University “Politehnica” of Bucharest

Automatic Control and Computers Faculty,
Computer Science and Engineering Department

MASTER THESIS

Intertial sensors in crowd sensing applications

Scientific Advisers:
Cristian Chilipirea
Ciprian Dobre

Author:
Vlad Ioan Nistorica

Bucharest, 2016

Abstract

Having access to up-to-date and accurate information about the environment can be very important in decision making for both environmental scientist, city planners, park administrators and also regular users. One of the easiest ways of capturing this type of data is through accelerometers.

Accelerometers have many application in the context of gathering environmental data, one of them being capturing surface movements like earthquakes or vibrations caused by diferrent factors, while other can include gesture recognition, user behavior in specific spaces or analysing the terrain that the user is going through.

Even more powerfull applications can be built from gathering inertial data from a large number of devices with accelerometers, which can be achived through crowd sensing. Other alternatives for acquiring such data by hand or by deploying sensor networks would be expensive and time consuming. Also, people without proper scientific training want to take part in the data acquisition process and assist with the needs of their respective communities.

Crowdsensing is a technology-driven area where ICT platforms are being developed which permit anyone to participate in processes that help expand our understanding and improve our surroundings. Crowdsensing refers to the process in which crowds (large number of people) measure specific features and share the resulting data or send it to a central location in which it can be used by the people that need it. Since the increase in popularity of smartphones, which are now ubiquitous, crowdsensing is more popular than ever.

A lot of people can now participate to crowdsensing but we still don't know how many are needed for a crowdsensing campaign to be successful. In order to answer this question, we built a simulator that mimics the characteristics of a crowdsensing campaign. We showcase three different scenarios in which we estimate the required number of participants and offer a discussion on the plausibility of having that many participants by taking into account factors such as accessibility to the area of interest (the area from which the measurements are needed).

Also, using crowd sensing and gesture recognition, a use case is proposed and analysed in more detail. We propose a way to gather information about location of stairs and elevators, that could help disabeled persons calculate paths to diferent destinations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Crowd Sensing

There is a need for up to date, detailed data on different metrics that characterize the environments we live in. A few examples are: air quality, noise pollution, traffic density and even WiFi access. This data is significant to environmental scientists, city managers and even citizens which can make better, informed, decisions. Data is the first step in enabling smart cities.

Standard solutions for gathering environmental data on a large geographical scale are expensive and difficult to implement. Even community driven projects such as the Air Quality Egg[1] come at a high price and have moderate usage (in the order of thousands for the entire planet). A recent alternative that is growing in popularity is crowdsensing (Section 2.1). Crowdsensing [17] proposes the distribution of the data gathering task to a large number of people. Any person carrying a device capable of measuring a certain characteristic of the environment and transmitting this data to a central database can participate in a crowdsensing campaign.

Smartphones represent the most convenient tool for a crowd sensing campaign. First of all, smartphones are now ubiquitous, meaning the cost of deploying a crowdsensing campaign is lowered because most people already own the hardware needed for such a campaign. According to [15], in United States the penetration of smartphones has reach a percentage of 72% and even in less developed countries the percentage of smartphone penetration is rising fast. They have a large variety of sensors, such as accelerometers, microphones and video cameras and can easily be extended with more through the use of Bluetooth. They have more and more powerful processors, which permit complex data processing, as well as clear methods of developing and deploying new applications that can run on a very large number of devices. Lastly all smartphones have WiFi and LTE modules which permit the transmission of the sensed data.

For crowdsensing, to be effective, it still has to deal with a number of pitfalls. Crowdsensing campaigns require a significant number of participants in order to permit the extraction of valid conclusions (a.k.a., a critical mass of people is needed in order to validate the measurements). These campaigns are either volunteer-based or offer special incentives, such as monetary compensation [16]. In either case, it is not clear how many participants are needed in order to have a successful crowdsensing campaign, where enough data is gathered to permit the organizers to draw valid conclusions. To our knowledge, there are no studies that offer a methodology for determining a threshold for an optimal number of participants of a crowdsensing campaign. The optimal number is the minimal number of participants for which enough data is gathered. The number of participants needs to be minimal in order to minimize resource usage and incentives costs.

[1] http://airqualityegg.com/

In the first part of this study, we propose a methodology for computing the optimal number of participants for a crowdsensing campaign depending on the campaign characteristics. Having this number is vital for the organizers of any campaign that uses crowdsensing techniques. First of all, it offers an idea of the possible success (as in, how representative is the measurement data to describe a particular environment) of the campaign. Secondly, a relation between the number of participants and the effectiveness of the measurement sensing campaign, can in fact help the organizers plan the use of incentives to either motivate participation, or keep the costs under control.

Determining an optimal number of participants in real life is difficult. Starting a crowd sensing application is currently based on the willingness of people to participate. It is unlikely and extremely time consuming to repeat the same crowdsensing campaign with varying number of participants.

No trivial solutions for determining the optimal number of participants are available. An obvious one would be to divide the size of the area of interest, by the size of the sensing area of a single sensor. The solution is appropriate for static sensors. However, it does not take into account overlaps between multiple sensors or movements of the sensors, as they are carry around by people, during a time period. When we take movement into account a single sensor can cover a far larger area then it is expected from its specification.

Our approach for determining the optimal number of participants is to simulate crowdsensing campaigns that take into account crowd movement (Section 3.1). We choose an interest area, Herastrau park in Bucharest, and simulate crowdsensing campaigns for air quality, WiFi access as well as people density as crowd sensing applications (Section 4.1).

We take a further look into the problem of the number of participants by offering a discussion on the number of possible candidates for a crowd sensing campaign. Possible candidates represent people for which taking part in the campaign is reasonable. To offer a clear example, it is wrong to assume that a crowd sensing campaign could have more participants than the number of citizens of the respective city the campaign takes place in (Section 4.1.1).

