Map matching tracking data

Kristof Ghys

promotor :
Prof. dr. Bart KUIJPERS

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica databases

universiteit hasselt

Dankwoord

Deze thesis is niet tot stand kunnen komen zonder een aantal mensen. Allereerst zou ik graag promotor Prof. dr. Bart Kuijpers en begeleider Bart Moelans willen danken voor de begeleiding doorheen het ganse jaar. Ook bedank ik Leen De Temmerman van de universiteit van Gent voor de vlotte samenwerking en voor het verschaffen van de noodzakelijke gegevens. Ook Prof. dr. Philippe Demaeyer en Prof. dr. Nico Van de Weghe van de universiteit van Gent had ik graag willen danken voor de gastvrijheid die ik heb mogen genieten toen ik in Gent was. Verder wil ik ook mijn vriendin Vera bedanken voor haar steun tijdens het ganse jaar! Uiteraard ook mijn ouders voor de steun die ze me hebben gegeven. En natuurlijk kunnen ook mijn medestudenten niet vergeten worden voor de onvergetelijke 4 jaren die we samen hebben doorgebracht.
Dus bij deze bedankt allemaal!

Kristof Ghys

Abstract

Route planners become more and more available for everyone. These intelligent devices are very popular and are very easy in use. The police of Ghent had a problem with new and transferred cops. They did not know the environment, so it would be better if they got a route planner in their car which showed the right way. The problem is that the ordinary route planners are not able to fulfill all the wishes for law enforcement. They do not take into account for instance where the school gate is situated and to avoid this area when they are on an intervention around school time if possible. This thesis tries to make a custom-made route planner for police officers. This thesis is about mapping the GPS points got from real-life trajectories of experienced police officers mapped to the real network topology. The analysis of these trajectories afterwards is very interesting and will be done in further work.

How do they get to the place of intervention? Do they take many risks by driving too fast or are they careful enough and do not drive too fast at all. We are trying to make a custom-made route planner for these officers especially for new or transferred cops who do not know the environment yet. We did this by analysing real trajectories from police officers on patrol. A first step is to record these real-life trajectories. Then we have these trajectories we can map to the real network topology, analyze it and finally we can make a custom-made route planner. In this work we try to map the GPS points recorded to the real network topology. We try a relatively new technique called beads. This is an estimate where the police car could have been between two succeeding points knowing that they do not exceed some maximum speed and knowing the time between both points.

Our recordings are pretty accurate in theory (every 10 meters a GPS point) but in practice this is of course not always the case. Sometimes there are even gaps of more than 400 meters. Most algorithms found in literature are based on real-time mapping and they have points time-based recorded for instance every 5 seconds a point. These are the biggest differences with our data. The fact that we have every 10 meters a point recorded is in our advantage because then our beads algorithm performs best. It can then better calculate where it could have been between two succeeding points, than if it was time-based recorded.

Dutch Summary - Nederlandstalige samenvatting

De politie van Gent had een probleem met nieuwe of overgeplaatste politieagenten. Deze agenten hebben namelijk nog geen ervaring in een stad als Gent en kennen bijgevolg de weg niet. Hierdoor duurt het natuurlijk langer dan ervaren agenten om naar de interventie te gaan en dit moet verbeterd worden voor een betere dienstverlening. Daarom besliste de politie van Gent om de universiteit van Gent hierover aan te spreken en vroeg hen om een oplossing. Omdat de universiteit van Gent samenwerkt in een Vlaams project *"Knowledge representation and database problems for spatiotemporal data: a calculus approach to representing knowledge about trajectories."* en omdat de universiteit Hasselt meer ervaring heeft op het gebied van de reconstructie en analyse van trajecten, alsook met het behandelen van onzekere of partiële informatie, werd dit probleem aan de UHasselt overgemaakt. Het hoofddoel van deze thesis is om een op maat gemaakte routeplanner voor agenten te ontwikkelen. Deze thesis is een eerste deel van vele waarin we de verworven data omzetten naar de 'echte' wegen topologie. Toekomstig werk zal vooral inhouden de trajecten te analyseren en tot slot dan een routeplanner voor de agenten te maken.

De universiteit van Gent vergaarde real-life trajecten van de agenten door hun een GPS toestel, dat enkel GPS punten opnam mee te geven op patrouille. Hierdoor krijgen we een beeld van hoe de agenten zich doorheen de stad bewegen en hoelang ze erover doen. Dit geeft ons een goed zicht op bepaalde feiten zoals bv. het vermijden van een schoolomgeving, het vermijden van zone 30 omgevingen en de voorkeur geven aan hoofdwegen. Dit zijn zeer interessante feiten om in rekening te brengen wanneer we een op maat gemaakte routeplanner willen ontwikkelen voor agenten. Zij zijn immers (als het om een hoog prioritaire interventie gaat) minder gebonden aan verkeersregels.

We stellen 2 technieken voor om de eigenlijke mapping van GPS coördinaten naar de eigenlijke topologie van het wegennetwerk te gaan. Eentje is een naïeve oplossing terwijl de andere een meer doordachte en betere oplossing is. De naïeve methode werkt als volgt. We kijken voor elk GPS punt welke straten er het dichtste bij liggen (5 dichtste) en deze straten behouden we, dan hebben we achteraf een klein netwerk waarop we een kortste pad algoritme zoals bv. A* [46] toepassen . De meer doordachte manier van werken is er eentje die gebruikt maakt van de techniek van beads. Met deze techniek berekenen we een gebied waarin de auto aanwezig zou kunnen geweest zijn in de tijd tussen punt 1 en punt 2. Dit doen we door een bepaalde maximumsnelheid vast te stellen (bv. 120 km/uur in bebouwde kom, wat realistisch is omdat deze snelheid waarschijnlijk nooit gehaald wordt in bebouwde kom en dus dat alle straten in dit gebied liggen) en dan de straten

te berekenen die in dit gebied liggen. Hier krijgen we dan ook een verkleind netwerk waarop we in plaats van **het** kortste pad, ***k***-kortste paden berekenen. Maar voor we de k-kortste paden berekenen gaan we aan de straten een gewicht toekennen. De gewichten worden toegekend door elk GPS punt te beschouwen en dan krijgt de dichtstbijzijnde straat gewicht n, de straat die het 2^{de} dichtste erbij ligt krijgt gewicht $n-1$ en zo verder tot de n^{de} straat gewicht 1 krijgt. Uiteraard kan men n zelf kiezen, maar wij kozen voor $n = 5$.

Een vergelijking van beide algoritmen is zeer interessant. De naïve oplossing, die op het eerste zicht geen enkele kans op slagen heeft blijkt achteraf toch een goede indruk te geven van de genomen route. De data die we hebben is vrij precies desondanks het feit dat er soms gaten in optreden. De naïeve oplossing is dikwijls evengoed als de beads oplossing, maar de naïeve oplossing is veel sneller (tot 5 keer gemiddeld en soms zelfs tot 10 keer sneller). Wanneer er echter gaten optreden geeft de naïeve methode geen oplossing terwijl de beads methode een juiste oplossing geeft. Waar beide algoritmen het moeilijk mee hebben zijn loops in de data. Dit is omdat we ervan uitgegaan zijn dat loops niet zouden voorkomen en dit dus zo geprogrammeerd is. Na grondige analyse van de beschikbare data blijkt echter dat loops af en toe wel degelijk voorkomen, vooral op het einde van het traject wanneer de agenten moeten zoeken naar het juiste huisnummer of de juiste straat. We kunnen dus concluderen dat desondanks dat de naïeve oplossing veel sneller is, ze niet altijd een juist resultaat geven. Daarentegen geeft de beads methode wel altijd een oplossing en een juiste maar duurt dit langer. Door het feit dat we de berekeningen niet in real-time moeten uitvoeren, is het niet erg dat de berekening langer duurt.

Zoals reeds gezegd houdt het algoritme geen rekening met loops. Dit moet nog ingebouwd worden. De analyse van deze data moet ook nog gebeuren om dan tot een goede op maat gemaakte routeplanner te komen. Verder kan er ook nog uitgezocht worden in hoeverre mate Google Earth een alternatief is om de routes te bepalen. Google Earth maakt van de opeenvolgende GPS punten ook al een mooie opeenvolging die op eerste zicht perfect op het stratennetwerk ligt, maar waarschijnlijk zal de output hiervan geen informatie bevatten over de straten die zijn genomen en dus zal dit onbruikbaar zijn om verder te gebruiken. Het zal wel bruikbaar zijn om de data te visualiseren. Wanneer de GPS systemen zullen in gebruik genomen kan het een mogelijkheid zijn om op het hoofdkwartier een real-time verbinding te houden met de patrouilles, zo kan er dan ook zeer snel gereageerd worden op de interventies en kan de dichtstbijzijnde patrouille gestuurd worden. Er kan ook voor gekozen worden om ook de files beter in kaart te brengen zodat ook daar beter op gereageerd kan worden. De mogelijkheden zijn enorm en dit zijn slechts enkele toepassingen die realiseerbaar kunnen zijn wanneer er een full-time link is met de patrouilles.

Contents

1 Introduction **3**
- 1.1 Problem description . . . 7
- 1.2 Map-Matching . . . 8
- 1.3 Available data . . . 11
  - 1.3.1 The real (imperfect) GPS data . . . 11
  - 1.3.2 How much data . . . 11
  - 1.3.3 Survey . . . 12
  - 1.3.4 Privacy and Security . . . 14
- 1.4 How the problem was handled . . . 15

2 Mapping GPS points to road segments **17**
- 2.1 Technical specifications of GPS devices . . . 17
- 2.2 Mapping data to road network . . . 20
  - 2.2.1 Input . . . 20
  - 2.2.2 Output . . . 21
  - 2.2.3 Naive . . . 22
  - 2.2.4 Limiting the scope . . . 23
  - 2.2.5 Special Cases . . . 26
- 2.3 More optimal algorithms . . . 31
  - 2.3.1 Score calculating algorithm . . . 31
  - 2.3.2 k-shortest paths . . . 33
- 2.4 Comparison of the used algorithms . . . 37
  - 2.4.1 Naive algorithm . . . 37
  - 2.4.2 Beads combined with k-shortest path . . . 37
  - 2.4.3 Results . . . 37
  - 2.4.4 Conclusion . . . 38
- 2.5 Future work . . . 41
  - 2.5.1 Implementation improvements . . . 41

3 Implementation **43**
- 3.1 Used Hard/Software . . . 43
  - 3.1.1 Hardware . . . 43
  - 3.1.2 Software . . . 43

3.2 Package structure 44
3.3 Grafical User Interface 48
3.4 External libraries 48
3.5 External Programs 49
3.6 Details 50
3.6.1 Overall view 50
3.6.2 ArcExplorer 50
3.7 Database structure 51
3.7.1 Tables 51

4 Problems with solution **52**

A Survey **60**

B GUI **63**

Chapter 1

Introduction

Nowadays, we use databases everywhere, from the local shop in our neighborhood to the big multinationals who store the information about products in it. Today we see that there is a huge growth in data from local-awareness devices. The most known localation aware device is of course the GPS device. GPS devices are relatively cheap to buy and therefore many people have one. The main advantage of these devices is that they can show the route to everywhere from your current location. Even if you miss a street the device will automatically recalculate the best route to your destination within a few seconds. That is why more and more households use a GPS device for navigation purposes.

Most people use the GPS device as a navigational tool but it can also be used for storing the GPS coordinates for analysis afterwards. We can for instance analyse the routes taken by a person and then we can see why they took one road instead of another road that seemed better. The main disadvantage of storing the GPS coordinates is that the GPS coordinates are not perfect. They do not match the road. Every GPS device takes these 'mistakes' into account by mapping the exact street driven on instead of just modelling the GPS coordinates got from the satellite. There are many algorithms described for use within the GPS devices like [45].

Since the beginning of times, people have observed moving objects, from insects to stars and planets and wondered how they moved. When we examine todays movements we have different ideas then in the past but there is still much to learn from those past ideas for movement. One of the most famous movements written down was this of Napoleons march to Moscow by Charles Joseph Minard (see Figure 1.1). In Figure 1.1 we see the march of Napoleons army to Moscow and how for instance the weather influenced the casualties in his army. There are even more aspects to be found in this picture (6 in total) but we refer you to [23] or [48] if you want to know more about it.

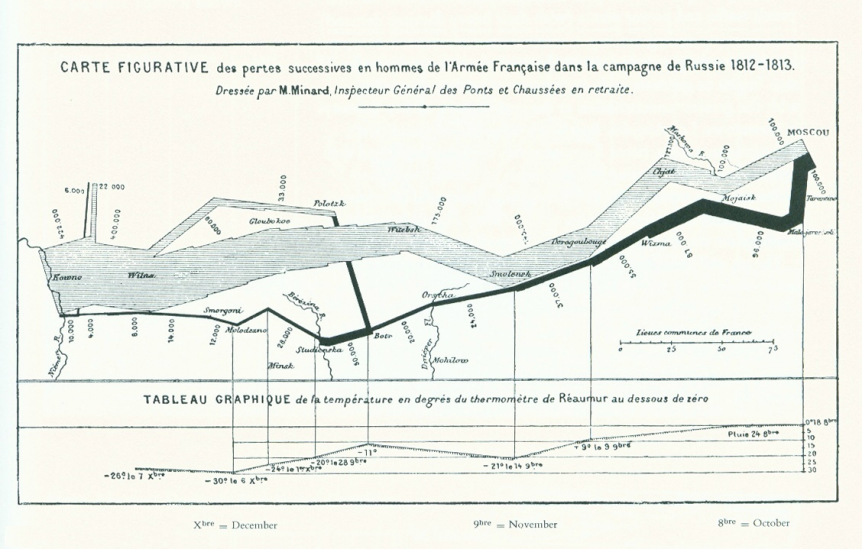


Figure 1.1: Charles Minard's "Carte figurative des pertes successives en hommes de l'Armee Française dans la campagne de Russie 1812-1813" from 1861

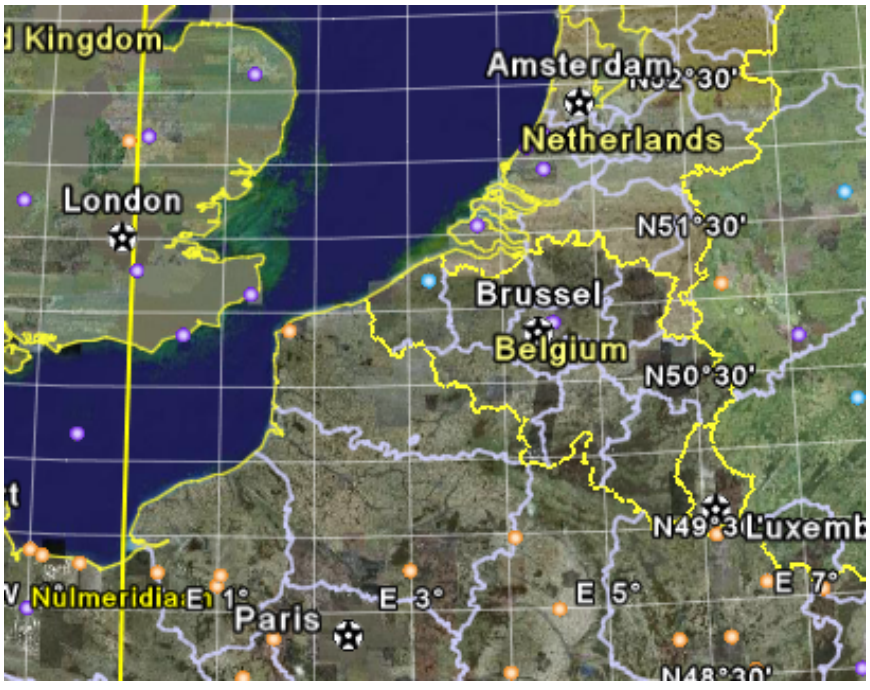


Figure 1.2: Reference system in Latitude/Longitude

Before we can go on, we define some of the most important words in the domain of Moving Object Databases (MODs). A strict definition of *movement* relates this concept to change in the physical position of an entity with respect to some reference system within which one can access positions. Most frequently, the reference system is in a geographical space (see Figure 1.2). Furthermore a *trajectory* is the path made by the moving entity through the space where it moves. For creating or following a path we need a certain amount of time. Therefore, time is an inseparable aspect of a trajectory. This is emphasised in the term 'space-time path' [28], one of the synonyms for 'trajectory'. When these trajectories are stored in a database we speak of moving objects databases. The area of MOD has become more interesting in the past years. In this domain and especially in trajectory databases, we try to extend the database technology to support the representation and querying of moving objects and their trajectory. Researchers in the field of MOD have been studied the representation issues of trajectories into computer systems aiming at keeping track of object locations, as well as supporting location-aware queries [49]. If, only time-dependent locations need to be managed (e.g., mobile phone users, cars, ships, etc.), then *moving point* is the basic abstraction; while, if the time-dependent shape or extent is also of interest (e.g. group of people, spread of vegetation), then we are talking about *moving regions*.

