Masters' Degree in Informatics Engineering
Dissertation
Final Report

AIDA - Trajectory prediction

João Alexandre Marques Rodrigues
jarod@student.dei.uc.pt

Advisor:
Dr. Francisco Câmara Pereira

Date: July 12, 2011

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

The AIDA project (Affective, Intelligent Driving Agent), a collaboration between Volkswagen of America and the Massachusetts Institute of Technology, is an intelligent driving navigation system with a robotic interface that aims to change the way people interact with the cars. In AIDA, a fundamental challenge is the problem of *Trajectory Prediction*, that consists in finding the route that the driver is more likely to choose to travel between the current location and a specific destination. AIDA's trajectory prediction model is able to predict routes with 92% of accuracy. However, it was only tested with three different drivers and, furthermore, the algorithm used shows some limitations. This thesis aims to statistically validate this model by testing it with more subjects, and to design a new and more realistic approach. In this report we present a description of this new algorithm, all the data processing methods used and the experiments performed on both original and new approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we will start by explaining the impact that the new generation of navigation systems could have in our daily lives, that represents the main motivation behind this thesis. Thus, in Section 1.1, we will analyze and discuss some facts about the driver's daily routines and limitations of the current GPS devices.

Then, in Section 1.2, we will present the AIDA project as one of these recent GPS systems and describe its concept and internal architecture. Since all this thesis's work belongs to this project, we will end the chapter by talking about what was already performed and what goals we defined.

1.1 Motivation

In our daily lives, much of our driving is routine and it is easy to detect patterns since we frequently go to the same destinations and follow the same routes over and over again. In fact, Krumm and Froehlich [20] found out that, for drivers observed during at least 40 days, about 60% of their future trips are repeated in the observations. Even if there are shorter or faster ways to reach the destinations, the drivers tend to stick with the same routes used in the past. For example, when a driver is traveling from work to home, there could be several reasons for him not to choose always the shortest path, like he might want to pass by someplace previously or due to traffic reasons.

Thus, by collecting and analyzing GPS traces of a driver, it should be possible to detect the patterns described above and predict what will be the driver's next destination and route taken in a specific time of a day. This information can then be used to improve the current navigations systems. These devices know how to compute and show the driver the shortest path for a given destination and some are also capable of displaying information about traffic and points of interest (POIs) nearby. However, as was explained above, the shortest way is not always the preferred by the drivers and without knowing their behaviors and preferences, the help these devices can provide becomes essentially limited to help the driver travel in unknown areas.

In contrast, if a navigation system knows beforehand what a driver's route and destination will be, it can provide better route guidance without forcing the driver to input any kind of information at all. For example, it should be able to warn the driver about the traffic and conditions of the roads that will next be taken and give different pieces of advice depending on them. Besides anticipatory driver warnings, these systems should be able to provide some other types of meaningful help, like showing the user environment information (e g. POIs, advertising) or automatically changing the vehicle's behavior (e g. automatic turn signals, cylinder deactivation, pre-braking). In fact, if a driver's next route is already known, it is possible to improve hybrid fuel economy by up to 7.8%, as researchers from Nissan [7] show. Tate and Boyd [30] also explored the benefits of knowing the route in advance in the hybrid vehicles control scheme.

So, during the recent years, the driver's trajectory and destination prediction research fields have been improving rapidly, in part due to the advantages we can take from them in the new car navigation systems. In this thesis, all of the work presented is integrated in the AIDA research project [1], that tries to combine these two fields in the best way in order to built an intelligent navigation system with similar features to the ones described above. In Section 1.2, we will describe the purposes of this project in detail, as well as its architecture, the work done so far and this thesis's goals.

1.2 AIDA project

The AIDA project (Affective, Intelligent Driving Agent), a collaboration between Volkswagen of America and the Massachusetts Institute of Technology, is an intelligent driving navigation system with a robotic interface that aims to change the way people interact with the cars. Over time, AIDA analyses the driver's mobility patterns, common routes and destinations. Within few weeks, it will have figured out, for example, the driver's favorite parts of the town, what time he gets off work or what he likes to do on the way back home. By merging this kind of information with car sensors and knowledge about the city such as road conditions, traffic, points of interest or gas stations, AIDA can provide precious suggestions and recommendations of the driver's interest.

For example, let's imagine a scenario where the driver just entered inside his car, on a week day, in his garage at home. As it is 9 o'clock in the morning AIDA predicts, with an high accuracy, that he is going to his work and that he will choose Route A to reach that destination. Moreover, in this scenario, AIDA just received real time information about a car accident that is expanding the traffic in Route A. In this case, even before he turns on the engine, AIDA can help the driver by suggesting him alternative ways to reach his work, in order to avoid Route A's traffic, save time and get to work on time.

One of the most innovative aspects in this system is the use of facial simulation for the communication, which is a very quick way to express information since it can be easily interpreted by the humans. AIDA communicates with the driver through a small robot embedded in the dashboard that expresses emotions through facial expressions (e.g. smile or by blink an eye). Over time, it is expected that a relationship develops between the driver and AIDA, where both learn from each other and establish an affective connection.

1.2.1 Architecture and work done

There are two groups currently working on AIDA: The Personal Robots lab and the SENSEable City group. The former is responsible for building and designing the robot while the latter is working on intelligent navigations algorithms and is divided in four main components: Driver behavior model, Destination and trajectory prediction, Dynamic routing algorithm and Semantics analysis of the city.

The Destination and trajectory prediction component is the one that shows more improvements until today since there are already two works performed on those areas that show promising results and conclusions. Di

Lorenzo [8] presents an appropriate destination prediction algorithm for AIDA with an average of 4.4 Km of error while Correia and Câmara, in Correia's master thesis [6], suggest an algorithm for this project that is able to predict a driver's route given a destination with 92% of accuracy. However, in the latter, the tests were performed with only three different drivers and the algorithm shows some limitations, that will be particularly discussed in Section 5.1. The ideas behind both these two algorithms will be deeply analyzed in Chapter 3.

1.2.2 Goals

The work that we will present in the next chapters is part of AIDA's destination and trajectory prediction module. One of the fundamental challenges in this component is the problem of *Trajectory Prediction*, that consists in finding the route that the driver is more likely to choose to travel between two points. Thus, the first goal of this thesis is to statistically validate Correia and Câmara's trajectory prediction algorithm [6], since the authors only tested their approach with just three different drivers. We want to study the behavior of this algorithm with a new dataset (presented in Chapter 4) and verify if the results obtained are as promising as the original ones (Section 6.2).

However, as was mentioned above, this algorithm has some limitations. Thus, our second goal is to design and test a more realistic algorithm, capable of maintaining or even improving the results obtained by the original one. This subject will represent our main concern in this thesis, so it is our intention to present a deep study on it, from the preliminary results to the worst case scenarios (Section 6.3).

Finally, we defined a third tentative objective, pending on overall progress, that consisted in exploring the destination prediction module and different ways of integrating it with the trajectory prediction one. As we will understand from Section 2.1, we only made a superficial exploration in this subject, given the much higher priority of the others and their considerable complexity.

Chapter 2

Methodology

In most of the research projects it is difficult to plan a whole year of work because unexpected results might be obtained and new discoveries can influence the next steps. So, in our intermediate report [26] we presented more than one possible working plan. In Section 2.1 we will explain which plan we actually followed and why.

Since this is an exploratory work, the type of development used somewhat differs from the traditional software development methodologies, like the waterfall model [35]. Thus, in Section 2.2 we will explain and talk about the working methodology used during this thesis.

2.1 Working Plans

After we deliver the intermediate report [26], the main ideas behind the new algorithm had already been discussed, most of the data had already been processed and the first goal, that corresponds to the statistical validation of Correia and Câmara's work [6], was almost completed. So, in that report, we presented two different possible working plans, since the next steps would depend on the experiments performed in the new algorithm.

In our Plan *A*, we would complete all these experiments in two months and thus, we would have time to start exploring the integration between the destination and trajectory prediction modules. Plan *B* (Figure 2.1) was more realistic since it was only focused in the new trajectory prediction algorithm, that is the main goal/concern of this thesis. In this plan, we reserved time to analyze the worst case scenarios and implement possible improvements.

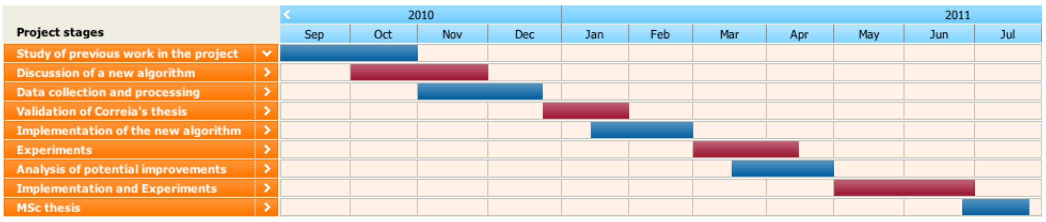

Figure 2.1: Gantt chart for Plan *B*.

Plan *B* was the one that we actually followed. The new approach revealed the need for a further in-depth exploration of our ideas. While they represented a much more realistic perspective than the earlier version, its design and calibration was more demanding. So, in that point, we decided to use the remaining time testing our algorithm in different ways, exploring the worst case scenarios and analyzing and implementing possible solutions/improvements. In conclusion, we preferred depth over breadth. I.e., instead of getting "yet another" functionality in the project, we decided to evolve the current one up to the maximum possible.

2.2 Development Approach

In a research project it is important to discuss and share opinions frequently in order to merge different ideas and negotiate directions. Therefore, in this thesis we chose an exploratory development approach, somewhat similar to an agile model [34] in a software project development.

From September to March, there were weekly meetings where the work developed during the week was reported and some different topics could be discussed. In the first ones the main priority was to define the thesis goals and the strategy for the rest of the year, while in the following the concern was in collecting new GPS traces and define different kinds of data filtering and processing techniques (Chapter 4) in order to build traces with the best quality possible.

After defining all these procedures, the meetings started to be focused in the new algorithm. They were more like brainstorm sessions where each person was supposed to bring new ideas and share the research work performed during the week with the rest of the team. These ideas were discussed and merged until we obtained a final algorithm, described in Chapter 5. Then, from the beginning of the experiments to the deadline of this thesis, the purpose of the meetings was only to discuss and interpret the results previously obtained and define the next steps and tests to perform.

After the end of March, when Francisco Pereira (my advisor) left Portugal, we started to meet only two times per month, via video conference. These meetings started to happen more sporadically, mostly due to the huge time difference between Portugal and Singapore (7 hours) and the difficulty in finding compatible slots. However, as most of the next steps and tests had already been defined and the thesis was already in its final phase, this fact did not affect the final outcome.

Chapter 3

State of The Art

In this project it is crucial to understand the driver's reasoning in order to simulate an appropriate representation of the mental city map of a driver, that is the base of our trajectory prediction approach. In Section 3.1 we will talk about some psychological studies that try to understand why these maps usually differ a lot from the real ones.

Then, in Section 3.2, we will examine previous work made on the trajectory prediction research field and particularly analyze Correia and Câmara's distortion algorithm [6], since it represents the first iteration of AIDA's trajectory prediction module and part of this thesis work is focused on it.

Finally, in the last section, we will take a look into some destination prediction algorithms that have been developed during the past years, specially Di Lorenzo's algorithm [8], that was used in AIDA's destination prediction module.

3.1 Spatial Cognition

Spatial Cognition is a research field that studies the way humans and other animals perceive and represent space in their heads. In this case, space refers mainly to cities, towns, metropolitan areas or other possible entities that can never be seen in their entirety. Because of their proportion, it is almost impossible for a person to perceive and represent such an entity in perfection, so the presence of distortions in this kind of mental representations is unquestionable. It is important to be aware that these distortions can be presented in both memory and perceptual representation [16].