1.2 Accelerometers

Accelerometers are one of the most used components, because of the multitude of applications that can be build from inertial information, low costs and easy integration in different types of hardware, like mobile phones, robots, drones and many more. An accelerometer is usualy integrated into an inertial system, which may also include gyroscops, magnetic sensors or gravitational sensors. Additional sensors can be used as a referance, in order to minimize the accelerometer drift. The role of an inertial sensor is to identfy phisical movement, which can be liniar displacement or rotation, and transform it to a readable set of analogical or digital data.

An inertial sensor commonly ranges from low cost, MEMS inertial sensors, measuring only a few square mm, that offer less precision, up to more precise and expensive sensors, like ring laser gyroscopes which can measure 50 cm in diameter. Most common types of sensors are MEMS (Microelectromechanical) inertaial sensors, which can be found in most smartphones, drones, head mount displayes, IoT applications and others. For example a Nexus 5 smartphone uses InvenSense MPU- 6515 six-axis, which is a MEMS MotionTracking device and includes capacitive gyros and accelerometers. The same IMU is used in [9] for detecting unsafe driving.

In an inertial system, the information from different sensors are fused toghether to have the result that is expected. In general, for fusing these types of information a Kalman filter is used, which is able to combine data from several different environments that have outputs with noise. The role of this filter is to use the combined data to reduce the weak point of sensors, so that combining the best parts of different types of sensors can result in better precision.

1.3 Accelerometer in crowd sensing applications

To give a more specific context to corwd sensing, we propose the use of inertial sensors to recognize when a user is using stairs or an elevator. Using data agregated from a large number of smartphones, a map of stairs and elevators can be built, with minimum effort and costs. The beneficiaries for this map could be people with disabilities, that could route their path to a destination according to this information.

For this use case, firstly, an Android application will be built for creating testing and training datasets. The training and testing data will be pre-segmented. If present, magnetic and gravitational sensor will be used to reduce drift and normalize the capture data.

In the second step, a gesture recognition pipeline will be proposed, which will contain algorithms for pre-processing, so that the noise for the captured data is reduced, for gesture recognition, which will identify 4 actions: stairs up, stairs down, elevator up and elevator down and post-processing algorithms.

The last step, will be to test how fast the pipeline learns the actions and compare different algoritms results.

Chapter 2

Related Work

2.1 Crowd Sensing

Crowdsensing is still a young scientific area but it has already gathered a lot of interest and support, as we show below. There are a great number of studies that show the diversity of scenarios in which crowdsensing can be applied.

As stated in [8] and [7], crowdsensing can provide micro and macroscopic analysis of cities, communities and persons. It can be applied in social networking, health, energy, monitoring human behavior and many others.

In terms of environmental analysis, the authors of [5] show that by using crowdsensing in urban areas one can collect traffic data or generate noise maps. A similar solution, regarding noise pollution in urban areas is presented in [19]. Air quality analysis represents another interesting use for crowdsensing. Authors in [23] claim that crowdsensing is the optimal solution for capturing data for large areas, although it is only feasible for CO_2 emissions, because of the complexity and cost of other sensors. [10] adds road surface monitoring and street parking availability. The former article also presents two categories of sensing: participatory, in which users have to get involved and be active in the measurement and data acquisition process, and opportunistic, where the user has a more limited role in the sensing. In fact, the second sensing approach tends to also be more visible in the literature and existing platforms for crowdsensing, because the user is relieved of much of the burdens associated with taking decisions where and how to collect measurement data. Our work focuses on the second approach, opportunistic sensing.

Without the use of crowdsensing the cost of implementing a similarly capable sensor system would be significantly higher. In [11] the authors show that to cover an area of approximatively $1km^2$, 100 sensors and 1096 relays need to be used. This method cannot scale for larger area of studies, so crowd sensing is a better alternative.

As described in [16], crowd sensing generally implies a requester, a campaign starter, which requires users willing to capture sensor data that is used directly or at a later time for various experiments. The authors acknowledge that in terms of user engagement, it is sometimes difficult to convince users to participate in a crowdsensing analysis. Incentives are proposed as a solution to this problem. They take the form of payments or gamification. Incentives are also used in [20]. The authors use micro-payments and gamification to attract enough users so that they can cover the area of a university campus. The scope was to measure WiFi signal strength across the entire campus. However, the authors make no analysis on what is the optimal number of participants to a crowd sensing campaign or how to manage incentives to reach that value.

The use of payments for crowd sensing participation without a previous analysis of possible number of participants could lead to high costs and little control over the results.

Access to a space of interest. It is not enough to determine the optimal number of participants for a crowd sensing campaign. One also needs to take into account the plausibility of achieving this number. Probably one of the factors that is most relevant is the accessibility of individuals to the area of interest. In the literature there are different approaches to determine the number of people that have access to an area, such as a park, some of which are presented below.

The authors of [13] present the travel cost approach. They show two methods of measuring the accessibility to a park using geographical information. In the first, the service area of the park is considered a circle, with the radius equal to the maximum desired distance between the center of the park and locations of visitors. The disadvantage of this method is that park visitors are assumed to be able to reach the park by using a straight line, which is not usually the case. The second method is based on the shortest path algorithm using the boundary of the park area and the streets and paths that the visitors can walk on. Their work takes into account both neighborhood parks (2 - 4 ha) and mini neighborhood and community parks (0.4 - 2 ha). The total distance accepted is less than 0.8 km of walking.

The main problem with this approach is that visitors do not always use the closest park and tend to travel more for a park that has a bigger area or more facilities compared to the ones offered by a closer park. An alternative would be consider the distance cost to all parks in the studied area, or a predefined set. Using this method, a set of 3 parks and limited number of people it has been demonstrated [6] that visitors prefer parks that offer more attractiveness even if they are not in close proximity.

The authors of [21] use the container approach to determine the equity and accessibility to public playground. An extension of this method is the kernel algorithm presented in [12]. The Kernel density estimation method can give an accessibility rating for every point in the studied area. The idea behind this method is to rate every acre that is studied as belonging to a park or not and using kernel density algorithm, the area is converted to a statistical surface. The problem with this method is that depending on the kernel bandwidth taken into consideration, there could be areas that have no accessibility, which is incorrect.

