An Off-line Map-Matching Algorithm for Incomplete Map Databases

Francisco Câmara Pereira, Hugo Costa and Nuno Martinho Pereira

Centro de Informática e Sistemas da Universidade de Coimbra (CISUC)
Departmento de Engenharia Informática, Universidade de Coimbra
Pólo II, Pinhal de Marrocos
3030 – Coimbra, Portugal
camara@dei.uc,pt, hecosta@student.dei.uc.pt, nfmart@student.dei.uc.pt

Abstract. The task of map-matching consists of determining, for a geographical point or sequence of points (e.g. obtained from GPS), where in a given map those points correspond. Due to many reasons, namely the noisy input data and incomplete or inaccurate maps, such task is not trivial and can affect the validity of applications that depend on it. This includes any Transport Research projects that rely on post-hoc analysis of traces (e.g. via Floating Car Data).

In this article, we describe an off-line map-matching algorithm that allows for incomplete map databases. We test and compare it with other approaches and provide guidelines for its use within other applications. This project is provided as open source.

Keywords*: Map Matching, Map Generation, Artificial Intelligence*

1. Introduction

Map Matching algorithms are needed in any system that needs to associate some sort of information to specific geo-referenced locations. In other words, while we may get exact maps that represent any portion of the planet, dynamic information obtained from common Global Navigation Satellite Systems (GNSS) devices (e.g. GPS) almost always carry errors that may affect their usefulness. For car navigation, for example, extreme care must be taken to find, for every second, where the driver really is, as opposed to what the GPS receiver says he/she is. Another example is the Floating Car Data probes (i.e. vehicles that periodically report their GPS position), from which it is possible to obtain information on the traffic situation [1] that can be used to generate real-time information as well as to provide traffic analysis and forecasting (e.g. [2,3]) or simply to analyse mobility behaviour within an area. A bolder example could be dynamic toll charging – charging each vehicle based on its profile, used roads and/or daily mileage. In any of these situations, accurate Map Matching algorithms become fundamental for the success of the applications. Furthermore, in some of these examples, the analysis involved can be taken in an offline, post-processing manner. While on-line algorithms have evolved to their limits recently,

essentially due to commercial car navigation applications, off-line approaches are still under explored. At a first sight, the former should be both a more challenging and a more generic task (solving the "real-time" problem often makes the post-processing solution simple), but a more careful examination shows that these are two different approaches to two different problems. Real time applications demand solutions that provide instant response and can only rely on "past" points. This implies a compromise of performance over accuracy. On the other hand, off-line applications can take advantage of "future" points and allow for slower performances in favour of accuracy. As a result, on-line solutions applied on an off-line basis show extremely poor results, thus specific research is needed for solving the latter problem.

The task of off-line map matching is thus to determine, for a sequence of geo-referenced points previously obtained (e.g. from GPS), where in a given map those points correspond. The difficulty of the challenge is inversely proportional to the accuracy of the localization technology, thus it could be said that with Differential GPS or with Real Time Kinematics (RTK), which allow centimetre level accuracy, the task becomes considerably simpler. However, these technologies still demand expensive receivers as well as a dedicated ground infrastructure, which enhances the importance of the common, off-the-shelf, GPS solutions that are so widespread nowadays. With even less accuracy, other very low cost localization approaches are becoming common, such as cell-phone based localization (e.g. [4, 5]). For these also, accurate Map Matching becomes a quite complex and determinant task.

Another aspect to mention is that, either for on-line or off-line applications, the available maps are often incomplete, due to the dynamics of the road networks almost everywhere in the world. Direction changes, areas under construction, new roads, off-road tracks and road closures are just some examples of phenomena that happen on a daily basis. When facing roads that are absent in the map, Map Matching algorithms typically take some time to become aware of it. They keep "glued" to the existing road links until they become too distant, and then they typically enter into an "initialization mode" that starts a new matching all over again when sufficiently close to a recognized map link. The "new road" segment becomes blurred in this process. For applications that demand some accuracy, this may affect the results, since it may imply, for example, that the vehicle has travelled slowly and stopped at one end while starting a new journey on the other end. There is at least one application in the market that covers some of these issues, TomTom Map Share. However, we should point out that the approaches are very different and this application focuses on *correction* of the map provided by TomTom (direction changes, areas under construction, etc.), as opposed to the *aggregation* of new roads or geometry *updates*. This is done with the intervention of human hands (as happens in OpenStreetMap.org [14]), and not fully automatically as in our project, YouTrace.

In this article, we propose M-GEMMA, an off-line Map Matching algorithm for incomplete maps. It is based on two other algorithms: an improved version of Marchal's algorithm [6] that allows incomplete maps; and the GEnetic Map Matching Algorithm (GEMMA), an algorithm based on the evolutionary computation paradigm of Genetic Algorithms (GAs), that intends to overcome the main problems raised by Marchal's approach. M-GEMMA was designed to combine the strengths of the two approaches and to become a versatile Map Matching tool.

We implemented and tested a total of four algorithms (Marchal's original and improved versions; GEMMA and M-GEMMA) and made a thorough comparison, reported in this article.

M-GEMMA's source code is available with a "creative commons license" and its use is free. We hope to provide information in this article that can help on its application and comprehension. M-GEMMA, Improved Marchal and GEMMA were developed within the context of the YouTrace platform (which will also be made available as open source), a project that allows for the collaborative incremental construction of trajectory maps[1]. The next section will provide an overview of the YouTrace project in order to provide some context to M-GEMMA.

The state-of-the-art in Map Matching is presented in section 3, while Marchal's algorithms (original and improved version) are described in section 4. Afterwards, we describe GEMMA in section 5. M-GEMMA is finally presented in section 6.

The experiments and a comparative analysis are shown in section 7, and the article is finished with a number of conclusions.

