Modelling and visualizing macroscopic movements

by

Ananya Chowdhury
Matriculation Number TODO

Piotr Mrówczyński
Matriculation Number 387521

Gabriel Vilén
Matriculation Number TODO

A project documentation submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

June 22, 2017

Supervised by:
Bianca Lüders
Friedhelm Victor
Prof. Dr. Axel Küpper

Service-centric NETWORKING

Technische Universität Berlin

1 Introduction

1.1 Macroscopic City Movements - problem definition

Here we should describe what problem project is about to solve. In the 1st term presentation I mentioned that we want to answer two questions:

Given a location, where and in what proportion people move to another location - Movement Graph?
Where and how long do people stay at certain location? - but we need to be more precise here.

1.2 Localization-based services and positioning technologies

TODO: we need here more details about what our Sudanian guy were supposed to research

1.2.1 Positioning technologies

TODO: http://geoawesomeness.com/knowledge-base/location-based-services/location-based-services-technologies/

1.2.2 Localization-based services

We should describe what are proactive and reactive LBS- https://www.move-forward.com/gps-everywhere-location-based-services/

Here we should describe how the data is obtained, and emphasise the fact that location updates are based on MOVEMENTS, so each next point is triggered by some mobile phone movement. Point updates are not triggered while being stationary.

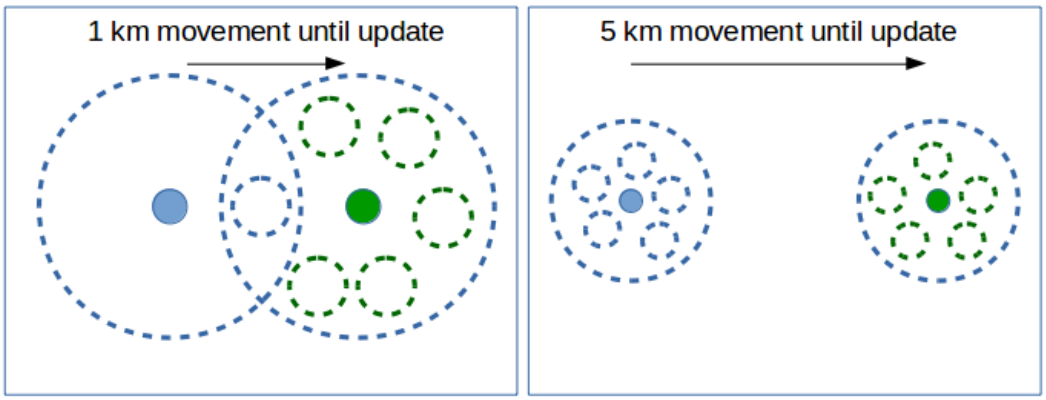


Figure 1.1: Continuous and Discontinuous location update after movement to section lacking area details

1.3 Human Mobility within the city

According to research [**HumanMobility1**][**HumanMobility2**][**HumanMobility3**], the average walking speed in the city for man aged 40 is 1.4 m/s (5 km/h), and slowest one is for elderly (with walking impairment) below 1 m/s (3.6 km/h).

TODO: elaborate a bit more

2 Related Work

2.1 Approaches to stop detection for localization services

2.1.1 Stop Detection analyzing repetitive appearance at location

2.1.2 Stop Detection based on continuous localization

2.1.3 Stop Detection based on mobility index

In the paper [**MobilityIndexGIS**], movement of mobile devices between cells (handovers) is considered. To detect the periods of slow movements of stops at the specific location, parameter called Mobility Index is considered.

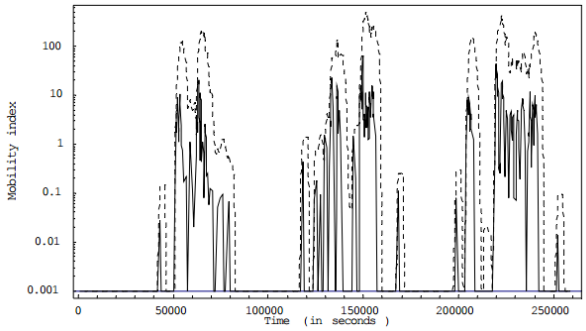


Figure 2.1: Mobility Index. Source: [**MobilityIndexGIS**]

In a populated area, with many GSM cells, the mobile terminal can change from one cell to another within seconds or after several minutes. Given a set of consecutive records, the mobility of a user can be estimated by calculating the Mobility Index over a pre-defined time period (sliding window).