Cognitive Map is the usual metaphor for the human's mental image of the maps of such environments. These maps differ from the real ones because they are distorted in different ways regarding each person. In fact, those representations are so complex that can not be defined with only this metaphor, as Tversky [32] states. She believes that there are sorts of non-maplike information that are also distorted and can be used to make inferences about spatial proximity. The author refers to these cases as *Cognitive Collages*, where *Collages* symbolize thematic overlays of multimedia that contain those types of information, such as figures, facts about distances or directions or even knowledge about the climate and time zones. *Spatial Mental Models* is the other metaphor suggested in her work and it is used in situations where the environments are simple and people do not preserve the metric information but do preserve most of the spatial relations among landmarks.

For our work it is crucial to understand the source of the distortions that originate the mental map of each person. In a study performed in some North America cities, Stevens and Coupe [28] realized that people tend to store in their memories not the exact location of all cities, but rather the relative location of the states, calling this type of distortion *hierarchical organization*. In a different study, Holyoak and Mah [13] placed two different groups of students in different sides of the american coast and asked them to estimate distances between some american cities. They realized that the judgments were distorting depending on which side of the coast they were, so different *Cognitive Perspectives* happened.

In 1981, Tversky [31] also detected other types of distortions like *Rotation* (people tend to rotate spatial locations so that they will correspond to the exact N-S frame of reference) and *Alignment* (two nearly-aligned locations tend to be remembered as more closely aligned than they actually were). In fact, judgements of spatial relations might be affected by political [22] and semantic [12] reasons too.

However, none of these studies seeks to understand the driver's reasoning and why some paths are preferred among others. The driver's route choices can make some specific routes be imaged as longer or shorter in the mental map and this fact is a source of distortion too. In 1995, Golledge [10] tried to understand the criteria most often used in route selection. Thirty-two subjects were selected and experiments were performed both in laboratory and real environment. In the end, the results showed that the top three criteria were shortest distance, least time and fewest turns. The author also states that the route choice varies depending on the direction traveled. This matches with Brunye's results [4] that showed that drivers prefer routes towards South and perceptually classify the North choices as being *uphill*. Similar studies [3] show that least-angle routes with an initial straight segment are often chosen. These results represent important clues for what should be AIDA's trajectory prediction algorithm, since the better we can understand the driver's reasoning, the better we will simulate and understand his mental map of the city.

In addition it should be mentioned that, besides this work, there are others that use a graph distortion approach (acting on the edge weights) followed by the application of a shortest path algorithm, but they all have different purposes.

Duckham and Kulik [9] used this procedure to generate the *simplest* route to a destination, in other words, the route that is easiest to explain, understand and memorize. In order to minimize the complexity of a route description, each pair of connected edges has an assigned weight that represents the amount of information required to negotiate their intersection. If the *decision point* is complex (e.g. T-junction) the weight will have an high value, otherwise it's value will be lower. After distorting the graph, they simply apply a shortest path algorithm and the *simplest* route is returned.

The same methodology is used in the works presented by Haque [11] and Caduff [5]. The former was inspired by Duckham and Kulik's work [9] and it only differs in the weighting function (in Duckham's work the weight reflects the instruction complexity of a *decision point* and in Haque's the weight describes the ambiguity of each turn at an intersection point). In the latter, each node of the graph has nearby landmarks associated with it. In this case, the weight of each pair of connected edges will depend on those landmarks' saliency. The distance/orientation of a driver with respect to them is also taken into consideration. In the end a revised version of Dijkstra is applied to the graph, returning the *clearest* route in terms of spatial references.

These works, besides their different goals, inspired AIDA's trajectory prediction algorithm, that will be analyzed in next section.

3.2 Trajectory Prediction

Trajectory prediction is a large research field that attempts to predict future trajectories of moving objects in several environments. Before taking a deep look into AIDA's trajectory prediction algorithm [6], we should first analyze some important works in the field, specially the ones that focus on the driver's context.

Karbassi and Barth [14] attempted to predict the route between two stations in a shared-use vehicle's system. They collected GPS traces during three years in order to create route frequency histograms between all pairs of stations during three periods of the day: morning, noon and afternoon. So, before entering the car, the users of this system should insert both origin and destination stations on the station kiosk touchscreen. Then, during the trip, as new location data were received (GPS positions every 30 seconds), a hierarchical tree data structure was used to recalculate the most probable route for the period of the day, based on the histograms described above and links traversed so far.

In 2007, Sang-Wook Kim et al. [15] proposed a similar approach to the one described above. Assuming that a driver started a trip from location S and reached the location C by a certain path, moving towards the final destination D, their algorithm starts by looking at all past trajectories taken from S to C. Then, for each one of them, their paths from C were investigated and the most frequent path (the one whose frequency is higher) that passes by the destination D is returned. The authors were not able to collect real-life trajectories, so they did not conduct performance tests.

The main limitation in the two works described above is the fact that they both require a database with a driver's past trajectories and routes. This can lead to enormous databases and affect the performance of the algorithm, besides the fact that it is impossible to predict a driver's trajectory from scratch (without acquiring prior knowledge), something that our algorithm should be capable of. Moreover, it should also be able to predict both the trajectory and destination without any user input and for a complete city map, in contrast to Karbassi's work.

There are also some authors that used pattern recognition techniques to predict future trajectories. Simmons et al. [27] trained a Hidden Markov Model with past routes and destinations in order to predict the next road segments taken by a specific driver until the destination arrival. They tested their approach on a driver with 46 trips, achieving an accuracy often above 98%. However, this is a misleading result, because 95% of the driver's road segments end points were connected to only one other road segment. So, the transition was forced and as there was only one option the prediction had to be correct. Thus, the relevant result in this work is the average of 75% of accuracy obtained when the transitions are unforced, that corresponds to 5% of the total predictions.

A similar approach was used in Krumm's work [19], where he used a simpler Markov Model (instead of a Hidden Markov model) to predict the chain of road segments that a driver will next take. In contrast to Simmons's approach, in this work only about 28% of the segment transitions are forced, which makes the results more believable. They show that the accuracy for predicting the next segments rises with the number of past segments matched and drops the farther the model looks into the future. For instance, 90% of accuracy is obtained when only predicting the next, single road segment, 65% when predicting the next five and only 50% when predicting the next ten segments.

In addition, both authors tried to add information about the time of the day and the day of the week to the models to see if it would increase the accuracy and both obtained negative results. However, Simmons et al. [27] found out that the current speed could be a significant factor.

In 2008, Krumm and Froehlich [20] tried to guess a driver's entire route, instead of making road segment predictions like in the last two described works. They gathered GPS driving data from 252 drivers resulting in 2.2 million GPS location points with an average of 15.1 days of worth driving data per user. The authors state that a trip is different from a route, since the trip describes a path through time and space and a route is a only collection of latitude and longitude pairs and lacks the time component.

Their algorithm starts by combining similar past trips into routes. In order to do that, they compare every trip in a driver's trajectory data and store the similarity scores between them in a *trip similarity* matrix. Then, they use a hierarchical clustering technique that recursively combines trips until the lowest score in the matrix exceeds a specific threshold. The similarity score between two trips is calculated by computing the average minimum point-segment distance between them (a version of the Hausdorff distance algorithm).

Thus, as a trip progresses, the distance between the current trip and all the driver's routes (that were calculated using the clustering method referred above) is updated in another *similarity* matrix. Then, two algorithms can be applied: the *Closest match algorithm* returns an ordered list with the most similar routes to the current trip and the *Threshold match algorithm* returns for each route a confidence measure given the trip's distance so far.

The results show that by the end of the first mile the percentage of repeated trips correctly predicted is 30% and that by that point, 70% of the times the correct route is already in the top 10 closest ones (increasing to 80% by the third mile). However, the results also show that by the first day, only 1.5% of the trips were considered to be repeated. Furthermore, we can observe that even after a month of route collection 40% of the trips are still over new routes so there will always be a lot of trips that will not have a correct match. These cases can not happen in our algorithm, since the system must be able to predict a driver's route even if it is new for him and that is one of the greatest motivations of our work.

Next, we talk about Correia's master thesis, that represents the first approach of AIDA's trajectory prediction algorithm [6]. In contrast to the works presented above, the idea of this work was to simulate the mental map of a driver by distorting the city graph acting on the edges weight. Then, like in Duckham's [9], Haque's [11], and Caduff's [5] works, a simple shortest-path algorithm (Dijkstra was his implementation) was applied in order to get the shortest path from the standpoint of a driver. Since shortest distance is the criteria most often used in route selection (Golledge [10], Section 3.1), the returned path should represent the one that the driver will choose.

For instance, if a couple of links are traversed 100 or more times in a month by a certain driver, it is because, probably, from that driver's standpoint, they represent the shortest ways to achieve specific destinations. Therefore, in order to make these links preferred over the others, their weight should be diminished. Thus, before the beginning of each trip, the weight of all the graph's edges is updated as the following:

$$w = (w_{max} - w_{min}) \times f(n_{tw}, N_{tw}) + w_{min}$$

where w_{max} corresponds to the highest possible value of an edge's weight (the same as the initial value) and w_{min} the lowest possible one, that depends on the "driver's reality perception factor" α that may vary between 0 and 1:

$$w_{min} = w_{max} \times \alpha$$

The function $f(n_{tw}, N_{tw})$ represents how an edge weight value decays with the number of times n_{tw} a driver transversed it (N_{tw} is a normalization factor). He tested this algorithm with different α values and six different decay functions: exponential, logarithms, sine, cosine, linear and inverse.

So, before the beginning of a trip, a set of probable destinations is returned by another algorithm [8], that will be deeply analyzed in Section 3.3. Correia and Câmara's algorithm starts by calculating the most probable route for each one of the destinations (the one with the shortest path in the distorted map). Then, during the trip, as the driver transverses a few links, some destinations will be ruled out of the set because their predicted routes can not be matched anymore by the driver. Anytime the set becomes empty, a new set of possible destinations and routes is computed.

As we can see, in contrast to Krumm and Froehlich's approach [20], this algorithm can predict a driver's route even if he never went through it. Even when the system is initialized and there are no graph distortions (every edge has w_{max} of weight), this algorithm achieves 86% (±7%) of accuracy when predicting the route that will next be taken. The results also show improvements of 6% after the map being distorted. However, the tests were performed with only three drivers from two cities, so the statistic evaluation is not a point in favor.

Moreover, we can detect some limitations in this algorithm. For instance, we can understand that it will never be able to represent the perfect mental map of a driver. Even if a driver chooses the same routes everyday, the distortions will never stop, the map will be changing infinitely and will never stabilize. This and another problems will be discussed in Chapter 5, where we will talk more deeply about the ideas discussed around this algorithm's limitations and the process we went through until reaching an improved distortion function.

3.3 Destination Prediction

In order to understand the idea behind AIDA's destination prediction algorithm [8], we first need to be aware of what inspired it and take a look into previous studies in this research field.

Terada et al. [29] proposed four different destination prediction methods that should be applied depending on the type of road driven on. The *Basic Method* returns the most probable destination by calculating the degree of concordance between the current route and every past trajectory stored in a database, using the following formula:

$$P_{ij} = (1 - \alpha)\frac{N_{ij}}{N_i} + \alpha P_{(i-1)j}$$

where P_{ij} represents the probability of a user at a specific road i going to destination j, N_{ij} the frequency of going to destination j by i and N_i the frequency of visiting edge i. The constant α may vary between 0 and 1 and controls the weight of considering past routes for calculating P_{ij}. The percentage of past drives that reached destination j will correspond to P_{0j}.

However, the authors noticed that the accuracy was decreasing when there was a use of alternative ways. Thus, for those cases, they decided to change the weight α dynamically calling it the *Alternative way Method.* In this new method, the α is not constant anymore and is defined be the following equation:

$$\alpha = \frac{N_{i-j}}{N_{i-1} + N_i}$$

The authors also propose two more methodologies: the *Departure Method* and the *Context Method.* In the former, the strategy was to use sorts of data of each departure to predict less commonly used destinations in situations of intense use of arterial ways. In the latter, they developed a similar approach to Patterson's [25] and trained a Bayesian network with driving context information (e.g. time of the day, the day of the week, weather or number of passengers) in order to predict the destinations.

