Abstract

The smartphones and the social networks become part of everyday life. Modern smartphones having a set of embedded sensors inside, are able to provide useful information such as location, slope, etc, which are enabling the emergence of personal and group sensing applications. However, most of existing sensing platforms for cyclists only gather data in a personal level. Quiet often do cyclist perform their activities in group. These groups activities present an opportunity to explore resources sharing among the group elements. This dissertation aims to develop a platform that provides several benefits to the cyclist users on a personal, social and public level. Using a mobile phone application, cyclists will be able to gather fitness and environmental data about their rides. When they cycle in group, the application will share resources with the rest of the group. It will also provide position estimation providing the localization of every cyclist relatively to others in the team. In case absolute location is available in the group, they are used as anchors to translate the relative location to absolute. In addition, if the one of the cyclists lose track of the rest of the team, the mobile application provides information about the location of his team mates in real time. Furthermore the user can use a web-based application that uses a web browser as a client to analyse his previous rides, or plan in advance his rides. We evaluated the system based on the device sensors/algorithm accuracy, power consume and feedback from cyclists.

Keywords: Social Sensing, PhoneSensing, Mobile application, P2P, Android

1. Introduction

The cyclist community has been demonstrating an increasing interest in discovering and sharing new and better routes. They take factors like security and enjoyability in consideration to choose an appropriate route. No only that, cyclists also display an increasing interest in gathering and collecting metrics about their rides, in order to monitor their own performance.

Until recently, cyclists had to not only acquire expensive equipement to collect their metrics, they also had to acquire several devices, since each was specialized in collecting a specific metric. As an example, they would have to acquire an odometer to measure the traveled distance, GPS to keep track of the traveleed route, and an inclinometer to measure the slope during the ride. Furthermore, these devices did not communicate among themselves, so it was necessary for the user to develop a pipeline to clean and integrate the data, in order to get a clear overview of the performance. However, smartphones have been slowly replacing these specialized devices, due to the fact that nowadays smartphones already have a set of embedded sensors and interesting computing resources, but also due to their proliferation.

Another problem faced by cyclists, is that it was difficult to collect and share metrics and routes, allowing the creation of communities that encourage sharing this knowledge. But due to the rising popularity of SNs, sharing information with other users has become incresingly easier. This means that cyclists now have access to platforms and communities that revolve around sharing and exploring new routes, organizing cycling events, and sharing metrics, therefore fostering the competition among members of the community.

Several already existing platforms already address some of the problems introduced. Consider the following as an example:

- **Bikely**: a route mapping and sharing website for cyclists\(^1\);
- **MapMyRide**: allows monitoring and sharing the cyclist performance, using social networks\(^2\);
- **Biketastic**: also allows monitoring and sharing the performance, but also monitors the

\(^1\)http://www.bikely.com/
\(^2\)www.mapmyride.com/
environment surrounding the cyclist, such as slope and the noise pollution of the route travelled[13].

But these platforms still do not take into account that cycling can be seen as a social activity: cyclists often perform their activities in groups. As such, these group activities present an opportunity to gather the collective data, using the sensors available in each smartphone that group has.

There is already some research that explores opportunistic connection between devices in order to: share resources, exchange information, and even execute tasks remotely. It is possible to use this research, and examine the possibilities of using opportunistic connections when cycling in groups. As an example, one could improve the capabilities of a smartphone, or even improve the accuracy of the metrics collected, given that the cycling group has several smartphones available.

This document describes a proposal, design, implementation and evaluation of the Social Sensing platform. This platform provides several benefits to the cyclist community, that can be categorized as: Personal, Social, and Public.

At the personal level, single cyclists (those who are not cycling in group) can use the Social Sensing platform to monitor their individual performance, such as fitness metrics, or get their real time location. The platform also allows to permanently store their metrics, providing means for the cyclist to analyse their metrics and routes over the time.

At the social level, meaning when cyclists ride in teams or groups, the Social Sensing platform provides all the services described above. Furthermore, it allows cyclists to see where each team member is in almost real time using a map, either using a mobile phone or even when they are not participating in the activity but want to track the team using the web. For each cyclist, information about the current location and route conditions are displayed, such as slope and noise. Using the Social Sensing platform, cyclists from the same team are also able to share device resources such as GPS or internet. This means that even though one of the team members might not be able to access the internet, the other members from the group can provide such functionality.

At the public level, each cyclist can share the data collected by the Social Sensing platform with the community. This way, others can search and discover routes that fit their criteria. This platform, also allows cyclists to share their fitness metrics in a social network such as Facebook.

