aid this dissertation.

\(^{24}\)http://www.openstreetmap.org/
Chapter 3

Bicycle Navigation in Urban Environments

3.1 Problem Description

The main problem in the context of this dissertation is the need for a routing system for bicycles that provides an optimal path in an urban environment to users that want to use their bicycles as their means of transportation. Their objective is to travel from a point to another in the most efficient way and they should be able to choose what the route properties they value the most are. Current solutions to improve the bicycle riders experience lack this kind of application.

The proposed solution should be able to find an optimal route for a specific bicycle rider, in other words, should be able to incorporate the user preferences about what properties the route should have. For this purpose the system should gather parameters about each road that should be taken into account when traveling by bicycle. The gathering of those parameters is also a problem because they must be obtained from different sources.

The cyclists should be able to indicate which parameters are more important to them, in order to find a more personalized optimal route. Therefore, the service should receive and take into account the different parameters weights when estimating that route.

3.2 Urban Routing Service for Bicycles

The solution to the problem on the context of this dissertation is a routing service to find optimal paths for cyclists. By necessity, the cyclists are, in most cases, people that always try to find new paths, in order to improve the efficiency of their journeys. On the perspective of this project this system improves their search because it is not a sharing of routes, it is not a logging application nor it is a geowiki, it computes the route based on the user’s choices, i.e, it finds the route fit to the needs of the user.

This service will also differ from other routing services on the parameters because, the type of road to choose for a bicycle to travel is a principal to find the best route. The choose of parameters
with which the routing algorithms compute the turning utility of the roads in a map create the biggest gaps between those algorithms. So, with the objective to generate optimal paths for each user as an unique person this service will use a system in which the users can insert weights to each of the given parameters in order to find a route fit to their needs. Therefore, on one hand, the service have its unique parameters that will not change and have predefined values. On the other hand, the service accepts weights to those parameters which are defined by the users in order to mold them to their needs.

The proposed routing service to find optimal paths for the cyclists is the combination of a server based system with a mobile application used by the user. This service must be narrowed by the choices of the users. As a user asks for the optimal path between two points he also indicates from a set of given parameters which are the ones he considers more or less important to better determine his path in terms of efficiency and/or safety.

Therefore, the system obtains the weights which the user gives to each of the given parameters. Then, it combines those weights with each road’s parameters which previously gathered (see Section 3.4). The combination of those weights with the parameters is a metric that serves as the utility to travel each road. The system also gets two pairs of coordinates from the user, the start and the end nodes. Finally, using the two nodes and the obtained metric the system finds the optimal path and shows it to the user.

The system have two modules that were developed under this project. On one hand it is the module to find, gather, treat and store the important parameters for bicycles and on the other hand, the module to find the optimal path and show it to the final user.

### 3.3 System Architecture

The system has three components: The Database Server where there are the SenseMyCity and the Urban Routing for Bicycles (URFB) databases, the Routing Server which is the main platform of the system and the Mobile Device which hosts the application that the final user will use to access the optimal path service. There is also an external platform which will be used that is the OpenStreetMap Server that is access over the Internet.

The Figure 3.1 shows the system architecture to the detail of which components are connected with each other.

#### 3.3.1 Parameters Module

The Parameters Module is located on the main server of this project: the Routing Server.

This module gathers, treats and stores the data to feed the system. This module is responsible for three tasks: gather the data about each road from the OSM API (Section 4.1.1), estimate the road gradient from the GPS data in the SenseMyCity’s database (Section 4.1.6) and lastly, compute the partial metric for all parameters in each road.
The SenseMyCity’s (Section 4.1.6) saves the data on the database as points without a road. Therefore, in order to get the points of each road to estimate their slope, first it is needed to use the Nominatim API (Section 4.1.2) in order to attribute a road to each of the points in the database.

The Figure 3.2 illustrates which tasks are within this module and their interfaces to other important modules.

The Find All Roads Data task is the one responsible for querying the OpenStreetMap API (Section 4.1.1), in order to get the parameters which characterize each road. Then, it chooses which are the ones that are important to this project and it stores them in the URFB Database (Section 4.1.6). The parameters to query the OSM API are the boundaries of the area for which this prototype is developed.
The Find Road task is responsible for obtaining a road to each individual GPS node on the SenseMyCity Database (Section 4.1.6). In order to achieve this, it gets the GPS nodes from the SMC database. Then, it queries the Nominatim API (Section 4.1.2) providing a GPS node at a time, in order to, get the address to which that point belongs. Finally, having received the correct information, it associates the road address to the GPS node in the database.

The Estimate Slope task shown on the Figure 3.2 is the one responsible to estimate the road gradient of each road in the defined area. Therefore, it connects to the SenseMyCity Database querying for all the nodes in each road. Then, it applies an algorithm which will be reviewed later on the Implementation in Section 4.2.1.3. Finally it attributes to each road their respective slope in the URFB Database.

The Compute Partial Metrics task serves the purpose of normalizing all parameters to the same interval. The method starts by querying the URFB Database, for all the parameters associated with each road. Then, it attributes a partial metric value to each parameter following the equations of Section 3.5. Finally it assigns to each road the final partial metrics connecting to the URFB Database.

3.3.2 Optimal Path Module

The Optimal Path Module is located on the main server of this project: the Routing Server.