A straightforward approach widely used is to model a moving point by generating periodically a location-time point of the form (l, t), indicating that the object is at location l at time t where l may be a coordinate pair (x, y) (see Figure 1.3). The point is stored in a database, and the database query language is used to retrieve the location information. This method is called *point-location management*, and it has several critical drawbacks, such as (1) it does not enable interpolation or extrapolation, (2) it leads to a critical precision/resource trade-off, and (3) it leads to inefficient software development.

For practical reasons, however, trajectories have to be represented by finite sequences of time-referenced locations. Such sequences may result from various ways used to observe movements and collect movement data:

- Time-based recording: positions of entities are recorded at regularly spaced time moments, e.g. every 5 minutes;
- Change-based recording: a record is made when the position of an entity differs from the previous one;
- Location-based recording: records are made when an entity comes close to specific locations, e.g. where sensors are installed;
- Event-based recording: positions and times are recorded when certain events occur, in particular, activities performed by the moving entity (e.g. calling by a mobile phone);

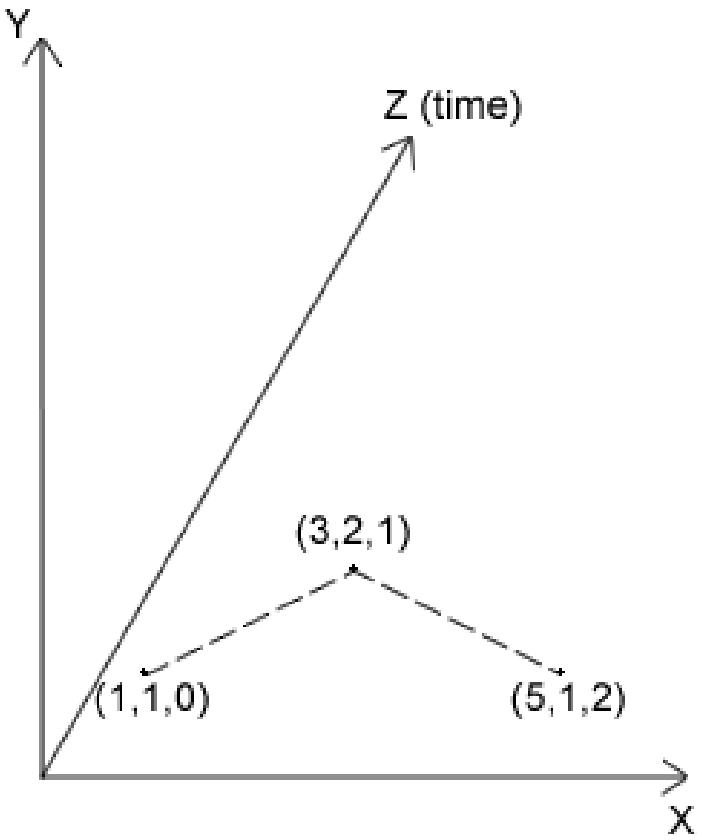


Figure 1.3: Point location based recording.

- Various combinations of these basic approaches.

For this thesis we work with GPS coordinates. They are stored using the point-location management method, so we have the triple (x, y, t) with (x, y) the location l and t the time. Another aspect of the data we got is that it is change-based recorded, we have every 10 meters a point recorded. Of course this is not always the case (for instance there are gaps as described in Section 2.2.5) but most of the time the 10 meters is respected. Take note of the fact that there is a point recorded every 10 **meters** instead of using **time** based recordings like often described in other papers (see Section 1.2).

The aim of this thesis is comparable to the knowledge written down by Minard (see Figure 1.1). We too want to track and know the movements of the police cars (see Section 1.1) for analysis afterwards. The purpose is knowledge discovery about the routes taken by the experienced police officers. Why did they take that particular route and not another shortest path? We hope we can answer this and many more questions afterwards.

For doing data analysis there is no straight-forward definition in handbooks. But most people involved in data analysis agree in following an iterative process [30]:

1. formulate questions;
2. choose analysis methods;
3. prepare the data for application of the methods;
4. apply the methods to the data;
5. interpret and evaluate the results obtained.

There is need for semantics about the data, and especially need for certainty about the trajectories driven on. For example we need the exact position where they drove because then we can see whether or not they passed the school gate and not just drove in the environment around the school. It is not obvious to see where the school gate lays, but it is easier to see where there is a school, it is because of this reason we need the semantics about certain locations and about the trajectories.

In this thesis we especially prepare the data for application because we do not have perfect data (see Section 1.3.1). We examine the behaviour, the configuration of characteristics corresponding to a given reference (sub)set, of traject. A trajectory of a single entity is a configuration of locations corresponding to a time interval. One of the main goal of the analysis of movement data is to characterize the overall movement behaviour of all the trajectories of these entities over the entire time period the data is consisted of, in other words to build a pattern representing the overall behaviour of the data. And furthermore concerning the movement of entities, we may be interested to know whether and how the movement is related to various spatial, temporal, and spatiotemporal phenomena such as weather, traffic jams, incidents, schools,...

1.1 Problem description

The police of Ghent had a problem with new or transferred police officers. As in every large city there are a lot of new police officers starting in the city. They do not know the city very well and because they do not know the environment yet, it takes them longer than experienced police officers to go from one place in the city to the place of intervention. You could solve this problem by putting an experienced police officer with an inexperienced cop, but there are not enough experienced police officers to put with the new officers. It would be better if we could develop a route planner specialized for the police so that the new cops too can quickly respond to an intervention and navigate quickly through the city at any given time at any given period.

One of the obvious questions is of course whether or not we should do all this research instead of buying a commercial route planner. The solution is simple, the shortest/fastest route (calculated by these commercial route planners) is **not** the best solution for the police because of several factors:

- They do **not** take into account the hour of the day, e.g. at 5 o'clock there is always a traffic jam at the station, so we avoid this area around 5 o'clock;
- They do **not** take into account certain locations (like schools), e.g. In the area around schools there may be a lot of unpredictable children for who we need to be more careful;
- They do **not** take into account extra information (like school routes [34]), e.g. tram lines embedded in the road where the police can drive on, for easier traffic flow;
- They do **not** take into account (local) traffic jams.

So these route planners are not flexible enough to deal with life threatening situations where even the smallest delay because of the software could cause serious damage and perhaps even death.
The police asked the university of Ghent (UGent) for some help with this problem but because they worked together with the university of Hasselt (UHasselt) in a Flemish project *"Knowledge representation and database problems for spatiotemporal data: a calculus approach to representing knowledge about trajectories."*[18] and because UHasselt had more experience in the field of reconstruction and analysis of trajectories and with the uncertainty of partial information, the problem was handed to the UHasselt.

As already described in Section 1 semantics are really important. It is for instance important to know if there is a school in the neighborhood but it is even more important where the school gate lays, at which road because it will be more dangerous to pass here than to pass the other side of the school.

Most of the data we got from the GPS devices is not perfect (see Section 1.3.1). There already has been done a lot of research in the area of map matching which we describe in Section 1.2 but most of the map-matching algorithms are real-time based because they are implemented for route planning within the GPS devices self and not for offline analysis afterwards like we need to do.

1.2 Map-Matching

The technique of mapping GPS coordinates to the road network itself is called map-matching. Here we describe some related work that already has been done.

In [3] three algorithms for map-matching are discussed. Here they too consider the real trajectory instead of matching algorithms that use current positions. Their data was consisted of samples every 30 seconds which means that the car could have travelled a distance of 417 meters before a next point is recorded! This is a big difference between their data set and ours (we have every 10 meters a point). They discuss an incremental map-matching algorithm which sequentially matches the next possible road segment and which tries to make a global-optimal solution by introducing a look-a-head. They tried to match the trajectory with the Fréchet distance and the weak Fréchet distance but they both had the same outcome, both were equally good. They were compared in running time and in quality of the results. The incremental approach was much faster but produced worse results.

In [45] a method for real-time map-matching is used instead of afterwards map-matching. This method is especially used within the GPS devices. Because the GPS signals have a systematic error of < 100 meters, thanks to the 'selective availability' imposed by the U.S. Military, digital mapping data is used. The system tracks a vehicle on all possible roads within a error region and then instantly computes which roads may be invalid. This method is called Road Reduction Filter (RRF) algorithm.

In [26] they analysed the GPS coordinates recorded from an ambulance. They got a new recorded point 'every 500 meters or 5 minutes' or 'every 333 meters or 15 minutes'. Their recorded GPS coordinates consisted out of a tuple with information about: position, time, velocity (given as speed and heading) and distance travelled since last point recorded. The velocity information is redundant but if we need to calculate it afterwards it takes more CPU time then when stored directly in the database. Because the GPS device recorded GPS points all the time the first job was to mark the point where they got a call from headquarters and mark the ending points. Then they define a network position out of the GPS coordinates that lays on edges for better handling these points. In the last step they try to compute a trajectory by choosing a sequence of positions from the candidate network positions calculated in the previous step. They do not say how well they have done it, and they conclude that they are currently improving the run times.

In [11] they wanted to identify the route taken by a traveller. They did not have information about the time travelled so far or between succeeding points and they did not have the travel speed. Their GPS coordinates are recorded every second. The main purpose of the article is to find the path that is closest to the GPS coordinates in minimum amount of computations. They calculate a score for the entire path by using the distance between the GPS points and the edges in the network. They did not state how they handled the gaps in the data, they just calculated the routes from which they had the GPS points and then tried to glue it together, if the distance between 2 succeeding points was to big they stopped with this path and tried

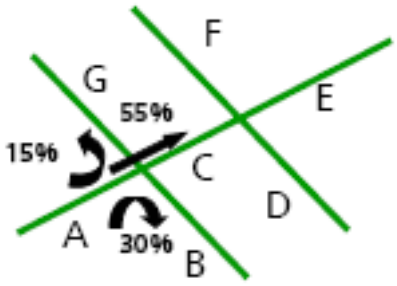


Figure 1.4: Probabilistic model (A → F are street segments)

it again with a fresh start. At every intersection they try the new calculated routes made by adding the routes of the intersection to this path. They concluded that their algorithm was very good of handling the GPS points but never stated what was exactly the accuracy of the GPS data recorded with the GPS devices.

Research has already been done in the field of mobile phone data converting to the street network. In [13] the idea was to introduce a new pricing system for poster sites according to mobile phone data. They used a probabilistic model for reconstructing the trajectories. They had prior knowledge about high-frequented streets. Therefore these streets were more likely to be walked on than less frequented streets. For track-to-street mapping they use the Hidden Markov Model [41], this is a model which finds the most likely sequence of states from observations and transition probabilities. They assign to each point in space a probability to belong to one (or more) street segments (see Figure 1.4). Then they combine the segments to the street network by calculating the sum probabilities of point over all the segments. Their main problem was to acquire mobile phone data and had to stop the project because they could not acquire the necessary data.

Many of these described ideas are for real-time mapping and not for mapping GPS coordinates afterwards. However some of the ideas could be used, for instance the look-ahead of the neighbour streets.

1.3 Available data

UGent provided us with the data for analysis. The dataset consists out of two parts: one with the recorded GPS points and one with extra information about the routes taken by the officers by letting them fill in a survey afterwards with questions like why they had taken that route, and whether or not it was out of experience (see Section 1.3.3). When we examine the surveys we see that not all surveys are filled in completely or with good info, e.g. reason was: 'according to me shortest', this was expected because the police have already enough paperwork and do not always want to fill in yet another paper. However some of the answer were very useful: 'no traffic lights on the road' is a very interessenting thing to know. You find the survey in appendix A. We wrote a program for quickly digitalizing these surveys on a computer (see appendix B) afterwards.

The data recorded by the experienced officers in the car where asked to start recording with the GPS device from the moment they got a call from headquarters and to stop recording when they arrived at their intervention. Afterwards they were asked to fill in the survey. Of course with the recorded data, you see obvious things like when they arrived they forgot to put the device out and half an hour later the device recorded one last point.

1.3.1 The real (imperfect) GPS data

UGent have chosen the GPS devices for the retrieval of GPS coordinates. They chose for a GPS device that only tracks a route and not gives a route (see Section 2.1). The device only records the route taken by storing every 10 meters a position (waypointing a route) (change-based recording). Normally this should work perfect but since the points recorded are not always on the road taken (95% is off-road). We need an algorithm for the mapping to the actual road taken. Another problem from the retrieval of GPS coordinates was the problem that there were gaps in the data. There was not always recorded a point every 10 meters, sometimes there even was a gap of more than 100 meters.

1.3.2 How much data

An algorithm can only be tested when we have enough data. We got from UGent 527 routes, taken over 3 months. The routes consist out of 94 GPS points at average which means we have nearly 50000 (49538) GPS points recorded. There are 388 routes which have less than 10 GPS points, so most likely these routes are not representative. 97 routes have more than 100 GPS points and 11 of these routes have even more of 1000 GPS points. There are even routes of 2200 GPS points or even 3400 GPS points. Most likely the users (police officers) did not turn off their device as requested.

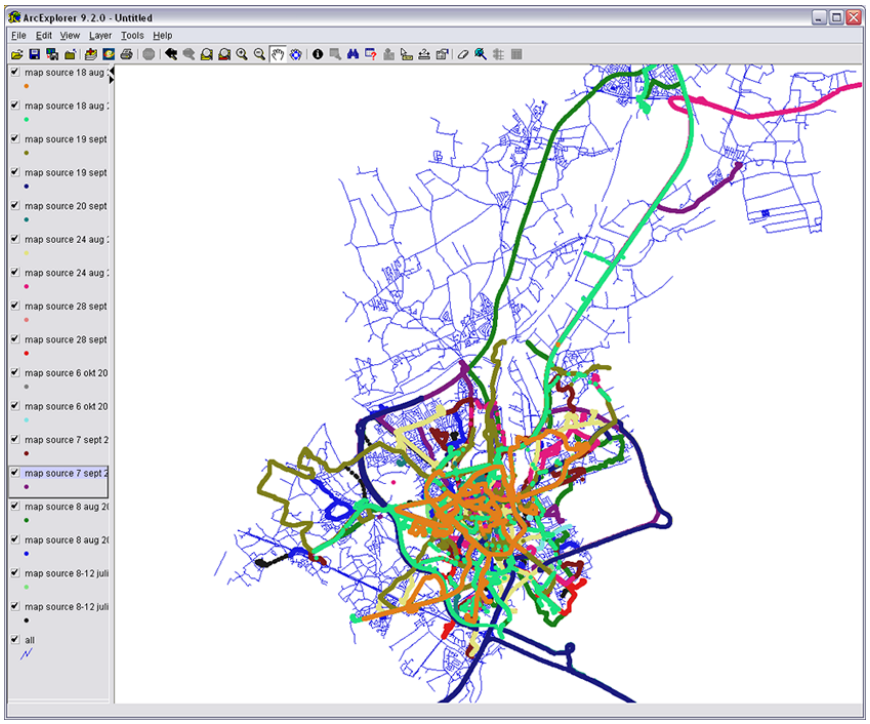


Figure 1.5: All trajectories recorded on one map

Another interesting thing to remark is that only 122 of the routes start at headquarters. In Figure 1.5 all trajectories are modelled.