The remainder of this extended abstract is organized into three main sections. Section 2 provides an overview of the implemented architecture, for the back-end, the web application and the mobile application. Section 3 details the implementation of the Social Sensing platform. The Section 4 evaluates the system based on the device sensors/algorithm accuracy, power consume and feedback from cyclists. Last section, section 4, summarizes and analyzes the obtained results drawing some final conclusions.

2. Architecture

The main concept of Social Sensing project was to design a system that would be both functional and easy to use. These guidelines are put into use to all situations of interaction with the system and can be classify in three perspectives: (1) Single user - mainly interested in individual data; (2) Social user - mainly interested in group data; (3) Web user - mainly interested in discovering new routes. All layers and modules will be introduced in a top-down way. By doing so, it will help to understand the next chapter, where we will detail the project’s implementation.

2.1. Back-end

Given the use cases described in the previous section, there is a need to keep the user data in a centralized way. This way is possible for the user to access his data from different places and devices.

Implementing a back-end application that abstract the access to the database allows to develop a service oriented architecture were virtually any client can request information from the database. Clients will make requests using the protocol Hyper Text Transfer Protocol3 (HTTP). We propose the back-end to use the web service architecture. This has three main benefits:

- It is possible to create an abstraction layer that simplifies the client application architecture, since these do not have to deal directly with the database.
- The client application is independent from the back-end: since they only interact through web services, the client application can be written in a different programming language from the back-end application.
- Only the client application can start an interaction by making a request to the back-end application. This makes the back-end application simpler since we only have to react to these requests.

The API provided to the clients applications will allow them to submit application requests/queries to authenticate the user, get team information, register new teams, upload fitness metrics to the back-end, among others. The back-end has a module

---

3http://www.w3.org/Protocols/
which is responsible for processing the database connection information and every time one of the clients modules want to connect to the database, will be through this module.

2.2. Web Application
The Social Sensing project provides a web-based application that uses a web browser as a client, improving the portability of the project. The web application has graphical presentation of a cyclist data and route health. The figure 1 illustrates the site map of the web application. As is possible to observe this application is divided in two different sections.

The public area include a presentation page which helps the guest users to understand the main characteristic of the mobile application, such as start screen, team edition, ride information displayed and track team functionality. It is also in this section the user is able to register or login to the Social Sensing project. To register the user simply needs an email and a password. We do not ask more information, since would be too much time consuming and susceptible to human errors that causes the user to to fill out the wrong details, reset and refill again, or worse leave the application without trying it. We follow a "plug and play" approach, if the user successfully registers, is automatically accepted without administrator confirmation. The private section, or login section, shows the full functionalities of the web application. The user can navigate through his past workouts and perform a critical analyze to the results or edit his profile with more complete information.

The user can also plan in advance the ride trip using our crowd-sensed environmental data, which provide rich information about the health quality of a given route, in terms of noise pollution, road quality and route steepness, and in a next phase with user comments. If the user is the type of cyclists that enjoys to cycle with friends, he can initiate a team with one click, add users to the team or delete. If the team is a professional team and possess a coach, he can track the team location in near real time, without leaving home. This functionality is also available for any type of team, in case the user leaves home late, for example, and desires to know where the team already is.

2.3. Mobile Application
One of the project goals for the system is for the smartphones explore resources from other devices using a distributed solution. Thus the application will be introduced to an environment of sparse connectivity and uncontrolled mobility. Given this characteristics, the Social Sensing mobile application needs a decoupled architecture and the protocol to be used must be fast and light. To achieve this, three different resources sharing systems were analysed. The required system needed to have a decentralized nature, so the devices were not dependent on any infrastructure support and the protocol used needed to be fast and light due to the mobility of the cyclists. In the end the PhoneSensing project/framework was chosen, since is the one that better fits the application requirements. To overcome some limitations of this framework, we added new modules. This new modules allows the framework to gather accurate audio and inclination data and map the environment ride of the user. This data can be later upload to the back-end to be summarized and visualized using either the mobile application or the web client application. Figure shows 2 a global over view of the architecture's stack structure. The characterization of the modules will start the local storage module and will end by describing the visualization modules.

Figure 1: Site Map.

Figure 2: General view of the Social Sensing mobile application.

Local Storage This component store the collected data after is has been processed. As the raw
data is treated, the data records are discarded. This is imperative to address the incorruptibility of the data and the privacy of the cyclist. Also none of the raw information is uploaded to the back-end.