This kind of driving context information was also used in Kostov's work [17], but with different goals. In this work, he started by organizing the driver's past trajectories in an *FP-Tree*, that allows an efficient search for moving patterns and facilitates the calculation of destination probabilities. Then, with the goal of predicting the driver's destination reliably, they proposed an algorithm that automatically assigns categories to each user based on the type of information referred above. For instance, if a user A often goes to a shopping on Saturdays and Sundays, it is assumed that *weekend* is a category to the user A. Likewise, if he has lunch one day at 12:50 and at 13:10 on another day, it is considered that his category for lunch time is around 13 o'clock. In the end, a group of common destinations is extracted by combining the tree data structure with these categories.

We can see in the results that the prediction accuracy increases as the number of history data used and the time since the engine started increases as well. So, when the engine is running for five minutes and the number of history data used is 400, they present an accuracy of approximately 75%, which demonstrates a good performance. However, the authors only applied this method on the history data of three subjects, which does not represent a strong statistical validation.

Besides Terada et al. [29], the use of pattern recognition techniques to predict a driver's destination has been explored by some other researchers as well. The works presented by Ashbrook [2] and Patterson [25] use a second-order Markov model and a Bayes network, respectively, to predict where the driver will go next, among a set of previous learned destinations. The main limitation in these works, is the fact that they both need to acquire prior knowledge of a driver's common destinations, like home and work. Although these ideas are interesting, AIDA needs a more complex algorithm that can work in a new car without knowing the driver's routines.

In 2006, Krumm [18] placed a grid 41 Km X 41 Km over the Seattle area, with each cell having 1 Km of weight. His approach is based in the idea that the time to reach a specific destination decreases monotonically as a trip progresses, which is not always true. The goal was to assign a destination probability to each cell on the grid by applying a Bayes rule. However, this algorithm can return cells that represent rivers, lakes or forests and that should not happen on AIDA's algorithm.

This shortcoming was solved by the same author and Horvitz [21], by using the United States Geological Survey (USGS) [33] to characterize each cell with ground cover information. In this way, cells that represent more attractive areas (e.g. commercial) have more probability to be chosen as destinations than the others (e.g. water, forest).

These authors use the same initial base but propose two different models for destination prediction. The *Closed-World* approach focuses on destinations that have already been visited by the driver, so the number of times a driver concluded a trip in each cell is computed. Then, a histogram over the cells is built and after the normalization they developed a probability function based on personal destinations. In contrast, the *Open-World* approach considers new destinations, that have not been seen before. Their approach is based on the intuition that the common destinations tend to cluster, in other words, the authors believe that they are always near each other so the driver can, for instance, save time. Thus, for each cell, a discretized probability distribution is computed over the distance from previously visited destinations.

These two models described above are further integrated with each other in order to create the main model. However, based on the results, this model is not able to predict unvisited destinations since it behaves essentially like a *Closed-World* model.

In some points Di Lorenzo's work [8] is similar to the one described above since it also uses square grids to modeling space and two different models for two different *Worlds*. Nevertheless, the ideas behind these two models are not the same and, in addition, this algorithm is capable of adaptively change the group of most probable unvisited destinations.

Therefore, in AIDA's destination prediction algorithm [8], the *Routine Model* corresponds to the scenario where the driver is traveling to a known destination on his daily routine, like home or work. Thus, the driver's driving history is extracted *a priori* and is used to train a probabilistic predictive model. In this case, the Markov model was chosen because there is access to temporal sequences. Then, in order to filter the set of possible destinations, an *expected trip time* t is computed and only those cells that are t time unit away from the origin stay in the set. Furthermore, as this model only focuses on visited destinations, the new ones are eliminated from the set.

The *Non-routine Model* is somewhat more complex than the former. As there are not enough temporal sequences to unvisited destinations, a Bayesian rule is applied over other relevant data, that takes into account the purpose of the trip and the familiarity with a particular area.

Given the day of the week and the time of the day, some temporal patterns can be extracted from the history of a driver, such as his time to lunch or to go to the gym. Thus, we can guess the purpose of most of trips. For instance, if a driver is not driving home at his lunch time, we know that probably he is going to some restaurant nearby. Therefore, in this approach, information about Points of Interest (POI) in each cell is used in the data training, so that the probability of a driver being at a specific POI category, in a certain time of a day, in a specific cell can be computed.

Moreover, if a driver is exploring a new area, information about its land use and population density is used to filter destinations, since drivers that are not familiar with a specific region tend to visit popular places. This filtering also discards the undesirable cells that represent ocean, forest or desert because they typically have zero human population.

The results show that the destinations can be predicted accurately with an average error of 4.29 Km. It is also noticeable that the choice of a Markov model in the *Routine Model* was appropriate because it yields a lower average error compared to the Bayes approach used by Krumm and Horvitz [21] in the *Closed-World* (3.93 Km vs 4.37 Km).

Although it has still many points to improve and future work ahead, the algorithm presented above is the one that was integrated, in a first phase, in AIDA's destination prediction module.

Chapter 4

Data Collection and Processing

As was explained in Subsection 1.2.2, the first part of this thesis consists in the statistical validation of AIDA's original trajectory prediction algorithm. The dataset used by Correia and Câmara to test this approach was just composed by three different drivers, two from Coimbra (Portugal) and one from Seattle (WA, USA). In the other hand, our new dataset is composed by GPS traces collected in San Francisco (CA, USA) by eight users in a period of one year (from May 2008 to May 2009).

So, our first goal is to test the original approach with the San Francisco dataset and compare the results obtained with the ones got by Correia and Câmara. In the end, if we succeed in the experimentations, we will have the algorithm validated, since we believe that the fact that it shows improvements in eleven different drivers from three different environments is enough to conclude that it can be used as a valid trajectory prediction algorithm. The same reasoning is applied to the second part of this work, where we intend to test the new approach with the same eleven drivers.

Both datasets recorded a broad range of users' outdoor movements, including not only home-work trips but also some extra-routine entertainments. However, they give no indication of when a trip begins or ends and AIDA's trajectory prediction algorithm needs knowledge of a driver's discrete trips to yield results. Moreover, some GPS points are noisy and contain invalid sensor data.

In the next four sections we will talk about the steps we performed in order to extract clean traces from the GPS raw measurements of both datasets.

Then, in Section 4.5, we will present some statistics and data comparisons and finally, in Section 4.6, we will talk about the limitations of our map matching algorithm and how they affected our traces.

4.1 Pre-processing

After some analysis of the original GPS raw data files, we found out that most of them were a variation of the NMEA format [24], where the coordinates are given in minutes and seconds. Thus, the first step was to convert the coordinates in degrees and compute a timestamp from the hour and the day provided, this way converting the data in a *latitude,longitude,timestamp* format. Next, each user's GPS data was sorted chronologically, by timestamp.

In the end an algorithm was processed for each user's datafile in order to understand what was the most appropriate bounding box for each subject. This last step is very important because we can not load, for instance, all the map of the state of California to our database (due to system memory problems), so it is crucial to find out which is the bounding box that best fits each driver's traces.

4.2 Trip Segmentation

The algorithm used to segment the GPS points in trips looks for gaps between two consecutive recorded points ($P1$ and $P2$) of 2 minutes or more. If a gap is found, $P1$ becomes the end point of the last trip and $P2$ the begging point of the current trip. The threshold value of 2 minutes, like some other GPS data processing techniques, was inspired by the work performed by Krumm and Froehlich [20], where they used GPS traces collected from Seattle (WA, USA).

In their work, they explain that a threshold value of 3 minutes would be optimal. However, after running some tests, we concluded that there was no problem in reducing the threshold in one minute. After a brief data analysis, we realized that during a common trip, the GPS points were being extracted with a very high frequency (from 1 do 5 seconds), so the 2 minute gap is enough to understand when a new trip begins.

4.3 Trip Cleansing

Every GPS trace contains noisy points that represent outliers and our dataset is no exception. So, once the traces were segmented, the GPS points needed to be cleaned. Outliers typically appear as wildly mistaken points along a reasonable sequence of GPS points and normally are far away from the temporally adjacent points in the traces.

Thus, like in Krumm's work [20], we decided to implement a cleansing algorithm based on speed and acceleration to remove these noisy points. We iterate over every GPS point in each trip checking if the segment formed by the current point with the previous one was traversed at more than 200 Km/h and, if so, the current point is removed. With this cleansing we achieved a reduction of approximately 1% of the data points:

Algorithm 1 Outliers Removal

```
 1: Load each user's GPS data files with the trips already segmented;
 2: Initialize empty List $l$;
 3: for all File $f$ do
 4:     for all Trip $t$ in File $f$ do
 5:         for all Point $p$ in Trip $t$ do
 6:             if speed(Point $p - 1$ , Point $p$) $<$ 200 Km/h then
 7:                 Add Point $p$ to List $l$;
 8:             else
 9:                 Point $p$ is outlier;
10:             end if
11:         end for
12:     end for
13: end for
```

4.4 Trip Filtering

In some cars the GPS receivers often produce intermittent streams of GPS readings, even when turned off, which can generate false trips in the user's data files. Thus, as our segmentation algorithm does not discriminate between these fictitious trips and the real ones, we applied some filtering algorithms to our traces in order to obtain valid results.

As was referred above, most of the GPS points are extracted every second. At first, we thought that this should be a positive fact because the more GPS points we have the more precision we can get. However, after analyzing the results obtained after the cleansing and segmentation stages, we realized that we were wrong. Each user's data file had an average size of 20 Megabytes and a lot of points were repeated, representing negligible distances and noise.

Thus, we decided to apply a filtering algorithm that eliminates every GPS point that is less than 10 meters away from the previous one. In this way, we were able to decrease the size of the files in 80%. However, there is one limitation. If a vehicle is more than 2 minutes stopped at a traffic light, all the GPS points extracted during that time will be within a circumference with 10 meters of radius, which means that after this filtering, these points will all be deleted except one. This way, when the car starts to move on again, the segmentation algorithm will consider that a new trip began because the next GPS point will be extracted 2 minutes after the previous one. So, our segmentation algorithm is not able to tell the difference between a 2 minute gap resulted from the car being shut off and a long traffic light waiting.

From Krumm and Froehlich work [20], we decided to use two filters: the *WithinBoundsTripFilter* and the *MinimumPointCountTripFilter*. The former removes the trips that take place outside the user's bounding box, that was defined the pre-processing stage. The latter removes the trips composed by less than 30 GPS points in order to eliminate small and fictitious trips.

4.5 Data Statistics

The application of these four stages described above to each user data file reduced the number of trips in roughly 73%. Thus, our final dataset is composed by 1892 trips from 11 different subjects. On average, each user took 6 (±3) trips per day and each trip lasted for 14 (±5) minutes and had 7 (±2) Kilometers of length.

If we compare these numbers with the ones showed by Krumm, we can conclude that our traces have a better quality. As we can see in Table 4.1, in our dataset, the number of days of data per subject represents more than one month, the double of Krumm's. Therefore, it allows us to perform a more completed training of our algorithms and obtain more truthful results, since there are more trips per user available to analyze. However, Krumm's dataset included 14468 trips from 240 users, which makes their work statistically more relevant.

Dataset	**AIDA (S. Francisco, Seattle & Coimbra)**	**Krumm (Seattle)**
Num subjects	11	240
Trip distance (Km)	7 (±2)	12
Trip time (minutes)	14 (±5)	16
Num trips / day	6 (±3)	4
Num trips / subject	172 (±130)	60
Num days of data / subject	36 (±35)	15

Table 4.1: Comparison between the quality of the traces presented in Krumm and Froehlich's work [20] and in this one.

4.6 Map Matching

The purpose of a map matching algorithm is to match each pair of coordinates of a trace to the correct road segment on the map. In this work, we did not need to worry about this phase, since AIDA's map matching algorithm was already been implemented by Correia and Câmara. Thus, the purpose of this section is not to describe how this algorithm works, since it is well explained in Section 4.1 of Correia's master thesis [6], but rather talk about its main limitation and how it influenced the composition of our traces.

What happens is that the algorithm was designed with too much perfectionism. That means that the matches only occur when there is 100% sure that the current point corresponds to a specific segment so, in theory, there will be no match errors. The problem is when there are more than one possible correspondent segment for a particular point and the algorithm does not know which one it should assign. In that case, it will simply not match the point, which will cause a separation in two traces. As this situation might happen more times during the matching, in the end we will have each trace divided in pieces, as Figure 4.1 demonstrates.