2. Giving some context: The YouTrace Project

The YouTrace project intends to be a social platform that allows users to collaborate in the construction of a map-of-the-world [7]. A key element in the platform is the Map Generation Engine that is responsible for aggregating the users' traces into a single map. A YouTrace user can upload his/her traces contributing to the construction of a joint map of the world. The users can then receive an updated map, which will allow, for example, for more efficient car navigation applications. An innovative characteristic of collaborative mapping is its dynamics, as opposed to the current static maps. Roads are constantly being updated and aggregated as new traces are introduced. The collected traces can also provide information for a more efficient route planning, as the traces are a useful and realistic source about road/trajectory usage, average speeds and user preferences on road alternatives. Besides providing a dynamic map of the world, YouTrace can also be a useful source of information about users mobility and city dynamics. This information can be extremely valuable to urban planners, which can base their planning decisions on more realistic information (as opposed to surveys or probabilistic reasoning). YouTrace users can access the system through a web portal that will be responsible for feeding the Map Generation Engine with traces, which will in turn add them to the map.

[1] We refer to trajectory maps since the geometry obtained corresponds to the driving trajectory as opposed to the road infrastructure geometry. Particularly in curves, the visual result can become very distinct.

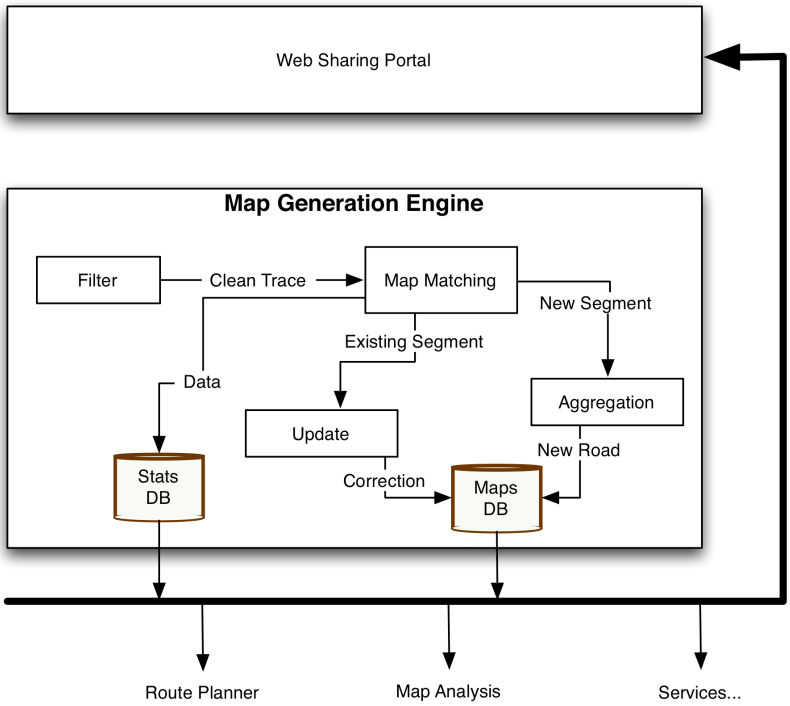

Figure 1 - YouTrace Map Generation Engine architecture

The first step of trace processing is filtering. The filtered trace is then addressed to the Map Matching, where GPS points are matched to the map, in order to find the existing segments on the map. The matched points of the trace are used to update the existing segments on the map, improving the road precision. The non-matched points of the trace are aggregated on to the map, creating new roads (or trajectories). Two databases are generated from this process: the map database and the statistics database. These databases serve to provide data for external services (e.g. route planning, traffic analysis). For more information on the YouTrace project, please read [7]. As can be understood, the Map Matching is a key element for YouTrace. It is responsible for finding the parts of the trace that already exist on the map, allowing for the distinction between the parts that should be aggregated and those that should be updated. Therefore the quality of the final map is very dependable of the quality of the match.

3. Current Trends on Map-Matching

Map-Matching algorithms are used to fix location data into a spatial road network. They are used in the most varied applications. The most common ones are certainly the GPS car navigation devices, which are constantly indicating the road segment where the user is located based on information retrieved from GPS satellites. The purpose of a Map-Matching algorithm can be divided in two parts. Firstly, the algorithm determines which road segment, from a given network, corresponds to each given position. Afterwards, it will determine the exact location of the same position inside the segment previously selected [8, 9].

There are algorithms designed specifically for given applications and others that are generic. In some situations, the path is known in advance so the set of roads to perform the matching is

restricted to them. For instance, the match of bus location data can be improved by restricting the road network to the known path taken [9]. Generic algorithms can also be one of two types: online or offline [8]. Real-time applications, such as GPS navigation devices use online algorithms, meaning that the matching is performed as the data is being received and thus it is based only on past matches. On the other hand, post-processing applications use offline algorithms, they can use online ones as well but they become less effective. Offline algorithms can take the advantage of not only matching each point according to past data but also based on the following "future" point, which helps the algorithm to select the correct road near to junctions. The literature review made by [8] states that the majority of the existent algorithms are for real-time applications, since the demand here is higher than in post-processing ones. In fact, only one offline algorithm is presented [6]. These algorithms use, as input, GPS coordinates to perform the matching, but most of them consider using an integration of GPS data with Dead-Reckoning (DR) in order to improve the matching accuracy [10]. DR systems use some sensors like odometers and gyroscopes in order to calculate subsequent positions in relation to the initial one. In these systems, the probability of the estimated position being wrong increases exponentially as more readings are made, since the new position given is calculated based on the previous one and on the readings from sensors that might have errors as well [10]. In [8], the author classifies Map-Matching algorithms into four groups, depending on the techniques applied by each one to perform the match. They are: geometric, topological, probabilistic and other advanced algorithms.