**Application Core** This module acts as the brain of the application. It is responsible for orchestrating the user input and configurations. For instance, the user desires to constitute a team using Bluetooth, the GUI sends a message to the core module, which will then send a message to PhoneSensing to discover other devices running an instance of the same application. When PhoneSensing finishes the Bluetooth discovery, sends a message back to the Application Core, which will take the appropriate decision, in this case will generate a pop up with the name of the devices discovered. This block is purely representative since implementation wise most of the code is divided up in independent classes so scalability is not an issue.

**Internet Manager** This module is in charge of handling the connections to the back-end. It does so in an asynchronous fashion. It uploads data from the local storage, and other primitives, such as user current location, fitness metrics, and environment data (e.g., steepness and sound pressure level). As soon as the data is uploaded, the connection is torn down. It may also download data if required (e.g., team formation, team location, etc). It is using this module the user is able to register or login.

**Social client** The social module allows the user to share his information with friends. This module is responsible for allowing the user to register/login and share data using a Social Network API.

**Location provider** Knowing the location of the user, allows the application to provide superior information to the user. This location can be either absolute or relative. The absolute location refers to the positioning coordinates of a receiver at an “unidentified” point with respect to the earth’s reference frame by involving the consultation of one, or more, dependable beacons or servers. The most usual of these systems if the Global Position System (GPS) [9][6]. The relative location describes those systems that allow the location of a node to be obtained, relative to other surrounding nodes.

Thus the two main functions of this module are:
1. to obtain the absolute location of the user;
2. to obtain the user relative location. Furthermore, if some nodes may have absolute geographical references, they can be used as anchors, so that relative coordinates may be converted to absolute ones.

**Map Client** This module will allow the user to explore his location embed into a map with rich information, such as tracked points plotted or custom markers. Although there are several map clients, such as Bing\(^4\) or OpenStreetMap\(^5\), Google Maps\(^6\) is by far the most used map client by the Android developer community, also is the one that offers better API information. Also since it is widely used, is easier to find solution for possible issues. Given these facts, the project will use the Google Maps API V2.

**Quick Action** This module is a personalized Android action bar\(^7\), which provides quick actions and guiding modes to the user like team formation, navigation, quick requests and log out.

**Ride Information** The fitness metrics and other services such as location, needs to be shown in an intuitive way. Using the information gathered by the application, this module will display useful information to the user. The usefulness of the information is decided by the user, since he can select what is shown.

**Team Information** This module will display rich information about the user current team, such as the elements names. Also allows editing / deleting the team in the device.

**Profile/Settings** The mobile application allows the users to enable and disable the sensing modalities available on the phone, such as audio, accelerometer, Bluetooth, GPS etc. With these functionalities the user is able to decide what portion of their presence is shared with his team or the cycling community.

Thus, the user is able to customize his cycling experience by choosing what is more important for him, and the application is able to remind each configuration for their respective profile. This is important, since the user can settle once the different profiles for his rides. For instance, if the user is going to cycle in the city he may want a different settings configuration, than when going to the country side, the same can be said about when cycling with friends or alone. And most important, the user only needs to configure the settings once, and every\(^8\) time he returns to the application, they will be there.

3. Implementation

The Social Sensing platform is divided in three main parts, the mobile and web application and the backend. The first section will talk about the resources used and the second will describe the implementation of the back-end. The third section talks about the web application, while the fourth is going to describe improvements done to the PhoneSensing Framework. The last section defines the implement-

---

4http://bingmapsandroidsdk.codeplex.com/
5http://wiki.openstreetmap.org/wiki/Android
6http://developer.android.com/google/google-api/services/maps.html
8Unless the user deletes the information.
tation of the mobile application functionalities.

3.1. Back-end
One of the biggest problems in software development is the integration between the applications that were built over different platforms and/or different languages, where these applications should communicate between each other in order to exchange data. To tackle this problem, we proposed a technology that goes by the name of Web Service. This Web Service was developed using HyperText Preprocessor (PHP). The PHP is a simple and powerful language which provides high compatibility with leading operating systems and web servers, thus allowing it to be deployed through several different platforms[11][1]. The data exchange format chosen was JSON. JSON is more lightweight than XML, allowing a faster parsing and reduce bandwidth necessary, which is useful for mobile applications.

The API provided to the clients allows them to submit application requests like user authentication, upload important information, between others.