The attractiveness of a park can be modeled by using as little as only the size of a park. The assumption is that larger parks have more facilities and respectively are more attractive. In [22] the authors propose the population-weighted distance, which should be a more precise method of determining the accessibility to a park. This is the method we chose for our analysis and discussion.

2.2 Accelerometers

Gesture recognition.

Chapter 3

Experimental analysis

3.1 Simulating a crowd sensing campaign

3.1.1 Use cases

In order to simulate a crowd sensing campaign, we require a set of characteristics for the campaign. A campaign can be characterized by the area of interest, the zone from which the data needs to be gathered, the range at which a sensor can measure data and the time the campaign is expected to take place.

A WiFi sensor can detect hotspots at ranges of about 100m, according to the WiFi 802.11 standards[1]. In contrast, an air quality sensor can take measurements of the air that directly touches it. For the later one we assume the measurements are relevant for a distance of 3m.

The area of interest can have highly irregular shapes. Consider a park, not only is the outer perimeter of the park often irregular, but a park may contain features that are not accessible to pedestrians such as small lakes. In order to have an accurate representation we extract the zone of interest from Open Street Maps[2]. The simulator takes the given area and the number of participants and generates their movement. We use random walk in order to simulate the movement of the individuals over a period of time.

In order to verify the correctness of our simulator we built a small visualization tool. A NodeJS server offers the simulation data to a JavaScript tool that makes use of the Google Maps[3] library in order to display the locations of individuals inside the area of interest. An example of the visualization can be observed in Figures 3.2. The area of park Herastrau, as described in Open Street Maps, is represented with red and the small white discs represent pedestrians.

Because the shape of the interest space can be bounded by an irregular polygon, in order to calculate its area, we use grid sampling. We take an outer rectangle that encapsulates the entire area and split it in a grid of 1000 by 1000 cells. We then count the total number of cells that are inside the polygon. Finally, we multiply the total number of cells inside the area of interest with the area of a cell.

To determine if a cell is inside the polygon describing the interest space we use a ray casting algorithm [18]. In ray casting a set of parallel lines are drawn over the rectangle. We start in the upper left corner and use vertical lines. The intersections between these lines and the

[1]http://standards.ieee.org/about/get/802/802.11.html
[2]https://www.openstreetmap.org
[3]https://www.google.ro/maps

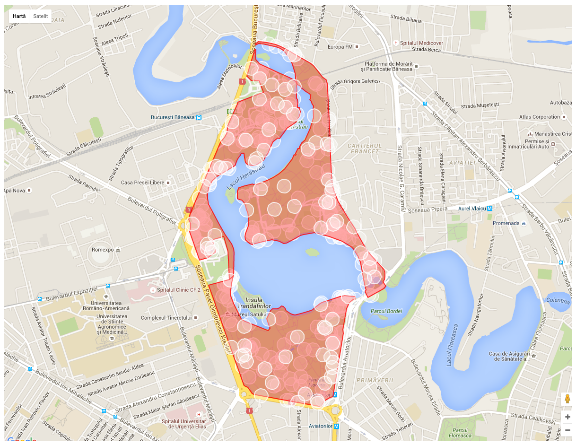

Figure 3.1: Simulation, 100 persons, sensing area radius of 50m

polygon are calculated. For each cell, if the number of intersections is even, it means the cell is inside the polygon and thus inside the area of interest, otherwise it means the cell is outside.

In order to simulate pedestrians, we randomly choose their starting location as points inside the area of the park. We generate random locations inside the outer rectangle and keep only those that are inside the polygon. In order to determine if the location is inside the space of interest we used the same ray casting method.

To have a more realistic simulation we added movement. Crowdsensing campaigns use pedestrians or even vehicles. In our experiments we focused on spaces such as parks through which only pedestrians can move. We set the same speed of 5km/h for all simulated individuals. The speed was chosen according to [2]. The movement was simulated using the Random Walk [14] algorithm. In random walk, pedestrians move in a randomly chosen direction until they reach the edge of the interest space. In our case we consider collisions with any element of the polygons. Because we focus on parks, there can be multiple polygons. As we stated earlier, parks can have features that are not accessible by pedestrians, such as small lakes. Each of these are bounded by a polygon. When we detect a collision for a pedestrian another direction is chosen randomly and the pedestrian continues its movement in the new direction with the same speed.

In order to simulate the data gathering process we use discs of a set radius centered on each of the pedestrians. In reality the shape of the detection area for a sensor is highly irregular. Take WiFi, where the shape of the area in which frames can be correctly received varies with irregularities of the antenna, features of the environment and even weather. Because the shape varies from sensor to sensor and there is no model that describes the irregularities found in nature a disc can be used as an acceptable approximation.

In order to calculate the area of the surface of the interest space covered by at least one of the sensors carried by the pedestrians we use the same grid sampling method. In this case, a cell represents an area that we consider to be covered by a sensor if there exist at least one sensor whose disc covers at least half of the cell. In order to determine if a cell is inside a disc we calculated the Euclidean distance [3] from the center of the cell to the position of any pedestrian. If, for any pedestrian, this distance is smaller than the sensing radius we consider the cell to be covered by the sensors. This is described in equation 3.1, where x_c and y_c represent the location of the center of the cell and x_p and y_p represent the location of a pedestrian.

$$\forall i; \sqrt{(x_c - x_{p_i})^2 + (y_c - y_{p_i})^2} < sensingRadius \tag{3.1}$$

The final goal of the simulation is, given a sensing radius, an area of interest and a time period, to determine the percentage of the area of interest that is covered by sensors. After we count the number of cells covered by the sensors and the number of cells inside the area of interest we use equation 3.2 in order to determine the percentage of the area covered by sensors. For a cell to be considered covered it needs to have been inside the disc of a sensor at any point in time during the simulation.

$$coveredPercentage = \frac{count_{coveredCells}}{count_{spaceOfInterestCells}} * 100 \tag{3.2}$$

3.2 Accelerometers

3.2.1 Use case

In terms of use cases that can be taken into consideration, which can combine the power of crowd sensing with the low costs and popularity of accelerometers that are embeded in smartphones, there are a multitude of application that can be built. One example can be taking real time readings of the magnited of earth movement caused by earthquakes or other sources of vibration like subways. Another great application can be using accelerometers and crowd sensing to see where users are most likely to enage in phisical activities, so that this area can be further be incrised. Also, smartphones in cars could capture data associated with incidents and using crowd sensing, a map of the most dangerous areas can be made and different actions can be taken to reduce them.