3.1. Geometric Algorithms

Geometric algorithms tend only to base the match on the geometry of the road segments, preferring the closest one to the point. They also ignore the way in which the network is connected, leading to various topological errors. There are three types of geometric algorithms, generally named: point-to-point, point-to-curve and curve-to curve. The first one will match the point to the closest point belonging to the network. The second one will prefer the closest map link. The last one is based on the point-to-point match where it selects a set of candidates and then the final curve chosen is the closest one to the composed by the current points being matched. According to [10] the point-to-curve tends to be the best choice and the curve-to-curve the worst. Since curve-to-curve depends on point-to-point, it usually produces bad results due to outliers. Moreover, the algorithmic complexity involved becomes prohibitive for large segments. Figure 2 shows an example of a point-to-curve match with a topological error.

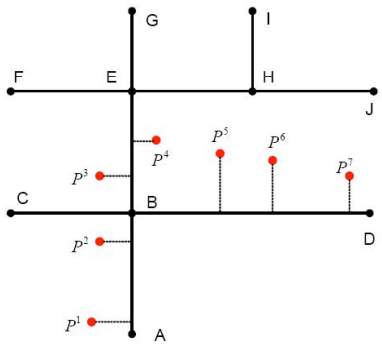

Figure 2 – Point-to-curve match. Points P3 and P4 are topological wrongly matched. Example from [10].

3.2. Topological Algorithms

Since maps are usually represented as graphs, topological algorithms tend to preserve continuity in the matching, avoiding this way those kinds of errors. However, they generally ignore additional readings from GPS data such as speed or heading and might be sensitive to outliers as well. One example of a topological algorithm is the Marchal's algorithm [6], which will be explained below in detail. These algorithms can generally be divided into two stages. The first one is the initial matching process, where the algorithm will select the most suitable link from the closest to the initial points. At the second stage, the algorithm will continue matching the points keeping in mind the network topology. In [8], the author also adds that these kinds of algorithms have some problems at junctions where the direction of links is not similar. This can only be solved via a sub-routine that selects the appropriate subsequent road segment. Since this routine runs in a post-processing mode, these algorithms tend to be useless in real-time applications.

3.3. Probabilistic algorithms

Probabilistic algorithms use a region, the error region, usually an ellipse or a rectangle to match the given point. From that region, the matched link is selected according to the direction, speed, connectivity and proximity from the point to the link. Some algorithms create error regions for each trace point [7], others [8] only create them near to junctions improving in this way the performance of the algorithm and also avoiding mismatches in case of having other road segments nearby. Figure 3 shows an example of an error region.

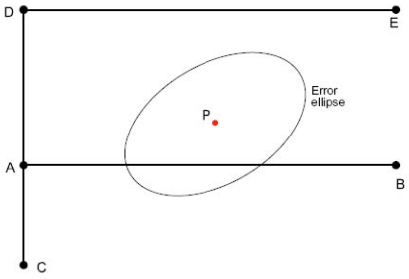

Figure 3 – Error ellipse having inside part of the link AB. Example taken from [8].

3.4. Advanced Algorithms

The advanced algorithms generally use the most varied techniques and approaches, or combine them with the simplest algorithms described above or even a simple combination of Map-Matching algorithms. The major goal is always to improve the accuracy of the matching. Aside from GPS coordinates, these algorithms are often aided with extra information such as speed, heading, connectivity of the roadmap, quality of the input data or even using correction errors from third party systems (e.g. Differential GPS). The approaches most used here are: fuzzy logic models, Dempster–Shafer's mathematical theory of evidence, Multiple Hypothesis Technique (MHT) or Bayesian inferences. Kalman Filters and Extended Kalman Filters are widely used as well, especially to integrate the data from GPS and from DR systems or, in other cases, to smooth the GPS data before proceeding to the matching.

In every algorithm, the accuracy of the matching highly depends on map resolution and completeness: the higher the resolution, the more accurate the matching will be. Some comparisons about map resolutions have been made in [6, 11]. By default, the majority of the algorithms assume that the map network is complete and that it is always possible to have a well succeed matching. This is often an incorrect assumption, and algorithms might have unexpected behaviours when there are no roads nearby to match.

4. The Marchal algorithm – an improved version

4.1. Overview of the Marchal algorithm for offline map matching

The algorithm presented by [6] is an offline topological algorithm inspired in MHT used on previous algorithms [10]. Authors say their algorithm is more focused on computational speed rather than on accuracy as opposed to the remaining ones. The algorithm only uses GPS coordinates to perform the matching in a road network represented by a directed graph.

The algorithm works as follows: firstly, map links nearby the first trace point are picked and each one will constitute a scored path candidate (a possible match sequence). The score of each candidate is based on the sum of the least Euclidean distance between each trace point and its matched link (also named as matching distance). Hence, best candidates have lowest scores.

After the initial matching process, for each point of the trace, it is assumed that the current point matches the last link of each candidate. Then, an update of the score occurs and the candidate is put into a new set. Afterwards, if the trace has reached the end of the link, new path candidates are created. To see if an intersection has been reached, a comparison is made between the travelled length through the trace points and the travelled length through the links of the path candidate. If the first length is bigger than a given percentage of the second, it is assumed that the next junction

has possibly been reached[2]. For this percentage, the authors fixed the value in 50%, which gives fair results in their opinion. New candidates are created similar to the current one and a link per new road segment starting on that junction will be added to each one of the new candidates. Their score is updated and they are inserted into the new set. When no more candidates to match the current point are available, the algorithm will pick only the best N candidates of the new set, it passes to the following trace point and does everything all over again. The authors tested some values for N and, based on these experiments, they say that with values above 30, improvements on matching accuracy are insignificant. The best candidate obtained gives the final match. This algorithm has some additional mechanisms that permits breaking the match and restarting it from scratch when the distance between two consecutive points or the difference between timestamps of two consecutive points is above given thresholds. The authors use, as an example, 300 meters for the distance and 30 seconds for time difference.