3.2. Web Application
The Social Sensing project provides a web application for the user to analyze and visualize fitness and environmental data. This front-end is a mix of different languages which together are able to provide important services to the cyclist.

The design or structure of the website are done using HyperText Markup Language 5 (HTML) and Cascading Style Sheets 3(CSS). If HTML specifies how the browsers should display a web page, the CSS is responsible for specifying the guideline of how a browser will present the HTML code[7]. This is important since the HTML code supported by the CSS is read in the same way by the different browsers, protecting the portability and versatility of the system.

To make the front-end more dynamic we used jQuery. JQuery is a javascript library that greatly simplifies JavaScript programming. The web application provide rich map information for the user analyze and visualize his fitness metrics.

The Social Sensing project is also a Crowdsensing platform[12][14]. Beyond the personal sensing, where the cyclist focuses on personal monitoring, the project also has public sensing. Public sensing is the type of data that is shared with all users for a common good. Since the cyclists already use the smartphone to monitor their performance, we leverage these devices by building large-scale sensing application efficiently (cost and time).

After the cyclist upload the environmental data stamped with time and location for context purposes, the front-end map system makes visualization, interpretation and sharing of the route information. To make data interpretation easy we added an layer to the map. We opted for using a Heatmap layer. When this layer is on, a colored overlay will be shown on top of the map, with areas with more intensity appearing red and less intensity appearing green.

Additionally the web application also allows the user to create, edit and track his team. The user can also see team metrics like max speed, average speed, distance, among others.

3.3. Extending PhoneSensing Framework
The Social Sensing mobile application will need to interact with several sensors in order to provide data about the Cyclist Performance and environment mapping. To overcome some limitations of this framework, we added new modules: audio and inclination.

It is secure to assume that all Android smartphones provide a microphone to the user. The mobile application uses the microphone to analyze the route noise pollution and security, since a high noise pollution in the route may indicate presence of cars nearby. The audio module runs asyncrony in background.

The mobile application uses a audio formats of 16bit since it offers a bigger frame window than the 8bit. The sample rate choose was the 44100Hz and the audio channel was Mono, since these are assured to work in any Android device. To avoid buffer overflows, the buffer size should not be small. The Android SDK provides a method that calculates the minimum buffer size for these configurations. Then the data is read form the microphone in chunks (temporary buffer) of length inferior to the total recording buffer size. We then use the root mean square to compute the quadratic mean value of the chunk, since it contains the peak value for each sample. The advantage of using root mean squared over finding the maximum value is that root mean squared takes into account all data points. This makes the calculation robust against single or short-lived time periods of high amplitude and allows only meaningful high amplitude signals to have an effect. We then proceed to calculate the sound pressure level (SPL), which is a logarithmic measure of the effective sound pressure of a sound relative to a reference value and it is measured in decibels (dB)[3], is obtained using equation 1.

\[
spl = 20 \times \log_{10} \frac{RMS}{REF VAL} + CAL VAL
\] (1)

In this equation the RMS stands for root mean square, REF VAL reference the sound pressure in the air. This value is commonly 20µPa, which is usually considered the threshold of human hearing.
The $\text{CAL}_V\text{AL}$ is the sensor calibration value inferred by the user. In case no calibration was done, this value is 0.

The characteristics of the cyclists fitness include the steepness of the road/trails that the cyclist covers during his ride. The slope of a route can provide useful information for the planning of a ride. The slope of a given route is obtained using the equation 2.

$$\text{slope}(%)=100 \times \tan \theta \tag{2}$$

The angle "theta" represents a slope from a ratio of a change of y-coordinates, also known as the rise, over a change in x-coordinates, known as the run[2]. A high percentage indicates a steeper degree of "tilt". The inclination module also runs asynchronously in background. To compute the inclination, we analyse three different ways of measuring the inclination. In the end we opted for using the following method:

The Geomagnetic Field Sensor provides information about the earth’s magnetic field and gives raw field strength data (in $\mu T$) for each of the three coordinate axes. Thus the application uses the accelerometer and geomagnetic field sensor in combination with the getRotationMatrix() method, provided by the Android API, to obtain the rotation matrix and the inclination matrix. Next the social sensing application use these matrices with the getOrientation() and getInclination() methods (also provided by the Android API), to obtain azimuth and geomagnetic inclination data.

Finally we apply equation 2 to the geomagnetic inclination, and we obtain the pitch, or steepness in percentage. To analyse the accuracy and the behavior of the sensors we put them stationary on a flat surface, such as a table and calibrate them. Then we proceed to several experiments moving the device between zero and eighty degrees. The values obtained by this sensor fusion were very accurate.