The use case proposed by us, wants to take advantage of the smartphones popularity and integration of inertial sensors to recognize gestures and diferentiate between normal walking and using an elevator or going up or down a stairway. This data gatherd from a large number of smartphones, through crowd sensing will result in a map of stairs and elevators, from which persons with disabilities could use when choosing a path to a destination.

This use case would imply minimal effort in terms of capturing data. The smartphone should be able to identify by itself different context of movement, withouth user input. In terms of security data gathered can be anonimous, as it is important to have a general idea of locations of stairs and elevators, not only for one person.

3.2.2 Data acquizition

For data acquizition an Android application is built, which will help with capturing data from inertial sensors, that will be later used as training and testing datasets. In this phase user input is important, so that datasets are segmented and the segment includes only one event of the following:

- stairs up - from the moment one leg leaves the ground, to the moment that the same leg touches the next step in the stairway
- stairs down - the same of above, only reversed
- elevator up - will contain only the first part when the elevator is starting to accelerate until it reaches a constant speed
- elevator down - the same as above, only the elevator is going down
- non relevant move - to test null rejection, which means that a gesture does not fit any of the above gestures. Can be a step in walkig, running or stanging stil

The first step in the data acquisition process will be, it is very important that the phone's orientaion should not give different readings. For this two methods are considered. One is using the acceleration magnitude form the equation 3.3, where x, y, z are the acceleration on the coordinates. Although this is a good method for general cases, in the usecase, provided the most movement is happening on the vertical axis, it would mean that the result would only add irelevant noise.

$$a = \sqrt{x^2 + y^2 + z^2} \tag{3.3}$$

A better solution is proposed by the Android framework and uses the reading from the gravity and magnetic sensors so that a rotation matrix is calculated, which can be used to transform a vector, more specifics the accelerations vector, from the device coordinate system to the world's coordinate system, as you can see in 3.1.

```
SensorManager.getRotationMatrix(R, I, gravityValues,
magneticValues);
android.opengl.Matrix.invertM(inv, 0, R, 0);
android.opengl.Matrix.multiplyMV(earthAcc, 0, inv, 0,
 linear_acceleration, 0);
// liniar_acceleration matrix will be transofrmed to
// world's coordinate system
```

Listing 3.1: Transforming to world's coordinate system

The second step is to isolate the force of gravity with the low-pass filter and remove the gravity contribution with the high-pass filter.

Another step includes calibration which will be used for normalization of the data. For this reading will be taken in the following cases:

- phone set on screen, which will be associated with +1g
- phone set on back, which will be associated with -1g
- phone droped, for exeample from one hand to another, which will be associated with zero value

With the results from the calibration phase, readings will be normalized so that captured data does not vary significantly from one device to another. For this, if we use the "zero value" from above as B, we can substract the measured values like in 3.4 and obtain Y, which is centered around B. Now, we want to scale this value into a rage that will be common to different devices. For this we use 1G read, as A, and 0 value and obtain 3.5

$$Y_{centered} = X - B \tag{3.4}$$

$$Y_{scaled} = \frac{X - B}{A - B} \tag{3.5}$$

Training and testing datasets will be pre-segmented using user input on the duration of a gesture or manualy segmented from a continios dataset and all gestures will be associated with a label that represents the type of gesture captured, wheter stairs, elevator or other.

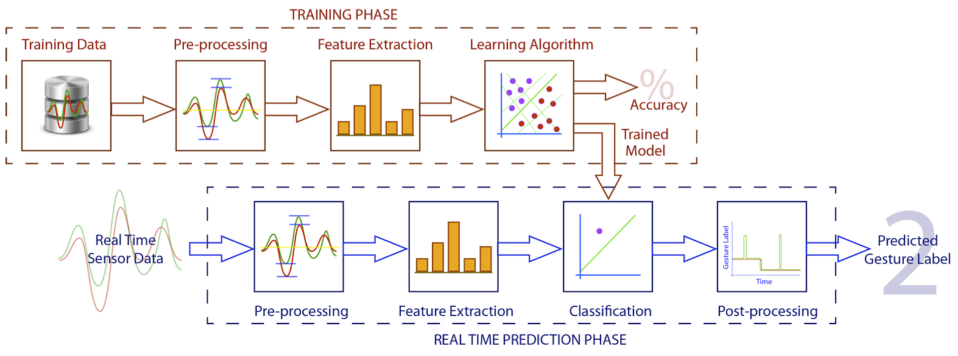

Figure 3.2: Gesture recognition pipeline

3.2.3 Gesture recognition

Problem type

Having the training and testing datasets it is important to understand the type of problem and types of gestures that the data falls into, so that gesture recognition algotithms are correctly selected.

As we want as a result a specific value, which indicates the type of gesture a person has made, the usecase falls into the classification problem. Further, the movement captured consists in a temporal gesture, which implies movement between two moments of time.

Pipeline

Recognising different gestures like walking or going up a stair imply building a gesture recognition pipeline which can support the following components:

- Preprocessing - it is used for noise reduction
- Feature extraction - prepares data depending on the necesities of the next components in the pipeline, like input data dimesions reduction
- Learning Algorithm - which are trained with data sets so that they can recognize gestures. They are tested using datasets that are different then the ones used for training and can also be easily interated in an Android application using tht Android NDK.
- Post processing - optional component used to have a better classification precision.