4.2. Our Implementation and Modifications

The implementation of the algorithm has been modified from the original version due to several reasons. The first one, and the more obvious, is the addition of a mechanism to detect whenever a new set of points corresponds to a road non-existing in the map. For this reason, every match distance (between GPS point and matched map position) should fall below a given threshold (fixed in 20 meters after several experiments). This condition must be verified in the initial matching process and during subsequent matches, for at least the best path candidate. This way, every set of unmatched points will be added to the map as a new road. If the subsequent matching is broken, then the best candidate (without considering the last point) is saved and the algorithm restarts at the initial matching with the point that caused the break. After processing the entire trace, we have two distinct sets of points: the first one with the matched segments and another one with the unmatched segments. These two sets correspond also to the output of our implemented Map-Matching algorithm module.

[2] To understand more clearly this reasoning, the reader should know that, in the map representation used in [6], each link connects two intersections (at each end). Therefore, if the trace has far exceeded the length of the link, then it is very likely that one of the intersections was reached.

Figure 4 – Matching a new trace. Yellow lines represent the existent map and arrows represent its direction. The magenta line is the new trace. Blue lines represent the matched trace points and the triangular signs the unmatched ones.

Other major modifications were made recently owing to some differences in map representation that affect the matching process. For the authors, the map is represented as a graph where each node is a junction and the links are "polylines" representing the curvature of each link [6]. On the other hand, our map is also represented as a directed graph but with two layers. In the upper layer, we have one node per junction (called super-node) and each link (called super-link) connects two super-nodes. This layer is the closest one to the representation in [6] and it corresponds to the notion of "road network", where each link connects two junctions (as opposed to being connected to another link). At the lower layer, super-nodes are also represented as nodes but super-links are replaced by a set of nodes connected by directed links that represent the pattern of the super-link. This way, we keep the road geometry as close to the obtained from the traces as possible and at the same time maintaining a flexible representation in terms of geometry corrections. Since the matching process makes use of the lower layer, the algorithm had to be adapted in order to improve the final results. Another modification relates to the detection of whenever it is necessary to jump to the subsequent links. The original function did not produce good results in our case because of two distinct situations: the length of the path through the trace points and through the path candidate can be slightly different; GPS readings are subject to errors and differences between both (GPS readings and road segment pattern) are very common in curves – see figures 5 and 6 as examples. Another reason has to do with the map representation. Since we are working with the lowest layer of our map, the function is triggered on every link and not only in junctions; this lead us to have a considerable set of candidates, quite similar between them, making the matching break unnecessarily near some junctions. This break occurs because the pattern of the trace and the correct path candidate were different and away from each other leading the algorithm to remove such candidates from the set since they had the worst scores when compared to the majority of the candidates. Then, two or three points ahead, matching would stop because none of the candidates corresponded to the correct path and a restart had to be performed. Since distances between consecutive trace points and link's length are not homogeneous, we introduced a new concept, tolerance link (see figure 7). The idea is to give the possibility to match accurately a point to a link without the need of the previous link being matched with any point. This way, we are

allowing that a reduced number of non matched links is included in the matched output without affecting the accuracy of the overall match - since these links can only exist between two matched ones, there is a high probability that the user passed over there. This also avoids the need of the matching process to stop and restart unnecessarily. At this moment, the number of tolerance links is fixed to 2.

Figures 5 and 6 – Difference between map and trace patterns in curves. The yellow line represents the map and magenta the trace.

Figure 7 – Concept of *tolerance link*. There is one link without being matched by any trace point between two consecutive matched links.

Due to all those new situations, we decided not to include the condition that tests if a jump to the following link is performed or not. Instead of this, at each trace point we create a set of candidates per current candidate unconditionally. Each new path candidate has a distinct link to do the matching. The links are: the link that matched the previous point of the past candidate and all reachable links from there at a maximum depth level of the number of tolerance links plus one. After scoring all the new candidates, they are filtered before passing to the following point. Two restrictions are applied in order to avoid similar path candidates that only would increase the number of candidates exponentially without having any improvement and to invalidate matches that, although being topologically correct, are far away from the trace points and so we assume that it is a new road segment instead. For the first restriction, only the best candidate passes per most recently matched super-link. This way, the number of candidates is highly reduced and we guarantee to have the best candidates from all possibilities available. On the second restriction, the distance between the last trace point and its match link must be lower than a given threshold (fixed in 45 meters). All candidates where the last match is above this threshold are simply removed, so these candidates will not be considered better than the "real" accurate ones on the following points in unexpected and rare situations.

5. GEMMA – GEnetic Map Matching Algorithm

5.1. Overview

The creation of a new genetic algorithm was conceptualized by us, since we did not find any reference, in the literature, to Map-Matching algorithms using evolutionary approaches. We knew that in terms of computational performance it would be less efficient than other types of algorithm, especially Marchal's, but the main concern was on improving the quality of the matches. The goal was to design an algorithm that would not have the same problems commonly seen in other algorithms, as described in the state-of-the-art section (e.g. matching errors due to topological situations or outliers) and also one that could perform smooth transitions between a matched segment to an unmatched and vice-versa, and also in transitions between two matched segments that are not interconnected yet. In our genetic algorithm[3], each individual consists of a matching sequence (from beginning to the end of the trace). Each gene corresponds to a trace point. The possible alleles for each gene are the links that are close to the respective trace point. It is also inserted a special value to give the opportunity not to perform any match for the given point. Then, and having an initial population randomly created, the algorithm will run for a given number of generations and the best individual of the last population is considered to be the correct match. In each generation, individuals have the possibility to be recombined and mutated. After that, they are evaluated using a fitness function that considers many factors (described below). Since the search space (and thus the program complexity) increases exponentially with trace length, we decided to break the trace into small segments inspired on [12]. In so doing, better individuals are obtained in less time. The break points are selected based on a score function that prefers less crowded areas, away from junctions.