1.3.3 Survey

The second dataset we can refer to is the survey afterwards filled in by the police officers. In this survey we hope they tell us why they have driven the way they did. The questions they had to answer were the following and we explain why these questions were important to know:

1. Number of device: there were 2 devices in use for quicker acquiring the data;

2. Start and end positions: we need to know from where to where they have driven, as detailed as possible so therefore we asked for the house number if they knew it;

3. Time of departure: this is especially interesting because most likely the route at 9 am is different from the route at 8 pm (e.g. because of less traffic);

4. Date: a sunday, weekday, saturday, these are all very interesting data

to collect (e.g. a great event happened on a particular date and therefore some roads were closed);

5. High priority: when the intervention is of high priority they can skip traffic lights, drive faster than when the intervention is of low priority;
6. Weather circumstances: are there roads they avoid when it has rained because they are more slippy than other roads;
7. State of road: Did they drive slower on roads which are in bad condition;
8. How was the route selected: Did they use any aid like a own GPS device, did they had to drive because they got directions from headquarters, or other;
9. Why route chosen: This is probably the most important question, why did they drive the way they did;
10. Special circumstances: Did they encountered problems, like road blocks, double-parked cars in a one-way street;
11. How well did they know the route: If they knew the route very well the trajectory could be more important because they knew how to drive to for instance avoid traffic jams than if they did not knew the environment;
12. Remarks: At the end they could leave their remarks.

When looked at some of the surveys there are some interesting things to remark. Almost every intervention was not a priority, this means that they could not take much risks and could not discard the traffic lights. Another interesting point to mark is that in the choice of the route taken they frequently said that the route they had chosen was the shortest or easiest to take. At the question if there were some special circumstances most of them did not answer this question but the few that did, stated that some roads were blocked because they were working on that street.

Some of the above issues could have been avoided by using a custom-made route planner (the custom-made route planner could have stored and processed the inaccessible roads and could have avoided these roads). This is once again a good reason why we should research and try to make a better route planner. Therefore it will be important that these surveys will be evaluated and analysed. It is also important that the evaluation of these surveys happens quickly after the analysis of the route because then the police officers could remember why they drove the way they did. These analysis of the surveys can be very important if the surveys are filled in completely because then you could have detailed information about the

routes taken and why they took that road, and could give a first impression where the problems lay and what needs to be given special attention...

1.3.4 Privacy and Security

When we record the GPS signals of a person, vehicle or something else, we need to keep in mind that we could invade the privacy of these people but because in this work we record the GPS signals of public law enforcements we do not need to keep in mind privacy. What we can do with regard to privacy is keep in mind that the place of intervention could be filtered and only the street is recorded and not the house number for the privacy of the people who needed the help of the police because of domestic problems or other embarrassing problems.

The surveys taken is another aspect of privacy. Of course they are anonymously taken.
All these aspects of privacy could be discarded if and because the data is kept internally and no external companies can record or can obtain these valuable information.

We could ask ourselves the question what should happen if the software is publicly used. Then we need techniques for ensuring the privacy and security of the data. There are four themes to classify the existing approaches, namely *Policy-based approaches, Cryptography-based approaches for data access and release, k-anonymity approaches for personal location collection,* and *Location privacy-based approaches for trajectories.* Here we shortly introduce the main characteristics of these approaches, if you want to know more about them we refer you to [50].

Policy-based approaches is very straight-forward to understand. We define a *policy*, stating a set of rules about who is allowed to access their personal data and for which purpose. This approach is both used in security as in privacy fields. In the security context it is used to allow system administrators specify and apply operational policy that will tie privileges to selected users while in the privacy context it allows the user to define his own policy. For our purpose we can use this technique because the only persons who will use it are the police officers.

When *Cryptography-based approaches for data access and release* is used we cipher the identity of the user prior to sending it to the service provider. This is not very usable for us because there is only one person namely the police, so we will not use this technique.

The main idea of *k-anonymity approaches* is to subdivide the area around the user's location and delay the request as long as the number of users in the specified area falls below the desired value of K.

Most of the above techniques were not useful for our purpose. We have trajectories and most of the previous methods describe the privacy of individuals and how they can prevent that their privacy is not violated. The

last approach *location privacy-based approaches* seems to be the best for our trajectories. Here we have two principal categories: *k-anonymity for location position collection* and *Confusion-based techniques*. The first category aims at preserving the anonymity of a user by obscuring his route. It uses 2 zones, *application zone* where the application can trace the users' movements and *mixed zone* where it cannot trace the movements. When a user enters the mixed zone the applications do not receive any location information but they receive a pseudonym. Since applications do not receive any location information when users are in a mix zone, the identities are said to be 'mixed'. The pseudonym of any user changes whenever he or she enters a mix zone. In this way, Location Based Services (LSB) applications that see a user coming from the mix zone cannot distinguish that user from any others who are in the mix zone at the same time and cannot therefore link users entering the mix zone, to others leaving it. The second category introduces fake routes or modifies the true users' trajectory. One possible way of doing this is *path confusing*. Here for every trajectory there is made another trajectory and the algorithm chooses one point in space where the 2 trajectories meet each other. By doing it this way there is no knowing of which trajectory is used by the user because you can not know which one is followed by the user and which one was generated by the algorithm. These 2 ways of anonymity are illustrated in Figure 1.6

1.4 How the problem was handled

In the next chapter we describe which data was available, which algorithms we used (and which we did not use), we compare 2 algorithms (the naive approach and the combined beads k-shortest path algorithm) and we conclude to tell what can still be approved and how the work can be continued. In chapter 3 the structure of both the program as the database is discussed. To conclude we have combined some of the problems occurred in chapter 4. In appendix A you find the survey the police officers were asked to fill in and in appendix B you find the grafical user interface of the program written to digitalize the survey's.

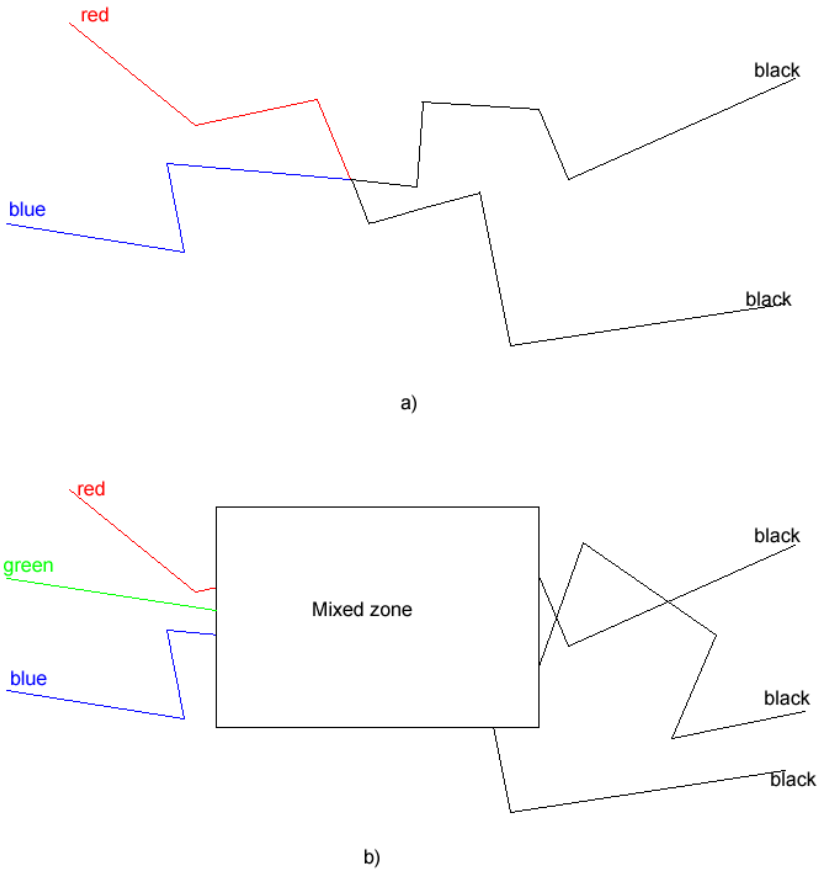


Figure 1.6: a) Confusion based trajectory (path confusing) b) k-anonymity for location position collection (mixed zone ⟷ application zone)

Chapter 2

Mapping GPS points to road segments

It is very important that we can work with *realistic* data instead of the *raw* data got from the GPS devices. As stated before it is more important to know for instance where the school gate is, at which road, then knowing that the police passes a school (perhaps on the back where there are no children or very few). Therefore we have to map the raw GPS coordinates to the actual roads.

Here we describe the hardware used for the retrieval of the GPS coordinates (Section 2.1) (hardware specifications done by and info got from UGent). Afterwards in Section 2.2 we describe the Input and Output we produce and we describe the algorithms we looked at, in Subsections 2.2.3 and 2.2.5 we describe a naive algorithm and some special cases with their solution. In Section 2.3 we give the algorithms used in this work. To conclude we have a comparison between the algorithms in Section 2.3.2.

2.1 Technical specifications of GPS devices

[21] Because we would like to examine the routes taken by the officers for analysis, we needed to track their routes. A device that could track a route is a GPS device, but there were some specifications it had to have and some it may not have.
The idea was that the experienced police officers were asked to start recording from the moment they got a call from headquarters and to stop recording when they arrived at their intervention so the device must have a button to start and stop recording the route. The device should not show a background map on the screen because otherwise they got an aid for knowing how to drive. The coordinates of the tracks need to be read afterwards on a computer, by preference directly in Lambert72 coordinates [2]. These are the main criteria. For all the GPS devices found on the market there were

Figure 2.1: Garmin GPS 60 (left) and Magellan Explorist 210 (right)

only two who satisfy these main criteria: Garmin GPS 60 [31] and Magellan Explorist 210[24] (see Figure 2.1).

Magellan has the largest memory but the batteries from both devices are not big enough to let the device track for more than a week. So it is not necessary to have a very big memory if the devices can not use more memory than tracked in a week. Therefore the memory from both devices were big enough to store all the track points for one team during one week. The recording of the track points happens automatically by the Magellan, so we have to guess when the GPS device records the track points, if we are driving fast he will record few tracking points and if we drive slowly it will record more tracking points. The advantage of the Garmin GPS was that we could choose how many track points needed to be recorded per time- or length measure. We have chosen for recordings of every 10 meters.
By examining the advantages as well as the disadvantages, criteria 4 (the user friendliness) was the most important one for choosing the Garmin GPS 60, in spite of the extra 20% cost. You find the rest of the specifications in table 2.1.

Criterium	**Garmin GPS 60**	**Magellan Explorist 210**
1. Register track points per time or length measure	yes	no
2. Memory capacity	max 10000 track points (number of tracks per track log file is limited to max capacity, not less)	max 150 track log files of each 2000 track points
3. Life cycle batteries	max 28h (dependent of brand battery)	max 18h (dependent of brand battery)
4. User friendly	very simple; format into which the tracks are recorded is clearly and easily convertible to Lambert72 and to shapefile format	easy; during testing phase the coordinates of the recorded tracks seemed to be incorrect (during navigation the coordinates were correct)
5. Accuracy	5 à 15m in x and y	5 à 15m in x and y
6. Price	227€	188€
7. Possibility to connect to 12V of car	yes (31 €)	yes (28 €)
8. Possibility/Necessity external antenna	yes (36 €)	no
9. Necessary time to initialize	cold start(after 8h of not used): max 1 min; hot start: min 20s	cold start: max 1 min; hot start: min 20s

Table 2.1: Specifications of the two GPS devices

2.2 Mapping data to road network

We describe in this section the input and output but also a relatively new technique of beads (see Section 2.2.4). Furthermore we describe several algorithms which could be useful. We also explain some of the problems which could occur and how to solve them.

2.2.1 Input

We have two available datasets, the road data and the GPS data. Here we describe both of them more in detail.

Roads

The route network is represented by a shapefile from TeleAtlas. We inserted it into the postgreSQL[36] database with PostGIS extension[38]. We limited the street network to only the district Ghent because the police of Ghent mostly operate within this district. We filtered the data of the routes outside of Ghent because they were not as important to analyse as the routes taken within the city. Another problem was the connection between the program and the database. The dataflow from the database to the program to load the road network in main memory was to slow (3 minutes). Because we know the data of the route network is static, we could preprocess this data, like calculating the neighbor relation, the length of the street; and write it then to disk (this takes about 30 minutes but it has to be done one time only). The next time we use the program the preprocessed data is read from local disk instead of calculated again.
For narrowing the search area of the network the calculation of the trajectory we use beads (see Section 2.2.4). This has a positive influence on the several algorithms described next (see Section 2.2.4).

GPS

The police officers needed to start recording when they got a call for an intervention and they needed to stop recording when they arrived at the intervention. The routes recorded are GPS coordinates. Every route is a sequence of GPS coordinates (GPS points). They are stored with (x, y, t) information, so we speak of point-location management. (x, y) is of course the location and t is the time. These coordinates form the route taken, sometimes with gaps, sometimes perfectly on the actual road taken but they can not be used directly without any kind of preprocessing. They need to be mapped to the road network for being analysed.

The reason why it is so important to map it to the actual road network is that we need to analyse the data afterwards. With the raw data we can examine some queries stating that the point is close to a school, but

we cannot express if it passes the school gate for instance. Therefore it is pertinent that we map it to the actual road network. The data-mining aspect will be more interesting on the actual road network than on the raw GPS data. Another reason is that when we work with the actual road network the intelligent route planner will be easier to make because we have exact roads they have taken otherwise it always will be a guess which of the roads they have taken.

2.2.2 Output

For the export of the route we used shapefiles because UGent was used to these type of files but we also used a GML[20] file for exporting these routes because this is more convenient to work with afterwards. The output is a $(streetId, t)$ tuple where the $streetId$ is the unique streetId given (internationally agreed) at a road segment and t is the time that they started at this road segment.
We have chosen to keep the time aspect in the data because this is a very important aspect of the data, by analysing the time we can see if they got stuck in a traffic jam, and we can determine their speed afterwards if needed because $streetId$ is stored as a geometry (MultiLineString). An example can be found in listing 2.1.

Listing 2.1: Part of GML file

```
<route>
  ...
  <gml:MultiLineString srsName="EPSG:31370">
    <gml:LineString>
      <gml:coordinates>
        103419.648276815,196982.992453492 103416.220612467,
        196982.11083326 103336.019739619,196973.81727373
      </gml:coordinates>
    </gml:LineString>
  </gml:MultiLineString>
  <gml:timeStamp>
    <gml:TimeInstant>
      <gml:timePosition>
        2006-10-03-T08:34:31
      </gml:timePosition>
    </gml:TimeInstant>
  </gml:timeStamp>
  <gml:MultiLineString srsName="EPSG:31370">
    <gml:LineString>
      <gml:coordinates>
        104102.778915869,193032.047188972 104104.719059269,
        193007.076320557 104108.048140889,192974.973109803
        104110.011252964,192960.570967148
      </gml:coordinates>
    </gml:LineString>
  </gml:MultiLineString>
  <gml:timeStamp>
    <gml:TimeInstant>
      <gml:timePosition>
        2006-10-03-T08:36:22
      </gml:timePosition>
    </gml:TimeInstant>
  </gml:timeStamp>
        ...
</route>
```

2.2.3 Naive

A first approach for calculating the road segments driven on is very straightforward. We calculate the road segments which are closest to the GPS coordinates and afterwards we combine them to a smooth following road. This would seem the easiest and best way to calculate the road but this is not.