As there are a lot of frameworks that integrate this components, there is no need to implement it from scrach, so for this purpose we choose to use Gesture Recognition toolkit, described in [4], which suits the best the studied use case.

Algorithms

In terms of preprocessing, Gesture Recognition Toolkit offers different filters like low and high pass filter, derivative, dead zone. For our current application low and high-pass filter was used so that the gravitational force was isolated from the accelerometer data. Also, to reduce the noise a double moving average filter was used.

For the learning algorithm, two algorithms are proposed: Hidden Markov Model and Dynamic Time Warping.

Hidden Markov Model algorithm is applicable for temporal pattern recognition, like gesture recognition, which consists in a sequence classifier and has the ability to assign a label or class to each sequence unit. In the training process, the algorithm maps a sequence of observations to a sequence of labels, using weighted finite automata. The algorithm can also use KMeans Quantizer, which is a feature extraction algorithm, to convert the input signal from N dimensions to one dimension.

Dynamic time warping can be used for temporal gesture recognition with an input that has more than one dimension. In the training phase, the algorithm builds a template from the input dataset, taking into consideration the Euclidian distance between gestures of the same label. This distance is also used in the testing phase when the template of a label is compared to an unlabeled gesture.

For post-processing, some of the algorithms present in Gesture Recognition Toolkit are class label filter, for gesture recognition in a continuous data stream, class label change filter or class label timeout filter which stop gestures in the training set or that were already classified to reenter the prediction phase.

3.2.4 Testing

The main metric analysed is the precision of classification and how it depends on the number of training data sets. For this K-Fold Cross Validation is used.

K-Fold Cross Validation separates the input dataset in K smaller sets and uses one of them to created the testing data set and the other subsets form the training data sets. These tests are repeated K times, with the subsets created randomly and also the testing dataset being chosen randomly.

To test the speed of learning, the size of the training dataset is raised incrementally for every tested class. This means that in total there will be K*N tests, where K is the value used for the number of subsets in the K-Fold Cross-validation algorithm and N is the number of incremental tests.

Another test could analyse the execution time for the recognition pipeline, or different components in the pipeline. Also, on mobile devices, metrics like memory consumption or processor load, could be useful.

Although Gesture Recognition Toolkit framework can be integrated into an Android application, and implicitly also the test, for this study the focus is put on the applicability of the use case. The tests will be run on a Linux machine, using real training and testing datasets.

Chapter 4

Experimental results

4.1 Simulating a crowd sensing campaign

We used our simulator in order to determine the results of multiple crowdsensing campaign located inside a city park. We used park Herastrau[1], in Bucharest, Romania as a case study. The geographical data, that describes the spatial mapping of the park was obtained from Open Street Maps. To be more precise the area consists of three polygons, that describe the shape of the park and the lake that spans across the park area.

To offer an idea on the difficulty of determining the number of required pedestrians to cover the entire area of the park, take the air quality campaign. It has a sensing radius of $3m$. This means that a pedestrian is able to cover approximatively $28m^2$ at a moment of time. We calculated the area of Herastrau park to be $1404409m^2$. This is consistent with the area calculated by the park managers. The result includes errors from the grid sampling method and from transformation from geographic coordinate system (GPS coordinates) to the metric system. Simply dividing the two values $1,404,409/28$ shows that we would need over $50,000$ participants. As we show later in this section, because in this solution mobility is not considered, the resulting number is far higher compared to the value obtained through simulations.

We run our simulator using three different scenarios. These scenarios pertain to three distinct crowdsensing campaigns. For all scenarios we simulated an hour of movement. We varied the number of pedestrians from 1 to 3,000 (the results did not differ much after this value) and calculated the percentage of the covered area for each value. We chose one hour as this is an appropriate time period for which a participant would be willing to visit a park. The results for all simulations are presented in Figure 4.1.

Air quality - with a sensing radius of $3m$ This is one of the most difficult crowd sensing campaigns to implement. While the other two examples only require special software on the smartphones of regular pedestrians, this one also requires a specialized device. The smartphone still serves the purpose of communication and data processing. This has been done before [1] in Zagreb.

For air quality the range is the smallest, technically the sensor can only measure air that directly touches the device, however, it is safe to assume that the air in the surrounding area has the same quality, so the radius was approximated to $3m$. In order to reach a percentage of 98% covered area, we require around 3000 persons.

[1] http://www.herastrauparc.ro/

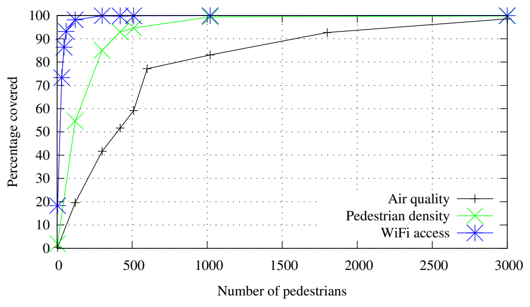

Figure 4.1: Percentage of area of interest covered for different scenarios and different number of participants

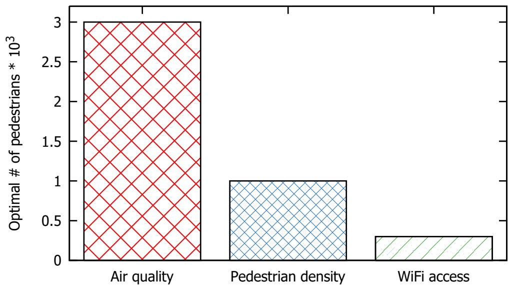

Figure 4.2: Comparison of scenarios, required number of participants in order to cover 95% of the interest space

Pedestrian density - with a sensing radius of $10m$ In order to estimate pedestrian density, we can take advantage of the fact that smartphones are ubiquitous. All smartphones have Bluetooth modules and a lot of users simply leave them active. This module regularly sends frames that can be detected. Based on these frames we can estimate the number of pedestrian around a sensor. We estimate an appropriate sensing distance to be of $10m$, according to the Bluetooth specifications. We can see that a coverage of almost 100% is reached at about 1000 participants.