3.4. Mobile Application

In the Architecture the PhoneSensing framework was introduced as separated from the application implementation. When we improved the framework we joined up together with the mobile application. That is to say, the mobile application has the framework built in. This choice is due to structural aspects related to the PhoneSensing development, this framework was then incorporated in the application.

The application Core extends Android’s native Service class. This class operates in the background and is used to perform long-running tasks. Most of the functionalities will be achieved using this class. Also most of the requests from the GUI will go to this module. That will process it or forward to one of the PhoneSensing services. This service use handlers to send objects of type Message$^9$ to the activity or the other services offered by the PhoneSensing. All tasks in this module are executed asynchronously. For instance, to filter the devices that belong to a team already registered, or to find which device in the team has a requested resource.

The mobile application use "Preferences" to save user preferences, like how the screen should look or the settings menu is configured. It is fastest way to save or access persistent data. The application also use a database to store the cyclists data, since they are ideal to save repeating or structured data. The application will use the SQLite database. The SQLite is open source and has supports relational database features like SQL syntax, transactions and prepared statements and is embedded into every Android devices.

The Social Sensing mobile application offer real-time interaction between the back-end and the biker in support of real-time location. Also the user can collect sensed data and then upload it into his personal repository. All the interactions with the back-end are done in background using asynchronous HTTP requests, handle responses in anonymous callbacks, and as mentioned before, HTTP requests happen outside the UI thread. This is particularly important because connecting with the back-end may take some time, and can block the GUI or crash the application.

Since usually the internet connection is provided by a Telecom operator, and the user pays per Megabyte, the application tries to upload data in burst. For instance, if the user has several rides to upload, instead of uploading one at a time, it uploads all at the same time.

The mobile application allow the cyclist to map his route and track his activity. This allows the user to know where he is going, where he has been and record his workouts. To do so, the application uses a Network Location Provider. Deciding which provider is the most suitable source of location data is a necessary compromise, since requesting updates from all of them would drain off battery. The cycling activity often occurs in places where the Wi-Fi access point is not in range, and although the cellular network provider consumes less energy than the GPS, it has much less accuracy. Since we decided to choose just one provider, the GPS also has the benefit of providing extra valuable accurate services, such as speed and distance data. For the aforementioned reasons, there is no doubt that GPS is the most suitable location provider for a cycling application.

\footnotesize{\textit{\textsuperscript{9}}http://developer.android.com/reference/android/os/Message.html}
As with any project targeted at mobile devices, energy consumption is a major concern [10]. If the location updates are too frequent the battery will be drained faster, on the other hand if they are too infrequent will increase the delay of the user location and accuracy. The social sensing accuracy tries to reduce the updates without significantly degrading the location and metrics accuracy by requesting GPS updates with a frequency of 2 minutes or a distance of 20 meters.

If the user has selected to share his location and environment conditions with friends, this data is sent to the Upload Manager. If the user does not have internet, the application search the local database to find a device with internet. If any is available, tries to connect, to send the data. The remote team device, if has internet forwards the data to the Upload Manager, otherwise says it is not possible to forward the data. But the mobile application provides more added value services to the cyclist, in special when he cycles in group.

The cycling team should not be dependent of the GPS embedded in the smartphones to know the team location. In case none of the team devices have a GPS, the application will generate a two-dimensional layout of nearby team elements.

There are several techniques to obtain the relative position, which will be described briefly. The localization of nodes frequently relies on determining the distance from these nodes to dissimilar referential points with well known locations[5][8]. The most often cited techniques for measuring the distance between nodes are the following: Time of Arrival (TOA), Time Difference of Arrival (TDOA), Angle of Arrival (AoA) and Received Signal Strength Indicator (RSSI).

The TDOA and AoA methods requires expensive implementation and infrastructure. The Time of Arrival provides high accuracy but requires a high complex system of time synchronization between the nodes. Thus the Received Signal Strength Indicator that depends on the theory that received signal power is inversely proportional to the square of the distance is a much more practical technique to implement in the Social Sensing project.

The Android API only allows to read the Bluetooth RSSI from the discovery process. This process commonly include an inquiry scan of about 12 seconds, followed by a page scan of each found device to retrieve its Bluetooth name. Although we rely on RSSI, methods such as propagation time become possible, the mobile application can use them alternatively. The RSSI unit returned by the Android device is dBm, which is an abbreviation for the power ratio in decibels (dB) of the measured power referenced to one milliwatt (mW).