This module has two tasks. First, it waits for the Android application to establish the communication with it in order to know what are the start and end nodes, and the weights which the user provides. The second task is to get the optimal path between the two nodes considering the weights given by the user.

The optimal path is computed recurring to a Java library that will be reviewed on Section 4.1.4. Therefore, whenever it is needed, this task communicates with the URFB Database in order to get the partial metrics of each road. Then, they are used together with the weights given by the user to compute the utility to travel each road.

The Figure 3.3 illustrates the tasks within this module and their internal and external interfaces.

The Android Handler task is one that is always running and waiting for the mobile application (Section 3.3.3) to establish a connection to it. When it connects with the mobile application it receives the start node, the end node and the weights which the user gives to each parameter. Then, it calls the Route Finder task and passes it the received information as parameters in order to start it.

The Route Finder task begins whenever it is properly called, i.e., it has the right parameters. This task will then estimate the optimal path through a Java library with the aid of the partial metrics which are stored on the URFB Database. Finally when the optimal path is found, the task returns it to the Android Handler which communicates it to the mobile application (Section 3.3.3).
3.3.3 Android Application Module

The Android Application module is located on the Mobile Device and it is the tool which the user uses to access the routing service. This module and its interface can be seen on the Figure 3.3.

This module have a GUI (Graphical User Interface) which the user accesses in order to choose its destination and to insert its weight on each of the five parameters. Then, the application sends the information to the Android Handler of the server. Finally, when the application receives the data about the path, it shows it to the user in the map.

3.4 Relevant Parameters for Cyclists

The base of this project is a module that, gathers, treats and stores, a set of parameters from different sources. This is done with the purpose of building up a database to the project with values which refer to those parameters in order to characterize each individual road.

There are two kinds of parameters. On one hand, there are those which are gathered directly from a known source. On the other hand, there are the ones which have to be estimated through the data available on the database. In this project there are five parameters of which four are gathered directly from the source and one other, the road gradient, is estimated through the available data. Those parameters are:

**slope**: It is the road gradient of the road;
**Bicycle Navigation in Urban Environments**

- **oneway**: It is a boolean parameter that distinguishes if a road can be traveled in one or two directions;

- **cycleway**: It is a boolean parameter that indicates if a road has a cycleway or not;

- **maxspeed**: It is the speed limit of the road;

- **width**: It is the number of lanes the road has.

The parameters are then used to compute the partial metrics (see Section 3.5). These values serve the purpose of normalizing the parameters to the same interval. This is done because in order to compare the values on weight they have to be on the same interval of values.

This process is totally apart from the service that finds the optimal path and it is all done in background so the user has no interaction with it.

The equation which estimates the metric that serves as the utility to travel each road uses the partial metric of each road which are reviewed on Section 3.5 and also uses weights which the user can attribute to each parameter in order to personalize the route accordingly to his preferences. The equation of the metric is as follows:

\[
\text{metric} = \text{cyclewaypartialmetric} \times \text{cyclewayweight} \\
+ \text{maxspeedpartialmetric} \times \text{maxspeedweight} \\
+ \text{onewaypartialmetric} \times \text{onewayweight} \\
+ \text{slopepartialmetric} \times \text{slopeweight} \\
+ \text{widthpartialmetric} \times \text{widthweight}
\] (3.1)

where the equation parameters are:

- **cyclewaypartialmetric**: cycleway partial metric (see Section 3.5.1);
- **cyclewayweight**: cycleway weight assigned by the user;
- **maxspeedpartialmetric**: maximum speed partial metric (see Section 3.5.2);
- **maxspeedweight**: maximum speed weight assigned by the user;
- **onewaypartialmetric**: number of directions partial metric (see Section 3.5.3);
- **onewayweight**: weight of the number of directions assigned by the user;
- **slopepartialmetric**: slope partial metric (see Section 3.5.4);
- **slopeweight**: slope weight assigned by the user;
- **widthpartialmetric**: width partial metric (see Section 3.5.5);
- **widthweight**: width weight assigned by the user.

### 3.5 Metric Equations and Algorithms

The need to the use of equations on this project emerged with the aim to normalize all the obtained parameters to the same interval, in this case, to the [0,1] interval.
The algorithm that finds the optimal path used by this project is a maximization algorithm. Therefore the equations must maximize what is best for the bicycles to travel, e.g., if the road has a cycle way the partial metric of cycleway for this road must be 1.

Next are described those equations that compute the partial metrics based on the parameters.

### 3.5.1 Cycleway Existence

The cycleway is a parameter that indicates if a road has a cycleway or not. The value of this parameter is boolean, i.e., is either 1 or 0 if the road has a cycle way or not, respectively.

From a bicycle rider perspective it is better if a road has the cycleway which means that what happens is that the utility of this road is higher by comparing it to a road that has no cycleway, assuming an hypothetical case that the two roads only have this parameter to judge.

Therefore, as said before, the equation must maximize the better attribute so there are two possible values in this case as follows:

1. If the parameter is 1, the partial metric will be 1;
2. If the parameter is 0, the partial metric will be 0.

The chart on Figure 3.4 shows how this equation \((y)\) varies giving the cycleway \((x)\) parameter.