WiFi access - with a sensing radius of $100m$ In order to determine WiFi access we need to identify the number of available, open hotspots in the area. Any device with a WiFi module can receive beacon frames in order to determine what hotspots are around it and what WiFi networks it can connect to. The sensing radius, set according to the WiFi specifications, is of 100m and covers an area of $31415m^2$. We can see that approximatively 300 participants are needed to cover the area of the park in one hour.

To conclude we estimate that a successful crowd sensing campaign would require a coverage of at least 95% of the interest area. This is not always the case as some campaigns may only require an estimate on the measured values. When the goals are as high as 95% of the covered area, for the three case studies we proposed we obtain values for the optimal number of individuals in the order of hundreds or thousands of people. Figure 4.2 shows these results in more detail.

The number of individuals and sensing radius are not the only factors that affect the percentage of the covered area. The duration of the campaign also needs to be considered. In order to show the effect time has on a crowd sensing campaign we choose to exemplify it on the air quality scenario with 2000 participants. The results are presented in Figure 4.3. Here, time is varied in 10 minute steps, from zero to two hours. This shows the importance of the duration

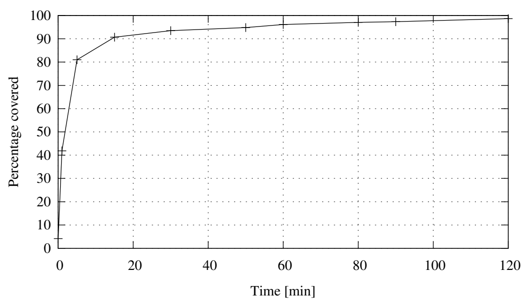

Figure 4.3: Duration of campaign, Air quality, 2000 participants

of the campaign. However, the duration of a campaign cannot be easily controlled.

4.1.1 Discution

In the previous section we showed how many people are required in order to have a successful crowdsensing campaign. Having this data, gives one a clear idea on what are the requirements, but it does not offer a clear view on what to expect. In order to set correct expectations, one would need to determine how many people can take part in a crowdsensing campaign. This also enables the organizers to set appropriate incentives.

The willingness of people to take part to a crowdsensing campaign is primarily influenced by the ease of access to the area of interest. People are unlikely to come from a different city just to participate in such an event. In order to determine the number of people that have access to the interest space we used a spatial interaction model.

We used an ArcGis[1] service in order to determine demographic information. From it, we extracted the number of people living in a specific area, in our case, the city of Bucharest. The map was split in a grid of 5 by 5 cells. The size and number of cells were chosen based on the minimum area accepted by the ArcGis service for demographics. Every cell was given an accessibility rating (A) using the model in equation 4.1.

$$A = \frac{S_{park}^{\alpha}}{d_{cell-park}^{\beta}} \tag{4.1}$$

where

- S_{park} - the total area of the park,
- $d_{cell-park}$ - Hamiltonian distance between the centroids of the cell and park,
- α - represents the effect that the dimension of the park has on the parks attractiveness,
- β - describes the will of a person to travel to a destination.

We chose $\alpha = 0.85$ and $\beta = 1.91$ according to a study [22] that modeled the spatial accessibility to parks.

In the grid with the maximum rating, it is considered that all the people living in that area have access to the park. Further, the percentage of the total population that lives in a zone and has access to the park is calculated relatively to the maximum rating. Figure 4.4 represents the results for the accessibility rating of the cells that compose the map of Bucharest. A more detailed view is presented in A.

[1] https://www.arcgis.com/

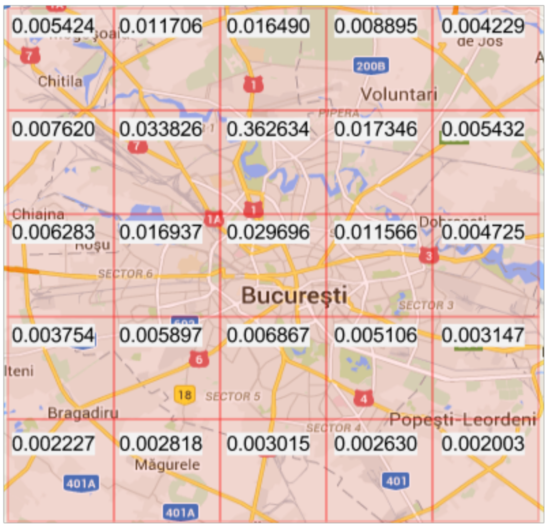

Figure 4.4: Accessibility ratings to park Herastrau

Considering the area of Park Herastrau, the results suggest that the estimated maximum number of possible participants to a crowd sensing study could be around 158,312 people. By taking into account the penetration of smartphones in Romania, of 27.9% as stated in [15], the number of possible participants is lowered to 44169.

This number is further decreased when we consider the willingness of people to partake in a crowdsensing campaign. However, the value can be used as a maximal boundary. If a crowdsensing campaign for park Herastrau requires a number of individuals larger than the 44,169 value, it is certain that the campaign will not successfully generate a complete data set. Even when incentives are used it is unlikely that they would be significant enough in order to reach this value.

In park Herastrau the scenarios we propose require different numbers of participants. These numbers represent between 1% and 7% (these percentages are obtained by comparing the results presented in Figure 4.2 to the value obtained in the previous paragraph) of the number of people that have access to the park and own a smartphone. We believe it is reasonable to assume that with proper incentives campaigns that require fewer participants can be successful (In our case the crowdsensing campaign for determining WiFi access). Campaigns that require a larger number of participants require stronger incentives.

4.2 Accelerometers

4.2.1 Gestures

The first part of implementing the use case in which we recognize gestures like walking, climbing stairs or using an elevator, is capturing real datasets with these movement, that will be used next in training the gesture recognition pipele and also when building test datasets.

