Acta Cybernetica **23** (2017) 441–470.

A Personalized Multi-Path and Multi-User Traffic Analysis and Visualization Tool

Cristian Babau[a], Marius Marcu[a], Mircea Tihu[a], Daniel Telbis,[a] and Vladimir Cretu[a]

Abstract

Traffic optimization is a subject that has become vital for the world we live in. People these days need to get from a starting point to a destination point as fast and as safe as possible. Traffic congestion plays a key role in the frustration of people and it results in lost time, reduced productivity and wasted resources. In our study we seek to address these issues by proposing a real-time road traffic planning system based on mobile context and crowd sourcing efforts. The first step toward this goal is real-time traffic characterization using data collected from mobile sensors of drivers, pedestrians, cyclists, passengers, etc.. We started developing a data collection and analysis system composed of a mobile application in order to collect user context data and a Web application to view and analyze the data. This new system will eventually give the users an automatically optimized route to the destination and predict the users' traveling route based on live traffic conditions and historical data.

Keywords: traffic optimization, mobile context, route analysis, visualization tool

1 Introduction

After the appearance of smart phones in the mobile telephones market an increasing number of applications are trying to provide specialized services for their users. A method for modifying the application's behavior is based on the context in which the mobile phone is currently functioning and the applications that use this context are called context-aware applications. Context-aware applications use the data sensed by the device sensors, such as a sound sensor, a light sensor, a temperature sensor, a movement sensor or a position sensor and they map it using certain algorithms to a real-life context (the user is inside or outside a building, the weather is cold or hot, the environment is quiet or loud); then, based on this context, different services can be provided by the application.

[a]Polytechnic University of Timisoara, E-mail: `cristian.babau@student.upt.ro`, `mmarcu@cs.upt.ro`, `mirceatihu@gmail.com`, `daniel_telbis@yahoo.com`, `vcretu@cs.upt.ro`

DOI: 10.14232/actacyb.23.2.2017.2

Integrating the mobile context of the drivers when analyzing urban traffic and planning transportation routes is the next step in urban intelligent transportation systems. The key goal of this study is to collect traffic data from users using their mobile devices and then to aggregate the raw data obtained in order to monitor, analyze the data and also characterize route segments and generate route predictions. In this study we present a mobile Android application designed for sample data from the built-in sensors and a Web application used to visualize and analyze the data sampled from mobile devices together with the conclusions resulting from using the applications. The visualization tool is public and it can be accessed at http://mobilebackend.cs.upt.ro/sensor-data-viewer/overview/index.php.

The Android application uses a service that runs in the background and collects sensor data from the most common sensors encountered in the mobile devices, namely an accelerometer, gyroscope, magnetometer, location from the network provider and GPS, linear acceleration, proximity, sound and light levels. The collected data is then stored in a local database, it can be sent on to a remote database on a remote server. The source code of the mobile client is open and it is published on bitbucket.org/lipan19/disertatie-v2.

The Web application uses both aggregated and individual data got from the remote sensor to help users visualize and analyze their collected data. All the locations collected are then displayed on a Google Map so the user can see his everyday routes and for each route he can analyze the speed, acceleration, light level, sound level for different parts of the journey. By splitting each part into atomic segments and performing segment profiling from all the data gathered for a particular journey, an accurate representation of the segments can be constructed, so the user can monitor his own driving habits. Based on this representation the user can later estimate and characterize a whole trip from one point to another by reconstructing the journey from the segments. The source code of the Web application is open and it is published on bitbucket.org/lipan19/sensor-data-viewer.

The paper is structured as follows. In Section II there we discuss the related work that has been carried out in the field. In section IV, we introduce our solution overview to the problem. Next, in section V we present the results of our experiments. Lastly, in Section VI we summarize our key conclusions and make some suggestions for future research.

2 Related work

Traffic advisory systems were introduced fairly recently to help people plan their routes and optimize their traveling distance and time. These systems are called Intelligent Transportation Systems or ITS for short [1]. ITS applies advanced communication, information and electronics technology to help solve transportation problems like traffic congestion, safety, transport efficiency and environmental conservation. Several aspects have to be addressed in any modern ITS, as suggested in Figure 1.

Traffic information data collection is one of the most important aspects of ITS

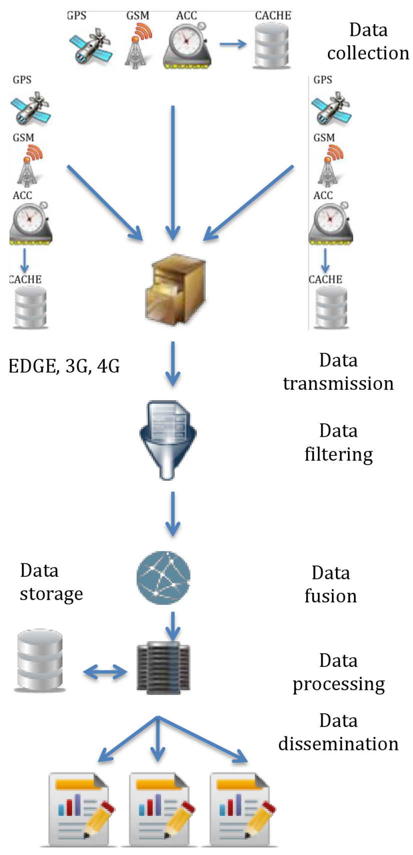


Figure 1: ITS overview

and the first that needs to be addressed when developing such systems. There are many ways that traffic data could be collected and subsequently analyzed and processed by traffic data analysts and automated systems. These methods can be grouped into two general categories depending on the source of the data: the first type involves colleting data from dedicated and reliable stations (both fixed stations and mobile agents) and the second type involves collecting data from the users participating in the traffic (e.g. crowd sourcing). The main role of data collection is gathering primary information about road traffic conditions in order to produce more complex knowledge, including traffic congestion, incidents that have occurred, the state of a section of the route they are traveling on and the weather conditions.

Most recent solutions use the mobile devices of the participants while using different navigation or context-based mobile applications. These sensors move freely with the vehicles and collect traffic data. A mobile sensor network of traffic participants must be able to accomplish three major tasks [1]:

- Location determination it must be able to determine the coordinates of the probe to same degree of accuracy;
- Map matching it must be able to pinpoint the coordinates of the probe in the road network to some degree of accuracy;
- Evaluation of the of traffic information the system must be able to process the data collected and generate traffic reports.

Google Traffic uses location and speed data collected from the users who have Android devices in order to provide information about traffic jams and the best routes in a specified area. The information is displayed on a Google Map, where the best route segments are shown in green, while the slow segments are shown in red. All the data collected is anonymous, and the sum of the data collected from more devices gives the final value of the segment, which is just characterized by its color, no other metric like speed or acceleration being provided. The service provides two operation modes, namely one in which the live-traffic conditions are displayed from data harvested in real-time from the users, and an hour-specific traffic, which displays data from the usual traffic for the given day of the week, for a particular time frame, between 06:00 and 22:00.

Waze is a platform which provides a mobile application that collects data from the users' smart devices and transfers this data to the application servers, where they are used to detect the roads where the traffic is congested and display it in real time. The users can also add traffic events to the mobile application system like accidents, roads under construction and closed roads. The platform also provides a Web application where the users can monitor on a map all the events added by other users and also note the most traffic-congested roads, while also having an interface to determine the fastest route between two points, the issue with the estimates often being that they are far too optimistic for the actual traffic.

In [2], the authors created an application that attempts to reconstruct and display the continuous traffic flow on different routes based on the data collected by sets of road sensors which measure the distances and speeds. The algorithm estimates the full state of the traffic flow by using statistical inference methods and being able to reconstruct a 3D version of the traffic at a particular date and hour on the particular routes, the main focus of the application being the highway traffic. EnviroCar [3] is a platform composed of a mobile application and a Web application which, using an On-Board Diagnosis adapter for cars, gathers car and road data like carbon dioxide emissions, noise emissions and when cars are stationary, which are automatically recorded in the mobile application. In order for the user to gather data, he needs to attach the On-Board Diagnosis adapter to his car, install the enviroCar application on his mobile device and let the adapter transmit the data. After gathering the data, it is uploaded to a remote server where it can then be analyzed using the Web application, which displays on a map the areas with high carbon dioxide emissions, high noise or long stationary times in the traffic.

Another approach for traffic prediction is outlined in [4], where the authors develop an application that allows one to analyze and predict the traffic jams on the main routes in North America. The application used the Microsoft JamBayes research project, which was started to permit the prediction of the traffic flow on the North American roads based on data gathered over a period of five years from different vehicle sources and it has an interface which displays the traffic flow in a color-coded way, green roads signifying a lighter flow, while red roads signify that a heavy flow traffic in present. By using screenshots taken from the JamBayes project interface, the authors perform an image recognition in order to determine the areas where traffic jams are present and the system displays them as a heat-map layer over a Bing map so the users can note which routes they should avoid.

Traffic Origins [5] is a visualization application created for the traffic management controllers in order to monitor and analyze the changes in traffic affected by accidents or other road transit events. The application allows the user to observe the routes during certain time frames before, during and after the traffic events occurred. A map is displayed where all the routes are color-coded from green to red, depending on the level of congestion for each particular segment. To observe the changes caused by an accident, a circle will appear on the map just prior to the event and it will remain on the map for a certain time after the event has occurred. This way, the traffic manager can observe how the conditions are changing during an traffic event and decide how best to handle it.