The Mobility Index is defined as the sum of the "distances" between each record and the previous ones, where the "distance" is the inverse of the time spent on each cell. If the value of MI is below certain threshold, it is assumed that the object stopped or moved slowly.

In the publication, sliding window of 10 minutes and a mobility index threshold value of 6 has been used for certain area. The obtained results were in agreement with actual movements during more than 90% of the time.

High mobility index tells that user have been moving from checkpoint to checkpoint within

very small time distance. Rapid drop in mobility index represents very long duration at the last point, and slight decrease in mobility index means that user changed position, but within longer duration.

3 Concept and Design

We need to describe here what do we want to achieve, what do we mean by modelling and visualizing macroscopic movements etc - thus general architecture that we have 3 components, Stop Detection from telekom data based on movements and updates of "areas" in your mobile phone, Clustering of Stops to find most popular "stops" in the city, Graph Generation to find where do the people move and in what amount

3.1 Stop Location Detection

3.1.1 Stop Detection for proactive localization based services

Analyzing different approaches for stop detection based on localization data (ref. section 2.1), we decided to design algorithm based on human behavior regarding mobility within the cities (ref. section 1.3) and reusing the concept of Mobility Index (ref. subsection 2.1.3)). Stop Location Detection Algorithm have to be able not only identify the time of start stop, but also the stop location.

3.1.2 Mobility index based approach

In related works about mobility index discussed in subsection 2.1.3, the approach to stop detection using mobility index is bound to the way the data is being sampled (handover from one GSM cell to another). The data is recorded whenever mobile device changes its GSM cell.

In case of the approach discussed in this paper (section 1.2), the situation is similar, however instead of GSM cells we have artificially created area around the location, and while new location is obtained at some distance after movement, new area might need to be updated and points is being recorded.

In a populated area, the user can move from one area to another after several seconds or only after minutes. Thus, if within a certain period of time (called mobility index time window) the user stays in the same area (mobility index is equal to 0) or changes the areas with small pace (mobility index is below certain threshold) it can be assumed that the user is immobile or approaching stop location and having stay somewhere within that time window.

Approach based on mobility index could precisely identify how mobile has been user within the certain time period and identify longer stop locations, however mobility index is not able to identify short stops at location where in the certain time window user been moving fast, stopped and moving fast again.

3.1.3 Human Behavior based approach

This approach targets the condition for movement between two points, which has to be satisfied, so that it can be considered as a stay within the certain distance.

The section 1.3 discusses, that in average, walking human is covering a short distances with the speed of over 3km/h up to 5km/h. However for movement on long distances, user will most likely use city or private transport, and thus his speed can vary much in that time/distance period.

This means that stop detection based on speed of movement over long distance might not be effective, however, on small distances it might quite precisely identify short stays at certain locations, in between high mobility of the user.

3.2 Clustering of Stops

Clustering can be described as "*the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)*" **[clustering]**.

After our stop detection algorithm has detected the proper stops we want to perform some analysis on these. Our goal is to get the overall movement patterns of the population of a certain area, which means we are interested in **macroscopic** movements and not microscopic (individual) movement. If we would consider every individual our analysis algorithms would get fed with a lot of noise (outliers) which would make our data mining and prediction difficult and inaccurate. Meaning, *Macroscopic Movements* are not interesting in tracking individuals but rather seeing the big picture.

Once our clustering algorithm has been applied, and we have the overall picture, we can see which stops are most popular, where people are moving from them, and how long they stay at a certain location. One could then perform some interesting graph analysis on these locations, answering questions such as "what is the probability that a person moves from point A to B?", or "what is the average stop time at location A?".

To achieve these clusters, which represents our interesting stop locations, we need to apply some clustering algorithm. There are different types of clustering (such as centroid, distribution, or density-based), and different algorithms with parameters that can be applied (such as K-means, DBScan, or OPTICS). However, they all work to achieve the same goal of structuring (clustering, grouping) similar objects and neglecting outliers from these groups. The algorithm and parameters are application sensitive and our choice will be handled in the *evaluation* section.