5.2. Segmentation

As said before, the segmentation was inspired on [12] in order to speed up the algorithm and get better results. Before starting the matching process, the algorithm segments the trace in the following manner: Firstly, the algorithm scores every trace point, lowest scores mean less ambiguous areas to the matching process; then, it looks for sets of four consecutive points that are under a given threshold (fixed in 0.9); finally, and using the previous sets as segment borders, the algorithm will try to make the widest segments but restricted to a maximum number of points per segment, fixed in 50 at the moment. The score is based on the sum of four distinct variables, which are normalized according to their units. The variables are: the difference of heading between the point and the closest map link, the distance between the point and the same link, the number of map links that are nearby the trace point (the defined distance is 20 meters) and the heading variation in the neighbourhood of the trace point - the closer trace curves are, the larger heading variation is.

[3] It is far beyond the scope of this paper to describe how Genetic Algorithms function. There are many smooth introductions. For a thorough course, please read [13]

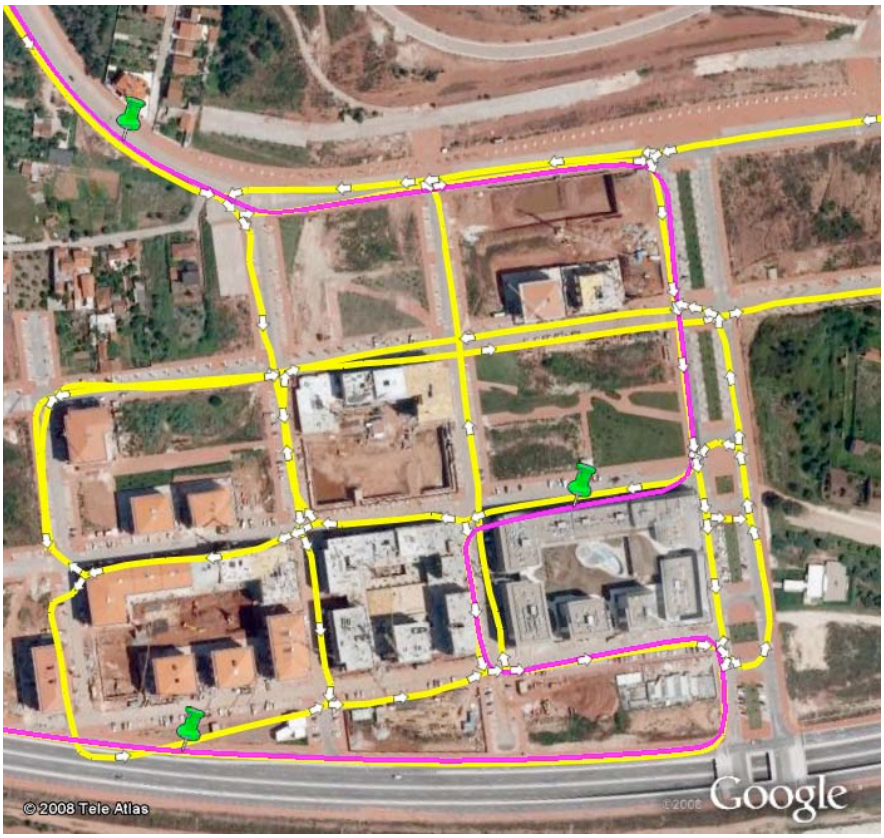

Figure 8 – Green pushpins represent the borders of each segment, which are located in less ambiguous and crowded areas. Trace goes from the top left corner to the bottom right corner.

5.3. Link Candidates

For each trace point, a set of link candidates is available as alleles. A special allele is also inserted in order to give the possibility of no match for that given point. At this moment, these link candidates can be collected according to two distinct methods. In the first one, they come directly from the map. Firstly, links in the area of each point are picked up. Then, the closest one per super-link is selected. The maximum distance allowed is 20 meters and links with opposite heading to the trace point are discarded in order to avoid matches with roads running on the opposite direction. For the second method, Marchal's algorithm is run first for candidate search. The candidates of each point are all links to which that point matched during the entire running of that algorithm. Afterwards, candidates are filtered using the same rules applied in the first method (maximum distance of 20 meters, opposite heading and one link per super-link). This second approach brought deep improvements, especially close to junctions, when compared to the first one, during the first versions of the fitness function, but at the moment, the difference between them is minimal.

5.4 Individual Representation

Individuals of the population are match candidates for a given trace. Each gene of the individual corresponds to a trace point and everyone has its own set of alleles that are unchangeable between themselves. Candidates are ranked according to a fitness function.

5.5. Fitness Function

Each individual is scored with a set of criteria that intend to evaluate the geometric part of the match, the continuity and the transitions between an unmatched and a matched part and vice-versa. The transitions between road segments that are not interconnected are evaluated as well. The fitness function is constituted by the sum of eight parameters that are normalized according to their dimensions. Since this is a minimization problem, best individuals have lowest scores, which are always greater or equal to zero. One of these parameters is the average distance between each trace point and its matched link. Unmatched points are also considered here with a default value of 20 meters. The maximum distance previously obtained is another parameter and the other is the maximum distance of the minimum distances of each matched segments. The latter tends to penalize matched zones where every matched point is too distant from the road segment. A parameter is also used to penalize candidates that privilege not matching rather than matching on acceptable conditions. This is measured with the length of the trace that is unmatched. The concept of tolerance link also exists here (see figure 7). In fact, it was created firstly for this algorithm and was later adapted to our Marchal's implementation. Another parameter consists of the sum of distances between two links when it is not possible to reach the second from the first one. The number of tolerance links used in the genetic algorithm is currently 5, so if it is not possible to go from one link to the other one at a maximum deep of 6, the Euclidean distance between the end point of the first link and the start point of the second link is added to the specific parameter. In order to keep matching continuity as much as possible, another parameter is used to store the sum of the square of the matched lengths of each segment. Since we want to minimize the score, the inverse of the obtained value is used. Finally, the last two parameters are used to smooth the transitions between matched and unmatched areas and vice-versa. One of these parameters stores the average distance between an unmatched trace point and the following matched link or the distance between the last matched link and the following unmatched trace point. The other one stores the maximum of these distances. Each parameter is weighted, thus allowing for different orders of importance. For example, we prefer continuity to geometric proximity, so the three parameters that measure the unmatched lengths, the distances between unreachable links and the square of the continued matched lengths have the highest weights.