To address practical problems in ITS, a suite of schemes and tools are required for realizing and analyzing big data in social transportation systems, among which of paramount importance is visual analysis [6]. Visual analytics is the science of analytic reasoning supported by interactive visual interfaces. In the core of visual analytics is data visualization, which transforms various types of data into appropriate visual representations, and greatly improves the efficiency of data comprehension and analysis [6]. Powerful tools are needed to enhance the user experience and proficiency in manipulating and understanding huge amounts of data collected from many users in various conditions and qualities. One such tool is the Iris geographic information system (GIS) - based a platform using the smartphone Android default API for geolocation [7]. The server side of the Iris platform complements the mobile system to collect data by storing, processing, analyzing, managing and presenting the results. Additional metrics related to the travel time and vehicle's speeds contribute to the assessment of traffic management issues [7].

In [8], the author presents a Web application which uses live traffic congestion data from Google Maps traffic layer for real-time congestion calculation. This application focuses in particular on urban roads congestion analysis. Besides the spatial aspect, the study concentrates on the temporal dimension of road link traffic by providing stored-time-of-the-day analysis data of when the congestion level was highest [8].

The most common method used to estimate the possible routes between two points is Dijkstra's algorithm, which was created for finding the shortest paths between different nodes in a graph, after supplying the initial node and the destination node. The algorithm starts from the initial node and advances through the graph, while keeping the distance count for each route until it reaches the destination node. To estimate the routes between two points on the map, a modified version of Dijkstra's algorithm can be used, which instead of starting from the initial point and going to the destination point, starts from both the start and end points and expands the routes on both sides until the routes meet in the middle. This way, the computational time is reduced because the operations can be performed in parallel. Depending on the distance between the start and end point, the algorithm can use street layers to avoid taking into account small streets for routes traveling from a city to another; the algorithm can use a layer for the highways, one for European roads, national roads and so on [9].

The IRMA project [10] seeks to implement a software framework that targets personal mobility that is green oriented, shared and public transportation and provide an Integrated Real-time Mobility Assistant. IRMA is an extendible modular platform using multiple sources of data (crowd, open, social, and sensor data) and it provides traffic services to several stakeholders, which includes the municipality, citizens, and transport providers. To access data, a service-oriented architecture based on an event driven platform has been adopted.

In [11] the authors provide a comprehensive survey on the state-of-the-art in visualization techniques for large-scale vehicle traffic data. Existing techniques include conventional ones such as 2D and 3D maps or more innovative ones, such as geometric projections (which display projections of multidimensional data sets), pixel-oriented views (which map each data value to a colored pixel) and hierarchical presentations.

The next step in ITS infrastructures is based on space-air-ground data, as introduced in [12]. The authors examine the new key data and technologies for ITS, naming satellites data and helicopter collected traffic data. The future approaches on ITS will probably include space-air-ground data acquisition sensors, dynamic data transmission, massive data storage, multi-source data fusion, massive data mining and analysis.

In contrast to existing solutions, we provide a complete system for collecting and analyzing urban traffic data. On our data, many specific and detailed algorithms or metrics can be obtained, in contrast to the limited data provided by Google Traffic or Waze. The main difference is that our work seeks to provide open traffic data augmented with contextual information. The present paper presents the basic low-level infrastructure we will use to analyze and label data gathered from the users.

On the client side, there are multiple applications in which the context-awareness is used to provide personalized services for the users, each researcher trying to develop new ways of determining the context while minimizing the battery consumption. Several related applications and techniques used to improve the energy efficiency of mobile context-aware data gathering applications are discussed below.

In [13], a model is proposed that is based on the user's context provides appropriate security for different applications The authors think that the same user, in different contexts, should have different roles and different levels of permissions in the applications, since some environments are considered safer for accessing certain kind of data than others. By mapping the raw data received from various sensors from the mobile device to the user's context, the authors created three classes of roles (R1, R2, R3), each of them having different degrees of permission for different actions, depending on the level of privacy required, each user having then assigned a different role based on his context. Three types of locations are defined, each with three sub-types, which are mapped to the role classes that determine how crowded the environment is: indoor, with the classroom(R2, R3), quiet room(R3, R3) and market hall(R1, R2) sub-types, outdoor, with a square(R2, R3), park(R2, R3) and street(R1, R2) sub-types and transportation with a bus(R1, R2), the metro(R1, R2) and a train(R1, R3) sub-types, where the first value represents that for a high crowd, and the second value that for a small crowd and R1 <R2 <R3, R3 granting permission the most often.

Seeking to minimize the energy consumption when detecting and processing the context, a new approach is proposed in [14] that modifies the general way of processing the context data. The general flow starts from the sensors, through the pre-processing feature extraction and context recognition, the context change being detected at the final step. The authors attempt to identify the change of context earlier in this flow, based on a lower level query, to avoid the processing of more computationally intensive procedures like decision tree logic. A middle-tier framework, called SeeMon was implemented, to provide an easier way to map certain actions to certain contexts. SeeMon provides a context monitoring query, which permits the user to map an action with a certain duration when a certain context is active, by using the following format: `Context <context element> ALARM <type> DURATION <duration>`. For the context element the equality and inequality operations are supported, while there are two alarm types an instant one that is triggered when the context conditions are met and a timed one, that will be triggered after a specified time once the context conditions are met. An alarm will be triggered if all the context conditions are met and the `<context element>` value changes from false to true, the duration parameter specifying how long the query should run.

In [15], the authors carried out an experiment to measure which sensors are the most energy efficient and which are consume the most energy; after completing the experiment and determining the most energy-efficient sensors, they implemented a system called SenseLess that relies heavily on these sensors, while using the most energy-consuming ones as little as possible and thus increase the battery lifetime.

The energy used for sampling the sensors and sending the data to a server, together with the actual cost of the transfer operation is also discussed in [16]. The author proposes a solution that leverages the costs for non-data-plan users, while also mining the energy consumption cost for data-plan users, by using WiFi or Bluetooth transfer to send the sensed data, saving 45%-50% in the data cost and 55%-65% in the energy consumption for each type of user.

3 Solution overview

The architecture presented in Figure 2 is composed of modules that represent the main characteristics and functionality of the system. To determine the user context changes and to measure the energy efficiency of the operation, we first needed to gather actual data from mobile devices used in different environments, then process and analyze this data and identify patterns corresponding to each user context. To achieve this, we implemented a mobile application that was used for data sampling, and it also collected input concerning the current context.

To analyze the data collected and aggregated, a Web application was implemented, that was used to visualize the user routes, calculate the distances and user speed and also analyze the data collected, which was plotted by the application. Based on all the data sampled by the sensors, metrics are then computed for each segment and stored in order to characterize each segment from the city by its speed, acceleration and sound level, all the data being displayed on the map so as to have an overview of an entire area.

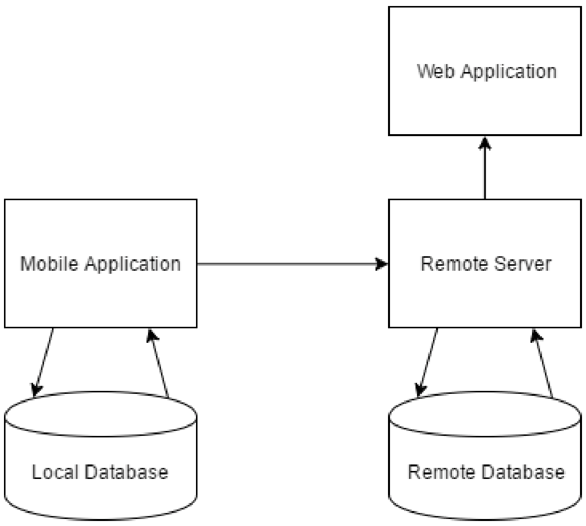


Figure 2: Solution overview

3.1 Our Mobile Data Sampling Application

In order to collect data, we created an Android application that will determine which sensors are available on the current device, the user being able to configure in the application which sensors should be enabled and how often they should be queried. The user can enable or disable sensors and also configure the querying period. By default, the following sensors are enabled with the following sampling time resolutions, as presented in table 1.

For the mobile application, the goal was to collect raw sensor data in a reliable way and to study the impact on the power usage. It is known that the GPS has a higher energy consumption compared to other device sensors. The mobile

Table 1: Sensor default sampling rates.

Sensor	Time (milliseconds)
Accelerometer	100
Gyroscope	100
Magnetometer	100
Linear Acceleration	100
Proximity sensor	1000
Light sensor	1000
Barometer	1000
Sound level	1000
Foreground application	2000
Location	5000

application will be used to determine the erxtent of its impact on the battery life is.

Another goal of the mobile application is to collect location data (especially from the GPS) in a more efficient manner by turning off location sampling if the device is not moving.

The measuring of the battery usage the internal Android battery usage monitor was performed and also a third party application called GSAM Battery Monitor, that also relies on the internal battery monitor. The data provided by the battery monitor displays the battery usage for the application, for the GPS and for the Google Play Service. In order to determine the total battery consumption, these need to be added together.