The top figure shows the appearance the trace should have when perfectly matched. The problem is that the algorithm was not able to match the red points, so the red segments will not be part of the trace after the matching.

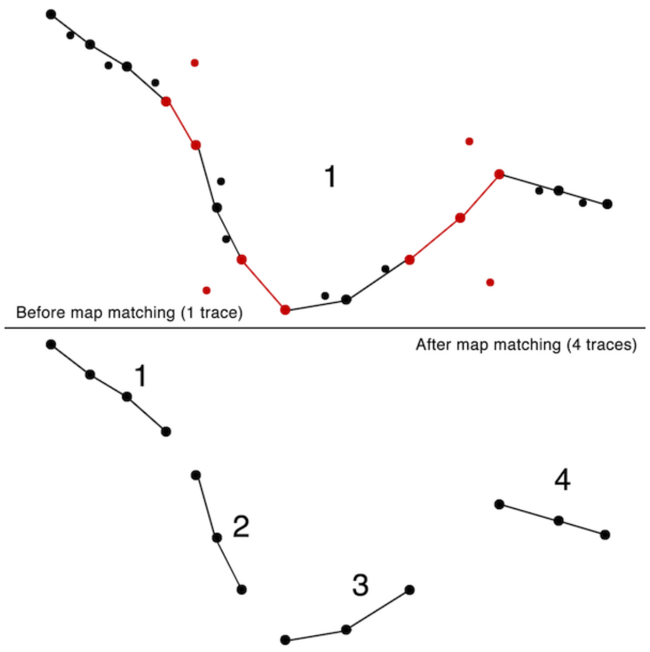

Figure 4.1: The map matching algorithm limitation.

Thus, in this case, we will have four smaller traces to train and test our distortion functions instead of a bigger one. On average, each trace is divided in 3.35 smaller traces after the map matching so, due to this limitation, the number of trips per driver has increased in roughly 335%, as we can see in the Table 4.2.

	Before Map Matching	**After Map Matching**
Trip distance (Km)	7 (±2)	1.1 (±0.7)
Num trips / subject	172 (±130)	576 (±497)

Table 4.2: Comparison between the number of traces before and after the map matching.

So, the traces that we are going to predict in the experiments phase (Chapter 6) have, on average, 1.1 (±0.7) Km of length that correspond, on average, to 19 (±9) segments. Although this number of segments is already quite significative, we also would like to study the performance of our algorithms with a different map matching algorithm, less perfectionist but that would not cause separations in the traces. If our traces had 7 (±2) Km length as initially, the number of segments that we predict would increase a lot and the predictions would be more difficult and the results more realistic. However, as we saw in Chapter 2, working on the map matching algorithm is not part of this works's plan and it is not our concern, so that represents future work.

Chapter 5

The new approach

In this chapter we will start by talking about the limitations of AIDA's original trajectory prediction algorithm (Section 5.1), by describing some scenarios where the results obtained might not be as promising as expected. Then, in the following sections, we will explain how the new approach works and how it is able to have a good performance when facing the original algorithm worst case scenarios.

5.1 Original algorithm limitations

In Section 3.2 we explained the basic ideas of Correia's master thesis [6], particularly how the weight updating function works and distorts the graph. Basically, before the beginning of each trip, the weight of each edge of the graph is updated by a decay function that receives a value x by argument. This x is the normalized quantity $\frac{n}{N}$ where n is the number of times a link was traversed and N is the highest n between all links. The higher the value of x, the lower the new weight value will be.

With the map being distorted by this function, the authors achieved about 92% (±5%) of accuracy. Although these are very promising results, their algorithm contains some limitations. As it was introduced in Section 3.2, one of the problems is that the map will be distorting infinitely. The main goal of using a distortion function approach is to get an approximation of each person's mental map of a city. This map is supposed to stabilize after a couple of weeks, something that is impossible to happen in this case. Even if a driver chooses the same routes everyday the distortions will never stop, since the edges weights are all updated before the beginning of a trip. These persistent updates can lead to other problems.

Let's imagine a scenario where a driver chooses always the same path from home to work during one month. After this month, a new and faster road is built and the driver decides to change his routine and starts to choose always the new road to go to work. The problem, in this case, is that the algorithm will take too long to realize that the driver has a new preferred route to work. It is easy to understand that the weight of the edges that compose this new route will rapidly diminish, because the number of times that they were traversed will quickly increase. So, in theory, this route should represent the shortest path. However, this is not reflected on the graph. What happens is that the edges of the old route were distorted during a long period (one month) and there is no way to increase its weights as quickly as they were previously decreased. In this case, the growth of these weights will be too slow, because the only way to increase them is to wait for the N to increase, so the x can be lower. Therefore, it will take too long for the weight of the new route to be lower than the old one, so the latter will represent the shortest path for a long time.

In conclusion, although this approach presents promising results, we believe that it will never be able to produce a stable map. We need a more complex algorithm that should be able to quickly adapt and react to changes in the driver's mind and stop the distortions when the best approximation to the mental map is achieved. In Section 5.2 we will describe how the new approach is going to work, particularly the *map stabilization effect*, which we believe to be a possible solution to overcome the limitations described above.

5.2 The map stabilization effect

As was explained in Section 5.1, one of the main limitations of Correia and Câmara's algorithm is the infinite distortion. Therefore, we decided not to update all the links weight before the beginning of a trip in our new approach. We realized that, when the route chosen by a certain driver is the same as the predicted by the algorithm, there is no reason to update the weight of any links. In contrast, if they are different, only the weight of the edges of both routes should be adjusted. So, we believe that the best way to achieve a stable map is not to change it when the trips are being predicted correctly.

Another limitation described above was the slow reaction of the algorithm when a situation of an abrupt change of routine happens. Let's imagine a scenario where the shortest-path between a driver's work and home is represented by Route **A**, that has 800 meters of length. However, due to some reason, this driver always chooses Route **B** (1200 meters) to travel between these two locations, so the algorithm must make Route **B** shorter than Route **A**. Our idea, differently from the original algorithm, is to decrement Route **B** and increment Route **A** in the same proportion every time the driver completes the trip, that corresponds to one iteration. The higher the difference between the weights of these two routes the higher should be the *strength* (proportion) applied, so more abrupt will be the distortions. In contrast, the last rectifications will be more accurate, in order to avoid distorting the links more than what is necessary and compromise the map stabilization. In this way, we believe that the algorithm will quickly rectify the weights when a case of abrupt change of routine happens. Figure 5.1 illustrates this idea, the *map stabilization effect*.

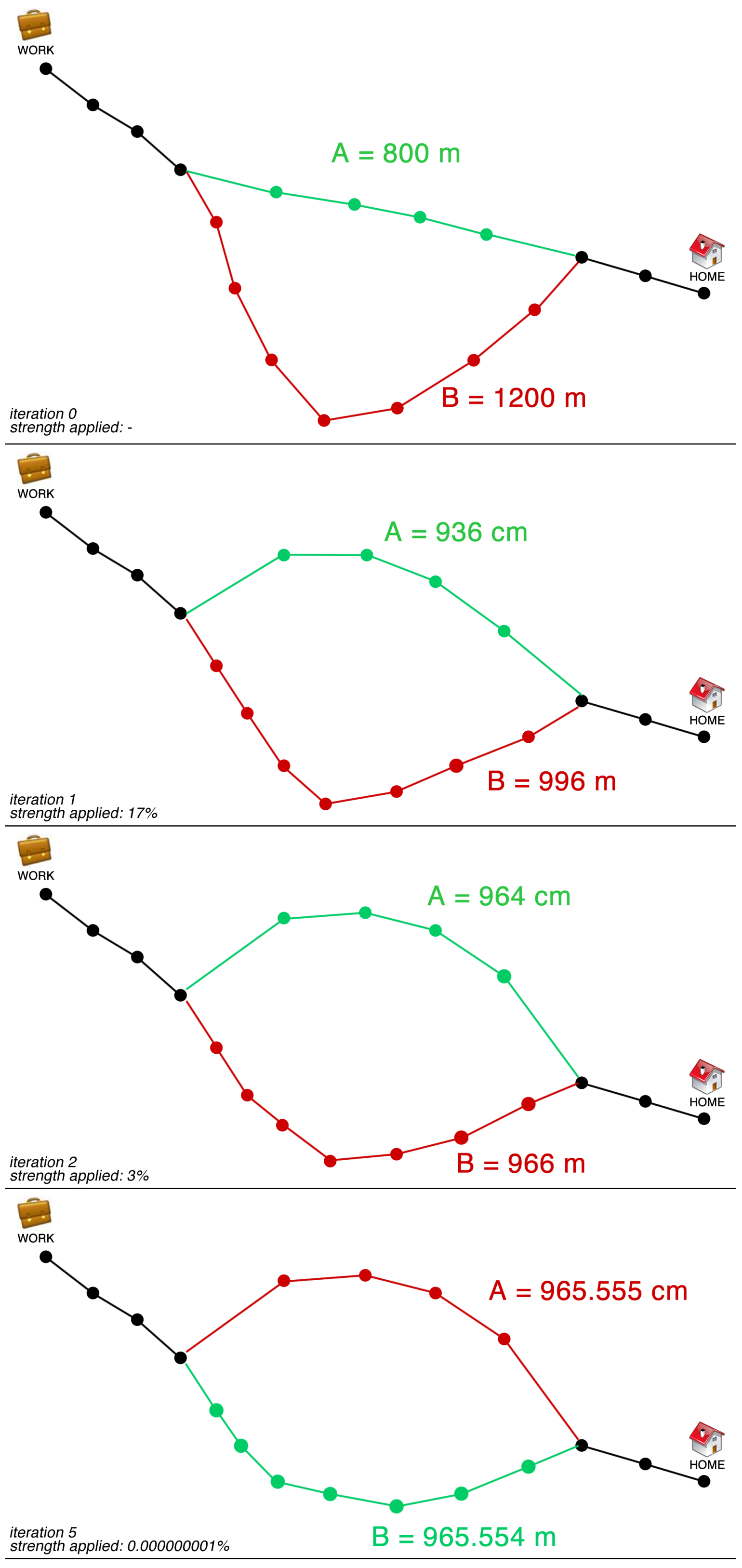

Figure 5.1: The *map stabilization effect*.

In order to compute the *strength* value, we start by calculating the difference (in percentage) between the weight of both routes. Then, we simply divide this difference in two equal parts to obtain two equal pieces of strength. One piece will be applied positively in the shortest route (the one we want to increment) and the other one will be applied negatively in the longest route (the one we want to decrement). So, this variable can be defined as

$$strength = \frac{1-\frac{a}{b}}{2}$$

where a is the total weight of the shortest route and b the total weight of the longest one. In this way, we will never distort a link more than 49.9(9)% of itself, since the difference between two routes will always be lower than 100%. In the opposite, the lowest possible *strength* value will be 0%, that corresponds to those situations where both routes have the same exact length. As we can see in Figure 5.1's example, the new weights for Route **A** and **B**, after the driver passes by Route **B** twice, are:

After one passage (iteration 1):

$strength = \frac{1-\frac{800m}{1200m}}{2} \approx 0.17(17\%)$

$Route\mathbf{A} = 800m + (800m \times 0.17) = 936m$

$Route\mathbf{B} = 1200m - (1200m \times 0.17) = 996m$

After two passages (iteration 2):

$strength = \frac{1-\frac{936m}{996m}}{2} \approx 0.03(3\%)$

$Route\mathbf{A} = 936m + (936m \times 0.03) = 964m$

$Route\mathbf{B} = 996m - (996m \times 0.03) = 966m$

In the examples described above we used $f(x) = x$ as distortion function, since the *strength* calculated is directly applied to the links so, for instance, $f(17\%) = 17\%$. However, despite of representing well the idea of the *map stabilization effect* (the lower the difference between the weights, the lower the *strength* applied), this function does not yield the results needed in some situations, as we will understand in Section 5.3.