In theory, we can obtain the distance based on the radio propagation model and power level using the equation 3:

\[ \text{RSSI(dBm)} = P_{TX} + G_{TX} + G_{RX} + 20\log\left(\frac{c}{4\pi f}\right) - 10\log(d) \]

\[ = P_{TX} + G - 40.2 - 10n\log(d) \]  

(3)

where \( P_{TX} \) is the transmitted power; the \( G_{TX} \), \( G_{RX} \) are the antenna gains and \( G = G_{TX} + G_{RX} \) is the total antenna gain; 'c' stands for the speed of light in the free space (approximately \( 3.0 \times 10^8 \text{m/s} \)); 'f' is the central frequency (2.44 GHz for Bluetooth); n is the attenuation factor (2 in free space) and 'd' is the distance between transmitter device and the receiver device in meters. Thus d is given by equation 4.

\[ d = 10^{\frac{P_{TX} - 40.2 - \text{RSSI}_{\text{dB}} + \text{G}}{10n}} \]

(4)

In theory the signal strength falls off logarithmically with the distance, but this model can only be used as a theoretical reference, since the measurements are corrupted by reflection, obstacles, noise, antenna orientation and device context (e.g., inside a backpack, on the pocket) leading to a more complex relationship between received signal strength and the distance.

To convert each RSSI reading to a distance estimate with an error bound, we use experimental methods that we developed by running several propagation experiments in outdoor. We positioned the devices at known distances and measured several samples of the RSSI. Since the android device used does not mention the power transmitter or the antenna gain, we use the values predicted by the propagation model with \( P_{TX} = 2.9\text{dBm} \) and \( G = -4.82\text{dBi} \) [8].

The results obtained were different from the theoretical values, as expected. In average the RSSI was approximately different -26.9739dBm from the expected value. After introducing this error in the equation we obtained a relatively close results to the theoretical values. Due to the environment conditions where the RSSI is measured and the phone context it is not straightforward to draw a definitive conclusion about the relation between Bluetooth RSSI and distance.

The algorithm used to measure the distance estimation is as follows: First a device starts a Bluetooth discovery process (lets assume is the team leader), measuring the RSSI and converting it to distance. This process is done asynchronously. Then a filtering phase takes place, since the goal is to generate a two-dimensional layout of nearby team elements. Afterwards the device will contact the team members in range of the Bluetooth to start a discovery and filter process, in order to also measure the RSSI of nearby devices. The numbers of devices contacted are saved, so the application knows...
the number of replies expected to receive in order to proceed to the next step. Also the remote devices do not forward this requests. This way is possible to prevent the devices that already received the request to receive the same request again, flooding the network. Also this allows shortening the time to execute the algorithm.

Since the devices may go out of range from the team leader and the application is time sensitive, we execute a thread in background that will run for limited period of time (e.g. 2 minutes, since a bigger delay of time would make this process slow and when tested with 2 devices, both of them replied in less than 1 minute.) in order to assure the continuity of the process. When the location provider finish running the algorithm responsible for the distance estimation, it proceed to the spatial placement algorithm to produce a 2D map of where peer devices are.

Most of the examined algorithms use geometrical calculations based on successive triangulations or multilaterations to assess a position relatively to some reference points. The algorithms may proliferate possible measurement mistakes even though they use minimization methods so typical in these algorithms. Given these circumstances, we choose the MultiDimensional Scaling (MDS)[4][5][15] as the most qualified for our systems requirements. It is possible to make a matrix "Y" of Cartesian coordinates of points in Euclidean space, knowing the Euclidean distances "D" amongst those points. Thus the MDS is a set of data analysis techniques that converts a distance matrix into a set of coordinates, such that the(Euclidean) distance derived from these coordinates approximate as close as possible the original distances. The following steps explain the functionality of the MDS[5].

A distance matrix "D" receives the known measured distances between "N" devices in a network. The double-centring method is used, in which the matrix is centered around the average value of its entries, in this manner acquiring the B Matrix is done with equation 5.

\[ B = -\frac{1}{2} \times (I - \frac{U}{N}) \times D^2 (I - \frac{U}{N}) \]  

where \( I \) is an identity matrix with \( N \times N \) dimensions and \( U \) a matrix with the same dimensions as \( I \) but filled with '1's. Later using 6 we compute the eigenvalues (matrix \( A \)) and the eigenvectors (matrix \( V \)) of the B matrix.

\[ B = V A V^T \]  

The \( m \) biggest positive eigenvalues and its correspondent eigenvectors are retained. The \( m \) is the desired number of spatial dimensions (2D, \( m=2 \)).