The aim of these measurements is not to give an absolute result that should be used as a reference for other applications and devices. The main purpose of these measurements is to compare the different location gathering methods with each other under similar conditions. Other measurements focused on determining the impact of sensor data collection, network location collection and sound level collection.

In order to be able to receive network location data, the device had to have an active SIM card and also be connected to the network at all times. Because the test must be performed under similar conditions, the network connection was left active during measurements that did not require it. The accuracy of the measurements changed slighty because of the device usage for activities such as phone calls and messaging, but these were kept to a minimum for the duration of the measurements. In all the tests the device was fully charged, with the configurations already set. After this, the data collection service was started. An alarm was set in order to notify the user to stop the process and save the results, this having a minimal impact on the results.

Three kinds of tests were run in order to estimate the efficiency of the sensor readings within the mobile client during normal usage conditions. The internal battery monitor provides the usage percentages relative to the percentage of the battery used. In order to display the battery consumption relative to the total battery the results had to be normalized. Energy accounting focused on built-in sensors, sound level and network location. The percentages of these measurements are of course relative to the total battery level divided by the power used during these measurements. The first test configuration focuses on the main built-in sensors (accelerometer, gyroscope, light sensor, proximity sensor, pressure sensor and magnetic field) used altogether at default sampling rates shown in table 1. Network location and noise level are activated separately during the second test and third test, respectively.

Table 2 presents the results of the six hour tests executions repeated three times, with the previous mentioned configurations. The total battery energy used indicates the amount of battery energy consumed by the device overall. The percentages for battery energy consumed by the application are relative to the total available energy, not the total energy used.

Table 2: User application battery usage.

Test type	Total battery used by system	Fraction of the energy used by app	CPU time
Built-in sensors	11.5%	0.43%	55.5s
Network location	14%	0.14%	17.5s
Sound level	7.5%	12.5%	24.5s

The second set of tests were run in order to measure the impact on the battery consumption using several location gathering methods: Google services with high accuracy priority (GS-HA); Google services with balanced priority (GS-BP); a dynamic location sampling algorithm (Dynamic); and basic location sampling (Basic). The energy consumption while using the dynamic collection mode depends on the amount of actual movement type.

Table 3: Location sampling battery usage.

Test type	Total battery used by system	Fraction of the energy used by app	CPU time
GS-HA	91.5%	57%	5039s
GS-BP	39%	3%	357s
Dynamic	29%	16%	541s
Basic	84.5%	44%	1991s

The results in table 3 are based on the battery profiling tool which gives the percentages for each application and system service. The total battery energy consumed by the application is the sum of energy consumption for the data collection application, the GPS location service and when the Google Play Service is used.

3.2 A Personalized Multi-Path Analyzer

The data gathered from all the devices that have installed the data collection application are stored in a database to be used afterwards by computing it different metrics for the different routes used and also determine different patterns for each user. The database contains an individual table for each sensor queried and uploaded by mobile clients.

In addition to the data collected from the default device sensors, we query data related to the application that is currently active on the device and the battery status, together with the context data entered by the user in the application, hence three extra tables were created for this.

The battery level data table stores the ID of the device from which the data was collected, the timestamp and the energy level of the battery at the given time. The contexts table stores the ID of the device from which the data was collected, the timestamp and a string representing the context entered by the user in the data-collecting application. Along with to all these tables, an extra devices table was added to store the ID of each device and the name of the users using it.

A Web application was created that helps the user analyze all the data gathered by the mobile sampling application and it allows him to visualize the locations from each particular route on a map based on the Google Maps JavaScript API and also analyze route segments, splitting the route into atomic parts with the Google Maps Directions API and then recomposing new routes for which we calculate different metrics. The personalized multipath analyzer consists of of three main tools, which are presented later on. The Web application is described in the next section.

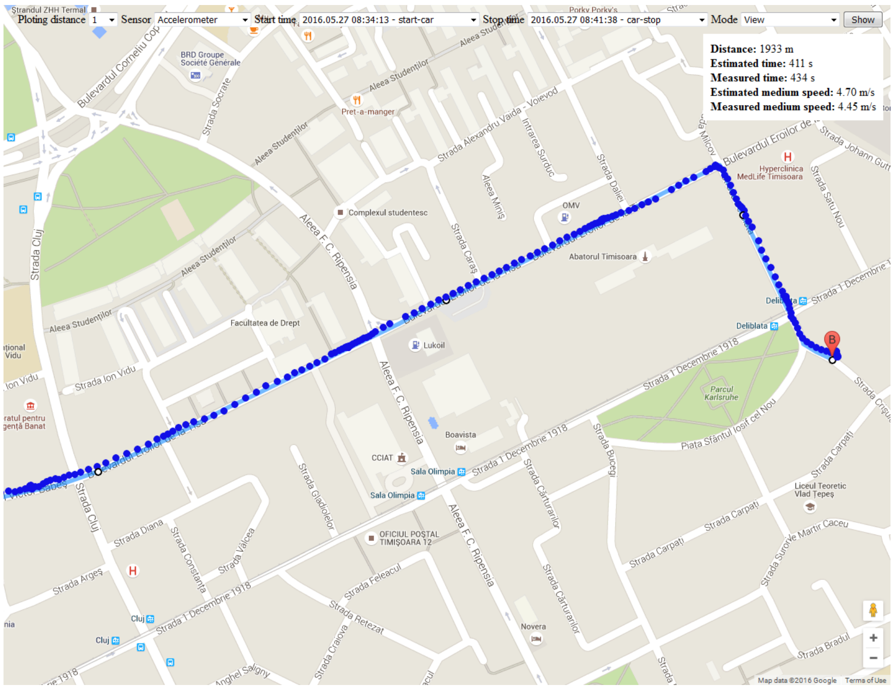
4 Traffic Visualization Tool

4.1 Visualization of raw data

The first tool we implemented was the visualization tool, which allows the user to visualize on a Google Map the route used by the user when moving from the start point to the end point based on the collected locations, while also allowing the user to analyze the data gathered from all the sensors.

Before starting the visualization tool, we need to choose which user we want to analyze the routes and the data, after which two select boxes will be displayed, one is called Start time and the other is called Stop time, each one containing the context start and end events introduced by the user in the mobile application. The options in the two select boxes display the date and hour when the event started or ended and the current context, for example: 2016.05.27 08:34:13 start-car, 2016.05.27 08:41:38 car-stop, 2016.03.27 08:34:13 start-pedestrian, 2016.03.27 08:41:38 pedestrian-stop. When the user chooses a start event from the Start time select box, the corresponding event from the Stop time select box will be automatically selected by the application. After the user has selected the desired start and stop time we use their timestamps to detect all the locations that have been stored for that user between the two timestamps and we display a marker on the map for each location so we can observe the route the user used.

In addition, we also make a query to the Google Maps Directions API and we can also display on the map the best route suggested by the API, together with some metrics so we can compare the actual route with the one recommended by the API: we display the distance between the start and stop location, the time estimated by the API, the actual time that passed while travelling from the start point to the end point, the average speed estimated by the API and the actual average speed measured by the device sensors, like that shown in Figure 3.

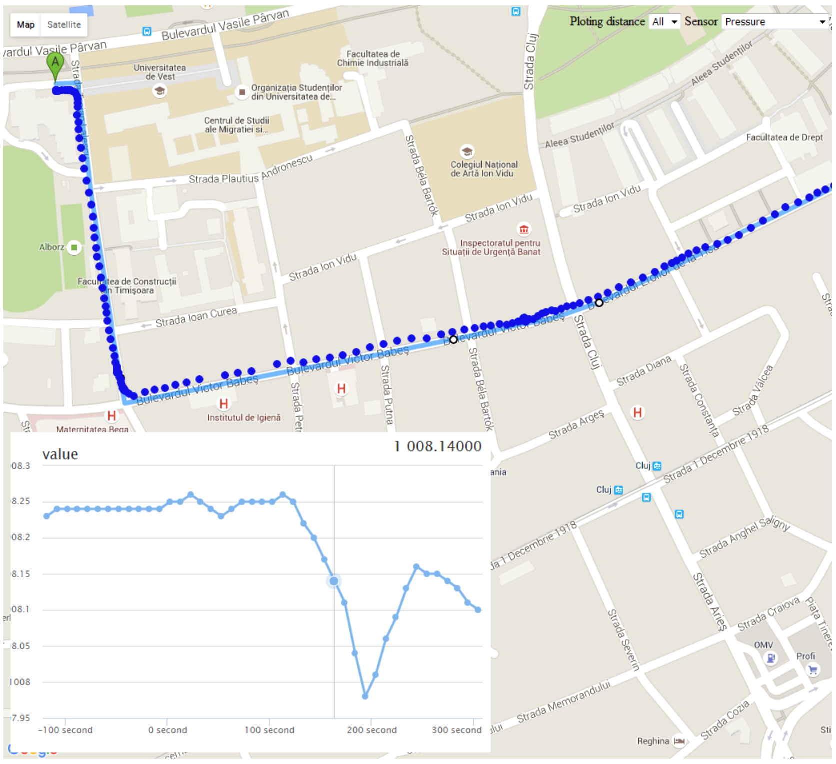

Ploting distance 1 Sensor Accelerometer Start time 2016.05.27 08:34:13 - start-car Stop time 2016.05.27 08:41:38 - car-stop Mode View

Distance: 1933 m
Estimated time: 411 s
Measured time: 434 s
Estimated medium speed: 4.70 m/s
Measured medium speed: 4.45 m/s

Figure 3: Visualization of the raw data

Having got the timestamp corresponding to the start time and the timestamp corresponding to the stop time we can also query all the data stored in the database for all the sensors between the two timestamps and we can plot this data on a graph over the map so the user can analyze patterns on the different routes. We added two extra select boxes, a sensor decide box and a plotting distance select box. From the sensor select box the user can select which sensor he wants to analyze, then the application will plot the values for that particular sensor on the graph. The second select box was because for long routes the amount of data that should be displayed on the graph is too large and also impossible to analyze this way, so the user can select from the box how many distance-points from the selected point the plotted graph should display. The available options for the plotting distance are: 1, 2, 3, 5, 10, 20 and all. After the user has selected the desired sensor and the desired plotting distance he can hover over the desired location point on the map and the data corresponding to the plotting range and that point in the middle will be displayed on the screen.