![Figure 3.4: Cycleway Partial Metric Equation](image)

### 3.5.2 Speed Limit Equation

The maxspeed parameter serves the purpose to indicate the speed limit of a road. From the viewpoint of someone riding a bicycle, the faster the traffic in a road for motorized vehicles, the greater is the danger to travel on it.

Given the fact that, the speed limit inside an urban environment in Portugal is 50 kilometers per hour, then, the equation that maximizes the minimum velocity, given the variable \(p_{\text{maxspd}}\) be the partial metric of the maxspeed parameter, is:
Bicycle Navigation in Urban Environments

\[ p_{\text{maxspd}} = 1 - \frac{\text{maxspeed} - 10}{40} \]  \hspace{1cm} (3.2)

Figure 3.5: Maxspeed Partial Metric Equation

The chart on the Figure 3.5 shows the equation of this partial metric varies giving the maxspeed \((x)\) parameter. By consulting the referred chart one can ascertain that this equation do not assign values to a maxspeed lower than 10 kilometers per hour because it is assumed by default that the lowest speed on a road inside an urban environment is 10 kilometers per hour and that is the best speed limit that a bicycle might encounter on a route.

### 3.5.3 One Way vs Two Ways

The oneway parameter indicates if the road has one or two directions. If a road has two directions then this is almost like indicating that the danger is traveling both ways, from the front and the back of the bicycle rider. This is the case where the rider has to be fully aware of both traffic flows, so, the danger is doubled compared to a one way road.

Therefore, the equation to compute the metric from this parameter has to maximize the one way road and minimize the two ways road:

1. If the road has two ways, the value on the parameter is 0 so in this case the equation will return 0;
2. If the road has one way, the value on the parameter is 1. The equation will return 1.

The chart presented on the Figure 3.6 shows how the equation \((y)\) varies giving the oneway \((x)\).

### 3.5.4 Road’s Slope Equation

Accordingly to the Institute of Roads and Infrastructures\(^1\) the slope of a road in Portugal should not exceed the 10 per cent mark. Therefore, if the estimated slope of a road exceeds this value it

\(^1\)http://www.inir.pt/portal/
must be an error of estimation, most probably due to an error on the altitude on one of the points. The algorithm to estimate the slope is described later in the Section 4.2.1.3.

Through some study was made the decision of using a symmetric equation to the slope in order to treat equal roads that fall or rise too much because on one hand traveling a road that rises is more exhaustive for the cyclist that a road without slope. On the other hand if the road falls too much the danger of traveling it increases because the cyclist can’t easily dodge obstacles which may appear in the way. So, the partial metric $pmetslope$, for the slope of a road being the $maxslope$ the maximum slope that any road can have, is computed as follows:

$$pmetslope = 1 - \frac{e^{slope}}{e^{maxslope}}$$  \hspace{1cm} (3.3)

The chart on Figure 3.7 presents as this equation varies, being $x$ the slope.

### 3.5.5 Road Width Equation

The width parameter that appears on the URFB database is the number of lanes that a road has. The probability of having more traffic volume in roads increases with the number of lanes that
Bicycle Navigation in Urban Environments

each road has. Therefore, the greater the number of lanes of a road, the higher the risk for cyclists to travel in it, assuming that a road’s danger increases as the traffic volume increases.

As this project fits into an urban environment, it was decided that the maximum number of lanes in a road would be 5. For this reason, the equation of the width’s partial metric ($pmetwidth$) that maximizes the value as the number of lanes ($width$) decreases is:

$$pmetwidth = \frac{|width - 5|}{4}$$

(3.4)

The chart in Figure 3.8 represents the Equation 3.4.

Figure 3.8: Width Partial Metric Equation
Chapter 4

Implementation

4.1 Technologies

This section reviews the APIs, Java libraries and the Databases available to aid this project.

4.1.1 OpenStreetMap API

The OpenStreetMap API\(^1\) is accessed to use one of its tools named *map* for two different purposes. On one hand to get the information about each road on the intended area. Also, to provide the optimal path module with the road network, turn restrictions and road relations in the same area. This tool uses RESTful technology and can be accessed by following the link:

\[
\text{http://api.openstreetmap.org/api/0.6/map?bbox=left,bottom,right,top} \quad (4.1)
\]

The parameters of the link in the Expression 4.1 are:
- **left**: is the longitude of the left (westernmost) side of the bounding box.
- **bottom**: is the latitude of the bottom (southernmost) side of the bounding box.
- **right**: is the longitude of the right (easternmost) side of the bounding box.
- **top**: is the latitude of the top (northernmost) side of the bounding box.

The return data which this tool provides is a *XML type file*. Also, it might return two types of errors: *HTTP status code 400 (Bad Request)* when any of the way/node/relation limits are crossed and *HTTP status code 509 (Bandwidth Limit Exceeded)* when too much data was downloaded.