First of all we have the problem of the inefficient retrieval of the GPS coordinates (see Figure 2.5), they do not match the road segments driven on. Like you see at Figure 2.5 the GPS points start at the wrong road segment but they appear to recover themselves by going to the right road which they actually have driven on. The same happens at the right top of the figure where there is a mismatch too. A second problem is that there are gaps in the GPS points, they do not following each 10 meters as promised. When we look at Figure 2.6 we see that there is a serious gap (marked by an ellipse) of 140 meters. Another common problem is seen in Figure 2.7 where there are two possible roads. This problem could be solved by examining the timestamp given by each GPS point and looking at the direction of both road segments (one-way or not) for determining which road we are driving on.

2.2.4 Limiting the scope

For mapping from the recorded GPS points to the actual road network there are several possibilities, we explain here an algorithm using beads and one with k-shortest paths. To compare we tried a second algorithm and hoped the algorithm mapped the GPS coordinates well to the actual road network. We hoped that the k-shortest path problem could solve our case. The reason why we have chosen for the k-shortest path algorithm was a guess. But it turned out to be a well-weighted guess because it seemed they took a shorter path.

Beads

Describing trajectories brings uncertainty that either results from the interpolation methods used to build the trajectory or from the measurement errors introduced by sensors that captures objects movement. These kinds of uncertainty will be referred to as: *interpolation uncertainty* and *measurement uncertainty*, respectively. Normally with the help of GPS devices, the measurement uncertainty should be very small compared to the *interpolation uncertainty*, but the GPS coordinates are not always perfect (see Section 1.3.1). *Interpolation uncertainty* can be managed by bead-model. However, in order to reduce the complexity of evaluating certain queries, minimal bounding mechanisms must be used which requires to determine the maximal speed of the object. In case of network constrained movements, like cars in a road network the uncertainty between two succeeding sampled positions could be further reduced by exploiting the network topology.

There are various ways for reconstructing the GPS data or trajectory samples (finite sequences of time-space points), of which linear interpolation is the best-known [37]. But linear interpolation relies on the assumption that between two succeeding points the object moves with a constant minimal

speed. It is more realistic if the object moves within a bounded speed limit. Given such upper bounds, an uncertainty model has been proposed which construct *beads*. We calculate the possible locations where a certain object, in our case a car, can be given the time between succeeding points and a maximal speed. Normally this would give us an ellipse but for complexity reason we calculate a bounding box by determining a point $R(X_1, Y_1)$ and a point $U(X_2, Y_2)$ (see Figure 2.2).
R is calculated by moving away from b as followed:

- X_1 is the maximal point at the X-axis you can reach if you move away from b and still want to be on time at b, so if drive at maximal speed (say v) away from b in X direction where do we need to return for being just in time at b;
- Y_1 is the maximal point at the Y-axis you can reach if you move away from b and still want to be on time at b, so if drive at maximal speed v away from b in Y direction where do we need to return for being just in time at b.

U is calculated by moving away from a as followed:

- X_2 is the maximal point at the X-axis you can reach if you move away from a and still want to be on time at a, so if drive at maximal speed v away from b in X direction where do we need to return for being just in time at a;
- Y_2 is the maximal point at the Y-axis you can reach if you move away from a and still want to be on time at a, so if drive at maximal speed v away from b in Y direction where do we need to return for being just in time at a.

S is calculated by taken the X-coordinate of U and the Y-coordinate of R, and T is calculated by taking the X-coordinate of R and the Y-coordinate of U. The hardest thing to determine is the maximal speed v. We have concluded by experimenting that a good value of $v = 120$ km/h

As we can see in Figure 2.3 the bead calculated is very big although the distance between points A and B is only 13 meters but when we examine the time needed for travelling from A to B it is very big nl. 35 seconds (possibly they had to stop for a traffic light). This is exactly the reason why the bead is so big. The bead calculates the region where the police car could have been when it travelled at a maximum speed of 120 km/h over a time of 35 seconds. Therefore the bead is so big and the reason why so many streets are found. On the other hand when we look at Figure 2.4 the distance here between succeeding points A and B is 11 meters but the time between them is only 1 second, therefore the bead calculated is very small but yet enough to contain the 2 road segments who surround the points.

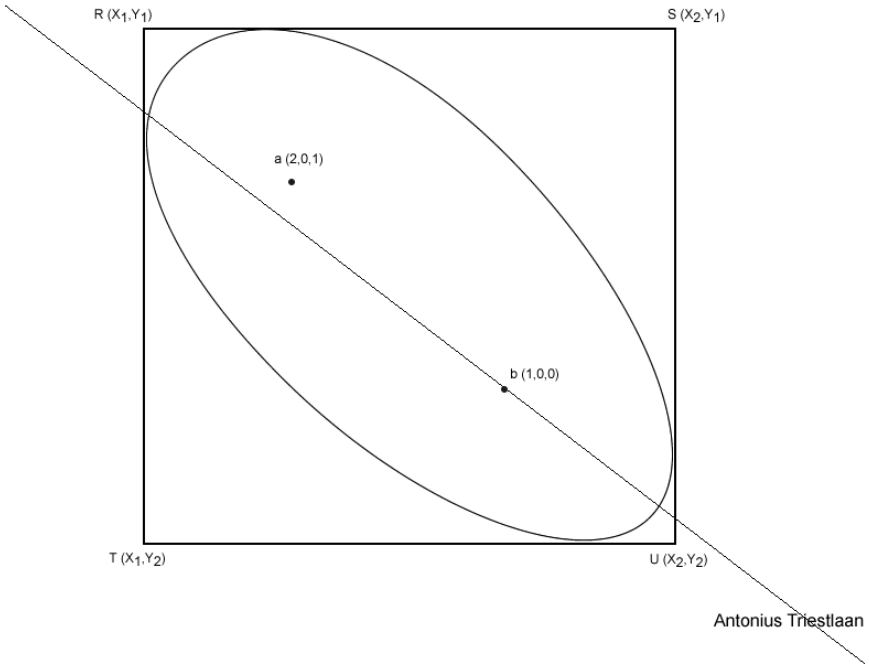


Figure 2.2: Representation of the ellipse used by the beads method (bounding box is also displayed). a and b are GPS coordinates.R,S,T,U are as described in Subsection 2.2.4

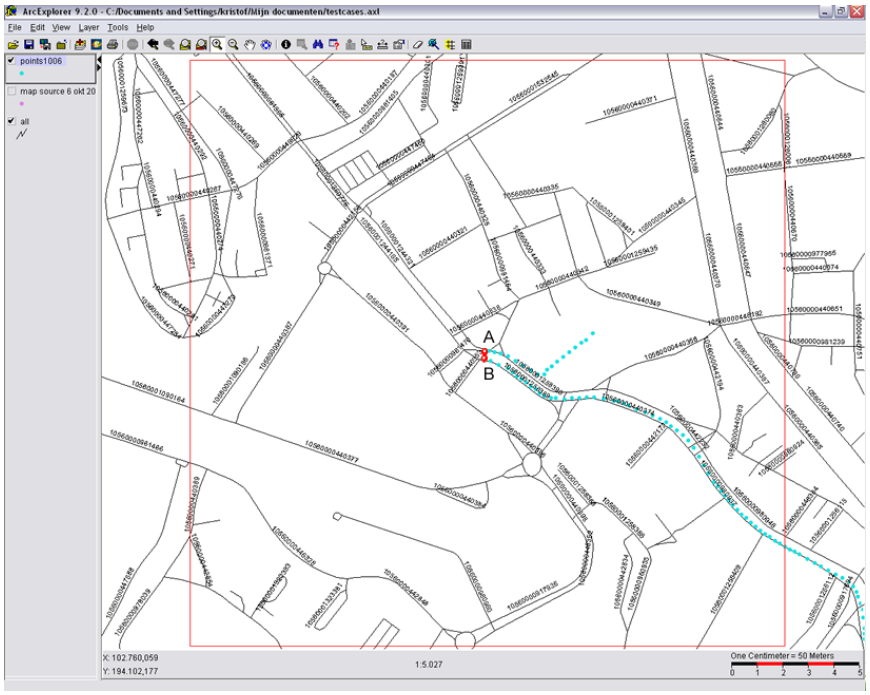


Figure 2.3: Real-life example. Seconds between 2 succeeding points (dotted points A and B) is 35 seconds

Algorithms

Here we describe different methods for trying to map the GPS coordinates to the real topology of the road network. First we try it with a naive algorithm then we explain an algorithm we use in several other algorithms. Then we examine several k-shortest path problems.

2.2.5 Special Cases

There are several special cases which are difficult to handle, we describe each of them and present a solution for solving these special cases if needed.

Triangle problem

When we have a triangle as we can see in Figure 2.8 we can clearly see that they drove the outer lines of the triangle and not the opposite tick line. But logically the computer estimates the shortest/fastest path and therefore the thick line is chosen. We can solve this by examining the score calculated (as described in Section 2.3.1) of each of the lines and make according to these scores a better route, but once again this is not optimal. When we

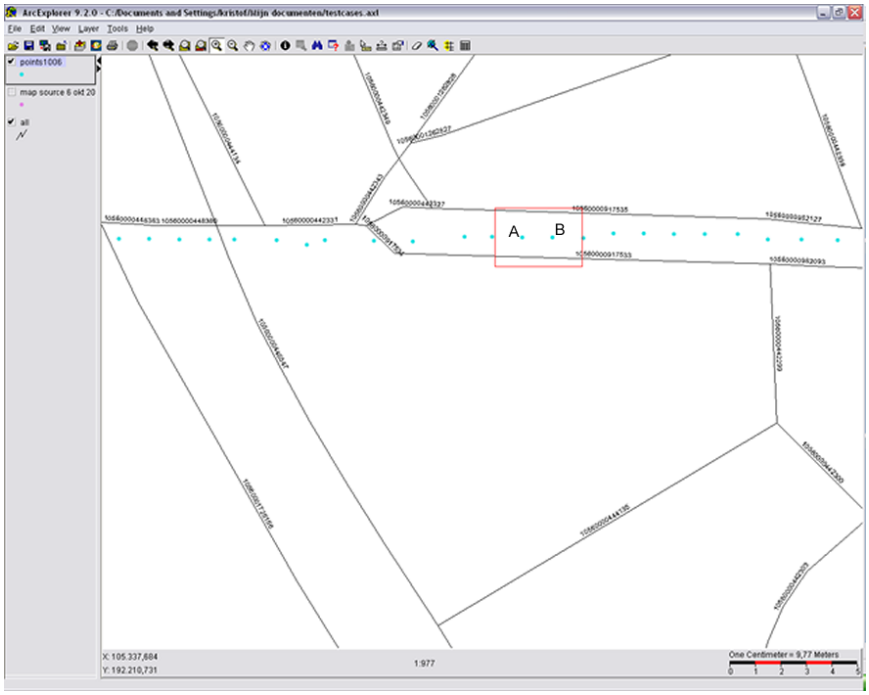


Figure 2.4: Real-life example. Seconds between 2 succeeding points (points A and B) is 1 seconds

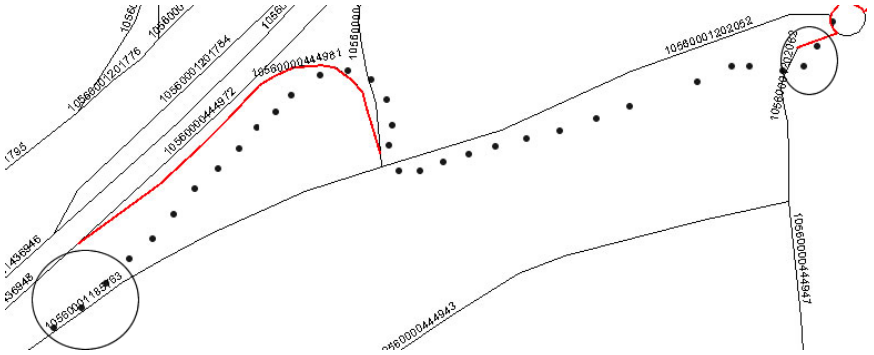


Figure 2.5: Faulty GPS signals are circled

examine line 1 and presume the following: line 1 has a score of 6, line 2 has a score of 5 and line 3 has a score of 6, because the Dijkstra algorithm works backwards we have the choice between lines 2 and 3 but because line 3 has a higher score the algorithm will choose this road segment instead of line 2. This could be solved by making a globally optimal solution and not a locally optimal solution by examining all routes within a certain threshold (see Subsection 2.3.2).

Traffic jams

A second problem is seen in Figure 2.9. As we clearly can see the road segment at the top is chosen, but there is no obvious reason why they have chosen this route. Perhaps there was a traffic jam on the straight road segment, or are the GPS signals faulty, less likely. This problem we only can fix by asking the police why they drove like they did but probably they will not remember it anymore because the data was taken almost a year ago. For this reason it is pertinent that we examine these routes as quickly as possible and get feedback from the police as soon as possible.

Gaps

When we examined the data we concluded that sometimes there were gaps in the route (see Figure 2.6 and Figure 2.10). There are several reasons why there could be gaps in the routes:

- A first explanation could be a faulty GPS signal because of bad reception of the coordinates. Normally this should not occur so we may exclude this option.
- Secondly there could be an interruption between the GPS satellite and the device because of a dense area of forest or a tunnel or because of high buildings (if the GPS signals are blocked because of tall buildings we speak of urban canyons).

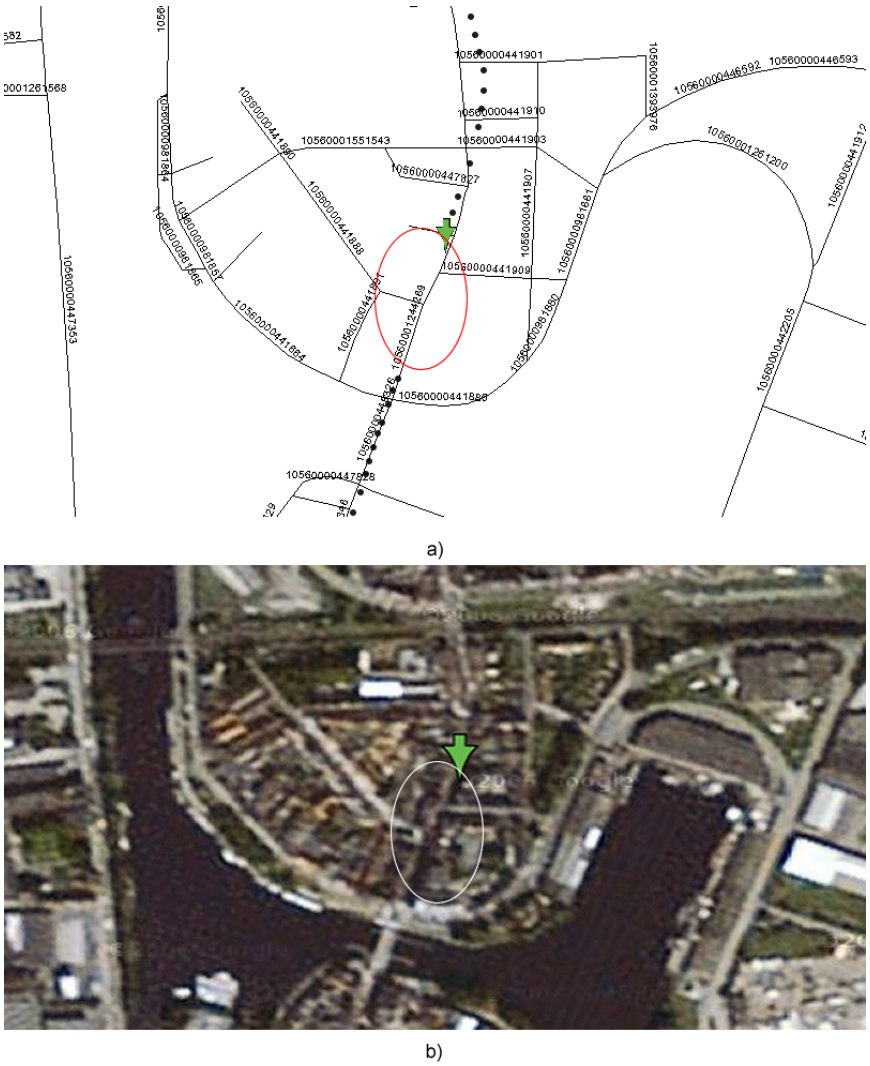


Figure 2.6: a) Extract from the data with a gap in the ellipse. b) Google Map extract from the same area.