There are two types of plots that can be analyzed, namely a single plot for the sensors that retrive single values, as shown in the Figure 4, like the light sensor, the pressure sensor, the proximity sensor and for the sound and the battery level values stored. The graph displays all the data existing in the database for the current selected user between the timestamp corresponding to the start location and the timestamp corresponding to the stop location, when hovering over one particular location on the graph the value for that location being displayed for the user. The second type of plot is a multiple graph of sensor values (see below)


Ploting distance All Sensor Pressure

1 008.14000

Figure 4: Visualization details of single sensor data

For the accelerometer, the gyroscope, the linear acceleration and the magnetic field a three-plot graph is used because each of these sensors return a set of data with three components, one for each axis, so we need to plot them accordingly so the user can visualize all this data from the data set the same graph. Here we display three interconnected graph, while hovering a point from one graph corresponding to one of the axes, and the corresponding values for the other two axes are also displayed so they can be analyzed together by the user, like that seen in Figure 5.

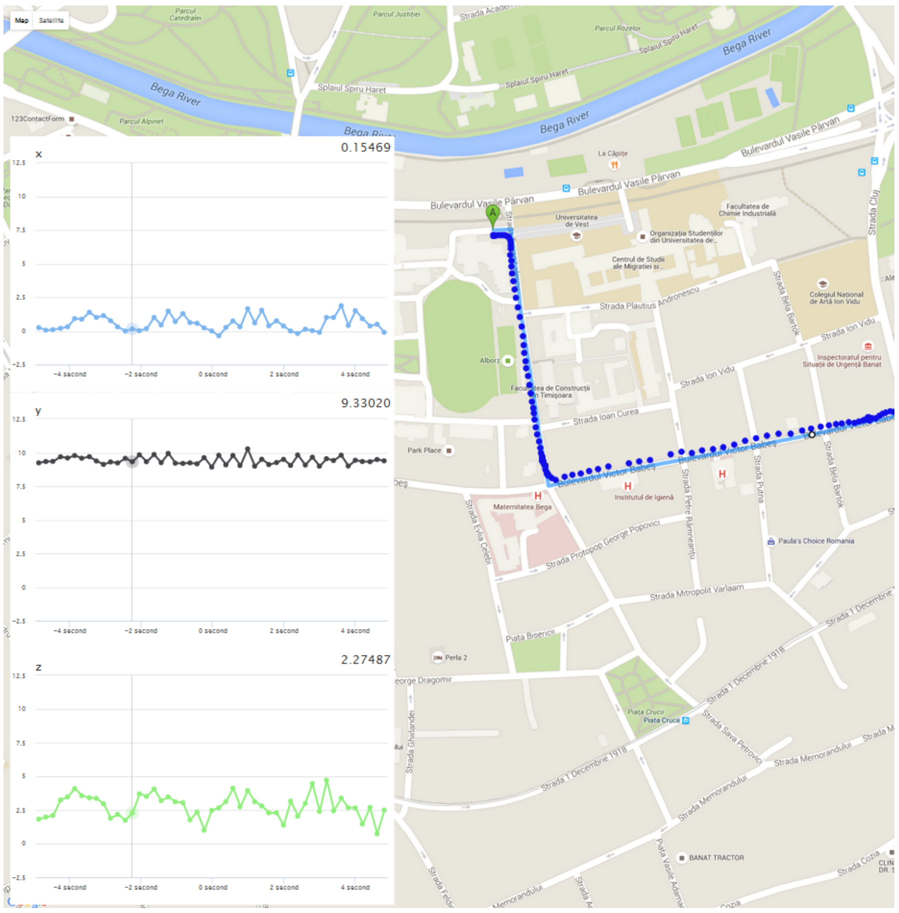


Figure 5: Visualization details of multiple axis sensor data

4.2 Segmentation Tool

The second tool that can be used in the analysis application is the segmentation tool, which performs a map matching between the route taken by the user and the route provided by the Google Maps Directions API, characterizing each segment from the route based on the data previously collected by the mobile application.

The segmentation tool also provides two select boxes: a Start time select box and a Stop time select box, that allow the user to decide which route he wants to process. After selecting a route, the user can start the process by clicking the Process segments button and he will have to wait until the progress bar displayed near the button is full so he can see that the entire process is completed and all the data has been stored in the database.

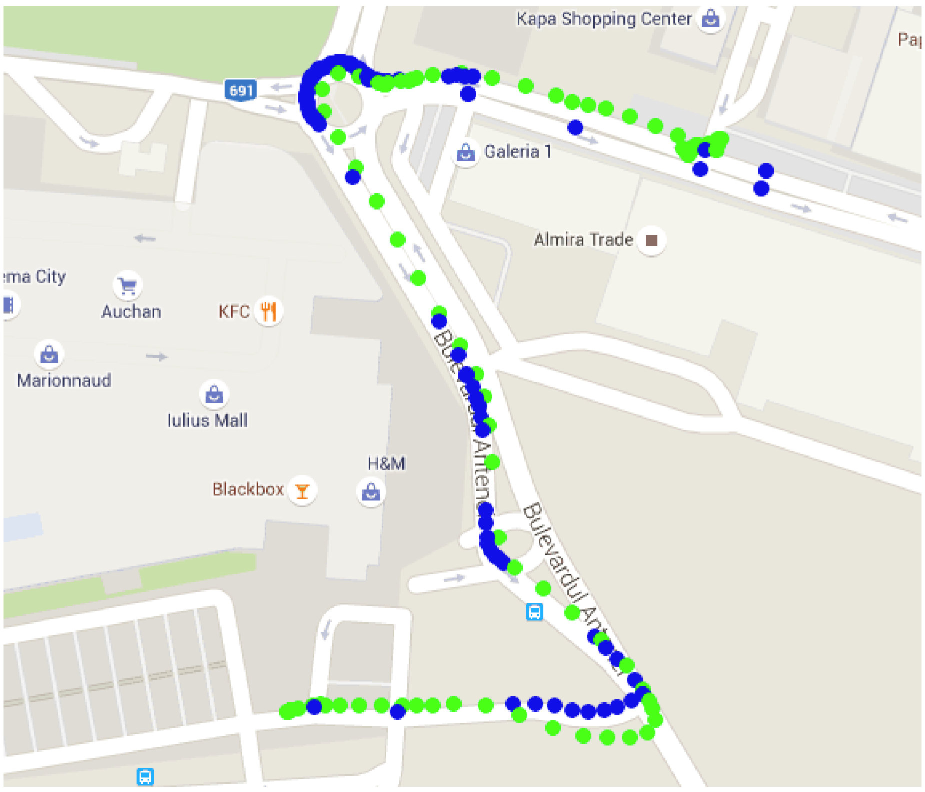


Figure 6: Visualization of segment limits

The segmentation process starts by sending a request to the Google Maps Directions API containing the start location, the end location and a number of eight waypoints, which are points from along the route followed by the user, the provided response being then used to map the data collected from the mobile application to the route segments. The response from the Directions API contains an array of locations that are the same if the user uses a certain road, regardless of the starting point or ending point of the route; anytime a user drives along a certain road the response from the Google Maps Directions API will always contain the same succession of locations. By storing these segments in the database we will be able, after segmenting many routes from different users, to compute different metrics for each segment based on the data from one user or all the users, using different time combinations. In Figure 6, the blue points are the location values received from the Directions API and the green ones are the locations got from the mobile application.

After requesting the segments from the API we had the entire route used decomposed into segments and the actual locations provided by our application, which each had its own timestamp. In order to calculate meta-data to characterize each segment, we needed to estimate the timestamps of each start and end location of each segment so we can then query the data from the sensors and determine the segment characteristics. To do this we attempted to find a segment determined by the locations from the mobile applications that is similar to the segment we are looking for, so we can approximate those segments have the same speed and then to determine their timestamps based on the timestamps of the points from the mobile application. First we calculated the bearing of the segment from the Directions API determined by a point with the same bearing, but by 5 meters from the start point and a point with the same bearing but right with 5 meters from the end point of the segment. Then from each one of the determined points we go left and right by 5 meters with a ninety degree bearing, creating a rectangle that contains in its center the original segment given by the Google Maps Directions API, like in Figure 7. A rectangle is used so we only take into account when approximating the time stamps the points that are actually part of that particular road. If two roads are really close to one another and the route is active on both roads we have to ensure that we are not using locations that will add time errors to the approximation.