An example of XML file retrieved from the OpenStreetMap API is:

```xml
<node id="25632373" lat="41.1737404" lon="-8.6206539" user="ZZZZZZ" uid="2345678"
visible="true" version="1" changeset="207073" timestamp="2007-02-03T12:23:48Z"/>
```

\(^1\)http://wiki.openstreetmap.org/wiki/API_v0.6
A road is called *way* in the OpenStreetMap’s XML file. This *way* has a key called *id* that is its identification number and is unique. Each *way* has a set of items which characterize it and are explained below:

1. **nd**: It is a reference to a node, it has an element called *ref* with the identification number of the respective node. Each way can have more than one node associated to it. Some of these nodes are the intersections between two or more different roads;

2. **tag**: It has specific information about the way. By default, it is composed by two elements, *k* that is the key that defines the type of the element and *v* that is the value associated with the item. A *way* can have various of this items but each one has only one type at a time. Some of these types are:

   (a) **cycleway**: This type is to define if a road has a cycleway. The item might not exist because it is not mandatory. The values available in this type are, *lane*, *yes* or *no*. Either the value *lane* or *yes* are both to say that the road has a cycleway. On the other hand the value *no* indicates that the road doesn’t have a cycleway;

   (b) **highway**: Defines the type of the road. It has various denominations, the most important ones include: *motorway*, *motorway_link*, *trunk*, *trunk_link*, *primary*, *primary_link*, *secondary*, *secondary_link*, *tertiary*, *tertiary_link*, *living_street*, *residential*, *pedestrian*, *unclassified*, *service*, *track*, *bus_guideway*, *raceway*, *road*;

   (c) **lanes**: Determines how much lanes a road has e.g. 2. This is not a mandatory tag.

   (d) **maxspeed**: Determines the speed limit of the road. The various speed limits can be:

   %Number%

   Stands for %Number% of kilometers per hour (e.g. 50);
Implementation

%Number% mph
For %Number% of miles per hour (e.g. 30 mph). The %Number% and the metric are always separated by a blank space;

%Number% knots
When the speed limit is in %Number% of knots (e.g. 60 knots). The %Number% and the metric are always separated by a blank space;

walk
Walking speed;

none
No speed limit in this section of the road;

signals
Indicates that the speed limit can vary depending on which numeric value is marked on an electronic sign.

It is not mandatory.

(e) name: It is the name of the road. It most probably exists but it is not mandatory;

(f) oneway: It determines if the road has one or two ways. The values are not only boolean: yes, no and -1 (which is applied when the road had one direction and then changed it);

(g) width: It’s the width of the road. The values associated to it can be:

%Number%
The width of the road in meters. This %Number% belongs to the set of rational numbers;

%Number% mi
Stands for widths in miles. The %Number% belongs to the set of rational numbers. The %Number% and the metric are always separated by a blank space;

X’Y”
Defines a width in X feet and Y inches.

The XML obtained from OpenStreetMap has also a node. This one has a set of keys that are relevant to describe next:

1. id: it is the identification number of the node. This is sometimes referenced in a way;

2. latitude: it is the latitude of the point in degrees;

3. longitude: it is the longitude of the point in degrees.
Implementation

The *latitude* and *longitude* referenced above is useful too, to test the Nominatim (Section 4.1.2) API because, if a node is referenced in a way, the pair of coordinates must belong to the road that is this way. So, if the road given by the API for this pair of coordinates is the same, the API is providing the correct data. Then by doing this to a set of 100 points if all of them give the correct data it’s more then enough to conclude that the Reverse Geocoding of this API can be used in this project.

### 4.1.2 Nominatim API

The Nominatim API\(^2\) is used to provide Reverse Geocoding which returns the address to which a given pair of coordinates belongs. The given information can be one of two. On one hand, it returns a road to which the given node belongs and on the other hand, it returns an exact address within a road if the node is an unique place.

An example of a call to this API with the pair of coordinates \((41.15591049,-8.61321640)\) is:

```
http://nominatim.openstreetmap.org/reverse?format=json&lat=41.15591049&lon=-8.61321640, and replies with the following JSON file:
```

```
{
2 "place_id":"24255269",
3 "licence":"Data \u00a9 OpenStreetMap contributors, ODbL 1.0. http:\/\www.openstreetmap.org\copyright",
4 "osm_type":"way",
5 "osm_id":"8367092",
6 "lat":"41.1560605",
7 "lon":"-8.6179574",
8 "display_name":"Rua da Boavista, Miragala, Cedofeita, Porto, 4200, Portugal, European Union",
9 "address":{
10 "road":"Rua da Boavista",
11 "neighbourhood":"Miragala",
12 "suburb":"Cedofeita",
13 "city":"Porto",
14 "postcode":"4200",
15 "country":"Portugal",
16 "country_code":"pt",
17 "continent":"European Union"
18 }
19 }
```

---

Listing 4.2: Nominatim JSON Example

In the above JSON (Listing 4.2) can be seen a field called osm_type and another called osm_id. These two fields have both information regarding how the object interacts with OpenStreetMap. First, with the osm_type field it can be known if this object’s type is a road or a node. On the other

---

\(^2\)http://wiki.openstreetmap.org/wiki/Nominatim
Implementation

hand, the osm_id indicates the identification number of the object in the OpenStreetMap. So, if the API returns a node it is automatically discarded because the aim is to obtain roads; if the object is a road then its osm_id is stored.