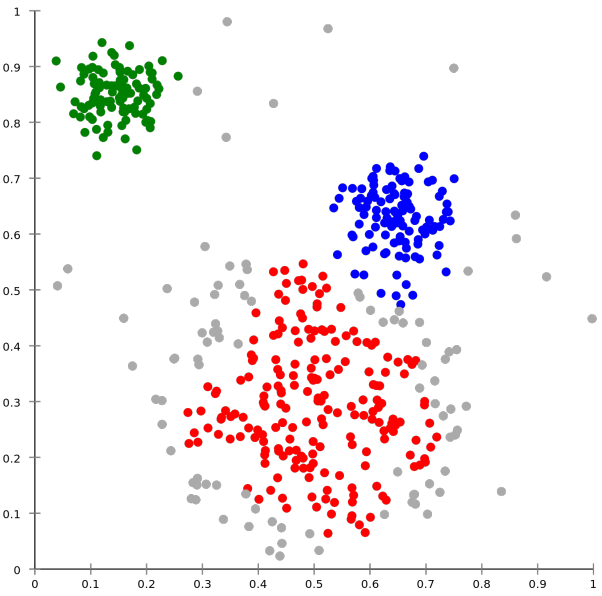


Figure 3.1: Clustering applied to a dataset.

3.3 City Movements Graph

4 Implementation

4.1 Scripts for data analysis - usage

TODO:

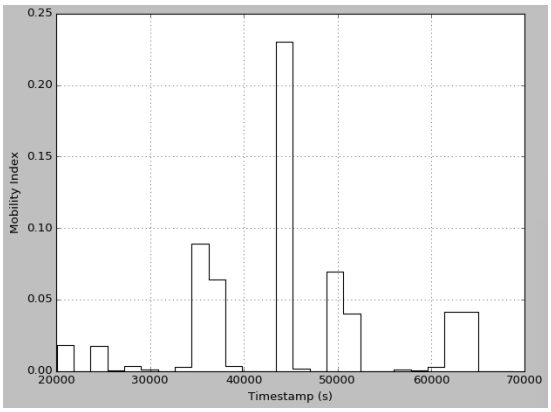
4.2 Stop Location Identification - batch processing in Apache Spark

TODO: final implementation and algorithm should be discussed here

4.2.1 Calculation of mobility index

Firstly, for a single user, minimum and maximum timestamp of sorted sequence is being found. Having value of MobilityIndexWindow (e.g. 30 minutes), function is creating equaly spaced bins of that value within maximum and minimum timestamps.

Secondly, each point is being visited and is assigned to the proper timestamp bin according to its own timestamp and duration to the previous point.




```
[],
[122.99999999999999, 157.0, 297.0, 3505.0],
[],
[582.0, 69.0, 949.9999999999999],
[1699.0000000000002],
...
[422.00000000000006, 71.0, 111.0, 286.0, 82.0, 3439.0000000000005],
[],
[]
```

Figure 4.1: Mobility Index values in different time windows (left) and fragment of duration values (in seconds) over which mobility index has been calculated (right) - window of 30 minutes

At the end, mobility index, as defined in subsection 2.1.3, is being calculated over the assigned values - Figure **??**, and assigning mobility index to each of the bins - Figure **??**.

4.2.2 Algorithm for stop detection

TODO

4.3 Clustering algorithm

For clustering we started off with DBScan, described in **??**. The difficulty with DBScan is figuring out the two parameters, the minimum points per cluster (*minPts*) and the radius of the clusters (*eps*).

4.3.1 DBSCAN on Spark

We wanted a scalable implementation of DBSCAN that could handle large data sets, preferable in parallel. We decided to use Apache Spark, a scalable engine for large-scale data processing [**spark**]. For this we used an implementation by Irving Cordova, based upon an algorithm called MR-DBSCAN built for the MapReduce framework [**dbscan_on_spark**]. The implementation of DBScan runs in parallel by splitting the data space into boxes, using the number of boxes as a cost estimator for the algorithm. Each box then grows to include one *eps* in it. After each box and its points has been determined the traditional DBScan algorithm is run on the points in each box. Finally it examines the intersection points betweeen boxes and merges the result together [**vis_dbscan_on_spark**]. The figures below visualises the execution steps.