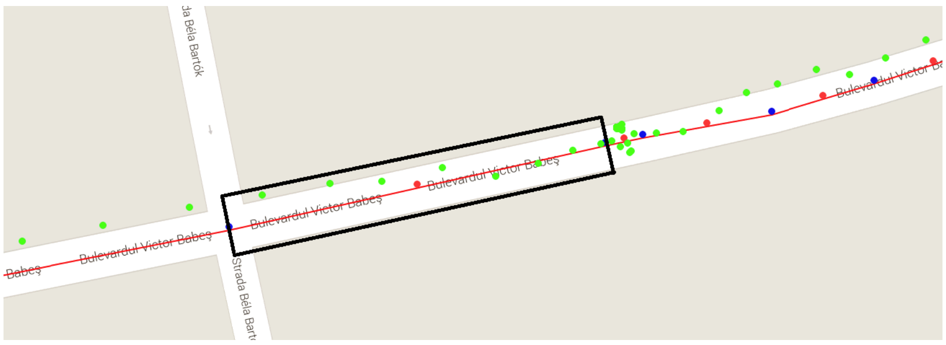


Figure 7: Segment mapping algorithm

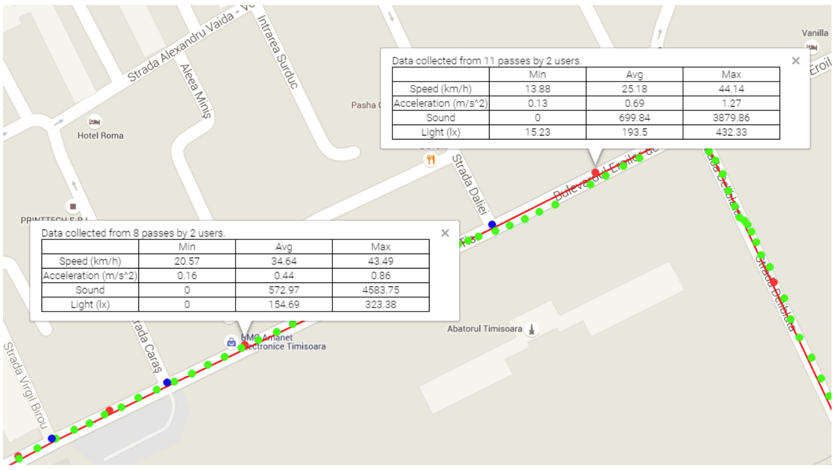
After determining the rectangle for a segment, we determine which of the location points sampled by the sampling application are inside the rectangle and then determine the location that is closer to the segment start point and the location that is closer to the segment end point. Based on these two points, we then determine the speed on this particular segment and then, from the first point sampled by the application that is inside the rectangle we approximate the timestamp for the segment start and from the last point sampled by the application that is inside the rectangle we approximate the end point.

Knowing the coordinates of the extremities for each segment that is part of the current route and also the timestamps for each one of them, we are making an AJAX request to the server where we transfer all the data mentioned above about each segment. The server checks whether the received segment is already stored in the database in the segments table and if it is not it is added. Then a query is assembled to get the data collected by the mobile application between the timestamps corresponding to the segment start and end point; and by using it we calculated the speed, the acceleration, the light intensity and the sound intensity for on the segment and we store it in the segment_data table. In this table we will have multiple entries that characterize the same segment and different periods of time and in order to have access to quick data about a particular segment without performed any intense calculations we calculate the metadata for the current segment and store it together with the segment data, as specified by the following attributes associated with path segments:

- The minimum, average and maximum speed
- The minimum, average and maximum acceleration
- The minimum, average and maximum light level
- The minimum, average and maximum sound level
- The number of unique users that have passed through the segment
- The number of passes through the segment, independent of the user

After the processing of the entire pool of segments for the current road is completed on the map, red segments will be drawn from each segment at the start location to each segment at the end location representing segments successfully characterized. A segment can be characterized only if in its rectangle there are at least two location points sampled from the mobile application, otherwise there will be no red polyline between the start point and end point of the segment, meaning it has not been characterized. In the center of the red polyline a pin will be displayed which will open an info window with the metadata that characterizes each segment. The info windows tell us how many users contributed to the segment characterization and the number of passes through the segment; and it is structured as a table containing minimum, average and maximum data about the speed, acceleration, light intensity and sound intensity calculated for the segment, like that shown in Figure 8

Additionally, on each location point sampled from the mobile application we included an info windows containing the point's timestamp and the speed at that particular point as retrieved from the GPS sensor, so the user can analyze the differences between the minimum, average and maximum speed calculated for the segment and the speed at a specific point, as shown in Figure 9.



Data collected from 11 passes by 2 users.

	Min	Avg	Max
Speed (km/h)	13.88	25.18	44.14
Acceleration (m/s^2)	0.13	0.69	1.27
Sound	0	699.84	3879.86
Light (lx)	15.23	193.5	432.33

Data collected from 8 passes by 2 users.

	Min	Avg	Max
Speed (km/h)	20.57	34.64	43.49
Acceleration (m/s^2)	0.16	0.44	0.86
Sound	0	572.97	4583.75
Light (lx)	0	154.69	323.38

Figure 8: Visualization of the segment details

The speed variable used to calculate the minimum, average and maximum speed is computed from the segment start and end point timestamps, while the speed in the green points is provided by the GPS. The fact that these speeds are correlated demostrates that the approximation made for determining the timestamps for the segment start and end points did not introduce any significant errors.

4.3 Analysis Tools

Having got all the data about all the segments from the routes taken by the users stored in the database, we implemented a tool that displays a Google Map with all the segments processed, so the user can have an overview of the entire city. The tool presents several boxes from which the user can modify the data he wants to be displayed on the map. The following options that are available are listd in table 4.

Each segment on the map is color-coded, and it has one of four colors: green, light green, orange and red, green representing highest values and red representing lowest values as shown in Figure 10.

The available modes for the overview tool are listed in Table 5.

Along with the visual representation of different modes, the user can also analyze each segment by clicking on it; an info window will be displayed that displays data concerning the number of passes through the segment, the number of users collecting the data, the number of time intervals in which data was collected, the minimum, average and maximum speed, minimum, average and maximum acceleration, minimum, average and maximum light level and minimum, average and maximum sound level, as indicated in Figure 11.

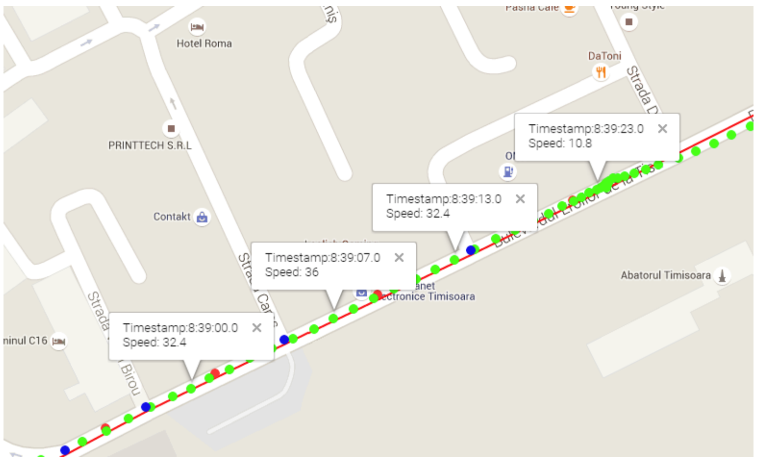


Figure 9: Visualization of the point details

5 Experimental Results

After finalizing the sampling application and testing it to ensure that all the data are sampled and stored correctly and after setting up the remote server we uploaded the sampling application in the Google Play repository so we could start collecting data from the daily routes that the users were taversing. The application was uploaded as a beta version, and only users that allowed us to collect data were allowed to use it. We present below some statistical data, to have an overview over the amount of data on which the research is based and also describing the three result components that can be determined by using the application:

- The user profiling,
- The area profiling,
- The best route estimation.

5.1 Statistical Data

The data that can be visualized and analyzed using the Web application and is presented here was sampled over six months, between the begging of January 2016 and end of June 2016 (6 months), but the process of data collecting is still in progress and only the processing of the routes has been temporarily put on hold to ensure data integrity and consistency.

So far the remote database has stored the following amount of data collected from user devices with the sampling application:

There were 364 routes recorded by the users in the application, routes which had the necessary data in order to characterize the resulting segments, and from the

Table 4: Analysis tool options.