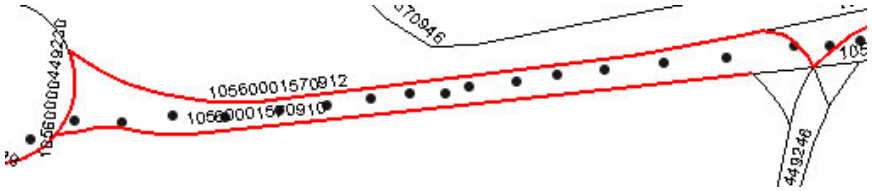


Figure 2.7: More then one possible road segment

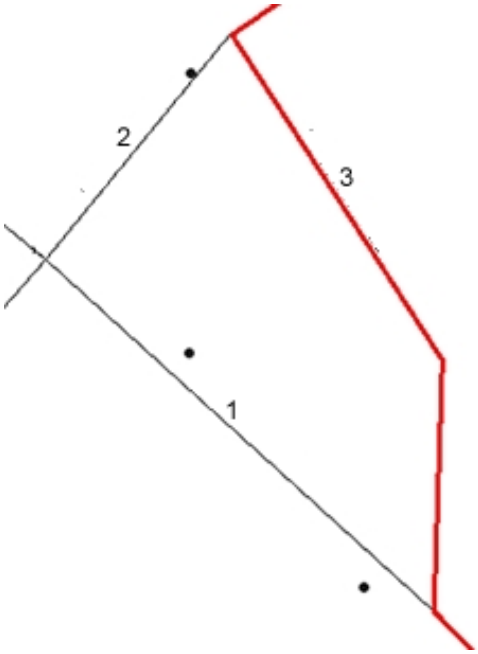


Figure 2.8: More then one possible road segment

The solution to these possibilities could be answered by Google Maps[15]. We could try and search for satellite photo's of the area and see what could be the problem. We did this for the gaps in Figure 2.6 and in Figure 2.10. For the first gap it could not be seen why there was a gap (see Figure 2.6) but for the second gap it was obvious (see Figure 2.10): they drove into a tunnel that is why there were not GPS coordinates recorded.
Without these semantics we have to guess why there were no GPS coordinates recorded. Thankfully we have these aid and this can help us for determining which road they have driven on, for instance if they drive into the tunnel they have to get out of the tunnel at the other side and the GPS coordinates followed just outside the tunnel could be faulty because of satellite search of the GPS device.

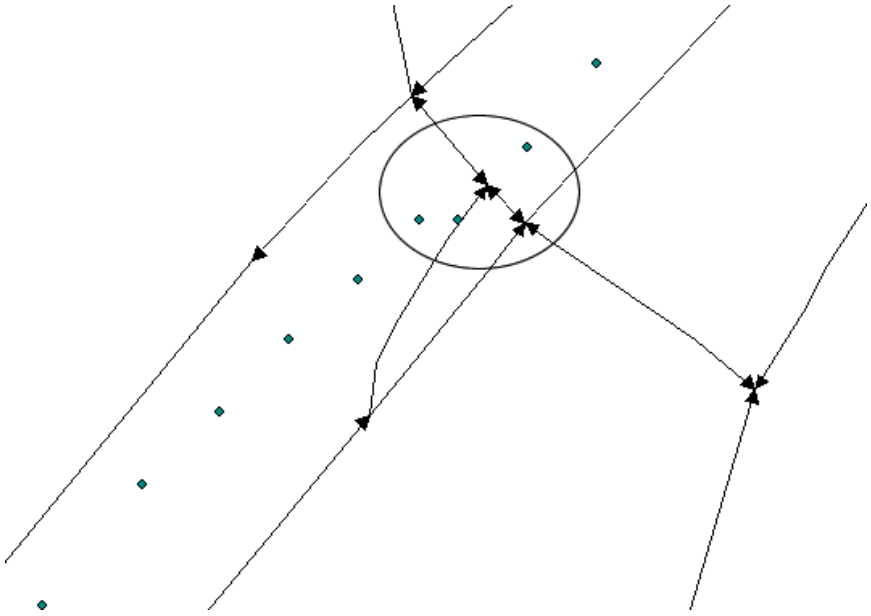


Figure 2.9: Strange GPS point, possible traffic jam or not?

2.3 More optimal algorithms

In this subsection we describe the more optimal algorithms which were useful and which we have used in this work.

2.3.1 Score calculating algorithm

We describe here the algorithm used in several of the following sections. When we got the GPS points we calculate which road segments are closest to these points. The street that is the closest to the point gets weight n, the street that is secondly closest receives weight $n - 1$ and so on till the n-th closest street receives weight 1, here too we calculate with beads the possible road segments because otherwise we have to check which roads are too far away to be reliable. Another approach could be to check that the streets has to be within x meters for being reliable. We have chosen by experimenting for n to be 3 and x to be 15 meters. By doing it one of these ways each of the road segments driven on will receive relatively high scores in comparison with other routes not driven on.

An example of the algorithm can be found in table 2.2 and in Figure 2.11. Here we clearly can see that this is a very good algorithm to start from. You see that the ending scores will follow the actual route taken. The score of streetId 1 is 5 when we continue we come to 2 neighbours one with score 3 (streetId 3) and one with score 7 (streetId 4). Here we immediate see that streetId 4 is the best. One we continue this algorithm we see that

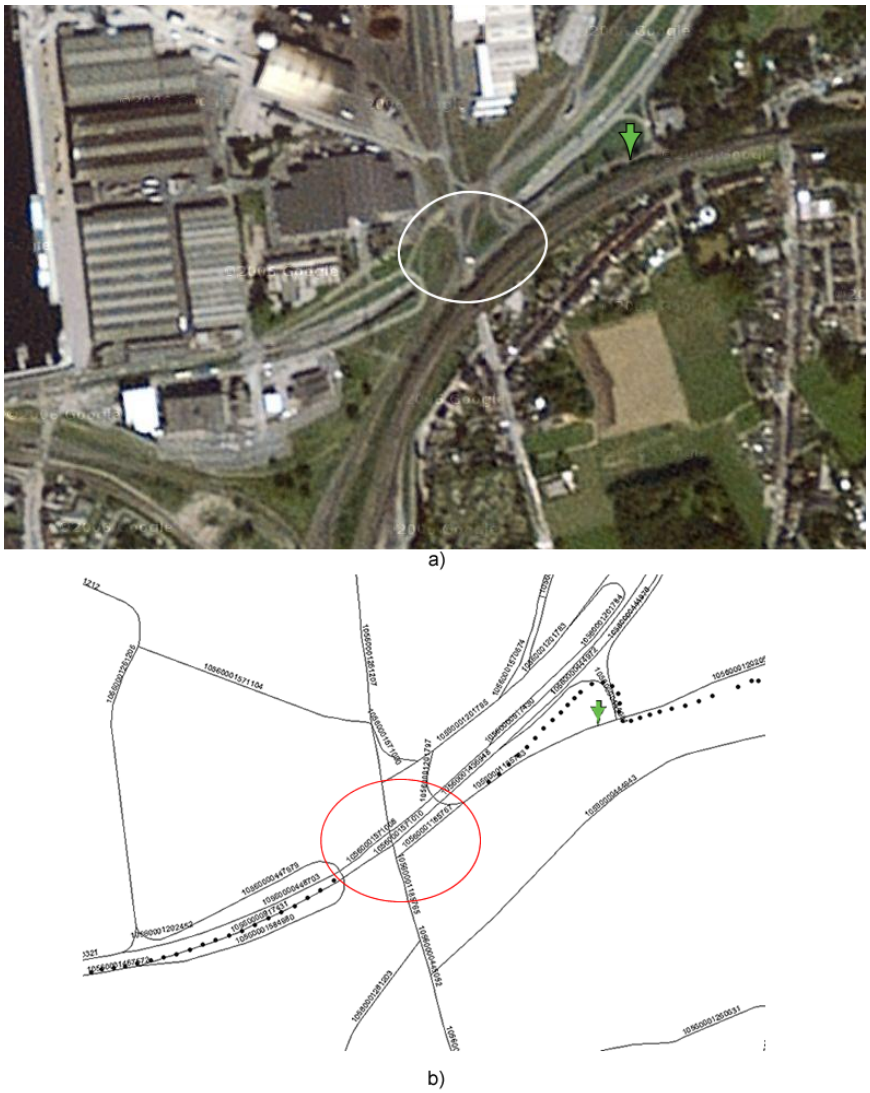


Figure 2.10: a) Google Maps extract from the location in b where we see a tunnel within the ellipse. b) Extract of the data with an obvious gap within the ellipse. (Arrow for orientation purposes)

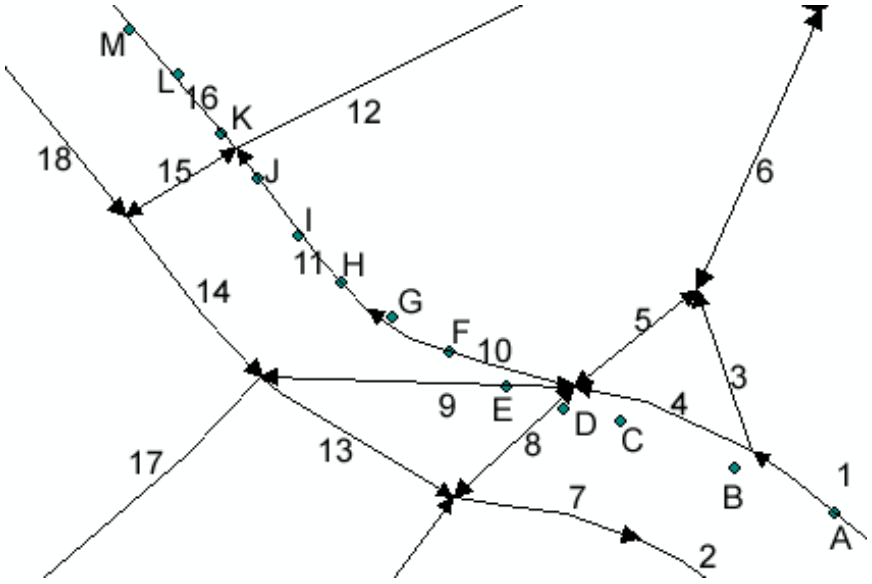


Figure 2.11: A → M are GPS coordinates. 1 → 18 are streetIds

this is a pretty good start. Therefore we use this algorithm in the following section to make a first good impression of the route taken and tune this further.

Listing 2.2: Pseudo code for calculating score

```
closestPoints = getClosestStreetsFrom(Point p)
for(every street found && i < n)
  street.score += n - i
```

2.3.2 k-shortest paths

The k-shortest path problem is a well known problem in networks. It is used when there are more constraints then only the distance of the path taken (as is the case in our work, the second constraint is the scores calculated as described in Subsection 2.3.1). Instead of finding one shortest path there are k paths found between 2 fixed nodes. We explain several possibilities and conclude which is used in this work.

Double Sweep

One of the possible algorithm was the Double Sweep algorithm which was mentioned at [5]. There we found a schema of the algorithm but not the entire algorithm. In [39] we found that this was especially used for routing airplanes because there were several parameters to keep in mind like refueling points, navigational waypoints or technicians who were based on one airport

Id	Initial	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	3	5	5	5	5	5	5	5	5	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	2	3	3	3	3	3	3	3	3	3	3	3	3
4	0	1	4	7	7	7	7	7	7	7	7	7	7	7
5	0	0	0	2	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	4	5	5	5	5	5	5	5	5	5
9	0	0	0	0	2	5	7	8	9	9	9	9	9	9
10	0	0	0	0	1	3	6	9	11	13	13	13	13	13
11	0	0	0	0	0	0	1	3	6	9	12	12	12	12
12	0	0	0	0	0	0	0	0	0	1	3	4	5	5
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	3	5	5
16	0	0	0	0	0	0	0	0	0	0	0	3	6	9
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.2: Example of the calculating algorithm

but not the other (for repairs). Another reason why this was not the best solution was that this algorithm computed k-shortest paths from one node to every other node in the network, this was not needed because we knew the starting and ending point.

The problem of D. Eppstein

Perhaps one of the most famous k-shortest path problem is that of David Eppstein[6]. He finds the k shortest paths by generalizing it to finding the route from a start node (s) to every other node in the network because this is much easier: use breadth first search. Use a priority queue of paths, initially containing zero-edge path from s to itself; then expand it by one-edge paths and so on. At the end remove the shortest from the queue and add this path to the output paths. The complexity was $O(dk + klogk)$ if the bounded degree of the graph was d. Then we translate the problem from one with two terminals, s (startnode) and t (ending node), to a problem with only one terminal. One can find paths from s to t by finding paths from s to any other node and concatenating a shortest path from that node to t. But we cannot apply this idea directly, for instance because each path from s to t may be represented in many ways as a path from s to some node followed by a shortest path from that node to t.

The main criteria why we did not use this algorithm was that it could contain cycles, something that we did not want to have because it most unlikely that the police took the same road more than once.

Removing path algorithm

Santos [40] discussed several algorithms. One was the removing path algorithm. The main idea is that the second shortest path is the path in the network where the shortest path is deleted from, the third shortest path is the shortest where the second shortest is deleted from too and so on. We can summarize the algorithm as described in Listing 2.3.

Listing 2.3: Pseudo code for removing path algorithm

```
do {
        calculate shortest path;
        remove shortest path from the network;
        put it in result list;
} while(k-shortest paths found);
output result list
```

The problem of Yen

In [4] we find several other algorithms like Yen, Lawler, Katoh and Hoffman. We describe only Yen because that is the algorithm we have chosen to use. Yen's algorithm uses the Dijkstra algorithm for calculating k-shortest paths. First the shortest path is calculated using A* algorithm. Yen takes every node in the shortest path, except the terminating node and calculates another shortest path (spur path) from each selected node to the terminating node. For each such node, the path from the start node to the current node is the root path. Two restrictions are placed on the spur path: (1) It must **not** pass through any node on the rootpath (the paths are loopless) and (2) It must **not** branch from the current node on any edge used by a previously found k-shortest path. If a new spur-path is found it is appended to the root path for that node, to form a complete path from start to end node. Then the score is calculated for this new shortest path (as described in Subsection 2.3.1), if this score is greater or equal to that of the shortest path calculated at the beginning it is added to the results list. If k-paths are found we return the path with the highest score. The complexity of the algorithm is $O(n^3)$. Calculating the spur paths from each node is $O(n)$ and using a shortest-path algorithm (Dijkstra) $O(|V|^2 + |E|)$ with $|V|$ number of vertices (intersections) and $|E|$ the number of streets (edges).

To clarify the algorithm we have an example. The road network is seen in Figure 2.12. Assume that all arrows have weight 1. We try to find the shortest from A to D. It is clear that the shortest path is A, B, C, D so we

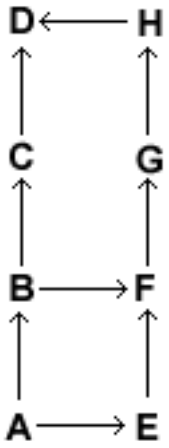


Figure 2.12: Roadnetwork used in the example of the problem of Yen

include this path in the result path. Now we try to find other paths starting from this shortest path. We proceed in every step with the next node and start with node A. These succeeding nodes we call the rootpath. We start with rootpath A, so we try to find a path from A to D that is not already in the result list. Now we find the path A, E, F, G, H, D (this is the only path we can follow because the succeeding node B has already been followed and is in the result list). We also add this path to the result list. Now we go a step further in the rootpath and start with the nodes A, B. Now we find A, B, F, G, H, D (here too we can not follow the succeeding node C after B because this is already been followed). We have found all paths possible in the road network and the algorithm comes to an end.