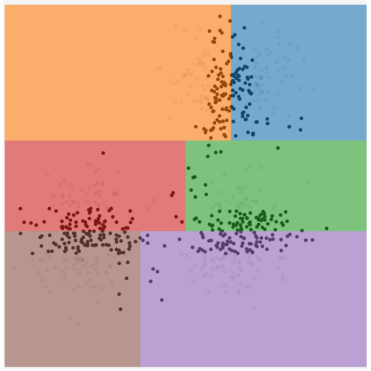


Figure 4.2: Step 1. DBSCAN on Spark assign the data space into boxes.

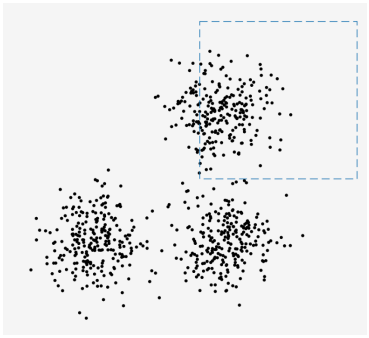


Figure 4.3: Step 2. Each box grows to include the points that are within one *eps* of it.

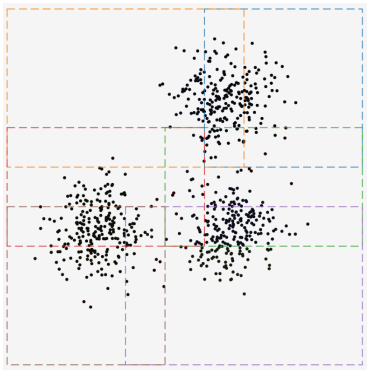


Figure 4.4: Step 3. Traditional DBScan algorithm is applied in parallel for each box. Each different color represents a different cluster.

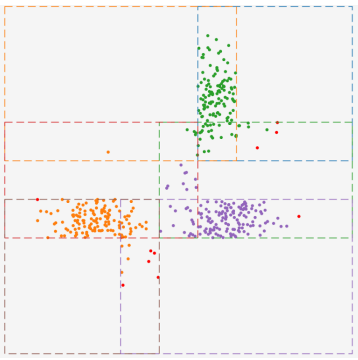


Figure 4.5: Step 4. After DBSCAN is done, all points witihin the borders of two clusters are examnied. If they are part of a cluster within two boxes they are merged to one cluster.

Figure 4.6: Step 5. Finally all remaning points are labeled with the global identified cluster and we are done. (red points are noise points).

5 Evaluation

5.1 Location data characteristics for Berlin area

5.1.1 General overview

Movement triggered data from mobile devices are spanned across the Germany as shown on Figure 5.1.

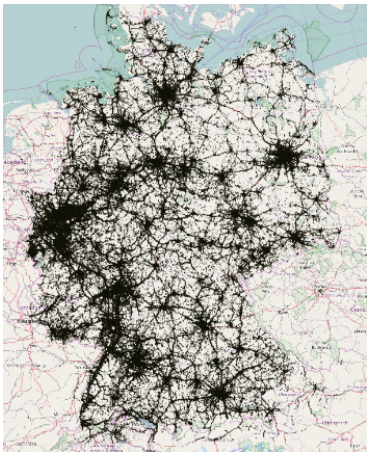


Figure 5.1: Visualisation of data points location in the geographical map of Germany

It points out, that most of the detected movements from one point to another are mostly gathered on the highways and inside the cities.