5.6. Running the algorithm

For each segment of the trace, the algorithm generates a random population. Each gene has a roulette wheel with the respective alleles. Every allele has the same probability except for the special one that represents an unmatched situation, which has a fixed probability of 15%. On each generation, the population has the opportunity to be recombined and mutated. Pairs of two

individuals are selected using the tournament selection method and have a probability to be recombined, fixed in 75%. The recombination method used is one point crossover, with the point being selected randomly as well. After that, each gene of the individual in the population has a slight probability to be mutated (0.5 %). The new link is picked up randomly from the correspondent roulette wheel built at the beginning. From one generation to the following one, we decided to include the best previous individuals without being recombined or mutated. 3% of the population passes directly, which corresponds to six individuals, since the population size is two hundred. There only exists one stop condition for the algorithm. It stops when the best individual has not being changed for the last given generations. Currently, this number of generations is three hundred.

5.7. Observations

The output of the algorithm is equal to the one presented above: a set with the matched paths and another one with the unmatched trace points. They are built from the best individuals of each trace segment. As of its nature, the genetic algorithm itself has been suffering some evolutions along the time, especially in what concerns to the fitness function. Consequently all the thresholds discussed here were defined based on observations in our set of traces, after a relatively large number of experiments (three months of daily tests, in which we refined both the algorithm and the parameters), meaning that new situations can always come up and new improvements to the fitness function or some thresholds adjustments might be needed. The same happens with the remaining parameters of the algorithm, including crossover and mutation rates, and size of population. The values have not been changed so frequently as in the fitness function, and it is less likely that they need new modifications. These adjustments were made after running several tests showing that modifications could improve the quality of the matching process. Currently, we have a set of traces with a total of 526 728 points that corresponds approximately to a length of 11 486 kilometres throughout Portuguese territory (essentially the central area).

6. M-GEMMA - Joining the best from two worlds

As we could see in the last section, the strengths and weaknesses of both algorithms are complementary. For this reason, we thought that an integration of both algorithms could lead us to better results than any of them separately. Since GEMMA is extremely time consuming, we need to restrict its usage to ambiguous areas where Marchal's have some difficulties. First, we run Marchal's algorithm on both directions (processing the trace forwards and then backwards). These two runs give us two possible path matches that can be different or exactly the same, depending on trace and map complexity. If they are exactly the same, then there is no ambiguity in the match and there is no need to run GEMMA. However, even when both runs give the same output, the final points of the trace (or/and the initial) can be wrongly matched as seen in figure 9, so GEMMA is run for the first and last few points (about 5) of the trace just to confirm that the output given by Marchal's is correct.

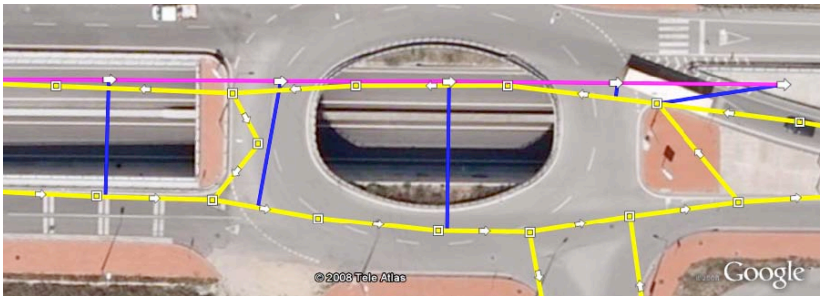

Figure 9 – Final points wrongly matched by Marchal Based Algorithm.

For areas where both Marchal's forwards and backwards runs produce different matches (and sets of candidate links), the whole set of candidate links for every participating point are added to a list. After testing all points, segments are created based on consecutive points that are on the list. For each segment, two unambiguous trace points are added in the borders (to force the start and end of the segment to "fit" into the remaining matches). This way, GEMMA can guarantee the continuity of the match and avoid topological errors. Candidates running on GEMMA are thus taken from the output of both Marchal's runs. Since continuity is guaranteed on GEMMA, the output for the ambiguous areas fits automatically in the remaining map. With this integration, some parameters on GEMMA had to be adapted, namely the population size, the number of generations of the best individual that lead the algorithm to stop and some weights in the fitness function. Since the new segments are commonly very small, the population size is now fixed to fifty individuals and the number of stabilized generations necessary for stopping is seventy-five. This way, the algorithm will find the best individual in the first few generations. Processing time is slightly bigger than running Marchal's alone, since we have to run Marchal's twice per trace here and GEMMA on some segments. Despite that, results show that the gained quality largely justifies the loss of performance, and as the map becomes more complete fewer ambiguous segments appear to run on the genetic algorithm – reducing even more the time.

7. Experiments and Comparative Analysis

Having implemented the four algorithms described, it is necessary to find which one adapts better to the objectives and performs better results. Each algorithm has benefits and drawbacks, either related to computational speed or to matching accuracy.