4.1.3 GeoNames API

GeoNames\(^3\) is a geographical database. It contains various web services through an API\(^4\) to search not only for postal codes and street names but also for the elevation of a pair of coordinates. To provide information about the altitude at a certain pair of coordinates it uses the NASA’s Shuttle Radar Topography Mission (SRTM) elevation data project. So, if there is a search for the elevation of a certain point (pair of latitude and longitude) the API searches in the Database of this project for it and returns an object in JSON with the altitude, the latitude and the longitude.

\[
\text{http://api.geonames.org/srtm3JSON?}
\]
\[
\text{username = USER&lat = LAT &lng = LON}
\]  

Listing 4.3: GeoNames JSON Example

The Expression 4.2 returns a JSON like the one on the example of the Listing 4.3. The parameters of this expression are: \text{USER}, \text{LAT} and \text{LON} which stand for user name, latitude and longitude respectively. The field \text{srtm3} provides the altitude of the node.

This API is used as a correction if the stored altitude referenced by a certain pair of coordinates shows a high disparity compared to the altitudes measured on other pairs of coordinates in the same road.

4.1.4 Traveling Salesman Library

Traveling Salesman is a Java library to aid in the creation of a navigation service fit to one’s needs. This library supports maps from the OpenStreetMap (Section 4.1.1) in XML format. This maps can be uploaded directly from the OSM or from a file. Whenever they are uploaded directly the box with the limits of the map in latitude and longitude must be defined.

It contains various abstract classes to alter the options of the routing algorithm, i.e., the developer can alter the metrics and other restrictions as it’s necessary. The choice of the metric is a very important step to change the default routing algorithm to the one in this project. Also, the

\(^{3}\)http://www.geonames.org/

\(^{4}\)http://api.geonames.org/
defining of some restrictions about the roads are of equal importance, i.e., the roads that a bicycle can travel and those that it cannot e.g. a bicycle can’t travel on a motorway so this is a restricted type of road.

The algorithm used by this library is a **Turn Restricted AStar (A*)** that **maximizes** the path, i.e., the best path it is the one that **has the higher utility to travel**. The library previously estimates the shortest distance between the two points. Then, the optimal route estimated through the bicycle metric can only exceed the shortest distance in a limited percentage of 6% (six percent) which is defined by the library and which cannot be changed.

### 4.1.5 OSMDroid Library

OSMDroid[^1] is a library for Android Applications to show OpenStreetMap information. Using this library one can paint various GPS points on a map and even paint an whole path between two nodes. It is used in the context of this dissertation to show the route found by the routing service to the user in the mobile application.

### 4.1.6 Databases

The databases which support this project are located on the local network Database server. Those are the SenseMyCity and the Urban Routing for Bicycles (URFB) databases. The SenseMyCity database is used to get all the GPS nodes from the defined area for this prototype. These nodes are used on the estimation of the road’s gradient. The GPS nodes available on the database which are in the defined area can belong to any road and are not marked yet. So, in order to properly estimate each road’s gradient, first each GPS node has to be attributed to a road. The URFB database, it is where the parameters about each road and their partial metrics are stored.

### 4.2 Developed Modules

In this section are described the two modules of the system and the scripts behind each on of them.

#### 4.2.1 Parameters Module

##### 4.2.1.1 Find Road: Reverse Geocoding

This project’s first step is to assign to all the GPS points on the SenseMyCity database (Section 4.1.6) the road to which they belong to, in the area of the city chosen for the prototype. This process is called **Reverse Geocoding** and can be achieved by using the API of Nominatim (Section 4.1.2).

A script was created to assign the points automatically to the respective roads. To facilitate the access to the GPS points this process begins by storing all of them in a local file. In order to attribute a road to a point this script had to go through the following steps:

[^1]: https://code.google.com/p/osmdroid/
Implementation

1. Read the point’s latitude and longitude from a file;

2. Get the information about the point using the API of Nominatim and returning it in JSON format;

3. Parse the JSON object to get the street’s ID number;

4. Write the way ID in the database associating it to the point that was read.

The first step tests if a file with all the rows of the GPS points table is created. If it is not created then the script uses the JDBC - Java Database Connectivity - library to make a query, to the table of GPS points, asking for all the rows in the table and storing them in a local file in order to facilitate the access to them. This is made this way so that if the script stops for some motive, it will not have to connect again to the database to continue the job. This way it saves time and resources and also, it’s better for the script to consult the local file instead of query the remote database all the time for the next point.

The second step is a bit more complicated. There are 3 different situations that can happen going through this step:

1. When the pair of coordinates do not have a street ID and the Nominatim’s server isn’t busy the step concludes without problems and does exactly as it was told, gets the information about the pair of coordinates using the API and returns that information in JSON format;

2. When the pair of coordinates already had a street ID associated to it on the database, this step stops and returns to the step one getting the next pair of coordinates;

3. When the server where the Nominatim’s API is located, is busy, the script does the second step over and over again with intervals of 5 seconds before it reconnects until it gets the data about the point.

These are three situations that can happen in this step and the script is fully prepared to treat them.

The third step is only to parse the JSON. It receives an object in JSON format like the shown in the Example 4.2, and it takes two actions to get the information it needs:

1. First it is tested if the object is a way. This is done because sometimes for specific nodes this API returns a node of an address and it does not have a way ID associated. So, it is a node nothing will be returned and the script returns to the first step with the next pair of coordinates. On the other hand, if it is way the last action will begin;

2. Lastly, the way ID is parsed and the script can proceed to the fourth step.

The fourth and last step is only as described, that is, write in the database the way ID associating it to the pair of coordinates that was read on the first step.