The spatial placement which refers to the devices coordinates relatively to each other is retrieved from the calculation of the X matrix provided by the equation 7.

\[ X = V A^\frac{1}{2} \]  

In this project we used the MDSJ library\(^{11}\), which is a free Java library for MDS. However if localization is opportunistically available, as we can see in the algorithm, the application uses it to translate the relative location to latitude and longitude coordinates of all the devices without absolute location. We need at least two nodes to report their absolute location. The application fixes the spatial location of these devices using their geographic coordinates in place of their relative coordinates. However, it may happen that the devices does not comply with the characteristics of the Euclidean space, and in that case we might have an "X" matrix filled with Not a Number (NaN). If such happen, we will try to minimize the situation by showing the location of the team members which uploaded their location to the cloud. Once again, in case the devices take too much time to reply, the application runs a thread in background to proceed with the algorithm. The devices contacted for the GPS request, will read their GPS data and reply back to the team leader. When the team leader receives all the requested GPS data, will lock the table.

The translation of Cartesian coordinates into Latitude and Longitude is as follows: Since we know the distance relation of the devices, we center the map in one of the devices with GPS. If only one GPS is available, it is fixed at the center, and the position of the other devices are translated from Cartesian to Polar coordinates, using the equations 8 and 9.

\[ r = \sqrt{x^2 + y^2} \]  

\[ \theta = \tan^{-1} \left( \frac{y}{x} \right) \]  

With 8 and 9 and knowing the distance relation in the 2D diagram, we can convert all positions to latitude and longitude. But with only one GPS, the real position of the other devices can be in any of the 4 quadrants. In case we have one more GPS, it is fixed at the center, and the position of the other devices are translated from Cartesian to Polar coordinates, using the equations 8 and 9.\(^{11}\)

\(^{11}\)http://www.inf.uni-konstanz.de/algo/software/mdsj/
Bluetooth to nearby team devices in order to share latitude and longitude of the team elements. This information is also sent to the screen responsible for sharing fitness and location data.

4. Results
The application was developed for cyclists and a set of tests were designed and carried out with that objective in mind.

4.1. Cyclist Experience Mapping
Cycling can be a social activity. Quite often do cyclists perform their activities in group. If everyone of a team uses the Social Sensing application, we could expect some correlation and consistency of the data collected by the sensors of the same team. This can be used for calibration purposes, irregularity detection and noise reduction. We conducted a pilot evaluation of system, which is carried out in Cascais. It takes cyclists 30 minutes to finish the ground truth route at a normal speed. Both the noise and the steepness evaluation were performed there. The ground truth route has a variety of urban cycling terrain, including busy roads and peaceful back roads with which we can detect the noise. As to the steepness, this route exposes the cyclists to flat roads, gradual downhill and steep uphill. In this ride, cyclist were asked to remain about 5 meters from each other. The team is constituted of three elements, Silva, Mansur and Yao. Silva and Mansur started the ride together and calibrated the devices before they started the ride. Yao joined the team in the middle of the ride and then she calibrated the device. The figures 3 and 4 illustrate Silva and Mansur ride. The data gathered by Yao was not so different from the rest of the team, although the noise events were gathered with less intensity. This is due to the fact Yao calibrate her device in a different context from Mansur and Silva. Thus we are only going to analyse Silva and Mansur data.

Figure 3: Noise data gathered along the route - Joao.

Analyzing the figures 3 and 4, it’s obvious that the data they collected share some correlations. For instance, areas of high noise are captured by both cyclists although the intensities may be different. As one can observe, the downtown area appears redder than other places on the map, which means the volume is higher there. The reasons are apparent: there are more cars and people in the center. As the cyclists move away from the downtown, the volume decreases as expected, indicating a more peaceful area. However, the traffic is not the only reason affecting the noise level. There are other factors related like multipath reflections of sound, changes of the wind, sound made by the cyclists, inverse square fading of volume with distance, cars passing by each cyclist at different times, etc.

The steepness ride data is accurate when the cyclist is idle. When the bicycle is moving, the error increases as the speed changes or the terrain becomes tougher. Apart from that, any slight movement in the device may cause a huge distortion in the measured slope, due to the non-linear nature of the inverse tangent function used to obtain the slope. We believe that if a filter is used to remove the bicycle vibrations from the data, a better result can be achieved.