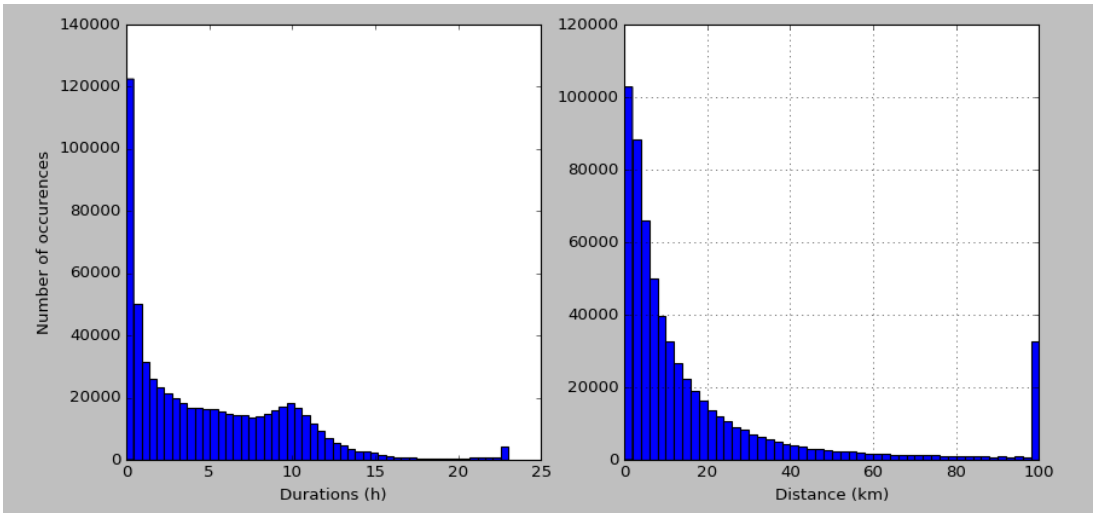


Figure 5.2: Histograms of durations and distances between consecutive points under and cumulatively over 100 km for the area of Germany

The given data batch has been gathered in the interval of 24h starting from 3am in the morning German time. It consists of over 675000 data points, belonging to 47300 unique mobile devices. Figure 5.2 shows, that for unique mobile devices, the durations between consecutive points in time are most frequent within 30 minutes and most of the data is in the interval 1-10h. Regarding distances, consecutive points are most frequent below 2 km and most data is in the interval 0-10km. Big chunk of distances is also over 100km.

Figure 5.3: Heatmap of data point densities within the area of Berlin

Figure 5.3 shows the heatmap of data points for Berlin area. We can see that most of the movements detected are on major train/tram/metro stations, highways and major living areas.

5.1.2 Data analysis

Statistics performed using tool written in Python (https://github.com/mrow4a/macroscopic-movements-algorithm-prototypes) shows, that for users at least once visiting Berlin, had in average 17 points gathered because of their movements in the period of 24h, harmonic mean of 4 points and maximum 100 of points.

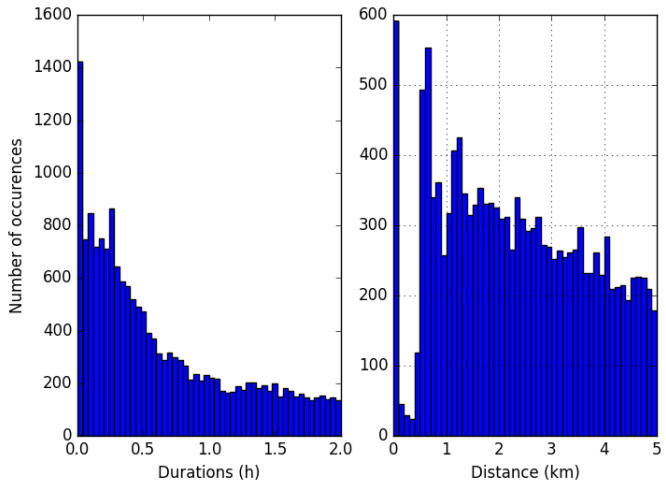


Figure 5.4: Histograms of durations and distances between consecutive points under 5 km and under durations of 2h for the area of Berlin

Furthermore, majority of durations between points is within interval of 30 minutes. For distances between points, there is significant "jump" at 500-1000m and 1100-1500m, which might mean that at these intervals, continuous points been gathered, and distances over that values might be discontinuous (ref. Figure 1.1). It is due to the fact that these distance intervals are most frequent and that might suggest that this is an average "update" distance for the moving mobile devices, as described in section 1.2. Less frequent values might suggest that updates were obtained with lower accuracy (e.g. by presence inside the building, metro line or simply mobile device lag in obtaining its location) and are result of discontinuity between consecutive updates.