5.3 The distortion function

In Section 5.2 we saw when and how we are going to distort each driver's city graph. However, we still need to define the function that returns the strength that will be applied because the function $f(x) = x$ has some limitations. For instance, the distortion function must not return zero when $x = 0$. If by accident two routes have the exact same length and the driver always chooses the one that is not the shortest for the algorithm, the rectification will never happen, since with $strength = 0$ no distortions will be made. Another limitation is the fact that this function always decreases in the same proportion, which can lead to situations where too small strengths are applied (e.g., 0.000001%) in each iteration, making the rectifications impossible to be completed in a valid period.

We believe that there are also other functions that make good representations of the *map stabilization effect* and that are capable of overcoming the limitations described above. Due to its flexibility and since it is commonly used in driver behavior modeling in transport research literature, we decided to try the *Sigmoid* as our distortion function. In our case, this function is defined as

$$f(x) = \frac{maxval}{1+e^{-\frac{x-mpoint}{grate}}}$$

where *maxval* is the maximum value (in percentage) the function can return, *mpoint* is the x value whose y corresponds to the medium point between zero and *maxval* and *grate* corresponds to the slope of the function, in other words, its growth rate. As the function's argument will never be lower than 0% and higher than 49.9(9)%, we will define the function in $x \in [0, 50[$.

The biggest challenge of using this function is to choose the right parameters. We started by defining $maxval = 50$, because we believe that we should not distort an edge more than 50% of its own weight in each iteration. By doing rectifications over that value, we would be distorting the links more than what is necessary to rectify them, which could compromise the stability of the map. Instead, we prefer to do smaller adjustments in each iteration and make more accurate rectifications. Moreover, if we want our function to have a similar behavior to $f(x) = x$ in order to have a good representation of the *map stabilization effect*, defining $f(49.9\%) \approx 50\%$ is a good starting point.

The remaining parameters are more tricky to choose and difficult to justify. We decided to define $mpoint = 25$ because, since we want this function to have the closest possible behavior to $f(x) = x$, this medium point value would guarantee that $f(25) = 25$. We also defined $grate = 5$ because the slope obtained with this value is not too abrupt nor too smooth and we believe it represents a proper learning curve. Moreover, the use of this growth rate implies that $f(0) = 0.3$, which represents the function's absolute minimum. Besides solving the situation of the two same length routes described above, in this way, we also guarantee that the rectifications will be completed in a valid period, since 0.3% is the minimum strength that can be applied in each iteration. We can see the function's appearance in Figure 5.2.

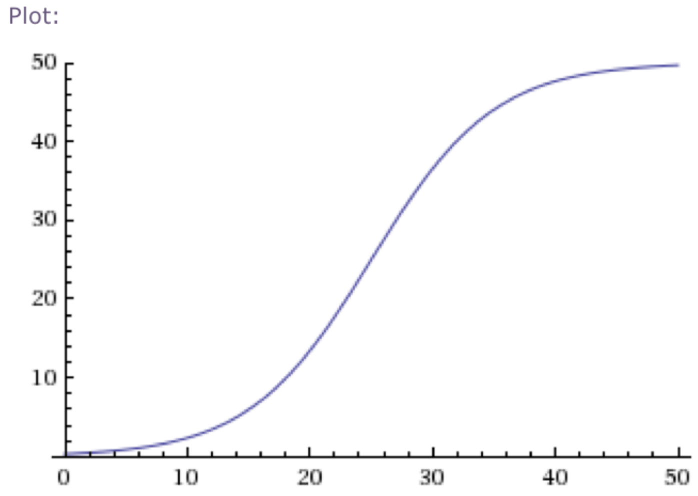

Figure 5.2: Distortion function's appearance with $maxval = 50$, $mpoint = 25$ and $grate = 5$.

By using this function and these parameters, we believe that we are maintaining the *map stabilization effect* represented by $f(x) = x$, but without the limitations referred. The results of the application of this new algorithm in the eleven drivers described in Chapter 4 will be presented in Section 6.3, where we will also understand why we were not so right about our initial theoretical assumptions (presented in Section 5.2) and why these parameters values, described above, are not the most optimal ones to this case.

Chapter 6

Experiments

In this chapter we will take a look to the tests performed and results obtained on both original and new approaches. We will start by talking about the performance evaluation of our algorithms. Then, in Section 6.2, we will describe the tests we made on the original approach and see if the results obtained are enough to statistically validate it.

Finally, in Section 6.3, we will see how we came up with an optimal distortion function for our new algorithm. We will compare its performance with the original one, show and discuss the results and see how to improve the accuracy in the worst cases scenarios.

6.1 Applied Measures

In order to compare the performance of their algorithm in the different tests, Correia and Câmara proposed four different measures, that are described in detail in Subsection 5.1.1 of Correia's master thesis [6]. In their work they claim that, if we look at them separately, we can not extract rigorous conclusions. Therefore, only by considering them as a group, we can further strictly evaluate the algorithm in the job of predicting the route that will be taken by the driver to a certain destination.

During the experiments, we performed a lot of tests on the distortion function and comparisons between different approaches, so we needed to find a simpler way of comparing the results and quickly understand what are the best ones. If we used all the four measures at the same time to compare two different approaches, we would take too much time to analyze the results and we would have a lot of difficulty in plotting them. Therefore, we decided to choose only one measure to describe the performance of our algorithm, this way reducing the problem dimensionality.

Thus, among those four measures, we decided to pick the *First Prediction Accuracy*. This measure aims to understand the accuracy of the first prediction made, that corresponds to the route predicted in the beginning of the trip when the shortest-path algorithm is for the first time applied. It corresponds to the ratio between the number of road segments that this route shares with the one that, in fact, the driver chose and the total number of road segments of that route that was taken by the driver. We believe that this measure is enough to express the algorithm's performance, since it allows us to have a good idea of how good the predictions yielded by the algorithm are.

6.2 Original Approach Validation

In the following subsections we will describe the tests we performed on the original algorithm and present the corresponding results. In Chapter 4 we explained that our dataset can be divided in two parts. The first part, composed by three different drivers from Coimbra and Seattle, has already been tested with this approach while the second one, composed by eight drivers from San Francisco, has not yet been. Our goal in this section is to demonstrate that the improvements shown in Correia's master thesis [6] with the first part of the dataset also occur with the second one. With this demonstration, we intend to statistically validate this approach.

6.2.1 Prior Prediction

Before testing the distortion function, we should first be aware of how the algorithm behaves when no distortions are applied to the map graph. So, the purpose of this test is to run every trace of a driver without making edge updates before the beginning of the trips. This test is very important, not just to understand the behavior of the algorithm in this situation but also to further compare its results with the ones yielded by tests that use the distortion function to distort the map in every iteration. Only by comparing both results we can understand if there are any improvements and if it is worth to use the distortion function or not.

After running this test on all the San Francisco drivers we obtained, on average, 83% (±4%) of *First Prediction Accuracy (FPA)*. This means that, if we just use a simple shortest-path algorithm to predict the route that the drivers will take, without making any distortions on the original city graphs, we can correctly guess, on average, 83% of the road segments by which they will pass over. In fact, in 62% (±7%) of the traces the algorithm was able to correctly predict all the segments, which means 100% of *FPA*. These are interesting results because they match with Golledge's studies [10], that demonstrate shortest distance is the criteria most often used in route selection. With this test, we can see that the base of our approach makes sense. However, our goal is to show that we can improve these results by distorting the city graph according to each driver's past route choices.

6.2.2 Re-substitution Method

The goal of this test is to understand the algorithm's performance when a distortion function is applied to the map edges before the beginning of each trip. This test runs every trip of a driver, starting with a non-distorted graph and distorting it in every iteration with the smooth log function $-\log 10(x)$ and $\alpha = 3$ (training phase). Then, the traces are run again, but this time using the graph previously distorted, in opposite to the *Prior Prediction* test where the traces are run using the original one. This is called, in the pattern recognition field, the *Re-substitution Method*, where the train and the test datasets are the same. We chose this function and this α because, when combined, they were the ones that presented best results in the tests performed by Correia and Câmara in [6].

In this test we obtained, on average, 90% (±2%) of *FPA* using the eight drivers of San Francisco as input, which represents improvements of nearly 7% regarding the 83% obtained in the *Prior Prediction* test. In Table 6.1 we can understand that these results are very similar to the ones obtained by Correia and Câmara in their work, using the Coimbra and Seattle dataset. If we join the two parts of the dataset and calculate a total average, we obtain 90% (±3%) of *FPA* all the eleven drivers.

	Coimbra/Seattle dataset	**S. Francisco dataset**	**Total Average**
N^o of drivers	3	8	11
Prior Prediction	86% (±7%)	83% (±4%)	84% (±5%)
Re-substitution	92% (±5%)	90% (±2%)	90% (±3%)
Improvements	6%	7%	6%

Table 6.1: Comparison between the improvements obtained by the original algorithm, regarding the *Prior Prediction* test, using two different datasets.

In Figure 6.1 we show how the map distortion affects the accuracy of first prediction in a typical driver. The X axis of the graph corresponds to all the traces of the driver (in this case, from 1 to 595) while the Y axis represents, for each trace, the difference (in percentage) between the *FPA* obtained using this test and the one obtained with the *Prior Prediction* test. Thus, a positive value means that the algorithm was able to correctly predict more segments of a trace using the distorted map instead of the original one. In the opposite, a negative value occurs when the distortions made in the graph deteriorate the *FPA* obtained in the *Prior Prediction* test.

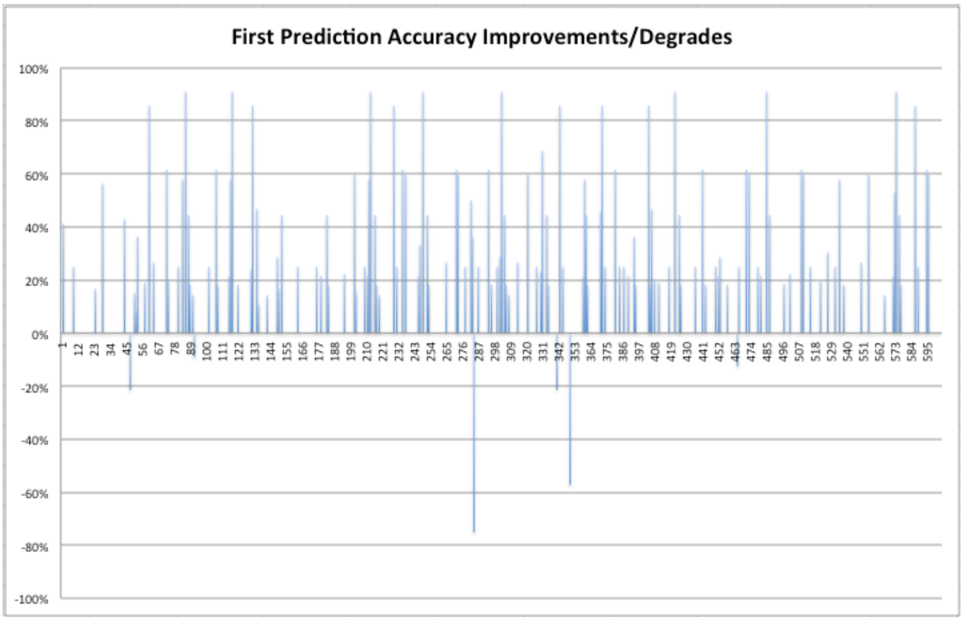

Figure 6.1: Improvements and degrades caused by the distortion function regarding the *Prior Prediction* test in a typical driver.

As we can see, if we run the traces using a distorted graph instead of using the original one, we obtain much more improvements than degrades. In fact, the use of a distortion function in this driver's map can cause improvements of nearly 90% of first prediction accuracy (36% on average), but may also cause degrades in some situations. These cases represent the worst case scenarios, that will be deeply analyzed in Subsection 6.3.6.

The study presented in this chapter is promising and confirm that, in fact, the results showed in Correia and Câmara's work were not obtained by chance. Thus, as we explained in Chapter 4, since this algorithm shows similar improvements in eleven different drivers from three different cities, we believe that it can be used as a valid trajectory prediction algorithm.