In conclusion, the Social Sensing platform provides not only noise pollution mapping to the cyclist community, but also, in a more personal perspective way, important information about safety and noise pollution. Also the system supplies information about route steepness so that the users can get some guidance for cycling routes, although additional studies need to be done in this area.

4.2. Evaluating Relative location algorithm
While the GPS is a natural solution to the design of the mobile application, not every device may be equipped with a GPS receiver. The mobile application provides a node-based relative positioning functionality that depend uniquely on the mobile device of the cyclist.

The primary goal of this algorithm is to accurately localize nearby peers. Therefore in this section we pretend to infer the location error provided by the algorithm. In order to achieve this goal, we started by placing three devices (e.g., let’s call them: Alfa,Beta and Gama) running the mobile application apart from each other. The separation among the devices was selected tanking into account a range of distances that is probably used by...
bicyclist when they cycle in small groups (e.g., three elements). This distances are also distances covered by the Bluetooth Technology present on the devices. The distances were measured using a tape. We expect the values measured by the tape to have a deviation from the exact ones, but be trustful enough for the tests.

Next we run the node-based relative positioning functionality, obtaining this way the position and distance among the devices. After successfully running more than thirty experiments, we proceed to do the same experiment, but this time using the GPS receiver embedded in the devices. The website csgnetwork\textsuperscript{12} was used to convert the latitude and longitude of the GPS coordinates into meters. The table 4.2, contains the error of the relative location algorithm and the error of the GPS. The errors are obtained by subtracting the distances between the devices measured by the relative location and the GPS location with the "real" location.

<table>
<thead>
<tr>
<th>Measure type (m)</th>
<th>A to B</th>
<th>B to G</th>
<th>A to G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Location (m)</td>
<td>4.20</td>
<td>3.50</td>
<td>4.15</td>
</tr>
<tr>
<td>Error of R. Loca-</td>
<td>1.04</td>
<td>-0.21</td>
<td>0.93</td>
</tr>
<tr>
<td>tion (m)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error of A. Loca-</td>
<td>23.72</td>
<td>4.20</td>
<td>25.74</td>
</tr>
<tr>
<td>tion (m)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Analysing the table 4.2, one can attest that the relative location algorithm is good to estimate distance, at least for small distances between the peers, as the ones used by cyclist when riding in small groups and not in competition mode. Additionally the algorithm supply more trustful measures than the GPS for this specific case.

In summary, the results presented in this section shows that the algorithm can be used to place the devices in a two-dimensional plane with low distance estimation error between them. Moreover in case some nodes have GPS, their location can be fixed and then translate the relative location to absolute of those who do not possess GPS receiver or wish to save battery.

5. Conclusions

The Social Sensing mobile application main focus is to provide accurate fitness measurements, route healthiness and advanced services like tracking team members, resources sharing and position estimation. To achieve this goals the application use and extended version of the PhoneSensing Framework. The position estimation was achieved using an algorithm that provides localization of every node relatively to others by placing them on a two dimensional plane without the need of a GPS receiver. In case some devices have a GPS receiver they can be used as anchors so that relative coordinates may be converted to absolute ones.

The back-end provides way of centralizing important information. This ways is possible the user is able to access his data from different places and devices. The web application has graphical information of a cyclist data and route health, in terms of safeness, noise pollution and inclination. This way the user is able to analyse his fitness data and plan in advance his rides. To evaluate the system a several of test were performed. We started by evaluating the accuracy of the route environment mapping and the correlation of the data when the bikers cycle in flocks. The tests show the microphone can provide reasonable accuracy, while the method used to obtain the inclination needs further improvements The usability of the system was tested with a group of cyclists that filled an online survey. In all due fairness we observe the users liked the system and were eager to use it, but also showed that it need some improvements, specially on the UI.

Although the Bluetooth is good for prototype development is not ideal for this application environment, due mainly to the high latency and short range. In this project the bluetooth was mainly chosen because of the devices limitation to establish ad-hoc WiFi connections. An interesting improvement would be to user other wireless technologies like the WiFi direct. Security is an important issue that was not properly addressed in this project. Therefore further optimizations at this level is desired.

Summing up, while the current experiments have concentrated on sensing for the cyclists, the platform can also be targeted to any user that enjoys to perform outdoor group activities or even to map traffic and noise pollution in a city scale.

References


\textsuperscript{12}http://www.csgnetwork.com/gpsdistcalc.html