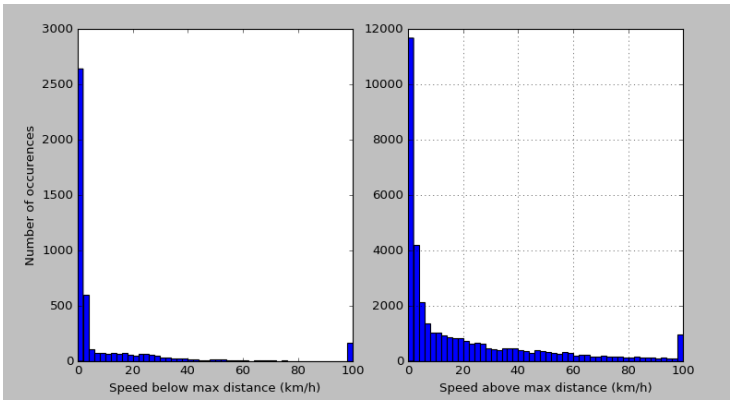


Figure 5.5: Histogram of speed between points with corresponding distance above or below 1.5 km

Figure 5.5 shows that considering the registered distances within range of 1500m, the wast majority of points are between 0-2 km/h, and also significant interval at 2-6 km/h. The rest of the values is sparse distributed in interval 6-100+ km/h. In case of registered point above range of 1500m, we observe that indeed most frequent occurrence is at interval of 0-4 km/h, however most are sparse distributed above 4 km/h, with most points being in interval of 4-20

km/h.

5.2 Pre-filtering of anomalies

Figure 5.4 shows that there is a fraction of points which duration or distance rapidly changed, thus resulting in very high speeds between the points. Furthermore, due to the methodology of obtaining points (movement triggered), there are some minimum distances and durations at which points can be collected, and values not matching stop or travel expectation according to gathered statistics, have to be filtered. Thus, the following predicates for filtered values have been set:

Jumps Points with speed over 83 m/s [300 km/h]

Duplicates Points with distance below 100m and at the same time duration below 10s

5.3 Evaluation of stop detection algorithms

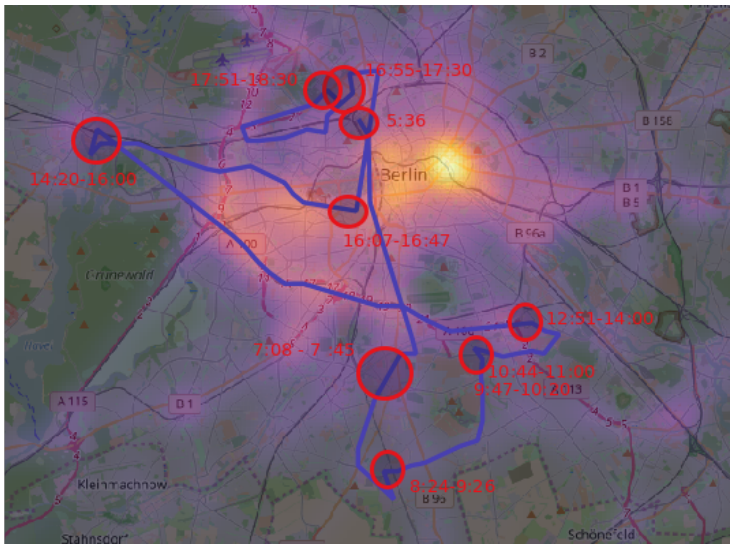


Figure 5.6: Visualization of distances, speed and mobility indexes for example from Figure 5.7 with identified stop candidates

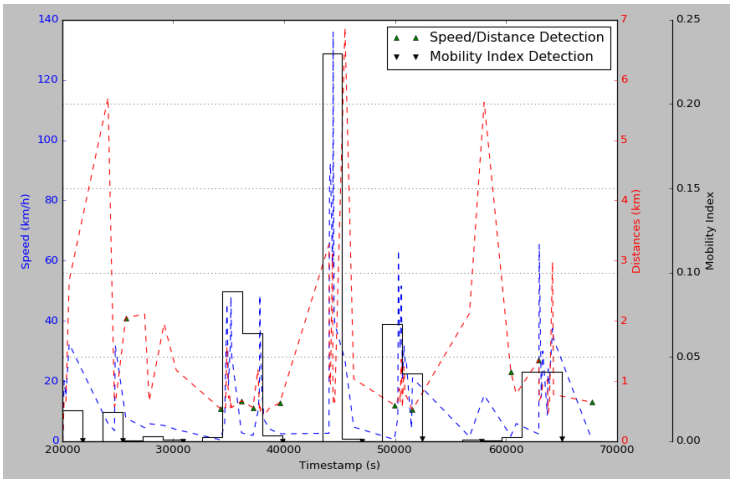


Figure 5.7: Movement example for one user in the period of 24h, with recorded locations due to the movement and annotated timestamps for detected stop candidates