This part was created using an Android application which enabled the user to record accelerometer data between two point in time. The device used for capturing data is a Nexsus 5, with InvenSense MPU-6515 as an Inertial system. The captured data is associated to one person, walking on different surfaces, using different stairs and elevators. The sampling rate is considered 2 microseconds.

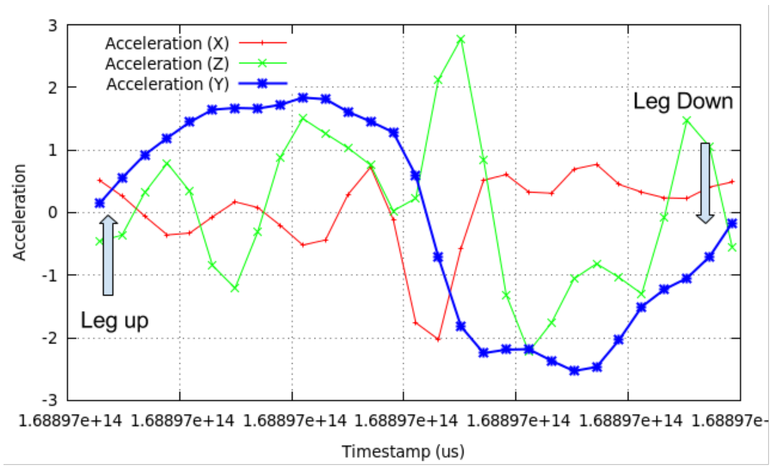

Figure 4.5: Accelerometer data for going up a stair

The important part on this phase was to understand the nature of the gestures. In figure 4.5 an example of recorded data when going up a stair is displayed. We can se that most of the movement happens vertically, so we consider only recognising gestures based on the vertical axis. Other captured data can be seen in 4.6, for going down a stair, 4.7 for one step in the process of walking.

One problem for this phase is to segment correctly the captured data, so that the training is correct. For this the Android application enables the user to delimit one step in the proccess of moving, but using only user input is not enought, as it is not very precise. This means that additional manual work is needed so that the training data is correctly segmented.

Another problem is the orientation of the phone, when the data is captured. For this the training and testing datasets include different position like in hand or in the pocket.

The results show that taking a step in the process of walking is simillar with a step when climbimg a stair. The difference sits in the initaial part of the step, when the leg getts up, as in stairs this action has a greater amplitude and also has a larger duration. Also, when the leg touches, when on stairs the acceleration is larger on the stairs.

When comparing going up or down a stair way, we can see that the movement are quite opposite and that going up taks more acceleration that going down.

In terms of elevator movement, the accelerations involved have the tendacy to jump from a positive value to a negative one fast.

4.2.2 Recognition performace

The porpose of recognising the seven action proposed in the current use case falls to the role of the gesture recognition pipeline.

The pipeline is trained using the datasets obtained in the data aquizition phase from the Android to create the training and testing datasets.

The recognition pipeline for implementing the proposed use case included two version. The first used Hidden Markow Model and the second used Dynamic Time Warping. This algorithms were choosen because of their popularity when dealing with temporal gesture recognition.

The Hidden Markov Model additionaly needs an algorithm for future extraction, which can covert from an N-dimensional input to a one dimennsional input. For this K-Means Quantizer

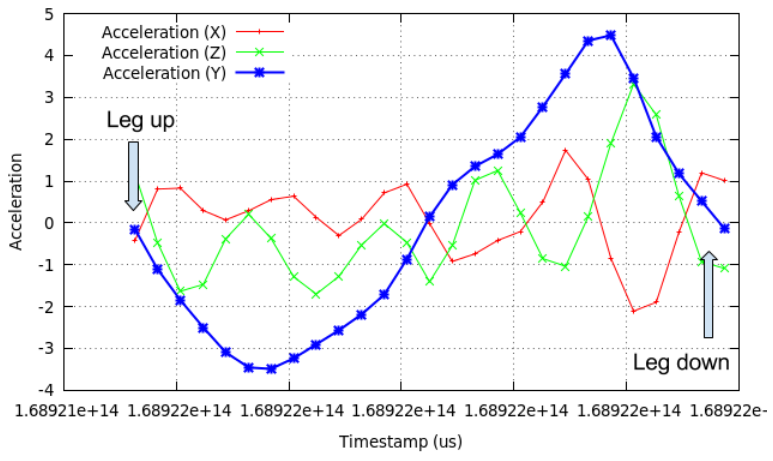

Figure 4.6: Accelerometer data for going down a stair

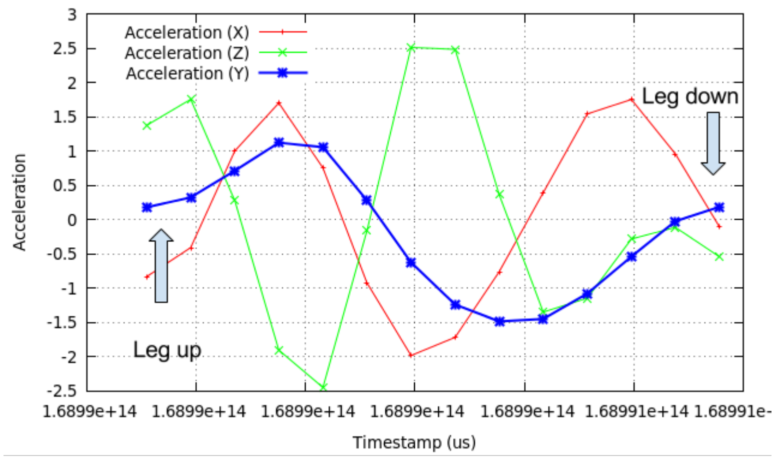

Figure 4.7: Accelerometer data for walking

was used.