The Figure 4.1 is the Sequential Diagram for this script and all that interacts with it through the four steps listed above.
Implementation

4.2.1.2 Find All Roads Data: Obtain Parameters from OpenStreetMap

The collaborative project OpenStreetMap, provides some information about the roads that can be used as parameters.

As reviewed before on this dissertation it is possible to download a XML file using the API of the OSM. This XML have a structure that looks like the Code Example 4.1. The keys of the tags that were taken from the OpenStreetMap data are: maxspeed, oneway, cycleway and lanes.

As were describe before, some parameters aren’t mandatory for a way to have. To do a proper selection of the parameters available in each way some tests had to be made to conclude which were the ones that a certain road had. Then, after selecting the correct ones for each road the database could finally be successfully updated for all roads. In the Sequence Diagram presented on Figure 4.2 one can better understand the flow of this gathering of parameters using the module Find All Roads Data.

Figure 4.1: Sequence Diagram for the Find Road Module

Figure 4.2: Sequence Diagram of the Gather of all OSM Parameters
4.2.1.3 Estimate Slope

To achieve the estimation of the road gradient were used the latitude, longitude and altitude parameters available on the table of GPS points described on the beginning of this chapter. This estimation was done following the next steps for each road:

1. Test if the road has two or more points associated to it. If it has at least two GPS points advance to step two. If the road do not have the minimum GPS points which are 2 then it is consider that its slope is 0 and the script passes to the next road;

2. Determine what are the two points that are farther apart from each other, i.e., pairing all points on the street at a time, in order to find out what is the farthest distance between a pair of points. There are two reasons for determining this:

   (a) The first reason is that there is no way of knowing the orientation of the route. So, with this determination the points that are closest to both ends of the road are found, assuming that the road’s length is greater than its width;

   (b) For the second reason it is assumed that the highest variation in the altitude that can occur in a road is between the start and the end points of the road. In this context it is also assumed that the road do not have any other variation in altitude between these two points, i.e., the same road can’t rise and then fall or vice versa.

The distance between the two points is calculated based on the Haversine formula as it was reviewed in Section 2.2.2.

3. Next, the difference of altitude between the two points found at the second step is determined;

4. The fourth is to estimate the slope based on the ratio between what was computed on the step three and what was estimated on the second step.

Calling \textit{Run} to the distance estimated on step two and \textit{Rise} to the difference of altitude computed in step three, the \textit{Slope} is calculated using the next set of equations by the following order:

\[
\begin{align*}
\text{DiffLatitude} &= \text{latitude}_1 \times \frac{\pi}{180} - \text{latitude}_2 \times \frac{\pi}{180} \quad \text{(4.3)} \\
\text{DiffLongitude} &= \text{longitude}_1 \times \frac{\pi}{180} - \text{longitude}_2 \times \frac{\pi}{180} \quad \text{(4.4)}
\end{align*}
\]

\[
A = \sin\left(\frac{\text{DiffLatitude}}{2}\right) \times \sin\left(\frac{\text{DiffLatitude}}{2}\right) \\
+ \cos\left(\text{latitude}_1 \times \frac{\pi}{180}\right) \times \cos\left(\text{latitude}_2 \times \frac{\pi}{180}\right) \\
\times \sin\left(\frac{\text{DiffLongitude}}{2}\right) \times \sin\left(\frac{\text{DiffLongitude}}{2}\right) \quad \text{(4.5)}
\]
Implementation

\[ C = 2 \times \arctan(\frac{\sqrt{A}}{\sqrt{1 - A}}) \]  
\[ R = 6,371 \]  
\[ \text{Run} = R \times C \]  
\[ \text{Rise} = |\text{altitude1} - \text{altitude2}| \]  
\[ \text{Slope} = \arctan\left(\frac{\text{Rise}}{\text{Run}}\right) \]

(4.6)  
(4.7)  
(4.8)  
(4.9)  
(4.10)

The Sequence Diagram on Figure 4.3 gives a perspective on how the estimation of the slope works.

![Sequence Diagram of the Estimation of the Slope](image)

Figure 4.3: Sequence Diagram of the Estimation of the Slope

4.2.1.4 Compute Partial Metrics

Normalize all the stored parameters, is an important step in order to use them as a metric because the parameters have different metrics, i.e., percentages or boolean and only normalizing them, they can be compared with one another. The routing service uses the partial metrics alongside with the weights given by the user to compute the metric of the road. The partial metrics computation uses the equations reviewed on Section 3.5. The Sequence Diagram of the Figure 4.4 shows how the partial metrics are calculated through the stored parameters.

4.2.2 Optimal Path Module

The optimal path module is the routing service for bicycles, which is the main module of the project. This service relies in a Java library called Traveling Salesman to get the optimal route between two points based on a metric developed to measure the utility of traveling a road based on the partial metrics that were reviewed on Section 3.5 as well as on weights given by the user. These
Implementation

![Sequence Diagram of the Calculation of the Partial Metrics](image)

Figure 4.4: Sequence Diagram of the Calculation of the Partial Metrics

are sent by an Android Application on a mobile device and treated by a module called Android Handler 4.2.2.1. Each time the handler receives a new request it calculates a new path.

In the following sections are described the steps to get the optimal route between two given points using the weights which the user gives to the important parameters.