5.3.1 Mobility Index approach with varying time window

TODO

5.3.2 Mobility Index approach with varying threshold

TODO

5.3.3 Human Mobility approach for long/short distances

TODO

5.3.4 Effectiveness of approaches

TODO

5.4 Clustering algorithm

In this section we will describe how we evaluated our dbscan algorithm and the choice of the *epsilon* and *minimum points per cluster* parameter.

5.4.1 DBSCAN

We visualised our results in QGIS, a powerful graph visualisation tool. Running the dbscan with different parameters resulted in different cluster sizes and number of clusters. We started off with the target area of Berlin. The *epsilon* parameter (radius of the cluster) was chosen by looking at the graininess (accuracy) of our given data set. The accuracy between points was

110 meters, this distance is refered to as *accuracy distance*. We took Alexanderplatz in Berlin as an example cluster (a popular train stop area in Berlin). We interpret one cluster as being one stop point of interest, and for this application want Alexanderplatz to be represented as one cluster. Setting the *epsilon* parameter to the *accuracy distance*, 110 meters, gave us good looking clusters whilst higher value of the parameter resulted in clusters being unnaturally large and distance below the *accuracy distance* resulted in only single-point clusters (points are stacked at the same location). Note that the *minPts*, minimum points per cluster, parameter is not taken into *consideration* here (it is set arbitary and only determines the number of clusters, while we here are interested in the *size* of the clusters). See figures below.

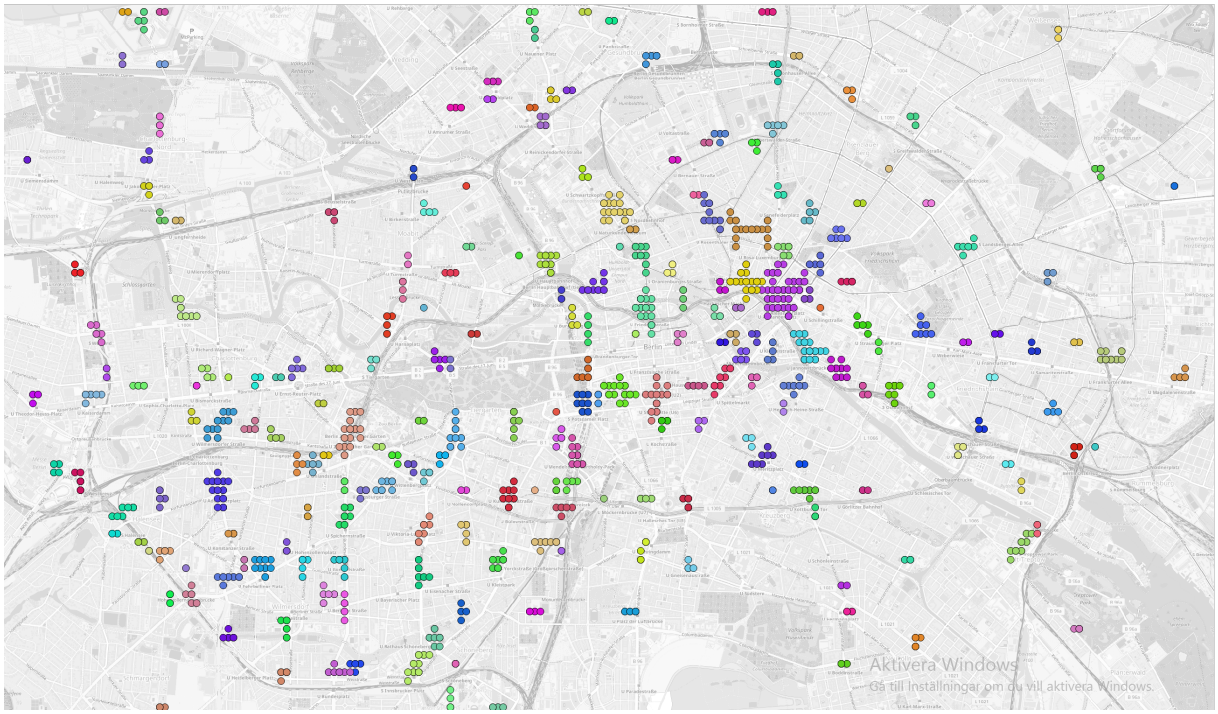


Figure 5.8: Clusters with good parameters *epsilon* = 110 meter, *minPts* = 5. Alexanderplatz (pink) has 85 data points in it.