Conclusion shortest paths algorithms

A shortest path problem like A* or Dijkstra do not match the exact route taken because not every route the police took is exactly the shortest/fastest, there are some influences for taking one road at a particular time of the day, for instance at 5 o'clock there will be a traffic jam at road x so we take road y. Therefore we need to search for an algorithm that is the best in polynomial time and can match the route taken by the police. A perfect solution is the k-shortest paths algorithm. First we calculate a shortest path by A* then we find several shortest paths and afterwards we take the route with the highest score calculated as in Section 2.3.1. We used the algorithm of Yen for implementation because this was the easiest to implement with the data structures used and there were only a few adjustments needed to the Dijkstra algorithm already implemented.

2.4 Comparison of the used algorithms

In this section we compare the naive algorithm with the new bead technique combined with the k-shortest path algorithm. We hope to conclude that despite the fact that the bead algorithm will work slower we come to much better results than when we worked with the naive algorithm. The naive algorithm is described in Section 2.2.3 while the bead algorithm is defined in Section 2.2.4. We summarize the working of both algorithms in the next sections and we examine the algorithms when processed on 30 routes.

2.4.1 Naive algorithm

The naive algorithm is very easy to understand. We just look at which streets are closest to the points and we keep these streets. Then afterwards we calculate the shortest path between start and ending node within this new network and keep this path.

The results of this algorithm matched exactly the routes taken by the police. That is when the GPS points are very frequent and when there are no gaps in the data.

2.4.2 Beads combined with k-shortest path

This algorithm is combined out of several steps:

1. At first we limit the network by calculating for each succeeding points which roads they could have driven on (as described in Section 2.2.4).

2. Then we calculate for each GPS point which roads are closest to this point (as described in Section 2.3.1).

3. Then the last step is to calculate within this limited network, as calculated in step 1, k-shortest paths and we take the shortest path with the highest score as calculated in step 2.

This algorithm ensures that the k-shortest path problem is calculated on a limited network instead of the global network which is very smart as calculating the shortest paths is the hardest thing to do the bigger your network is. The preprocessing step (step 1 & 2) does not take as long as calculating the shortest paths on a entire network. It takes 15 seconds in general.

2.4.3 Results

There were some routes which were not capable for testing purposes because they laid out of the district of Ghent with only a few streets but because we had not the data available it was impossible for mapping it to the actual

road network. We have examined 30 routes which had a lot of GPS points (more than 500) as well as routes which had only 50 GPS points. You find all details in the table on the next page. The routes are ordered from highest GPS points to lowest. We will highlight some of the results written down in the table.

We see that in half the cases the naive algorithm is as good as the beads algorithm and produces the same result much quicker but when there are gaps in the data the naive algorithm does not produce an answer whereas the beads algorithm produces a perfect road. Sometimes we see that as well as the beads algorithm as the naive algorithm does not include loops (because we have programmed it not to include loops), these loops are present because the police had to search at the end for the house number or even for which street to turn into, we supposed that the police did knew the environment well, but as is proven this is not the case. However this certainly does not happen in all routes but because it sometimes happens we have included this into the test results. In a rare case the network we got is not big enough to include all patrols from the police of Ghent, sometimes it felt out of the district of Ghent. Like said before this is a rare case but it happens once in a while.

2.4.4 Conclusion

As we can see in the table on the next pages the naive algorithm is very good for data which is nearly perfect and when the succeeding GPS points are close to each other but when there are gaps in the data, how small they are, the naive algorithm is not good enough anymore despite the fact that this is a very quick algorithm. The beads algorithm always performs a good answer but is a lot slower (double as slow but most of the time 4, 5 times and sometimes even 10 times slower).

Start	End	Length (m)	# GPS points	Remarks	Beads	Naive
					Seconds	
10560000445252	10560000439755	9000	661	None, both are equally good	408	61
10560001258396(HQ)	10560000438637	9650	550	Naive gives no output because of gaps	53	15
10560001258396(HQ)	10560001261207	5690	442	None, both are equally good	40	3
10560001258396(HQ)	10560000098214	5300	433	End lays on a road we do not have in database	67	31
10560000449904	10560000439128	6900	428	None, both are equally good	84	12
10560001258396(HQ)	10560000444811	6000	425	Strange return and strange beginning, naive gives no output because of gaps	80	12
10560001258396(HQ)	10560001260462	5600	423	Naive gives no output because of gaps	146	13
10560001258396(HQ)	10560000443248	5000	405	Roads taken which are not in network	35	13
10560001258396(HQ)	10560001362084	5900	403	Naive gives no output because of gaps	55	11
10560000446704	10560001263331	5200	401	None, both are equally good	64	12
10560001258396(HQ)	10560001266760	5200	377	None, both are equally good	34	11
10560001260482	10560000444275	4700	359	None, both are equally good	259	10
10560001184920	10560000982089	5200	340	None, both are equally good	25	10
10560001258389	10560000448755	5100	325	None, both are equally good	31	10
10560001258396(HQ)	10560000443238	4050	305	None, both are equally good	65	11

Figure 2.13: Real-life example. Seconds between 2 succeeding points (dotted points A and B) is 35 seconds

Start	End	Length (m)	# GPS points	Remarks	Beads Seconds	Naive Seconds
10560001258396(HQ)	10560000441411	3800	292	None, both are equally good	23	11
10560001258396(HQ)	10560001551544	3800	256	None, both are equally good	52	9
10560001258396(HQ)	10560000996107	3800	217	None, both are equally good	53	8
10560001258396(HQ)	10560000441385	2800	204	End fault (loop)	51	9
10560001258396(HQ)	10560000440597	2100	197	End fault (loop)	30	10
10560001344539	10560000443221	2500	195	None, both are equally good	24	14
10560001258396(HQ)	10560001374886	2200	185	None, both are equally good	17	8
10560000438637	10560001261477	2500	185	None, both are equally good	16	9
10560001258396(HQ)	10560000449144	3200	182	Naive gives no output because of gaps	91	9
10560001258396(HQ)	10560000442651	2400	176	Naive gives no output because of gaps	19	9
10560000442488	10560000440314	1500	96	None, both are equally good	10	20
10560001258396(HQ)	10560000444836	6200	83	Naive gives no output because of gaps	33	6
10560000447694	10560001260105	2000	65	Naive gives no output because of gaps	40	6
10560001089390	10560000980483	1000	63	None, both are equally good	20	6

Figure 2.14: Real-life example. Seconds between 2 succeeding points (dotted points A and B) is 35 seconds

2.5 Future work

There is still a lot of work to be done. We have mapped the imperfect GPS data to the real road topology for analysis. But the interesting part still needs to be done namely the data mining of the collected data. So the next step in the process is to analyze the surveys and the data together and find out some data mining rules why they drove that particular road. Do they avoid areas where are crowded people at particular times of the day, like schools, or do the police officers know when and where there are traffic jams and do they avoid these places? These are all very interesting questions to be answered. After this analysis the main goal is of course to come to a route planner which is ideal for police officers. This will be the main challenge to achieve.

For doing these kind of data mining we need to have the coordinates of schools, zone 30 and all other kind of interesting mining data, the real questions will be if these data is publicly available and if it is, if we can get it. Other interesting data is *when* traffic jams occur, in which area and how they evolve in time and if the police officers take these routes or not.

When the route planner will be used by the police, it could be considered to maintain a full time link between headquarters and the police cars to have a better look at where which patrol is. This can lead to a better reaction to interventions. Furthermore the real-time traffic information could also be modelled at headquarters to inform the police officers that there is a traffic jam where they need to pass or even build it right in the route planner to handle these situations.

Google Earth can be a possibility for displaying the data or even for mapping it to the real network. Here we both have the satellite as the actual road data available. It seemed that it even made a path from the succeeding GPS points, however I do not think it will map it with the real data like the name of the street or the coordinates of the street. This can be further explored to see whether or not this can be helpful in the mapping the GPS coordinates to the actual road network.

2.5.1 Implementation improvements

There are still some issues to be handled.

The software has a quite good error handling. Every time something went wrong the user is informed by pop-up messages but the messages are not always as clear as needed to be. Here there could be much improvement, normally the user does not come in contact with these type of error messages but it could be, so therefore it is pretty important that the messages are clear enough that the user can handle the problem himself.

It needs to be checked what happens when the police have to make a U-turn or when the police took a wrong street and returned on the same point.

Apparently this happens a few times especially at the place of intervention. Probably they did not get the right house number or they did not knew the environment very well. The algorithm is designed not to include loops (because we thought this would not occur) and therefore the calculated route is not entirely the same as the route taken by the police officers. This can be solved by allowing loops to occur in the path but this makes it more difficult and we need to be careful not to loop indefinitely.

Another problems that occurred when the data was closely examined is that nearly all interventions were of low priority. This means that the police had to respect the traffic rules but sometimes they drove into the wrong direction, of course the algorithm does not take these irregularities into account and a parallel street is chosen instead. Here too we need to be careful when we want to change the algorithm, this would mean that the network is not a directed graph anymore and the algorithm becomes a lot more difficult and has to take more time than when the network is directed.

Chapter 3

Implementation

For the implementation of the map-matching algorithm we have chosen for Java. Java is very flexible and can be run on every computer without any change of the code. As Database Management System we have chosen for postgreSQL [36] because the software needed to be run on the computers of UGent and we did not know whether or not they had a license to work with DB2 [16] from IBM. We used the PostGIS extension [38] for the spatial data. This was entirely new for me but it was pretty straightforward to work with and well commented on the forums on the internet. They have a very active community behind it who help where they can. This was a great aid in some difficult problem solving.

3.1 Used Hard/Software

3.1.1 Hardware

Everything was done on a Dell Inspiron 6400 laptop with specifications:

- double-core Centrino Intel processor 1,66 Ghz 1 Mb cache each
- 1024 Gb DDR RAM
- 40 Gb Hard disk size

3.1.2 Software

The software made for this thesis is written in Java [44]. The program used for writing the code is Netbeans [32]. This is a very good program that auto-completes your source code and can write all the necessary getters and setters for you, which is very convenient. The choice for a DBMS was at first a bit difficult to choose. At first we thought we should use the DBMS used at the university namely DB2 UDB from IBM [16] but because of network traffic we decided it had to be installed on my computer but there were a

lot of downloads available at the IBM website and it was difficult to know which was the best for me to use. Additionally we had the problem that all the software had to be run on a computer at UGent so it would be better if all the software needed was open-source. Therefore we tried other DBMS like MySQL [29] and postgreSQL [36]. Both of these DBMS are open-source projects and both of them have spatial extenders but postgreSQL got our preference because it had a better graphical user interface and because some of the team members already had experience with this DBMS.
Because UGent works a lot with shapefiles the output of the programs made are also shapefiles. For examining these shapefiles we used ArcExplorer [8] from ESRI[7] but we also exported the output to GML [20] format for conveniences afterwards.
The data from the road network was provided by TeleAtlas. The data was available in shapefiles, for the extra data (like the direction we can drive on the road segment for example) the data was stored in a .dbf file. However postgreSQL does not support these type of files there were open-source programs available who could convert these .dbf files to valid SQL-queries. These open-source programs had the disadvantage that they did not support the Long type. So we needed to go to the commercial market and found exactly what we needed with the software from sqlManager[43].

For converting the data directly got from server to a workable format we used GPSBabel[22] which transform all kinds of GPS data to other GPS data.

3.2 Package structure

Database package: db

In the package *db* we find everything that has to deal with the underlying database. It fetches from the database all necessary tables and puts it into the right datastructure. The main classes found here are:

- **Connectie:** This is the most important class we work with. This file includes **all** functions that deal with the database and it defines all needed queries.
- **DBStreet:** Street info like stored in the shapefile from TeleAtlas.
- **DBStreetWithInfo:** Some street info and completed with extra information about length of the street, maximum speed, one-way and with the removal of some redundant columns like *tollrd* which indicates whether or not you have to pay to access this road, this data is redundant for police cars.
- **GPSInfo:** GPS coordinate used for displaying the points in the GUI.

- **AlReadyExistsSameRouteException:** As the class name states this exception is thrown when there already exists the same route in the database. For instance when the user wants to load the same file twice.
- **CouldNotLoadFileException:** This exception is thrown when the file could not be loaded. Especially used by input of the GPS coordinates from file (.shp or .gdb file).
- **CouldNotLoadNetworkException:** Exception is thrown when the network could not be loaded. This is thrown when the shapefile for the network is corrupt or when it could not be read.

GPS package: GPS

Here we find everything that includes the algorithms of map matching like the beads algorithms or the naive algorithm.

- **Beads:** In this class we find the main algorithm for calculating the beads.
- **StreetScore:** A simple datastructure file for storing the streetId and the corresponding score. Includes all necessary functions like *compareTo()*, *toString()* and *equals()*.
- **StreetScoreTime:** A simple datastructure file for storing the streetId, the score and the time associated with it. Includes also the necessary functions and also a *Comparator* to compare 2 StreetScoreTime instances.
- **StreetScoreVector:** Datastructure for storing a *Vector* of *StreetScore* items. It is easier to make a new class for this type of datastructure to handle all the necessary functions like *add()* and *contains()* and more important override of the *toString()* function.
- **TimestampLong:** Stores a streetId with the time passed there.

GUI package: gui

In this package we find everything that deals with the Grafical User Interface (GUI).

- **GdbFilter:** Filter for the *FileChooser* to let the user only choose '.gdb' files.
- **ShapeFilter:** Filter for the *FileChooser* to let the user only choose '.shp' files.

- **InterfaceWithGUI:** The main *Panel* where all the buttons, labels and events are defined.
- **MyTable:** *Table* overridden by a own class for better handling the columns, events,...
- **MyTableModel:** *DefaultTableModel* overridden by a own class for better handling the data and for making the sorting easier.

Parser package: parser

This package includes the parser for converting the '.gdb' file (got directly from the GPS server) to a better format in XML (Google Earth KML [14]) (with the aid of GPSBabel[22]) and further to parse this XML-based format to store it in the database.

- **Parser:** The actual parser is found in here. It uses the SAX parser [27]. It filters out all the unnecessary tags and remains the data of the tags which are needed for the analysis.
- **Tags:** The main tags needed for extracting the data.

Road network package: roadNetwork

Here we find several basic datastructures especially for storing the road network. There has been taken into account the expansion of the datastructure and it will be very easy to extend the current classes and datastructures with other objects or to change them with own classes. The structure can also be found in Figure 3.1

- **Roads:** Head structure for the road network. Consist out of *Hashtable* of *Points* and *PointsData* for quickly knowing the *length* of the street-segment. Contains also the real roadnetwork as $Hashtable$ <Point, RoadData> *Hashtable* has been chosen because of instant retrieval of the points.
- **RoadData:** As the name states, the road data is stored in here. A *Vector* of *Out* objects and *SecondaryData* object. You can store everything in this datastructure what has something to do with a street.
- **SecondaryData:** Currently only used for storing the *Geometry* in the form of a *MultiLineString* but it can be extended to store much more.
- **Out:** Consist out of *ExtraData* and *Point*. It is used for knowing which points are neighbours of which other points and there can be stored additional data in the *ExtraData* variable.