Name	Description
Start hour	the data that was collected after a given hour will be displayed on the map
Stop hour	the data that was collected before a given hour will be displayed in the map
User	data collected by all the users or just the data collected by a particular user
Mode	specifies which property of the segment will be highlighted on the map

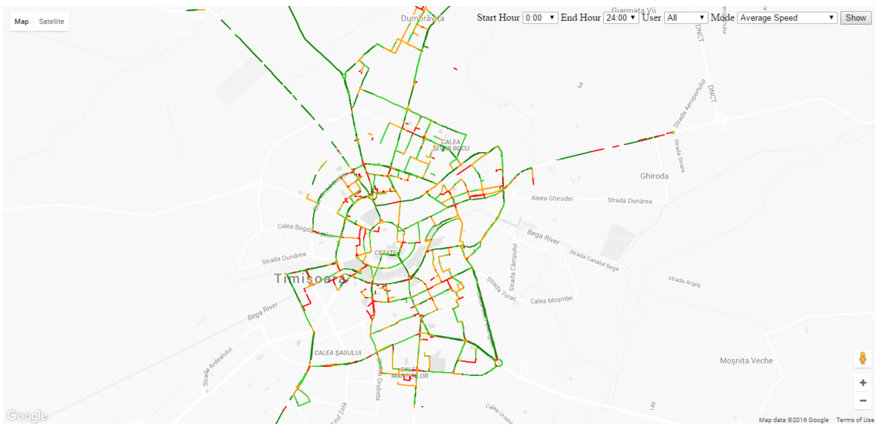

Start Hour 0:00 End Hour 24:00 User All Mode Average Speed Show

Figure 10: Aggregated data visualization

processing of those routes it resulted a total of 16,141 segments, most of them being in the Timisoara urban area. There are approximately 51.000 entries of segment data and 200,000 entries of segment metadata; in other words an average of four passes per segment. In this phase of the project 4 users who drive their own car, were involved in collecting daily traffic data. They installed the mobile client and let the application run for 6 months.

5.2 User Profiling

By using the overview tool in the Passes mode and selecting a certain user, we can see which routes and the areas that are most used by a given user, and which one determine his day-to-day travel route. In Figure 12, there are segments detailed in

Table 5: Overview of our analysis tool

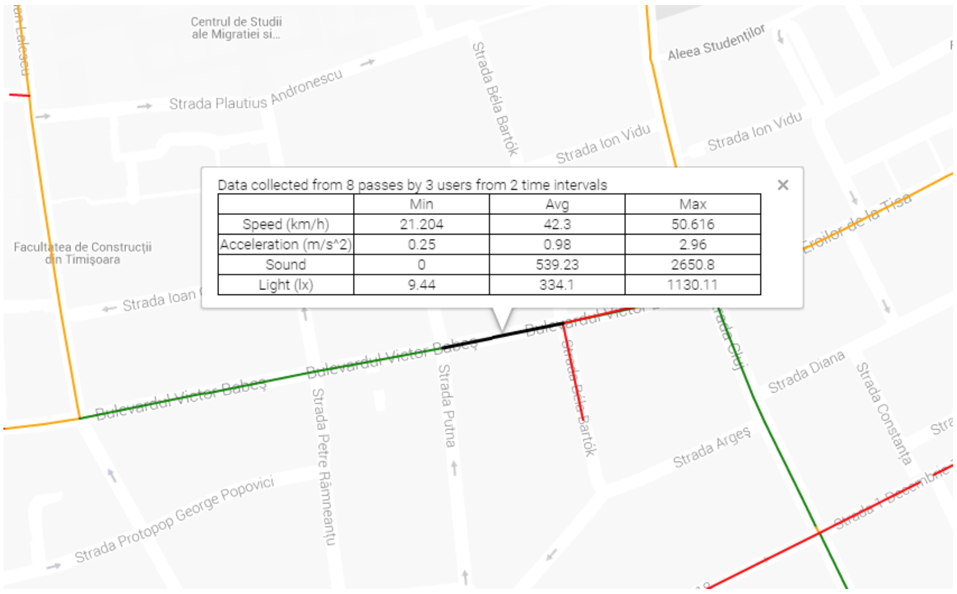
Name	Description
Passes	The segments marked in green are those that have been used more extensively and the segments marked in orange and green are those with a lower passing count
Users	green segments reflect the fact that there is data was sampled from many users, while the segments marked in red and orange are those with data samples got from fewer users
Time intervals	green segments denote data sampled from many time intervals, while the orange and red segments have data sampled from less time intervals
Speed	green segments have a high minimum, average or maximum speed, while the red and orange segments have lower values for speed
Acceleration	green segments have a high minimum, average or maximum acceleration, while the red and orange segments have lower values for acceleration
Light level	green segments have a high minimum, average or maximum light level, while the red and orange segments have lower values for the light level
Sound level	green segments have a high minimum, average or maximum sound level, while the red and orange segments have lower values for the sound level

the Passes mode for two different users, depicting the routes used the most by each user. The green segments denote the routes most used by each user, while the red segments denote the routes that that the users do not use regularly.

Using the Passes mode for a specific user in combination with the timeline where we can select the start hour and the stop hour for the data that is being displayed on the map, we can determine the routes that the user used for traveling from home to work and from work to home by looking at the green segments. Between 07:00 and 10:00 we assume that the green route shown on the map matches the home-work route, while between 16:00 and 19:00 we can assume that the green route shown on the map matches the work-home route.

By using the Average speed mode with the username selected and within the desired time frame the user can estimate the best route to use to get from one point of the city to another by studying the green segments, as shown in Figure 13, which are the segments with the higher speed. In this way, the estimate is based only on the current user's data, based on his own driving style.

If no user is selected, then the estimate is based on all the user data and some users may have a more aggressive driving style than others, progressing this way



Data collected from 8 passes by 3 users from 2 time intervals

	Min	Avg	Max
Speed (km/h)	21.204	42.3	50.616
Acceleration (m/s^2)	0.25	0.98	2.96
Sound	0	539.23	2650.8
Light (lx)	9.44	334.1	1130.11

Figure 11: Aggregated data details visualization

with an increased speed. Hence, by selecting to just use the data collected by him, the use can estimate the best route for his own driving style.

As can be seen in Figure 13, for two different users the colors of the segment may differ for the same segment, influenced by the number of cars on the road at a certain time, but they are also influenced by the user's driving style, for certain segments the right user is more aggressive in traffic than the left one, telling us that it is important for the best route estimate to be made based on the user's own data, where possible.

The validation of the proposed paths was performed with user intervention using two metrics (path duration and path length) and user feedback. Both routes were taken by one user one after the other, and then a discussion took place about the two routes.

5.3 Area Profiling

By using the Passes mode with the data collected from all the users, also specifying the time frame for the analysis, we can determine which routes from a city or area are the most used one during the whole day or which routes from the city are the most used at certain times. The most used segments are also the most reliable as regards their metrics because they are calculated based on more passes through that route. The segments colored in green are the segments that were used more and there are multiple data entries in the database so the metrics are more accurate, while the red and orange segments have less passes through them, so the calculation

Table 6: Experimental database.

Sensor	Records count	Size [MB]
accelerometer	over 18.200.000	1.488
magnetometer	over 10.000.000	660
gyroscope	over 9.430.000	620
linear acceleration	over 5.500.000	360
light level	over 2.600.000	148
barometer	over 1.380.000	80
location (network/GPS)	over 404.000 entries	36
sound level	over 241.000	15
battery status	over 58.000	4
foreground application	over 50.000	4

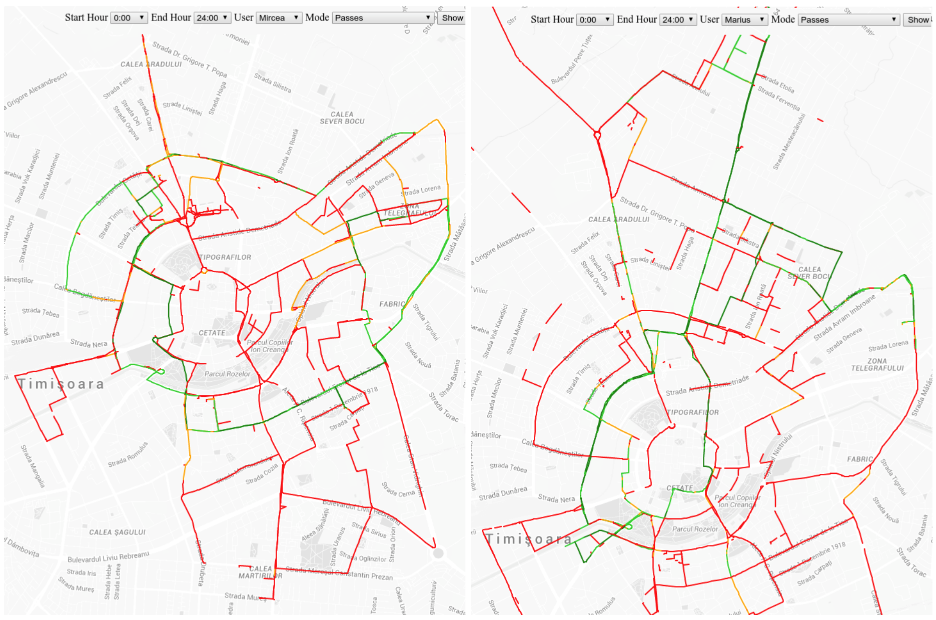
of the metrics is not as accurate as the first one.

The reliability is also based on the number of users from which are getting the data from; the most used segments by different users can be examined via the Users mode. The green and orange segments tell us that through specific segments metadata are based on data collected from more users, while the red segments tell us that the metadata for that segment is calculated based on data collected from only one user.