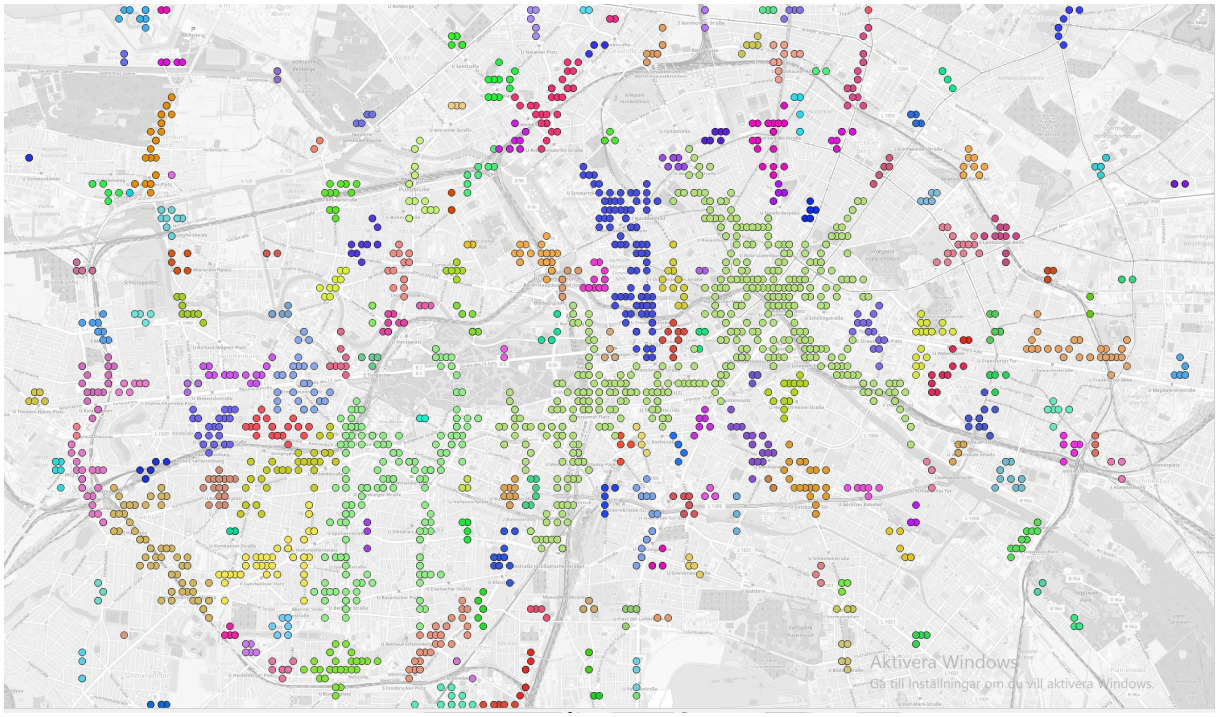


Figure 5.9: Example of a clusters with bad parameters, here *epsilon* = 220 meter is too large, *minPts* = 5. "Alexanderplatz" (light green) has 748 data points in it and stretches over the whole centre (Mitte) of Berlin.

Appendices

Appendix 1

Figure 1: Visualization of distances, speed and mobility indexes for example from Figure 2 with identified stop candidates

Figure 2: Movement example for one user in the period of 24h, with recorded locations due to the movement and annotated timestamps for detected stop candidates

Appendix 2

Figure 1: Visualization of distances, speed and mobility indexes for example from Figure 2 with identified stop candidates

Figure 2: Movement example for one user in the period of 24h, with recorded locations due to the movement and annotated timestamps for detected stop candidates