6.3 New Algorithm Behavior

In this section we will start by talking about the preliminary results we obtained with the new algorithm and how they influenced our next steps. Then, we will explain how we came up with our final distortion function and show some statistics and graphs about it. In the end, we will take a look to the worst case scenarios. As we are testing a new approach, all the traces of the eleven drivers described in Chapter 4 were used as input to our tests, in opposite to Section 6.2 where the approach was not new and one part of the dataset had already been tested.

6.3.1 Preliminary results

In Section 5.3 we explained why we believed 25 and 5 were the most suitable values for the medium point and growth rate, respectively, of our new distortion function. Table 6.2 shows the *First Prediction Accuracy* we obtained using our new approach with these parameters and compares its improvement regarding the *Prior Prediction* test with the one obtained by the original algorithm.

As we can see, we obtained more improvements (6%) using the original approach instead of the new one (3%). These results were quite surprising and disappointing for us. Since in theory our new approach should produce an higher *FPA* value, obviously something was wrong with the function pa-

	Original Approach	**New Approach (M.Point=25 & G.Rate=5)**
Prior Prediction	84% (±5%)	84% (±5%)
Re-substitution	90% (±3%)	87% (±4%)
Improvements	6%	3%

Table 6.2: Comparison between the improvements obtained by the original and by the new algorithm regarding the *Prior Prediction* test.

rameters. We realized that 25 and 5 might not be the most appropriate values for this case and we started to think about the best way to figure out what are the optimal parameters. That is what we are going to talk about in the next two sections.

6.3.2 Function Flexibility

Before taking a look to the results obtained, we should first understand how the parameters affect our distortion function. Since the maximum value of distortion is only a threshold and does not affect the function's appearance, we will only study the influence of the medium point M and growth rate G parameters. Figure 6.2 shows how the function's look changes with the variation of M and G values.

As we can see, the lower the growth rate the higher the function's slope and the sooner the function will start to make very small adjustments. In the opposite, a higher G makes the learning rate smoother. So, with a very small G value (e.g., 2) we are expecting to obtain bad results, since the small adjustments will start too soon and a lot of iterations will be needed until the rectification is completed. For example, let's imagine that there are two different roads (**A** and **B**) that end in the same destination: **A** has 112 meters and **B** 508 meters of length. For some reason, the driver always chooses route **B**, so we must make this route shorter than **A**. If we use our distortion function with the parameters G=2 and M=25 we will need 11196 iterations to complete the rectification, in other words, the algorithm will make the correct prediction only after the driver pass 11196 times by that path. However, due to the very small rectifications done in the final iterations (some lower than 0.0001%), the algorithm will correct the paths with a huge precision (**A**=206.6068m vs **B**=206.6065m).

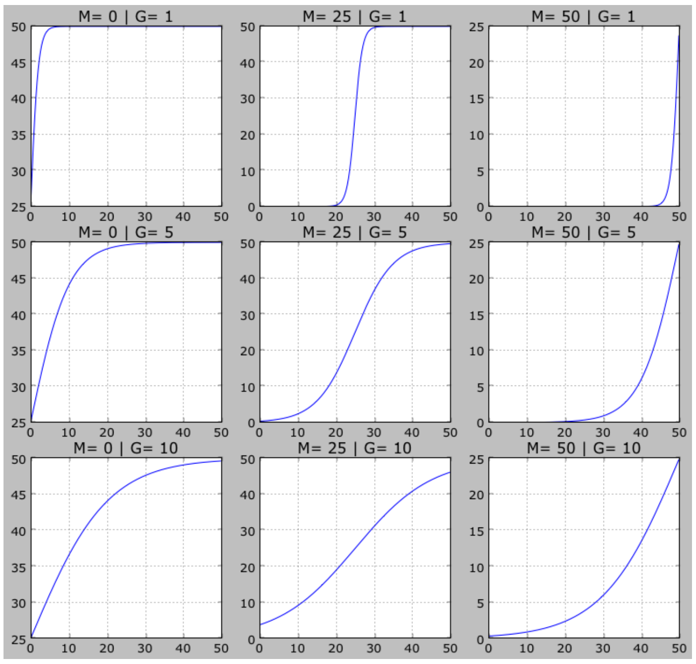

Figure 6.2: Function flexibility analysis.

We also believe that a very high G (e.g., 70) will cause a bad *FPA* too because, in this case, the small adjustments will never be made and the function will distort more than what is needed, which can affect traces that share links with the routes that are being distorted. If we maintain the M value, change the G to 70 and do the exact same test above, we see that we just need 3 iterations to get the weights fixed, due to the abrupt distortions done in each iteration (some over 40%), which means a fast rectification. However, we can also see that the difference between the two routes after the rectification (*precision interval*) is too high (**A**=222m vs **B**=208m). The fact that we are distorting the routes more than what is truly needed (e.g., route **A** has 222m but could have 206.6068m as we saw above) is not positive at all because some of the traces that share links with, for example, route **A** may no longer be the shortest ones for the algorithm after this rectification.

The same reasoning is applied to the medium point parameter. If it is too short the consequences are similar to ones verified when the G is too high. A very high M value has the same effect that a very short G value. Thus, we can understand that our G and M parameters should not be too short nor too high, so we need to find a balanced point.

6.3.3 Brute Force Search

In order to find the optimal parameters for our distortion function, we decided to perform a brute force style search. In this search, we varied the G value from 0 to 70 and the M value from -20 to 50 and performed the *Re-substitution Method* for all the combinations possible for all the different drivers. Figure 6.3 shows the *First Prediction Accuracy* obtained for each combination of G and M in a typical driver.

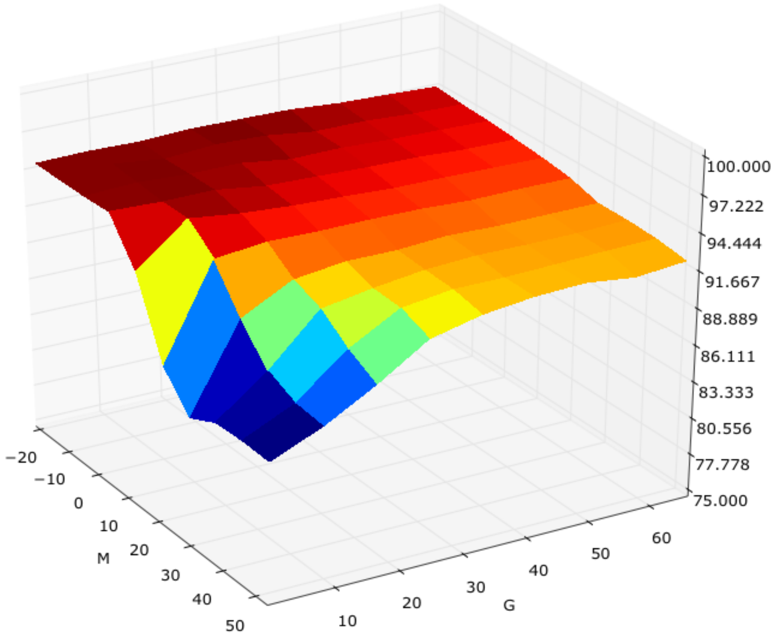

Figure 6.3: *First Prediction Accuracy* (in percentage) obtained for each combination of G and M in a typical driver using the *Re-substitution Method.*

This graph was quite surprising for us and we took a while to understand it. As we can see, typically, in each driver's graph there is a dark red zone where *FPAs* slightly above 95% are obtained, which represents improvements of 11% regarding the *Prior Prediction* test and 5% regarding Correia and Câmara's original approach. This zone corresponds to very low medium points and not too high growth rates. The best results where obtained with G=9 and M=-12 (95% $\pm$ 2% on average). We can see the function's appearance with these parameters in Figure 6.4.

After all, our initial idea was surprisingly wrong. It seems that we can get much more improvements if we make strong and abrupt rectifications in each

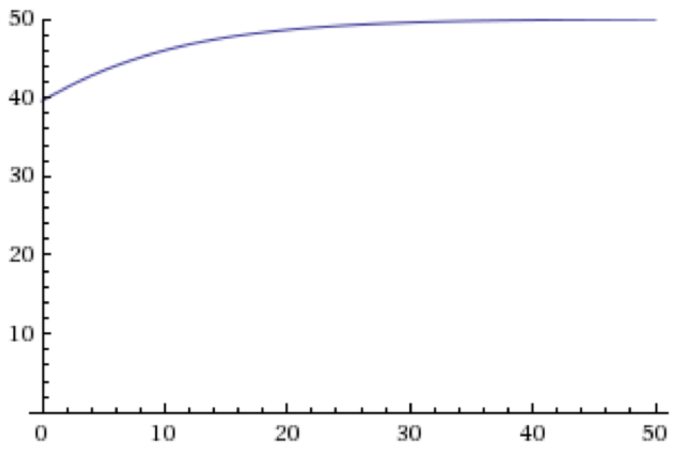

Figure 6.4: Distortion function's appearance with G=9 and M=-12.

iteration (typically above 40%) instead of trying to represent the *map stabilization effect* described in Section 5.2. In Section 6.3.2 we explained why we thought the use of very low medium point was a bad idea. Theoretically, we would get the weights fixed in very few iterations but we would compromise the precision of the rectification. If we repeat the same test performed in that section, but this time using the parameters that produced the best results, we will see that we just needed 2 iterations to fix the weights, but the total difference between them is too high (**A**=249m vs **B**=116m).

Apparently, the precision is not the problem that we thought. We were so worried about making precise rectifications and not to distort more than what is necessary when after all, a *precision interval* of, for example, 133 meters (249m - 116m) is not bad at all and it is not enough to jeopardize the predictions of traces that share links with the routes that are being distorted. In order to let the reader understand the difficulty of the predictions that are being made, we should say that about 33% (±19%) of the transitions are forced, meaning that there is only one next link for a given link in those cases. This statistic is very important, because if this value was too high, the promising results we obtained would not have relevant meaning.

6.3.4 Complete Simulation Test

To have an idea of how a simple weight rectification affects the average *FPA*, it is important to understand how it changes after each trip is completed.

In order to study those variations, we picked four of the eleven drivers and applied the *Complete Simulation* test, that simulates all a driver's routes in sequence, one route at a time, while checking how both the average *FPA* of all his traces and the number of traces with a bad *FPA* is changing. Figure 6.5 shows the graph resulted from applying this test to a typical driver.

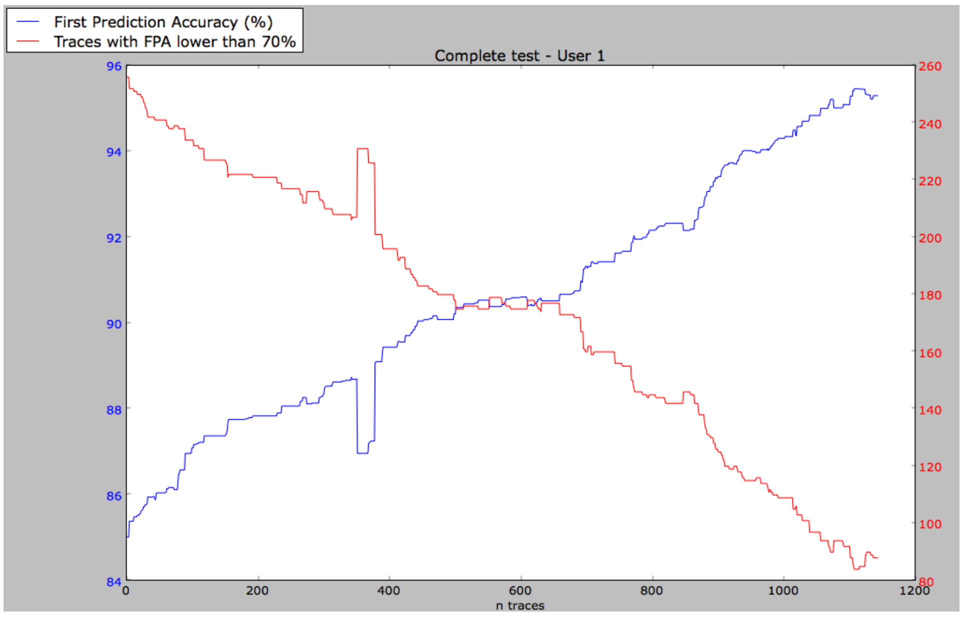

Figure 6.5: Complete Simulation Test on a typical driver (Driver number 1: 1144 traces).