4.2.2.1 Android Handler

The Android Handler is basically a thread which runs on the background of the routing service and waits to be contacted by the mobile application. This thread also has three other tasks. It treats the information, i.e., the received and sent data are always JSON, so, this thread on one hand, parses the JSON object it receives from the mobile application and on the other hand, creates the JSON object with the data from the optimal route which the service finds through the information received from the mobile application. The handler is also responsible for launching the method which finds the optimal path for each request.

The Sequence diagram of the Figure 4.5 shows how this thread works and its relation with other important modules.

4.2.2.2 The Bicycle Metric

The most important thing to distinguish a path created by the routing service on this project and all the others is the metric calculated to each road to serve as the utility to travel that road. This metric improves the path to be traveled by bicycles because it is calculated based on parameters that distinguishes a road by its safeness or efficiency from the perspective of a cyclist.

The user has also a word to say on this matter. Therefore, when asking for the optimal path between two points, the user can assign weights to each parameter, thus giving their opinion about what are the most important parameters to take into account when calculating the metric to find the optimal path. The five weights added all together should be 100 and should not pass this
Implementation

Figure 4.5: Sequence Diagram of the Android Handler

**value.** Each weight belongs to the domain of all real numbers between 0 and 100. The equation to compute the Bicycle Metric of a road is on the Section 3.4.

The Sequence Diagram of the Figure 4.6 shows how the metric is used and calculates the utility of the roads given to the Traveling Salesman Library.

Figure 4.6: Sequence Diagram of the Bicycle Metric
4.2.2.3 Route Finder: Traveling Salesman

This section describes the steps to obtain the optimal route using the Traveling Salesman Library. The distance is used on the equation as reviewed on Section 4.1.4. The traveled time is not considered:

1. The script gives Traveling Salesman (TS) library the bounds to the area of the project;
2. Then, asks TS for the best route giving two pairs of coordinates: the start and the end nodes;
3. Next, the TS asks OpenStreetMap API for the map of the area of the project;
4. Upon gathering all the data the TS starts calculating which is the best route. At this, it starts to question the given metric for the utility value of each road he visits;
5. Lastly, when it finds the best route between the two given points, it returns this router to the script.

![Sequence Diagram of the Route Finder](image)

4.2.3 Android Module: The Mobile Application

The mobile application is the way which the user has to communicate with the routing service. The application has three stages. First, the user has access to an activity where he has to insert
the weight to give to each of the five parameters. Then, when the user finishes attributing all the
weights, he passes to another activity in which he has to decide what is his destination, i.e., the
final point of the trip and the initial point in case he does not want to use the GPS to give it. Finally
and after the information is sent to the server and receives the reply, the user can visualize a map
with the route between the start point and the destination. This application sends information in
JSON format as follows:

```json
{
  "startnode": {
    "latitude": 41.1560605,
    "longitude": -8.6179574
  },
  "endnode": {
    "latitude": 41.14716339,
    "longitude": -8.60896683
  },
  "weights": {
    "slope": 20,
    "maxspeed": 20,
    "width": 20,
    "oneway": 20,
    "cycleway": 20
  }
}
```

Listing 4.4: Android Application JSON Sent Example

The application receives information in JSON as follows where \( N \) is the number of the last step:

```json
{
  "startnode": {
    "latitude": 0,
    "longitude": 0
  },
  "routingsteps": {
    "1": {
      "startnode": {
        "latitude": 0,
        "longitude": 0
      },
      "endnode": {
        "latitude": 1,
        "longitude": 1
      }
    },
    "2": {
      "startnode": {
```
The model on Figure 4.8 shows the sequence of steps to the user to see the route between two points with his weights. The Insert Weights Activity can save the chosen weights after the first usage but it is always shown to the user even if the weights are already saved.

**Note:** The Android application module was not implemented on this project.
Figure 4.8: Sequence Diagram of the Mobile Application
Chapter 5

Results

This chapter shows the results of some tests made to the routing service of this dissertation. All the tests were made with no road restrictions and with equal weights for all the parameters.

The reason to make these tests was to demonstrate the functionality of the routing service by comparing the optimal path based on the Bicycle Metric of this project and the path created by the shortest route metric. This last metric only considers the distance between the two points and follows the shorter path. So, in order to compare the paths of the two metrics, the route created by the shortest route metric is then, measured by the Bicycle Metric. The weights used on the tests were equal to all parameters, i.e., 20% each.

5.1 Source Data

In this section it will be reviewed all the important numbers of this dissertation (Notice$^1$):

Total Number of Roads: 1312;

Total Number of GPS points: 471456;

Total Number of GPS points with an Assigned Road: 223378;

Average GPS Traces per Road: 454;

Total Number of Roads with Slope > 0: 108;

Total Number of Roads with Maxspeed < 50: 10;

Total Number of Roads with Cycleway: 7;

Total Number of Roads with Width > 1: 24;

Total Number of Roads with Default Values: 1109;

$^1$All the values were taken from queries made directly to the database
5.2 First Test: From Rua de Egas Moniz to Rua Dom Frei Vicente da Soledade e Castro (Faculdade de Engenharia da UP)

In this first test the results numbers were:

**Bicycle Metric**
- **Total Utility:** 1538;
- **Total Distance:** 3507 meters;
- **Total Number of Roads:** 19;
- **Average Utility per Road:** 80.9%.