By using the Average speed mode and using the data collected from all the users we can determine which segments are the most crowded in an area or city. Using the map and the color-coded segments, the user can identify which routes should he use and which ones to avoid at certain hours in order to reach his destination more quickly.

In Figure 14, the segments colored in green are the ones with speeds over 50 km/h, the ones colored in light green correspond to speeds over 35km/h, the ones colored in orange correspond to speeds over 20 km/h and the ones colored in red are corresponding to speeds under 20 km/h.

Besides using the Average speed mode to estimate the best route to get from one point on the map to another, the user can use other two approaches: an pessimistic one, by using the Minimum speed mode, where for each segment on the map the lower speed ever recorded for that particular segment is displayed and an optimistic approach using the Maximum speed mode, where the maximum speed ever recorded on the segment is displayed on the map. Below, in Figure 15 the two different options (optimistic and pessimistic) are illustrated to highlight the difference between them.


Start Hour 0:00 End Hour 24:00 User Mircea Mode Passes Show

Start Hour 0:00 End Hour 24:00 User Marius Mode Passes Show

Figure 12: Visualization of the segments data availability

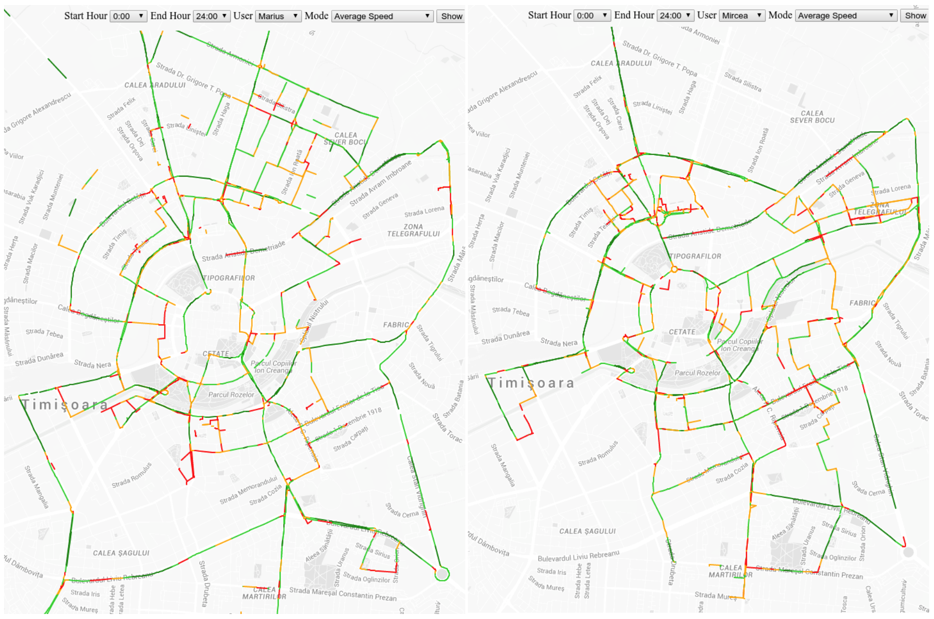
5.4 Best Route Estimation

By using the Average speed mode the user can study the possible routes between two points and then, based on the data gathered by himself or by other users, he can choose which ones from all possible routes he should choose and then calculate which one of these chosen is the best one in terms of distance or time, necessary to finish the route based on the average speed computed from the segment's historical data.

Assuming that the user wants to travel by car from point A to point B, as depicted in Figure 16, he can select the desired time frame and the segment data will be displayed on the map accordingly. He can also choose whether he wants the estimate to be based only on his sampled data, or whether he wants to use the data from all the users that have our application installed on their devices.

By analyzing the map the user can study some routes between the two points, but he can only choose some of them visually, by selecting the green roads for their greater speed. But having all the routes on the map composed of segments for which we have data collected, the user can choose and compare two actual routes, as in Figure 16.

Here, we called the two different routes between the point A and point B Route 1 and Route 2. By examining them visually on the map, the two chosen routes appear pretty similar, having almost the same distance and each one having more green segments than red and orange ones, meaning they are fast routes, see Figure 16.



Start Hour 0:00 End Hour 24:00 User Marius Mode Average Speed Show

Start Hour 0:00 End Hour 24:00 User Mircea Mode Average Speed Show

Figure 13: Average speed data visualization

By recomposing the routes segment by segment and having the length of each segment and the time needed to traverse it, we came to the conclusion that even through the route distances are not that different, Route 2 is much faster than Route 1:

- Route 1 is composed of 263 segments that total 6,675 meters and can be traveled on average in 19 minutes and 4 seconds with an average speed of 21 km/h.

- Route 2 is composed of 227 segments that total 6,276 meters and can be traveled on average in 11 minutes and 59 seconds with an average speed of 31.5 km/h.

Even though by looking at them on the map, the two routes look similar and they seem to have the same traveling speed, by calculating the sdtimates based on the aggregated data we find that the second route is thirty percent faster than the first one; by using the estimate and not just by looking at the map, the user can make a calculate choice based on this estimate.

The user can choose one of the three available modes to estimate the best route: the general mode, by using the Average speed mode, the pessimistic mode by using the Minimum speed mode and the optimistic mode by using the Maximum speed mode. The general estimation mode is depicted in Figure 19 and it is based on the average speed for each segment, computed from all the users that passed through

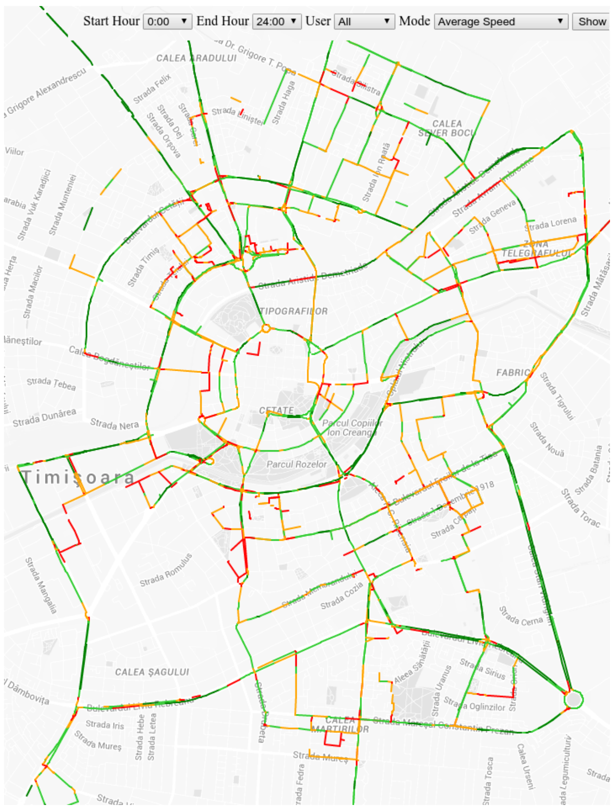
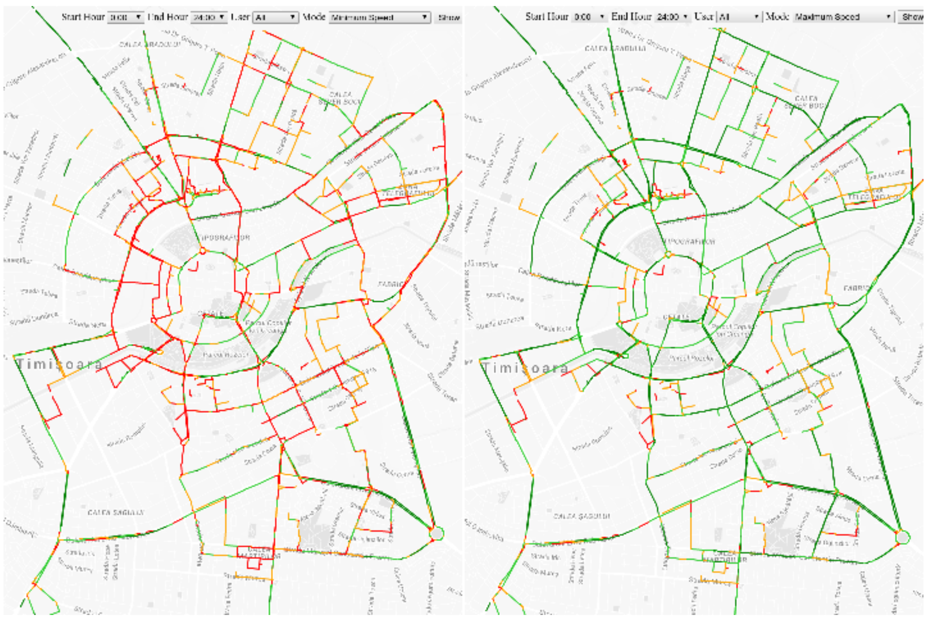


Figure 14: Visualization of the average speed data

that particular segment. The pessimistic estimation mode can be seen in Figure 20 and it is based on the lowest speed ever recorded for each particular segment. The optimistic estimation mode is depicted in Figure 21 and is based on the highest speed ever recorded for each particular segment.