The base map to work with was extracted from OpenStreetMap.org [14]. This choice was due to several reasons: it is open source and freely available; it is partially complete; in covered areas, it is comparable to TeleAtlas or NavTeq commercial solutions in terms of accuracy and completeness. For the sake of the experiments, we are confident that this choice is as valid as any other map database available (commercial or not). At most, it could be said that OpenStreetMap is globally less complete and more imprecise than those professional databases – which becomes more challenging to our purposes.

We made two main experiments, one on the large area of Coimbra (Portugal) to assess general performance issues; the other one on a smaller area, near our department, that contains many junctions, buildings, areas under construction and new roads (see figure 10). With the latter we intended to perceive accuracy issues.

For the first experiment, we initially built a base map with YouTrace with 225 km of traces in the area of Coimbra, originating 730 intersections and 15901 links (1264 super-links), then we applied both algorithms to match 11 traces with a total of 16 km. With an IntelCore™ 2 Duo processor running at 2.2 GHz with 2 GB of RAM, Improved Marchal's algorithm took 0.171 seconds to determine the entire match. M-GEMMA took 0.874 seconds to do the same task, of which 0.468 were necessary for the GEMMA part to process 152 ambiguous points in a total of 1.363 km. On average, M-GEMMA does need near 5 times more processing effort than Marchal's approach in areas with high density of intersections.

More experiments will be necessary (in other cities, rural areas, areas with plenty of multi-path effect, etc.) to achieve more conclusive results. However, these results are coherent with the experience we had during the development of the algorithms and with other experiments. We are also aware that a thorough algorithmic complexity analysis is needed in order to present a more explicit view of the efficiency involved. On a first analysis, Marchal's algorithm time grows linearly with the size of the trace, with a quadratic component for local search of Euclidean distance. GEMMA behaves in an $O(n * p * m * g)$, with n being the trace size, p the population size, m the average number of alleles and g the number of generations. M-GEMMA corresponds to a combination of these two measures. This is however a *naïf* analysis, since no attention is given to aspects such as distribution of segmentation, sensitive areas or other parameters on any of the algorithms.

Figure 10 – The base map. In yellow we show the OpenStreetMap Map Database links. New roads and junctions are observable (e.g. a new speedway from left to right of the bottom of the image, which is not yet in the main commercial maps).

Figure 11 – Marchal's original algorithm: Given the absence of a descending road in the center, the algorithm insists on keeping the match to the upper road[4].

Regarding the second experiment, we focused on a smaller area, extracting the exact base map from OpenStreetMap.org (the actual roads drawn, not the traces), as we can see in figure 10. Within this scenario, we will test a set of 10 small traces in order to raise and focus on the main problems found in each algorithm: Marchal "original version"; Marchal "improved version"; GEMMA; and M-GEMMA.

Figure 12 – Marchal's original version: Another situation with incomplete maps

[4] We recall that, in Yellow, we have the "base map"; in magenta, with arrows for direction, we have the incoming trace; in blue, we have the matches connecting the trace to the base map.

Although Marchal's algorithm behaves reasonably well with traces with regular samples (in space and time) and with a complete map, it becomes fragile when either of these premises fail. Aiming to solve some of those fragilities, we added the improvements described in section 4.2. As can be seen in Figure 13, we achieved some better results, particularly in unmatched areas. However, it still has strong fragilities in transitions, where the algorithm tends to insist in making matches (where it is already in a "new road"). See Figures 14, 15 and 16.

Figure 13 – Marchal's improved version: same scenario as Fig.11. Accepting "un-matched segments" brings drastic improvements.

Figure 14 – Marchal's improved version: Wrong match within un-matched sequence.

Figure 15 – Marchal's improved version: Transition between un-matched to matched area.

For both versions of Marchal's algorithm, U-turns are also an issue when, in the map, it is topologically impossible to move from a road segment to the one running on the opposite direction. The first points that correspond to the new direction still match wrongly the previous links (see Figure 17). This can simply be reduced to a problem of a transition between a matched zone and an unmatched one.

Figure 17 – U-turn matching by Marchal Based Algorithm.

The last problem found in Marchal's approaches has to do with small matched segments. Generally, paths that have only until three matched trace points correspond to wrong matches. The most common situation occurs when a trace crosses a perpendicular road segment. If that trace is a new road, the algorithm tends to match the closest points to the existent road links with them. Bridges are zones where this happens frequently. For this reason we decided to ignore all these small matched paths (that have at most three trace points). However, this is not always true. Figure 18 shows an example where an accurate match could have been made.

Figure 18 – Small matches being ignored by Marchal Based Algorithm.

In general, the changes we made to Marchal's algorithm allowed for an accurate detection of when unmatched areas occur. This may be sufficient for those applications that do not need the precise breaking spots and that are tolerant to the transition errors described above. However, for many applications, such as in the case of YouTrace, a better accuracy is necessary. The multi-objective properties of the Genetic Algorithm allow for a fitness function tuning such that smooth transitions (those that affect less negatively the several distance measures involved) are preferred. To better illustrate GEMMA's improvements in comparison with Marchal's, we present figures 19, 20 and 21.

In terms of accuracy, the major drawback of GEMMA happens when two (parallel|) matches consistently compete along a large width (Figure 22). In these cases, the algorithm tends to bounce repeatedly from one to the other. It could be said that this situation is rare: it needs two roads that keep parallel to each other along a number of links (typically at least 3) in the same direction. However, given the average error of the GPS (of around 10 meters) this may become common, for example when main and secondary roads parallel each other through entire avenues. It is clear that

GEMMA strongly fails here regarding topology, which is the main strength of Marchal's algorithm.