In this driver, the *FPA* starts at nearly 85%, meaning that without making any distortions on the original city graph, we can correctly guess, on average, that percentage of road segments by which he will pass over (*Prior Prediction* test). Then, in general, we can see that the average *FPA* increases as the trips are being completed and the map is being distorted, until reaching roughly 95%. It is interesting to see that, although the scales are different, the number of low *FPA* traces is inversely proportional to the average *FPA*. In this driver, the algorithm was able to reduce the number of these traces from 250 to 88 (from 22% of all the 1144 to 8%), which represents improvements of 14%. In fact, in all the remaining drivers the algorithm was able to reduce the number of low *FPA* traces in similar percentages (19% ± 5% on average), which demonstrates a good performance. However, there are still some cases where the rectifications made after a trip cause the decline of the *FPA*. We will understand why this happens in Subsection 6.3.6, where we will take a look to the worst case scenarios.

6.3.5 Unknown Routes

In Subsection 6.3.3 we demonstrated how we achieved, on average, 95% of *FPA* in all the eleven drivers, using the *Re-substitution Method*. However, as was explained in Subsection 6.2.2, in this method we are training our model with the same dataset that we are testing it, in other words, the traces that we are using to distort the map are the same ones that we are trying to predict in the testing phase. Thus, we are considering that all the traces that we are going to predict have already been traversed by the driver at least once before.

The problem with the *Re-substitution Method* is that it does not count the fact that the route that we are trying to predict may have not ever been traversed by the driver. So, in order to have an idea of the performance of the algorithm when counting with those situations, we decided to use the *ten-fold cross validation* technique, where the map is distorted using 90% of the driver's traces and then tested with the remaining 10%. This is repeated ten times, each time holding out a different set of testing data, with the average over all tests reported. Moreover, due to this method's randomness in building the sets, we decided to repeat this procedure another ten times for each one of the eleven drivers.

We should say that about 45% (±14%) of the traces collected for each driver are repeated at least three times in the corresponding dataset, meaning that they are probably part of routines. So, we can conclude that all the remaining 54% represent sporadic trips. This is important to refer because if the datasets were composed by mostly routines, the probability of finding a unknown route in the testing sets (that was not previously trained) would be very low, and thus this test would not make sense at all.

Table 6.3 shows that the final average *FPA* obtained with this test was 91% (±2%). This table is a new version of Table 6.2 where we can compare the relevant results obtained until this point and see that the new algorithm, even when considering the driver's unknown routes, can produce higher improvements than the original approach tested with the *Re-substitution Method*.

	Original Approach	**New Approach (M.Point=-12 & G.Rate=9)**
Prior Prediction	84% (±5%)	84% (±5%)
Re-substitution	90% (±3%)	95% (±2%)
Cross-validation	-	91% (±2%)
Improvements	6%	9% (±2%)

Table 6.3: Comparison between the improvements obtained by the original and by the new algorithm (tested with two different methods), regarding the *Prior Prediction* test.

6.3.6 Worst Case Scenarios

In this subsection we will try to understand why some rectifications made after a trip cause the decline of the average *FPA* and degradations in some traces. In a study performed in four of the eleven drivers, we concluded that the worst case scenarios represent 17% (±2%) of the total traces and correspond to those situations where a route that had its edges weight increased in previously iterations share links with routes that will next be taken by the driver. The first typical worst case we are going to analyze is when this situation occurs with the origin of both routes being the same (Figure 6.6). This situation happens in 60% (±9%) of the worst cases.

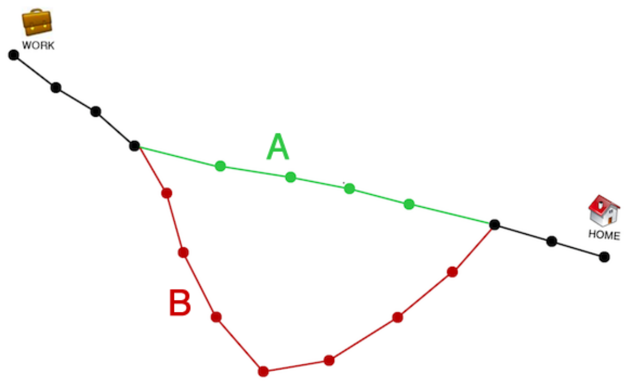

Figure 6.6: Typical worst case scenario (Same origin and destination).

Let's imagine that a driver goes from his work to home five times a week. From Monday to Thursday, he always takes route **B** but on Friday, due to some reason, he prefers route **A**. In this case, let's consider that four passages are enough to make the route **B** shorter than route **A**. So, on Friday's afternoon, when the driver is about to leave his work, the algorithm will wrongly predict that he will take route **B** to home. At this point, the average *FPA* obtained is 80%, because the algorithm can just correctly predict 100% of the links in $\frac{4}{5}$ situations (it is not able to predict Friday's route).

After Friday's trip is completed, as the rectifications that are being done by the new distortion function are very abrupt (always higher than 40%), route **A** will become shorter than route **B** again. At this point, the average *FPA* will decrease to 20%, because with this distribution of weights the algorithm is only able to correctly predict Fridays's route, that represents $\frac{1}{5}$ of all the routes. If this cycle is repeated in the next weeks, this inconsistency will last forever and the algorithm will never be able to correctly predict all these five trips at the same time. Now, we understand why there will be always degradations in some traces and why it is impossible to avoid some declines on the average *FPA* during the *Complete Simulation* test (Subsection 6.3.4).

Figure 6.7 shows the second type of scenario that can happen, that represents the remaining 40% of the worst cases. This situation occurs when the routes share the same destination but have different origins.

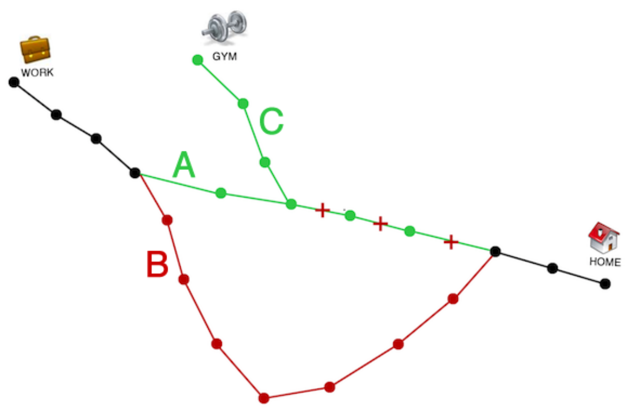

Figure 6.7: Typical worst case scenario (Different origin but same destination).

In this example we consider that the driver always chooses route **B** to go from his work to home. Thus, as route **A** is initially shorter than **B**, its weight will be increased. The problem is that route **A** shares three links with route **C**, that is the shortest and preferred one by the driver to travel from the gym to home. So, since our distortion function makes abrupt rectifications, those three links can have their weight increased in such a way that route **C** will no longer represent the shortest path for the algorithm in the way gym-home.

This fact was our main concern in Subsection 6.3.2, where we explained that the fact that we are distorting the routes more than what is truly needed should not be positive at all. However, due to the promising results obtained, in Subsection 6.3.3 we concluded that, despite the abruptness of the rectifications, the correct prediction of routes similar to route **C** was not in jeopardy. After analyzing all these cases in those four drivers, we confirmed that our theory was right, because in all of them an average of 100% *FPA* was obtained after few iterations, meaning that the increase of route **A**'s weight did not affect route **C**.

So, we can conclude that, at least given the current dataset, only the first typical worst cases we presented (Figure 6.6) truly represent a threat. Next, we will look more deeply into those cases and discuss some ideas about how to improve the *FPA* in those situations.

6.3.7 Possible Solutions and Future Work

Figure 6.6 showed us the worst case scenarios that truly represent a threat to our *FPA*. In these cases, it is impossible for this algorithm to always correctly guess which path the driver will take so, since only one represents the shortest, we will need to add some extra information to our model. Our approach to resolve this situations is based on the fact that people have different periods of the day where they tend to do similar trips.

If we pick the example shown in Figure 6.6, we would say that it is possible that, for some reason, the driver takes route **A** when he goes home at the lunch time and route **B** when he returns home afternoon. In order to confirm this idea, we decided to plot a histogram, for each driver, of the frequency of trips made in each half an hour interval of a day and then normalize the data in order to validate the periods. We decided to use a normal curve to represent each relevant period since it is a standard. Probably, there are other distributions that best fit a driver's behavior during a day but, since our only purpose is to detect periods, we believe that a normal curve works well in this case. Figure 6.8 shows the histogram obtained for a typical driver.

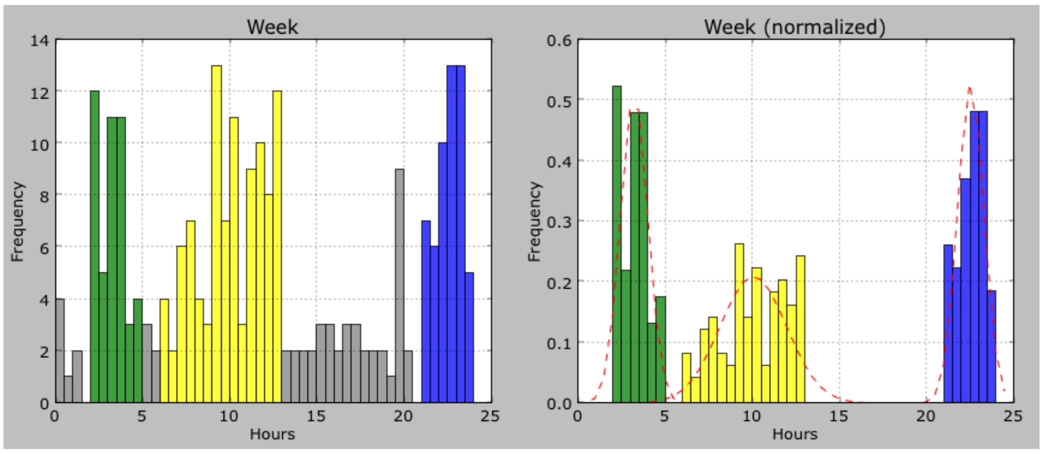

Figure 6.8: Left: Histogram of the frequency of trips made in each half an hour interval of a day; Right: Normalization of the data and period detection.

In the left figure we can see three different periods in green, yellow and blue colors. The grey intervals represent sporadic trips that rarely happen. In the right figure these intervals were discarded and the data was normalized. As we can see, the periods detected can be approximated without too much error to normal curve, which validates our idea. The algorithm used to detect these zones was the following:

Algorithm 2 Periods Detection

```
1: Detect which intervals represent a minimum in the histogram;
2: Calculate the mean value of the minimums;
3: Eliminate all the intervals whose frequency is below the average of the
   minimums;
4: Detect the periods resulted;
5: Join the periods that are separated by half an hour (1 interval);
6: Eliminate all the periods that do not last more than one hour;
```

We just saw that we were right about our assumption since we detected, in each driver, different periods of day where the trips are frequent. So, in theory, if we assign one individual map for each one of the periods and distort it according to the trips done during each one, we will obtain different maps adapted to different periods and so, in general, we expect to obtain less trace degradations. In Figure 6.9 we can see the results obtained by applying the *Complete Simulation* test to the same typical driver studied in Subsection 6.3.4, but this time using four different distorted maps, one for each relevant period detected during the week and one for all his weekend trips.