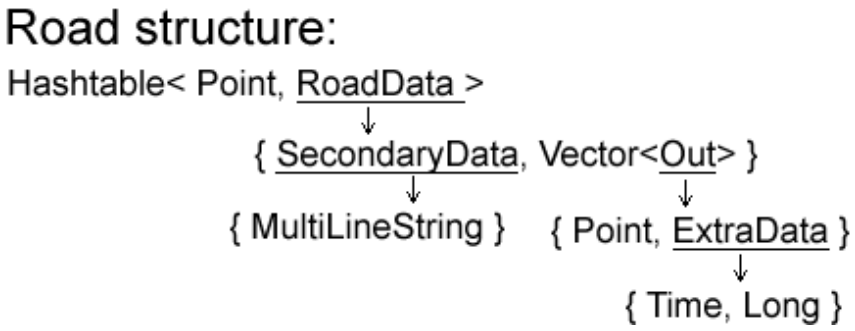


Figure 3.1: Structure of the *Road* class

- **ExtraData:** The *time* with a *streetId* is stored in this class.
- **GPSPoint:** Very basic (*point*, *timestamp*) tuple.
- **Points:** ($point_1$, $point_2$) tuple. Used for storing 2 succeeding points for storing with the *PointsData* class. Tuple ($point_1$, $point_2$) will be stored the same as the tuple ($point_2$, $point_1$).
- **PointsData:** Currently only used for storing the length but it is easy to work further with, for instance also store the one-way or not variable.

Useful structures package: util

In this package we find several useful structures like comparators and algorithms like Dijkstra, AStar or k-shortest path implemented.

- **Dijkstra:** The Dijkstra algorithm is implemented in this class, but also the k-shortest path is implemented in here (*AlgorithmMultiDijkstra*).
- **AStar:** The AStar algorithm is implemented in this file.
- **Direction:** Basic *enum* type with POSITIVE, NEGATIVE, BOTH or CLOSED status.
- **StreamGobbler:** Used for retrieval of output from *Process* objects like the export of shape files.
- **Comparators:** In this package there are a few *Comparator* objects to be found, I do not explain in detail because they are pretty straight forward.

3.3 Grafical User Interface

For the graphical user interface we tried to make it as easy as possible for the user. Therefore we have chosen for one screen with all the functionalities, few parameters to set and a clearly overview. The user can choose how he wants to name the output and where he wants it to be stored. For the input the user can choose between a shape file and the .gdb file directly got from the GPS server. When loaded the routes are numbered and displayed on the right together with their starting and ending timestamp. The user can delete a route if he does not want to use it anymore or when it is a invalid route (only 1 GPS point stored) or display it with one button in ArcExplorer. In Figure 3.2 you find the GUI.

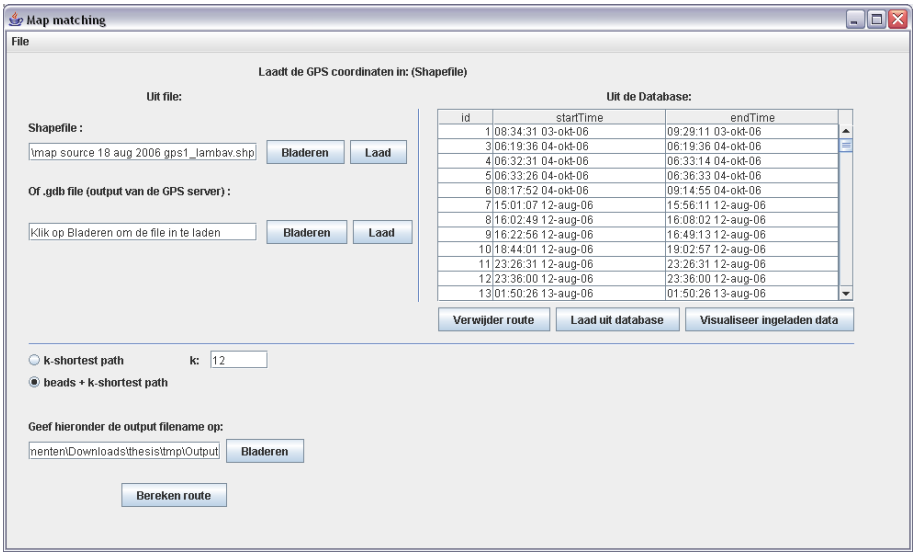


Figure 3.2: GUI of the map-matching program

3.4 External libraries

As with every project here too we used some other external libraries.

Swing layout extensions

The default layout extension when you work with Java, Swing is the most intuitive solution when you make a GUI for Java. Thanks to the very intuitive GUI builder in Netbeans[32] it is very easy to make a good GUI.

PostgreSQL [36]

For connecting and communicating with the database we need this library.

PostGIS [38]

Default library for working with geographical data and the postgreSQL database. Here we find the definitions and classes for *Geometry*, *MultiLineString* and *Point*.

SAX [27]

SAX is the Simple API for XML [27], originally a Java-only API. SAX was the first widely adopted API for XML in Java, and is a "de facto" standard. It is a very straight-forward and easy in use software packet for dealing with XML files and is very well documented.

3.5 External Programs

GPSBabel

GPSBabel[22] is a program that converts waypoints, tracks, and routes between popular GPS receivers and mapping programs. It also has powerful manipulation tools for such data. We used the program for converting the data directly got from the GPS server to a better interpretable format like KML[14] (XML based). The data from the server is a .gdb file and can be read by Garmin Mapsource [12] but it is much easier when we could read it automatically in a XML based format. This program handles these kind of conversions perfectly and has a GUI as well as a command-line based interface. We used the command-line interface because doing it this way is much easier for automation of the process instead of working with the GUI. GPSBabel is a open-source initiative and therefore free of charge.

ArcExplorer

ArcExplorer[8] is a light-weight GIS data[33] viewer for viewing, navigating and querying the GIS data. Very easy in use and written in java. We have used this program because UGent works with this program and because is the program for viewing GIS data. It's very straight forward to learn and to view the data. It's a bit more difficult to handle the data further but the main objective was to view the data. ArcExplorer is free of charge.

ArcMap

ArcMap is a part of the ArcGIS Desktop[9] software. Everybody can get a 60 day evaluation CD so I only used the software for 60 days but this was long enough to do everything I wanted to do. The software is used for knowing the direction of traffic flow. You could assign each road segment an arrow which indicated the traffic flow. This could not be done by ArcExplorer

which was too light-weighted for these kind of actions. The arrows were a great aid in testing the algorithms.

3.6 Details

In this section we describe more in detail which classes are the most important and we describe the most important functions described in these classes.

3.6.1 Overall view

The main algorithm is described in the file Beads.java. In this file you can call the function *calculatePath(Integer routeId)* to calculate the path of the GPS points with id *routeId*. The algorithm will then combine all steps. At first it will retrieve all GPS points from the database with id *routeId*. It will then limit some of the road network by calculating the streets within each bead. Then it will calculate the scores associated with each street. To conclude it will calculate the k-shortest path and return the path with the highest score. The k-shortest path algorithm is defined in the file Dijkstra.java with the function *algorithmMultiDijkstra(Hashtable <Long,RoadWeightedData> roads,Long start, Long end)*. It is made static because there is no need to define variables or to make an instance of this object. The arguments are very straightforward nl. roads in the form of a *Hashtable* because you need the road data when you want to form a shortest path algorithm, and furthermore we need the *start* and *end* streetId for calculating the path from *start* to *end*.

The file Connectie.java is very easy to understand. It handles the connection between the postgreSQL database and the program. We work with a typical 3 layer structure. We've got the database, the actual program and in between a file which handles the queries. The name of the functions are chosen to be very clear and to say everything.
We tried to be as complete as possible when made a own data structure. Therefore we have overridden in most of the datastructures the following functions *equals()*, *compareTo()*, *toString()* and made in some cases a *Comparator* class to compare 2 objects of the same class.

3.6.2 ArcExplorer

As output we have chosen to produce a .shp file because UGent was used to work with these kind of data. For viewing the .shp file we used ArcExplorer (see Section 3.5). In Figure 3.3 you find an example route.

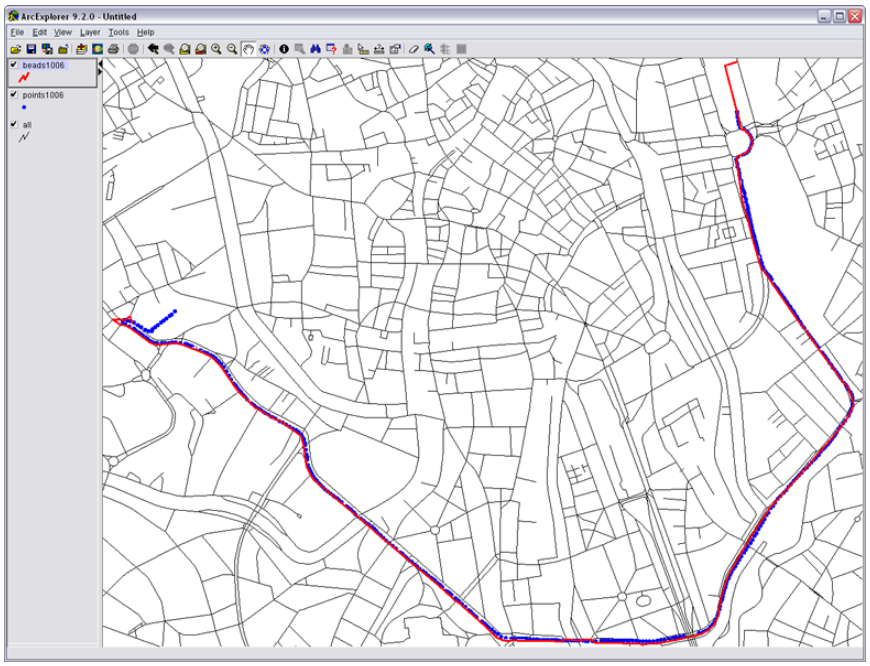


Figure 3.3: GPS points and their matching streets.

3.7 Database structure

3.7.1 Tables

- **streets:** This table contains all street information without extra information, just the .shp file inserted in the database.
- **completestreets:** This table contains all street information plus all extra information about a street, like the maximum speed, the length of the street, one-way or not.
- **gpspointsallinfo:** Here we find the GPS points with the right data types.
- **tmpsgpspoints:** This table is used for converting the GPS points from the original form (with almost all data types as *varchar*).
- **routes:** The computed routes are stored in here and are extracted from this table.

Chapter 4

Problems with solution

As with every project we came to some problems which where sometimes easy to solve but there also were problems which needed some special skills and knowledge to solve.

Choice of DBMS

One of the first issues was the choice of the DBMS. At the University of Hasselt we had experience with DB2[16] from IBM but because of network traffic we decided that the server could better be run on the localhost instead of a central server. Therefore we needed to find which version we could download for installing. There were way too many options available and we could not decide which was the best for installing on my computer. Another problem was that the server had to be installed at headquarters in Ghent because the project was made for them. Therefore we searched for a better, if possible open-source DBMS, we tried postgreSQL [36] of course with PostGIS extender (spatial extender for postgreSQL) and decided this was exactly what we needed!

Import .dbf file

For the extra information, for instance in which direction we could pass the street, we needed to input a dbf file, but postgreSQL does not support standard input from a dbf file so we searched for a program that converted the dbf file into SQL statements. We found an open-source program [1] but the program did not support the long type. We tried to adjust the source code but some of the basic functionalities where hidden in a jar file which was unadaptable. We had to go to a commercial product (EMS Data Import for PostgreSQL[43]) which supported the long type and could insert all these records (more than 365000) within a few minutes.

Displaying GIS data

When we think about GIS software we immediately think of Arc Explorer[8] the freeware software packet for displaying GIS data. During our education we only worked ones with this software packet. Learning to work with this packet was straight-forward. Just make a project and add layers and you could do basic things but we could not do more sophisticated stuff like display arrows according to the direction of traffic. We used ArcMap[10] instead for doing these kind of things. ArcMap needed more memory usage but the map was more intuitively in use thanks to the arrows which indicated the direction of the traffic. Another problem of ArcExplorer was that when we searched for the streetids we had to make a real query and not simply use the search function included in the packet.
We tried several open-source alternatives for displaying GIS data, one was JUMP[42] but as stated before here too we had the problem of displaying long objects and it could not be used.

Data

The data we used from TeleAtlas was in WGS84 coordinate system but the GPS coordinates got from the GPS devices were in Lambert72. One of the 2 coordinates needed to be transformed. UGent helped us by translating the WGS84 coordinates to the Belgian Lambert72 coordinates with which they were used to work with. Afterwards we needed to convert the converted shapefile to the postgreSQL database but there were several Lambert72 possibilities here too the expertise of the UGent was very useful in finding the right coordinate system. For the postgreSQL database the Lambert72 coordinate system was translated into the number 31370.

Start of routes

A frequent problem we had was that some of the police officers started recording when they still were at the parking lot of headquarters (see Figure 4.1). We solved this by making a polygon that contains the headquarters and when the route starts here, the street with id from 'Antonius Triestlaan' (10560001258396) has been chosen to start from because this is their main entrance.

Conversion WGS84 to Lambert72

Thanks to the postGIS extension of postgreSQL there is a function for transforming one coordinate system to another. However to convert from WGS84 (lat/long with srid:4326) to Belgian Lambert72 (srid:31370) there is mistake in the database which they promised to remove in the next version. There were a few other people with the same problem who tried to solve it [35]

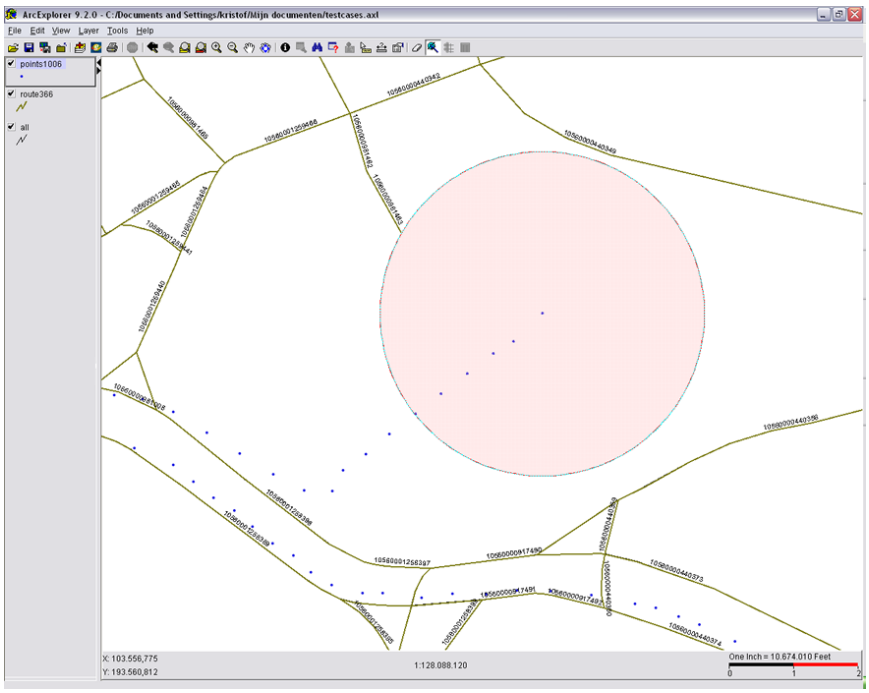


Figure 4.1: GPSPoints start at parking lot of headquarters, this area has been filtered out to start at road segment at the bottom left instead of closest roadsegment (see circle)

en [25] but I tried both references but both did not work. There is still a mistake of a few meters.

Bibliography

[1] Sebastian Baumhekel. DbasePsql. http://www.tv.com.pl/stepbystep/dbasepsql/ Last visited: 1 may 2007.

[2] J-P Beeckman. Wat men moet weten om zonder zorgen te navigeren met GPS. *Nationaal Geografisch Instituut.*

[3] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 853–864. VLDB Endowment, 2005.