Chapter 5

Conclusion and future work

We proposed a method to estimate the optimal number of participants to a crowd sensing campaign, on an area of interest, through the use of simulations. We used three scenarios for three different crowdsensing campaigns: air quality analysis, pedestrian density and WiFi access. The three crowd sensing campaigns were simulated over the same space of interest, Herastrau park, in order to show how the differences in the crowdsensing campaign, mainly the sensor type, affect the results. We extended our analysis by taking a look at how the duration of the campaign affects the results.

To show the relevance of the results we added a method to determine the number of people that have access to a crowdsensing area of interest, own a smartphone and may be willing to participate. We know that incentives can be used in order to increase the willingness of people to participate, but this is outside the scope of the article and can be considered future work.

As future work we need to offer more detailed simulations that better match specific crowdsensing campaigns. People do not normally move according to a random walk algorithm, but take specific paths, they also travel in groups and their behavior can be influenced by the campaign. The sensors can be modeled with higher precision, moving away from a disc radius. The simulation can take the environment into account as well as its effect on the sensors and the area that they cover. Finally, real life crowdsensing campaigns can be used to further improve the simulations.

As future work different segmentaion algorithms could be tested, which would have the purpose to help recognise the studied events in a continous stream of data. This algorithm could present ways of triggering when the data is captured, for example, the gesture recognition pipeline should not be used when the person is stationary. It can also include different proposals for using sliding window approach and methods to calculate the best sliding window size.

Appendix A

Accesibility rating results

The following tables present the results obtained for the accesibility ratings, where each cell is associated with a area on the map of Bucharest.

Table A.1: Accesibility rating

0.005424	0.011706	0.016490	0.008895	0.004229
0.007620	0.033826	0.362634	0.017346	0.005432
0.006283	0.016937	0.029696	0.011566	0.0047259
0.003754	0.005897	0.006867	0.005106	0.003147
0.002227	0.002818	0.003015	0.002630	0.002003

Table A.2: Procentage of population with access to parks

1.50%	3.23%	4.55%	2.45%	1.17%
2.10%	9.33%	100.00%	4.78%	1.50%
1.73%	4.67%	8.19%	3.19%	1.30%
1.04%	1.63%	1.89%	1.41%	0.87%
0.61%	0.78%	0.83%	0.73%	0.55%

Table A.3: Population that has access to the park

115	120	1756	143	30
485	7434	117927	1306	205
1021	4709	10318	3959	529
192	1239	2382	1777	972
27	22	813	787	40

Bibliography

[1] Aleksandar Antonic, Vedran Bilas, Martina Marjanovic, Maja Matijasevic, Dinko Oletic, Marko Pavelic, Ivana Podnar Zarko, Kresimir Pripuzic, and Lea Skorin-Kapov. Urban crowd sensing demonstrator: Sense the zagreb air. In *Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference on*, pages 423–424. IEEE, 2014.

[2] Nick Carey. Establishing pedestrian walking speeds. *Portland State University*, 2005.

[3] Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2009.

[4] Nicholas Edward Gillian and Joseph A. Paradiso. The gesture recognition toolkit. *Journal of Machine Learning Research 15.1: 3483-3487*, 2014.

[5] Bin Guo, Zhiwen Yu, Xingshe Zhou, and Daqing Zhang. From participatory sensing to mobile crowd sensing. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 593–598. IEEE, 2014.

[6] Pawinee Iamtrakul, Kardi Teknomo, and Kazunori Hokao. Public park valuation using travel cost method. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 5, pages 1249–1264. Citeseer, 2005.

[7] Wazir Zada Khan, Yang Xiang, Mohammed Y Aalsalem, and Quratulain Arshad. Mobile phone sensing systems: A survey. *Communications Surveys & Tutorials, IEEE*, 15(1):402–427, 2013.

[8] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[9] et al. Liu, Luyang. Toward detection of unsafe driving with wearables. *Proceedings of the 2015 workshop on Wearable Systems and Applications. ACM.*, 2015.

[10] Huadong Ma, Dong Zhao, and Peiyan Yuan. Opportunities in mobile crowd sensing. *Communications Magazine, IEEE*, 52(8):29–35, 2014.

[11] Xufei Mao, Xin Miao, Yuan He, Xiang-Yang Li, and Yunhao Liu. Citysee: Urban co 2 monitoring with sensors. In *INFOCOM, 2012 Proceedings IEEE*, pages 1611–1619. IEEE, 2012.

[12] Andrew R Maroko, Juliana A Maantay, Nancy L Sohler, Kristen L Grady, and Peter S Arno. The complexities of measuring access to parks and physical activity sites in new york city: a quantitative and qualitative approach. *International Journal of Health Geographics*, 8(1):1, 2009.

[13] Sarah Nicholls. Measuring the accessibility and equity of public parks: a case study using gis. *Managing leisure*, 6(4):201–219, 2001.

[14] Karl Pearson. The problem of the random walk. *Nature*, 72:342, 1905.

[15] Jacob Poushter. Smartphone ownership and internet usage continues to climb in emerging economies, 2016.

[16] Moo-Ryong Ra, Bin Liu, Tom F La Porta, and Ramesh Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 337–350. ACM, 2012.

[17] Oriana Riva and Cristian Borcea. The urbanet revolution: Sensor power to the people! *Pervasive Computing, IEEE*, 6(2):41–49, 2007.

[18] Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

[19] Matthias Stevens and Ellie D'Hondt. Crowdsourcing of pollution data using smartphones. In *Workshop on Ubiquitous Crowdsourcing*, 2010.

[20] Manoop Talasila, Reza Curtmola, and Cristian Borcea. Crowdsensing in the wild with aliens and micro-payments. *under submission in IEEE Pervasive Computing Magazine*, 2014.

[21] Emily Talen and Luc Anselin. Assessing spatial equity: an evaluation of measures of accessibility to public playgrounds. *Environment and Planning a*, 30(4):595–613, 1998.

[22] Xingyou Zhang, Hua Lu, and James B Holt. Modeling spatial accessibility to parks: a national study. *International Journal of Health Geographics*, 10(1):1, 2011.

[23] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.