We choose a route from one random point to another and we estimated the best route for each of the following cases: general, pessimistic and optimistic. For the route between the point A and point B, which is 2,594 kilometers long and is composed of 58 segments, the following times, were estimated:

- The general mode: 5 minutes and 20 seconds, with an average speed of 29 km/h
- The pessimistic mode: 28 minutes and 10 seconds, with an average speed of 5.5 km/h
- The optimistic mode: 4 minutes and 2 seconds, with an average speed of 38.5 km/h


Start Hour 0:00 End Hour 24:00 User All Mode Minimum Speed Show

Start Hour 0:00 End Hour 24:00 User All Mode Maximum Speed Show

Figure 15: Optimistic and pessimistic data visualization

The difference between the pessimistic mode and the general mode is significant compared to the difference between the general mode and the optimistic mode, but this can be explained by the fact that the pessimistic mode takes into account all the events that ever slowed down the traffic for that particular route, while the optimistic one is determined by a regular, no-event traffic flow.

6 Conclusions

All the improvements that have been added to the mobile devices over the past few years, like the built-in sensors and the GPS, can be used in the user's favor. By collecting data concerning his day-to-day activities we can perform a user profiling and then provide personalized features for each user. This user profiling can be carried out by using non-sensitive data, like the data from the build-in sensors, but the main focus when collecting the specified data should be on the amount of energy that this operation is consuming. It is important to improve the energy efficiency of the data collection for the applications that provide certain features based on sensor data querying by dynamically modifying the sampling rates or enabling and disabling the sensors completely, depending in the context in which the user is at a certain time.

By sampling data from the accelerometer, gyroscope, magnetometer, barometer, linear acceleration sensor, light sensor and the locations of the network providers

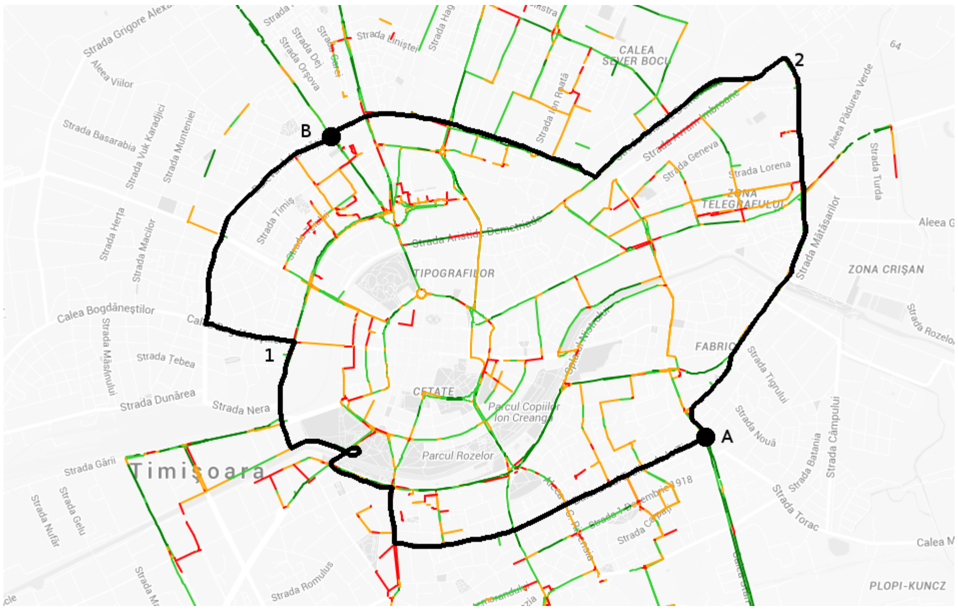


Figure 16: Comparative visualization of the routes

and the GPS while the user is driving, using public transport, using his bicycle or walking, we can compute all this data and characterize both the user and the routes he is using. The Google Maps Directions API allows us to query a certain route and it returns all the segments that make up that particular route, so after splitting it into segments we can compute the acceleration, speed, sound level, light level, distance and time frame metrics for each one of them. After gathering sufficient data from different users in different time frames traveling on different routes and after parsing all this routes, we can calculate metrics for each of the segments and then reassemble them to form different routes for which we can calculate the distance and estimate the average speed and required time to get from one point to another using either a general mode, based on the average speed of each segment, a pessimistic mode based on the minimum speed of each segment and an optimistic mode based on the maximum speed of each segment. The key aspect of the route estimation is that the user can choose the time frame in which he wants to travel and only the data sampled in that time frame will be taken into account to calculate the segment data. Another important point is that the user can select that the estimation should be based only on the data collected by him, this way ensuring that the estimate will be made according to his driving style.

Having 16,000 segments characterized and 51,000 segment data entries in the database means that we only have a small part of the Timisoara covered, so for the next time period our goal is to include more users in the collecting process and record and process as many routes as possible in order to characterize all the segments from the metropolitan area of Timisoara and also gather data from all the existing hour-intervals, tso that the estimates will evermore reliable and accurate.

The applications created allow the users to gather their traffic data and map it to road segments in order to get the route best estimates in a pessimistic, optimistic or general way, for particular time intervals, based on the user profile or on a general profile with data gathered from all the users. The next phase in the application's development will be to implement an interface where the user can specify a starting point and a stopping point and as many intermediate points he finds necessary, to be able to define the routes he considers as his best options between the two points. The application will need the to process each route and estimate in a general, optimistic or pessimistic way which routes is better in terms of distance and transit time. Another key task will be the optimization of the remote database, because the size increases with each recorded route, which at this time is over 3.1 Gb. After processing the segments and calculating their metadata all the unnecessary data sampled from the sensors should be trimmed so that one can prevent huge amounts of raw sensor data from building up and being stored.

References

[1] Smith, B. L., Zhang, H., Mike, F., Green, M Final report of ITS Center project: Cellphone probes as an ATMS tool", University of Virginia: Smart Travel Lab, 2003.

[2] Wilkie, D., Jason S., and Ming L. Flow reconstruction for data-driven traffic animation. ACM Transactions on Graphics (TOG) vol. 32(4), pp. 89-99, 2013.

[3] Broring, A., Remke, A., Stasch, C., and Mollers, J. enviroCar: A Citizen Science Platform for Analyzing and Mapping CrowdSourced Car Sensor Data. Transactions in GIS, vol. 19(3), pp. 362-376, 2015.

[4] Tostes, A.J., Duarte-Figueiredo, F., Assuncao, R., Salles, J., and Loureiro, A. From data to knowledge: City-wide traffic flows analysis and prediction using Bing maps. Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing ACM, p. 12, 2013.

[5] Anwar, A., Till, N., and Ratti, C. Traffic origins: a simple visualization technique to support traffic incident analysis. Proceedings of IEEE Visualization Symposium (PacificVis), pp. 316-319, 2014.

[6] Zheng, X., Chen, W., Wang, P., Shen, D., Chen, S., Wang, X., Zhang, Q., and Yang, L. Big Data for Social Transportation IEEE Transactions on Intelligent Transportations Systems, vol. 17(3), March 2016.

[7] Goncalves, J., Goncalves, J.S.V., Rossetti, R.J.F., and Olaverri-Monrea, C. Smartphone Sensor Platform to Study Traffic Conditions and Assess Driving Performance Proceedings of IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), October 8-11, 2014. Qingdao, China.

[8] Petrovska, N. and Stevanovic, A. Traffic Congestion Analysis Visualisation Tool Proceedings of IEEE 18th International Conference on Intelligent Transportation Systems, 2015.

[9] Dezani, H., Bassi, R., Marranghello, N., Gomes, L., Damiani, F., and Nunes da Silva, I. Optimizing urban traffic flow using Genetic Algorithm with Petri net analysis as fitness function. Neurocomputing, pp. 162-167, 2014.

[10] Motta, G., Sacco, D., Ma, T., You, L., Liu, K. Personal Mobility Service System in Urban Areas: the IRMA Project IEEE Symposium on Service-Oriented System Engineering, 2015.

[11] Gondim, H.W.A.S., do Nascimento, H.A.D., and Reilly, D. Visualizing Large Scale Vehicle Traffic Network Data - A Survey of the State-of-the-art International Conference on Information Visualization Theory and Applications (IVAPP), 2014.

[12] Xiong, G., Zhu, F., Dong, X., Fan, H., Hu, B., Kong, Q., Kang, W., Teng, T. A Kind of Novel ITS Based on Space-Air-Ground Big-Data IEEE Intelligent Transportation Systems Magazine, pp.10-23, Spring 2016

[13] Zhuo, W., Deng, R.H., Shen, J., Zhu, J., Ouyang, K. and Wu, Y. Multidimensional Context Awareness in Mobile Devices. MultiMedia Modeling, pp. 38-49. Springer International Publishing, 2015.

[14] Seungwoo, K, Lee, J., Jang, H., Lee, Y., Park, P., and Song, J. A scalable and energy-efficient context monitoring framework for mobile personal sensor networks. Mobile Computing, IEEE Transactions on 9, no. 5, pp. 686-702, 2010.

[15] Abdesslem, B., Fehmi, A.P., and Henderson, T. Less is more: energy-efficient mobile sensing with senseless. Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds. ACM, 2009.

[16] Wang, L., Zhang, D., and Xiong, H. effSense: energy-efficient and cost-effective data uploading in mobile crowdsensing. Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication. ACM, 2013.