**Shortest Route Metric**
- **Total Utility:** 1278;
- **Total Distance:** 3404 meters;
- **Total Number of Roads:** 16;
- **Average Utility per Road:** 79.9%.

Looking at the Figures 5.1 and 5.2 one can review that there are two major spots where the route differs on the two metrics. On the zone marked with blue the Bicycle Metric chose to go through the road Rua Ribeiro de Sousa instead of traveling through the road Rua da Aliança. This happened because the slope of the road Rua da São Dinis is 2% higher than the slope of the road Rua Ribeiro de Sousa. On the zone marked with red the Bicycle Metric chose to go through a different way than the Shortest Route Metric because of two factors:

1. First, the path that the Shortest Route Metric followed has two directions and the path that the Bicycle Metric followed has only one direction;

2. Second, the path that the Shortest Route Metric followed do not has a cycleway and the road that the Bicycle Metric followed has a cycleway.

So, looking at the numbers one can see that the Bicycle Metric is 100 meters off the shortest route (it is not that significant in a route of more than 3000 meters) but has a safer average percentage and a higher total utility which means that passes through more roads with the objective to make a safer trip. Also, it makes a more efficient trip since it passes through at least a road with a better slope value than the Shortest Route Metric.

Next on Figures 5.1 and 5.2 are the graphic paths of both this routes, respectively.
Results

Figure 5.1: Route of Bicycle Metric for the First Test

Figure 5.2: Route of Shortest Route Metric for the First Test

5.3 Second Test: From Rua de Santos Pousada to Rua da Boavista

In this second test the results numbers were:

Bicycle Metric

Total Utility: 1099.65;
Total Distance: 1744 meters;
Total Number of Roads: 14;
Average Utility per Road: 78.55%.
Results

**Shortest Route Metric**

- **Total Utility:** 939.65;
- **Total Distance:** 1641 meters;
- **Total Number of Roads:** 12;
- **Average Utility per Road:** 78.30%.

In this test the Bicycle Metric continues with a better average utility per road. The difference between the two routes is that the speed limit on the road "Rua do Bonjardim" - marked with the red circle - is 20 kilometers per hour which is a safety improvement on the route. The rest of the roads on the route are equal in utility and the total distance of the bicycle metric is only 103 meters higher than the shortest metric.

The Figures 5.3 and 5.4 show the route between the two points on the two metrics, respectively.

![Figure 5.3: Route of Bicycle Metric for the Second Test](image)

![Figure 5.4: Route of Shortest Route Metric for the Second Test](image)
Results

5.4 Third Test: From Praça do Marquês de Pombal to Rua de Delfim Maia (Universidade Fernando Pessoa)

The results numbers of the third test were:

**Bicycle Metric**

- **Total Utility**: 959.91;
- **Total Distance**: 1617 meters;
- **Total Number of Roads**: 14;
- **Average Utility per Road**: 68.56%.

**Shortest Route Metric**

- **Total Utility**: 959.91%;
- **Total Distance**: 1617 meters;
- **Total Number of Roads**: 14;
- **Average Utility per Road**: 68.56%.

In the third test the results were the same on the two metrics. This might had happen because of the heuristic used by the Turn Restricted AStar algorithm being based on the distance in meters. So, if there is a straight forward path, the two metrics might just use it straight ahead.

The next two Figures 5.5 and 5.6 are the route for the Bicycle Metric and Shortest Route Metric respectively.

Figure 5.5: Route of Bicycle Metric for the Third Test

Figure 5.6: Route of Shortest Route Metric for the Third Test
Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this dissertation it was developed a prototype of a routing service that fulfills a lack in the current ecosystem, by providing bicycle riders with a way to find better routes.

By successfully differentiating roads by parameters which are meant to improve the bicycle rider experience, the calculated routes are different from the ones generated by a shortest distance routing service and are potentially more interesting in a bicycle riding point of view.

This service meets the initial goal of letting users define different weights for the parameters that are taken into account when finding the routes.

A support system to feed the service with the different road parameters was successfully created by gathering information from two sources: the collaborative project OpenStreetMap and the SenseMyCity database. These systems were not connected before.

The contributions under the development of this project are:

- Map generic GPS points to OpenStreetMap’s roads;
- Estimate the slope of OpenStreetMap’s roads through generic GPS points from the crowd sourcing project SenseMyCity;
- Gather data from multiple sources on a system that merges the crowd sourcing project SenseMyCity and the collaborative map project OpenStreetMap;
- Develop a metric to estimate an optimal path for bicycles with a specific library;
- This project was successful in letting the users indicate the weights given to the different parameters used when calculating the routes. It intends to inspire existing and future services to adopt this kind of personalization, in order to better meet the users’ needs;
- The implemented service calculates an optimal route and returns it in JSON format allowing others to further develop applications in the benefit of bicycle riding.
Conclusions and Future Work

6.2 Future Work

One of the initial non-mandatory goals of the project was the development of an Android application to make use of the service. This objective was not fulfilled. However, if finished, this application would become a great asset for the future of this project.

To further improve the calculated routes, the service could make use of additional parameters like the road conditions. This could enrich the quality of the resultant routes. Another way to achieve better results would be refine the quality of the parameters on each road.

Following the completion of the Android application and the refinement of the service results, it would be important to test the routes with real cyclists and ascertain the quality of those routes in terms of efficiency, safety and the riders’ own needs.
References