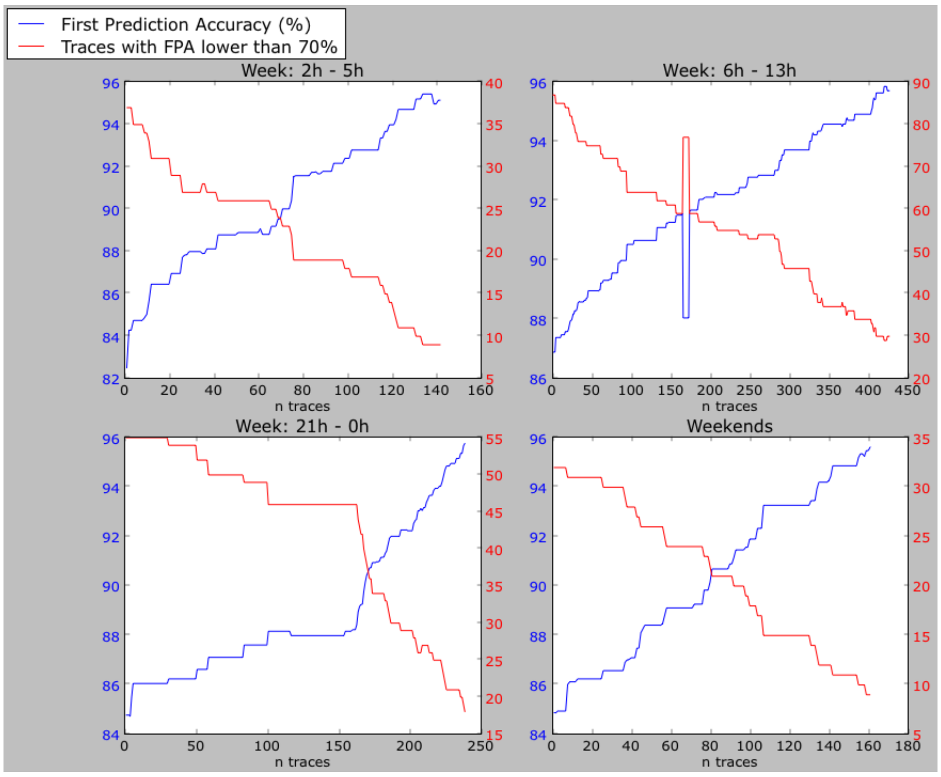

Figure 6.9: *Complete Simulation* test for each relevant period of a typical driver.

These results were quite satisfactory, since we can verify that there are much less *FPA* declines in each period's simulation regarding the simulation showed in Figure 6.5. In fact, these approach seems to work very well for this driver specially for the weekends, since there are no declines presented in the simulation for that period. This means that during the weekends this driver has a very consistent route choice and always chooses the same routes to reach the same destinations. We can find improvements in the other graphs too, but since they all correspond to week periods it is more difficult to eliminate all the declines, since the drivers tend to have a less consistent route choice during those periods due to their work, the traffic hours or family contretemps. However, we can see that in most of these periods the final average *FPA* obtained is nearly 96%, which improves the 95% obtained with the same simulation, but using the same map for all the traces (Figure 6.5).

So, we just showed how can we improve the results by using a individual map for each relevant period of each driver. However, the algorithm is still quite far from the perfection. We believe that there are more ways of improving the average *FPA* in these worst case scenarios and that represents our future work. For example, we believe that we could do more accurate predictions if we could get an idea of the rush hours of each city or traffic information. It is common sense that the drivers tend to avoid high traffic roads during rush hours and so, that information could help us to understand which route the driver will choose. Another idea is to have an individual map for each day of the week instead of each day period, or even join both. This could help us predicting those situations similar to the first example mentioned in Subsection 6.3.6, where for example the driver, on Fridays, chose a different route from the other days of the week.

Chapter 7

Conclusions

In this chapter we will start by introducing the team that contributed to the development of this thesis and the roles of each individual (Section 7.1). Then, in Section 7.2, we will make a brief summary about the work performed during this year, that is described in this report.

7.1 Contributions

During this thesis I worked with Pedro Correia and Francisco Pereira (my advisor), both members of AIDA's team. Since Pedro was the one that implemented AIDA's trajectory prediction algorithm, and one of my tasks was to validate it, he helped me during the first experiments phase by explaining me the architecture of his code. We also defined the tests to make on the original algorithm and analyzed its results together.

All the data processing phase and literature research was my entire responsibility. I looked for similar works in order to understand what type of processing should be applied to the traces and I also performed a deep research about scientific papers that could be the state of the art for this project and explained the basic concepts of the most relevant.

As was referred in Section 2.2, we obtained the new algorithm through brainstorm sessions where the three of us participated actively by discussing each one's ideas and merging them until we reached a final proper sketch.

From that point to end of this thesis, I was responsible to implement the new algorithm and all the tests/studies associated as well. In the weekly meetings, me and Francisco used to analyze and discuss together the results obtained and define the next steps to perform in the following week(s). Finally, I wrote this report and the intermediate one as well.

7.2 Summary

In this report we started by introducing AIDA as a new generation navigation system capable of changing the way people interact with the cars. We talked about its different components, particularly the destination and trajectory prediction model that is the one that shows more improvements until today.

We collected and processed new GPS data in order to statistically validate AIDA's original trajectory prediction algorithm, that had just been tested with three different drivers. We chose the tests we thought were the most relevant for this task and we successfully validated the algorithm since, on average, it demonstrated a good performance (90% of accuracy) when predicting the routes of eleven drivers from three different environments.

We also explained the limitations of this algorithm and proposed a more realistic one, based on the *map stabilization effect* described in Section 5.2. In the experiments phase, we realized that we were wrong about this initial idea and we demonstrated how to reach, on average, 95% of accuracy using the new algorithm (with the same eleven drivers) just by changing the parameters of the distortion function. We also performed tests like the *Complete Simulation*, where we can verify that the low accuracy traces and the total average accuracy are inversely proportional and the *Unknown Routes*, where we can see that the accuracy of the algorithm decreases to 91% when counting with unvisited paths. Finally, we presented a study about the worst case scenarios and possible solutions to resolve this kind of situations, like the use of different maps for different relevant periods of each driver.

In conclusion, this thesis contributed to the Trajectory Prediction research field by proposing a new approach, based on the distortion of the *cognitive map* of each individual, that is able to predict (with high accuracy) the route that will be chosen by a driver to travel between two different locations, without acquiring prior knowledge.

Bibliography

[1] AIDA: *Aida - affective intelligent driving agent.* `http://senseable.mit.edu/aida`.

[2] Ashbrook, Daniel and Thad Starner: *Using gps to learn significant locations and predict movement across multiple users.* In *Personal and Ubiquitous Computing*, pages 275–286, 2003.

[3] Bailenson, J. and D. Shum, M. Uttal: *The initial segment strategy: A heuristic for route selection.* In *Memory and Cognition, 28(2)*, pages 306–318, 2000.

[4] Brunye, T. T., C. R. Mahoney, A. L. Gardony, and H. A. Taylor: *North is up(hill): Route planning heuristics in real-world environments.* In *Memory and Cognition (in Press)*, 2010.

[5] Caduff, David, Sabine Timpf, and Alexander Klippel: *The landmark spider: Representing landmark knowledge for wayfinding tasks.* In *Reasoning with Mental and External Diagrams: Computational Modeling and Spatial Assistance, AAAI Spring Symposium*, pages 30–35, 2005.

[6] Correia, Pedro: *Trajectory Prediction*, 2010.

[7] Deguchi, Yoshitaka, et al.: *Hev charge/discharge control system based on navigation information.* In *SAE Convergence International Congress Exposition On Transportation Electronics*, 2004.

[8] Di Lorenzo, Giusy, Santi Phithakkitnukoon, and Carlo Ratti: *Context-aware navigation: Improving urban living experience with predictive navigation system.* In *UBI Challenge Workshop 2010, Copenhagen, Denmark*, 2010.

[9] Duckham, Matt and Lars Kulik: *"simplest" paths: Automated route selection for navigation.* In *COSIT*, pages 169–185, 2003.

[10] Golledge, Reginald G.: *Path selection and route preference in human navigation: A progress report.* In *Frank, A., Kuhn, W., eds.: Spatial Information Theory: A Theoretical Basis for GIS (COSIT '95). Number 988 in Lecture Notes in Computer Science, Berlin, Springer*, pages 207–222, 1995.

[11] Haque, Shazia, Lars Kulik, and Alexander Klippel: *Algorithms for reliable navigation and wayfinding.* In *Spatial Cognition*, pages 308–326, 2006.

[12] Hirtle, S. C. and M. F. Mascolo: *Effect of semantic clustering on the memory of spatial locations.* In *Journal of Experimental Psychology: Learning, Memory and Cognition, 12*, pages 182–189, 1986.

[13] Holyoak, K. J. and W. A. Mah: *Cognitive reference points in judgments of symbolic magnitude.* In *Cognitive Psychology, 14*, pages 328–352, 1982.

[14] Karbassi, Abdolreza and Matthew Barth: *Vehicle route prediction and time of arrival estimation techniques for improved transportation system management.* In *Intelligent Vehicles Symposium, 2003*, pages 511–516, 2003.

[15] Kim, Sang Wook, Jung Im Won, Jong Dae Kim, Miyoung Shin, Junghoon Lee, and Hanil Kim: *Path prediction of moving objects on road networks through analyzing past trajectories.* In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 379–389, 2007.

[16] Klippel, Alexander, Lothar Knuf, Bernhard Hommel, and Christian Freksa: *Perceptually induced distortions in cognitive maps.* In *Spatial Cognition Conference*, 2004.

[17] Kostov, V., J. Ozawa, M. Yoshioka, and T. Kudoh: *Travel destination prediction using frequent crossing pattern from driving history.* In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 343–350, 2005.

[18] Krumm, J.: *Real time destination prediction based on efficient routes.* In *2006 SAE International*, pages 1–6, 2006.

[19] Krumm, John: *Markov model for driver turn prediction.* In *Society of Automotive Engineers (SAE) 2008 World Congress, April 2008*, 2008.

[20] Krumm, John and Jon Froehlich: *Route prediction from trip observations.* In *Society of Automotive Engineers (SAE) 2008 World Congress, April 2008*, 2008.

[21] Krumm, John and Eric Horvitz: *Predestination: Inferring destinations from partial trajectories.* In *UbiComp 2006: Ubiquitous Computing*, pages 243–260, 2006.

[22] Maki, R. H.: *Categorization and distance effects with spatial linear orders.* In *Journal of Experimental Psychology: Human Learning and Memory, 7*, pages 15–32, 1981.

[23] Mark, D. M.: *Finding simple routes: 'ease of description' as an objective function in automated route selection.* In *Proceedings, Second Symposium on Artificial Intelligence Applications (IEEE), Miami Beach*, pages 557–581, 1985.

[24] NMEA: *Nmea format.* `http://www.gpsinformation.org/dale/nmea.htm`.

[25] Patterson, Donald J., Lin Liao, Krzysztof Gajos, Michael Collier, Nik Livic, Katherine Olson, Shiaokai Wang, Dieter Fox, and Henry Kautz: *Opportunity knocks: A system to provide cognitive assistance with transportation services.* In *UbiComp 2004: Ubiquitous Computing*, pages 433–450, 2004.

[26] Rodrigues, João: *AIDA - Driver behavior modeling (intermediate report)*, 2011.

[27] Simmons, Reid, Brett Browning, Yilu Zhang, and Varsha Sadekar: *Learning to predict driver route and destination intent.* In *In Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 127–132, 2006.

[28] Stevens, A. and P. Coupe: *Distortions in judged spatial relation.* In *Cognitive Psychology, 10*, pages 422–437, 1978.

[29] Tanaka, Kohei, Yasue Kishino, Tsutomu Terada, and Shojiro Nishio: *A destination prediction method using driving contexts and trajectory for car navigation systems.* In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 190–195, 2009.

[30] Tate, E.D. and S.P. Boyd: *Finding ultimate limits of performance for hybrid electric vehicles.* In *SAE Transactions - Journal of Passenger Car - Mechanical Systems*, 2001.

[31] Tversky, B.: *Distortions in memory for maps.* In *Cognitive Psychology, 13*, pages 407–433, 1981.

[32] Tversky, Barbara: *Cognitive maps, cognitive collages, and spatial mental models.* In *Frank, A.U. and Campari, I. (eds.) Spatial Information Theory: A Theoretical Basis for GIS, Procceedings COSIT '93*, 1993.

[33] USGS: *United states geological survey.* `http://landcover.usgs.gov/`.

[34] Wikipedia: *Agile software development.* `http://en.wikipedia.org/wiki/Agile_software_development`.

[35] Wikipedia: *Waterfall model.* `http://en.wikipedia.org/wiki/Waterfall_model`.