[4] A. W. Brander and Mark C. Sinclair. A comparative study of k-shortest path algorithms. In *Proc. 11th UK Performance Engineering Worksh. for Computer and Telecommunications Systems*, September 1995.

[5] Juliana Castillo. Analysis and Implementation of K-Shortest Path Algorithms in Geographic Information Systems. In . University of Texas at Dallas, 2005. http://charlotte.utdallas.edu/mgis/prj_mstrs/2005/Spring/castillo/website/index.htm Last Visited: 1 may 2007.

[6] David Eppstein. Finding the k shortest paths. *SIAM J. Comput.*, 28(2):652–673, 1999.

[7] ESRI. http://www.esri.com Last Visited: 2 may 2007.

[8] ESRI. ArcExplorer GIS Data Viewer. http://www.esri.com/software/arcexplorer/index.html Last Visited: 1 may 2007.

[9] ESRI. Arcgis. http://www.esri.com/software/arcgis/about/desktop_gis.html Last Visited: 29 july 2007.

[10] ESRI. ArcMap map-authoring application. http://www.esri.com/software/arcgis/about/arcmap.html Last Visited: 1 may 2007.

[11] F. Marchal and J. Hackney and K.W. Axhausen. Efficient map-matching of large GPS data sets - Tests on a speed monitoring experiment in Zurich. 2004.

[12] Garmin. Mapsource. http://www8.garmin.com/cartography/ Last Visited: 29 july 2007.

[13] GeoPKDD. Techn. meeting munich. Internal report.

[14] Google. Kml. http://code.google.com/apis/kml/documentation/ Last Visited: 20 july 2007.

[15] Google. Maps. http://maps.google.com/ Last Visited: 26 july 2007.

[16] IBM. DB2 Universal DataBase (UDB). http://www.ibm.com/db2 Last Visited: 29 july 2007.

[17] Jose Macedo, Christelle Vangenot, Walied Othman, Nikos Pelekis, Elias Frentzos, Bart Kuijpers, Irene Ntoutsi, Stefano Spaccapietra, and Yannis Theodoridis. Trajectory data models.

[18] Bart Kuijpers. Knowledge representation and database problems for spatio-temoral data: a calculus approach to representing knowledge about trajectories., 2005–2008. http://www.uhasselt.be/english/onderzoek/groepen/teams_p_dep/abstract_project.asp?id=R-0920\&pnummer=1265 Last visited: 27 april 2007.

[19] Bart Kuijpers and Walied Othman. Trajectory databases: Data models, uncertainty and complete query languages. In *International Conference on Database Theory (ICDT)*, pages 224–238, 2007.

[20] Ron Lake. *Geography mark-up language: foundation for the geo-web (GML)*. John Wiley and Sons, Published 2004. ISBN 0470871547.

[21] Leen De Temmerman. Technical specification of GPS devices. Technical report, Ghent University, 2006.

[22] Robert Lipe. GPSBabel. http://www.gpsbabel.org/ Last Visited: 29 july 2007.

[23] Kraak M.-J. Geovisualization illustrated. *ISPRS Journal of Photogrammetry and Remote Sensing*, 57:390–399(10), April 2003.

[24] Magellan. Officiele website van magellan. http://www.magellangps.com/products/product.asp?segID=355\&prodID=1267 Last Visited: 20 july 2007.

[25] Maptools. forum. http://lists.maptools.org/pipermail/proj/2006-October/002601.html Last Visited: 29 july 2007.

[26] A.J. Mason. Emergency vehicle trip analysis using gps avl data: A dynamic program for map matching. 2005.

[27] David Megginson. Simple api for xml. `http://sax.sourceforge.net/` Last Visited: 20 july 2007.

[28] Harvey J. Miller. A measurement theory for time geography. *Geographical Analysis*.

[29] MySql. Mysql dbms. `http://www.mysql.org/` Last Visited: 29 july 2007.

[30] Stefano Spaccapietra Natalia Andrienko, Gennady Andrienko Nikos Pelekis. Basic concepts of movement data.

[31] Garmin Nederland. Officiele website van garmin nederland. `http://www.gps-garmin.nl/gps60.html` Last Visited: 20 july 2007.

[32] Netbeans. Netbeans. `http://www.netbeans.org/` Last Visited: 1 august 2007.

[33] OGC, Open Geospatial Consortium. Open Geospatial Consortium. `http://www.opengeospatial.org/` Last Visited: 27 july 2007.

[34] Octopus plan. `http://www.octopusplan.be` Last Visited: 20 may 2007.

[35] PostGIS. forum. `http://postgis.refractions.net/pipermail/postgis-users/2006-October/013570.html` Last Visited: 29 july 2007.

[36] PostgreSQL. . `http://www.postgresql.org/` Last Visited: 1 august 2007.

[37] Ralf Hartmut Güting and Markus Schneider. *Moving Objects Databases (The Morgan Kaufmann Series in Data Management Systems) (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[38] Refractions Research. Spatial Extender for PostgresSQL. `http://postgis.com/` Last visited: 1 may 2007.

[39] Sundarapandian V. Rink K.A., Rodin E.Y. A simplification of the double-sweep algorithm to solve the k-shortest path problem. *Applied Mathematics Letters*, 13:77–85(9), November 2000.

[40] Jose L. Santos. K shortest path algorithms, 2006. `http://www.dis.uniroma1.it/~challenge9/papers/santos.pdf` Last Visited: 20 july 2007.

[41] Sean R. Eddy. What is a hidden markov model. *Nature Biotechnology 22, 1315 - 1316*, 2004.

[42] Vivid Solutions. Jump unified mapping platform. `http://www.vividsolutions.com/JUMP/` Last Visited: 1 may 2007.

[43] SQLManager.net. Ems data import for postgresql. `http://www.sqlmanager.net/en/products/postgresql/dataimport` Last Visited: 20 july 2007.

[44] Java sun. Java. `java.sun.com/` Last Visited: 1 august 2007.

[45] George Taylor. Road reduction filtering for gps-gis navigation. *Transactions in GIS*, 5:193–207(15), June 2001.

[46] Wikipedia the free encyclopedia. A* search lgorithm. `http://en.wikipedia.org/wiki/A-star` Last Visited: 29 july 2007.

[47] Yannis Theodoridis and Dimitris Papadias. Range queries involving spatial relations: A performance analysis. In *Spatial Information Theory*, pages 537–551, 1995.

[48] Edward R. Tufte. *The visual display of quantitative information.* Graphics Press, Cheshire, CT, USA, 1986, ISBN: 0-9613921-0-X.

[49] Stefanie Andrae und Stephan Winter. Summarizing GPS trajectories by salient patterns, Technikum Kaernten, Villach, Austria 2 Department of Geomatics, The University of Melbourne, Australia.

[50] Vassilios S. Verykios, Maria Luisa Damiani, and Aris Gkoulalas-Divanis. Privacy and security in spatio-temporal data and trajectories, GeoPKDD.

Appendix A

Survey

ENQUÊTE ROUTEKEUZES DOOR HULPDIENSTEN

Mijn doctoraatsonderzoek aan de Universiteit Gent (vakgroep geografie) heeft als doel de routes gevolgd door hulpdiensten te vergelijken met de routes weergegeven door routeplanners. Het is de bedoeling om uit deze vergelijking aspecten te halen die van belang zijn voor hulpdiensten, maar die routeplanners niet in rekening brengen. We denken hierbij bijvoorbeeld aan het vermijden van bepaalde straten wegens te veel verkeerslichten, verkeersdrempels, file etc. Het is dus absoluut NIET de bedoeling om te achterhalen of de 'beste' weg is gevolgd of om gemaakte routekeuzes te evalueren; integendeel, we gaan trachten om uit de praktijkervaring suggesties te doen om tot betere routeplanners te komen.

De gereden trajecten worden geregistreerd via GPS; van belang zijn enkel de ritten **NAAR** een interventie, dus niet de terugrit naar de thuisbasis.
De gegevens worden volledig anoniem verwerkt.
Per interventie dient 1 vragenlijst te worden ingevuld.
Voor verdere inlichtingen, opmerkingen of vragen kunt u mij steeds contacteren:
Leen De Temmerman, Krijgslaan 281 S8, 9000 Gent, 09/2644636.
Leen.DeTemmerman@UGent.be

*****ALVAST HEEL ERG **BEDANKT** VOOR UW MEDEWERKING!*****

VRAGENLIJST (1 enquête per interventie)

De vragenlijst bestaat uit 2 soorten vragen: open vragen en meerkeuzevragen. Indien bij deze laatste meerdere keuzes mogelijk zijn, is dit vermeld.

1) Nummer toestel (zie etiket op toestel - omcirkel): nr 1 nr 2

2) Straatnaam (+ eventueel ter hoogte van welk huisnummer of bij autosnelwegen ter hoogte van welke km-paal):

VERTREKPLAATS (dit kan Antonius Triestlaan zijn, maar ook andere lokatie) :
……..

AANKOMSTPLAATS (= plaats van interventie):……..

GEREDEN AFSTAND: …….. km

3) Tijdstip in uur (van 0 tot 23u) en minuten van:

VERTREK (tijdstip van oproep): ……..u .....min

AANKOMST (op interventie): ……..u .....min

4) Datum (dd/mm/jj): ……./……./ 06

Zie ommezijde aub

5) De rit is:
- O prioritair (met zwaailichten en/of sirenes)
- O niet-prioritair (zonder zwaailichten of sirenes)

6) Weersomstandigheden (meerdere keuzes mogelijk):
- O regenval
- O sterke wind/rukwind
- O mist – zichtbaarheid < 100m
- O hagelbui
- O normaal (droog)

7) Staat van wegdek:
- O droog
- O nat/plassen

8) Hoe hebt u deze route gekozen (meerdere keuzes mogelijk):
- O intuïtief (op het gevoel de juiste richting gevolgd)
- O ervaring/kennis (ik ken de weg van vertrek- naar aankomstplaats)
- O kaart/stratenatlas
- O GPS (afzonderlijk toestel of geïnstalleerd in wagen)
- O richtlijnen gevraagd/gekregen via radiocontact
- O andere – specificeer: ……..

9) Waarom hebt u specifiek déze route gekozen?
……..……
……
……
……..

10) Waren er speciale omstandigheden (bv. wegomlegging) waardoor u een andere weg hebt gevolgd dan dat u van plan was?
……
……..
……
……..

11) Hoe goed is uw kennis van het wegennetwerk in het gebied tussen vertrekplaats en aankomstplaats? Omcirkel het cijfer dat best bij deze rit past.

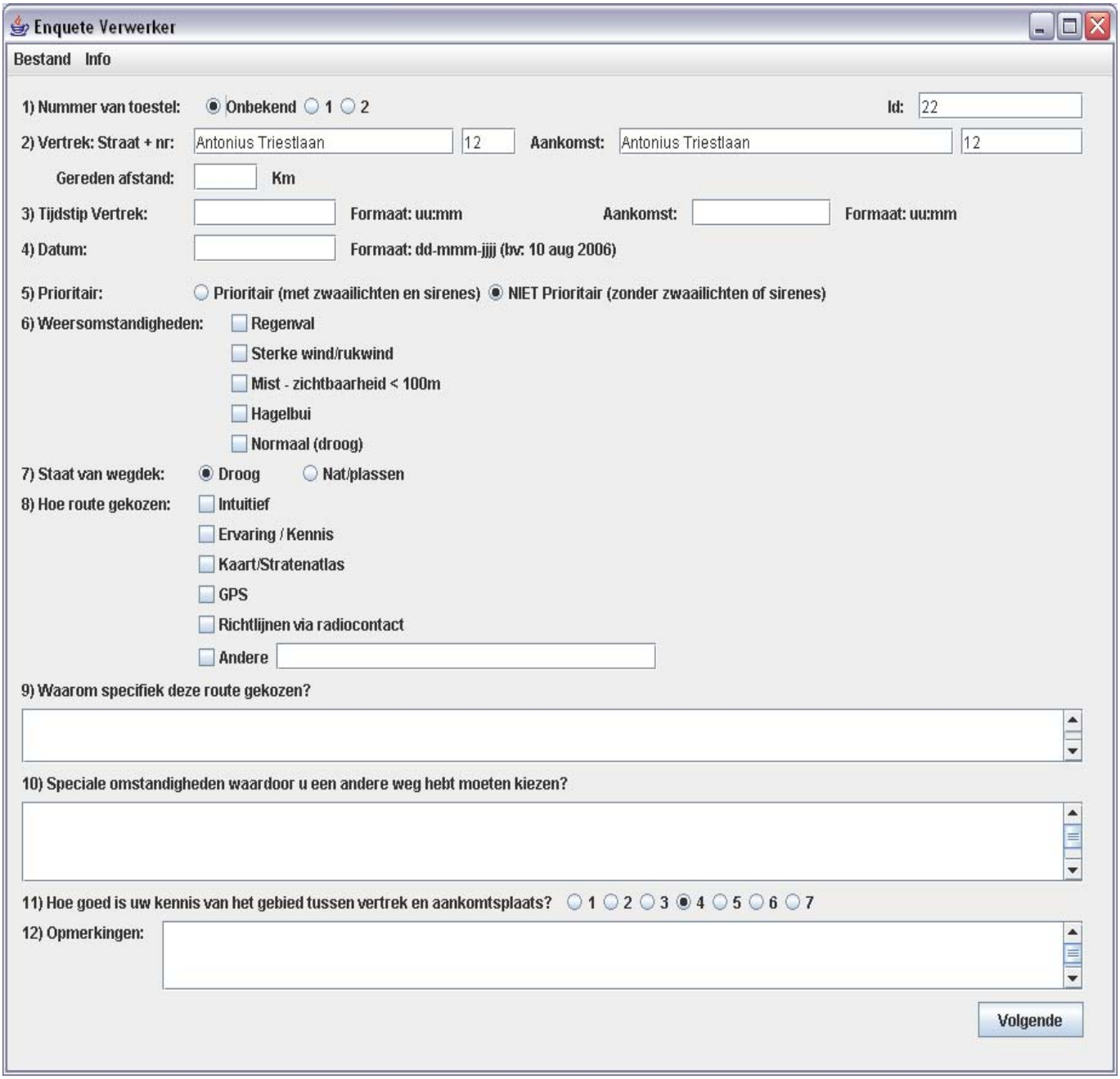
Ik ben niet vertrouwd met dit gebied *Ik ben volledig vertrouwd dit gebied*

1 2 3 4 5 6 7

12) Indien u opmerkingen of suggesties hebt, gelieve deze hier te vermelden.
……
……
……
……
……
……..

Appendix B

GUI


Enquete Verwerker
Bestand Info
1) Nummer van toestel: Onbekend 1 2
Id: 22
2) Vertrek: Straat + nr: Antonius Triestlaan 12
Aankomst: Antonius Triestlaan 12
Gereden afstand: Km
3) Tijdstip Vertrek: Formaat: uu:mm
Aankomst: Formaat: uu:mm
4) Datum: Formaat: dd-mmm-jjjj (bv: 10 aug 2006)
5) Prioritair: Prioritair (met zwaailichten en sirenes) NIET Prioritair (zonder zwaailichten of sirenes)
6) Weersomstandigheden:
Regenval
Sterke wind/rukwind
Mist - zichtbaarheid < 100m
Hagelbui
Normaal (droog)
7) Staat van wegdek: Droog Nat/plassen
8) Hoe route gekozen:
Intuitief
Ervaring / Kennis
Kaart/Stratenatlas
GPS
Richtlijnen via radiocontact
Andere
9) Waarom specifiek deze route gekozen?
10) Speciale omstandigheden waardoor u een andere weg hebt moeten kiezen?
11) Hoe goed is uw kennis van het gebied tussen vertrek en aankomtsplaats? 1 2 3 4 5 6 7
12) Opmerkingen:
Volgende

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Map matching tracking data
Richting: **Master in de informatica** Jaar: **2007**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

Kristof Ghys

Datum: **20.08.2007**