Figure 19 – GEMMA: The same scenario as in Fig. 17. The U-turn with the Genetic Algorithm

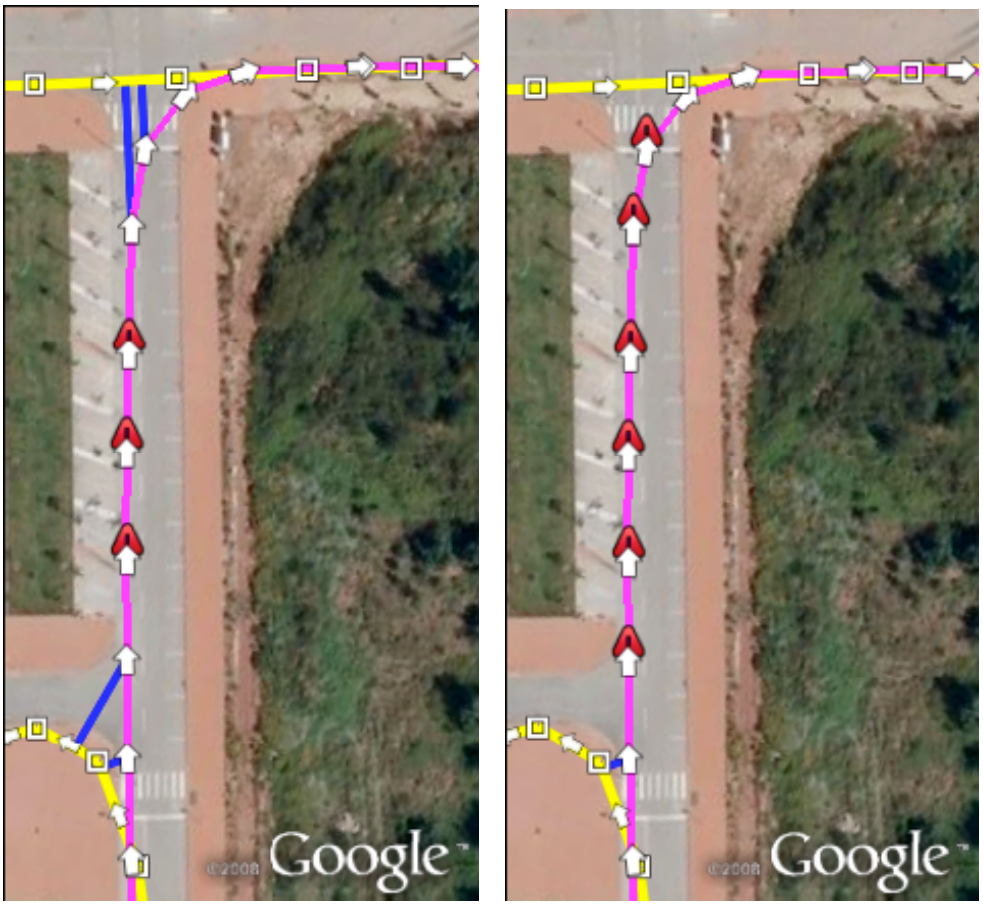

Figure 20 – On the left, Marchal's improved version; on the right, GEMMA's solution. The transition is clearly smoother and more realistic for the latter case.

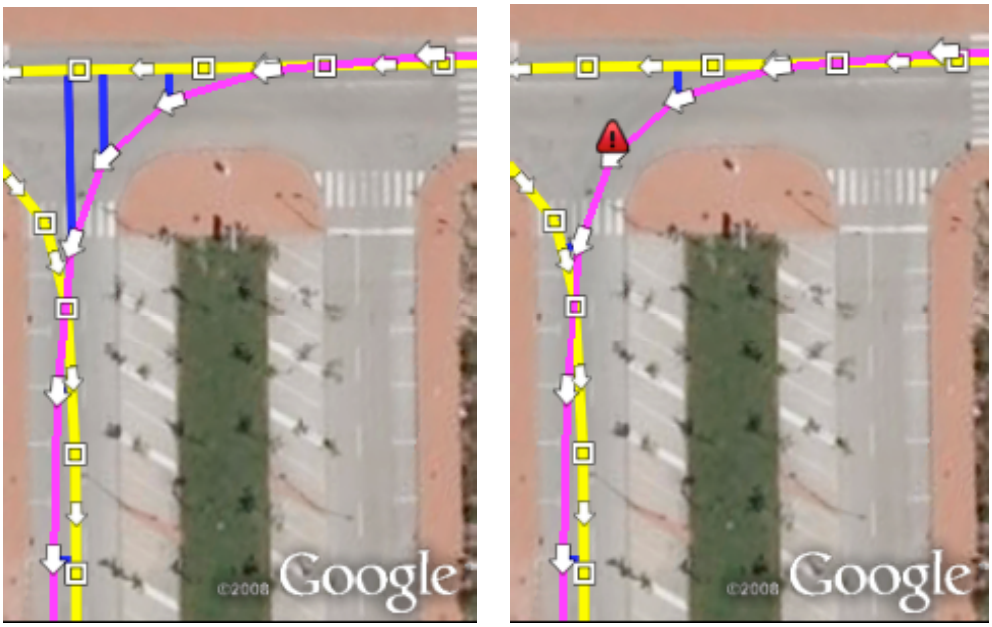

Figure 21 - On the left, Marchal's improved version; on the right, GEMMA's solution. The transition is clearly smoother and more realistic for the latter case.

Quite naturally, M-GEMMA takes advantage of the complementarities of the two approaches above-mentioned. It not only assures topological continuity but also finds smoother transitions between matched and unmatched areas. Figure 23 shows an example of a trace that includes many situations of transitions, matched areas, unmatched areas and irregular sized samples. The journey starts on the right and then goes around two blocks, finally moving out of the area through the speedway in the bottom.

Figure 22 – GEMMA – Parallel road ambiguity.

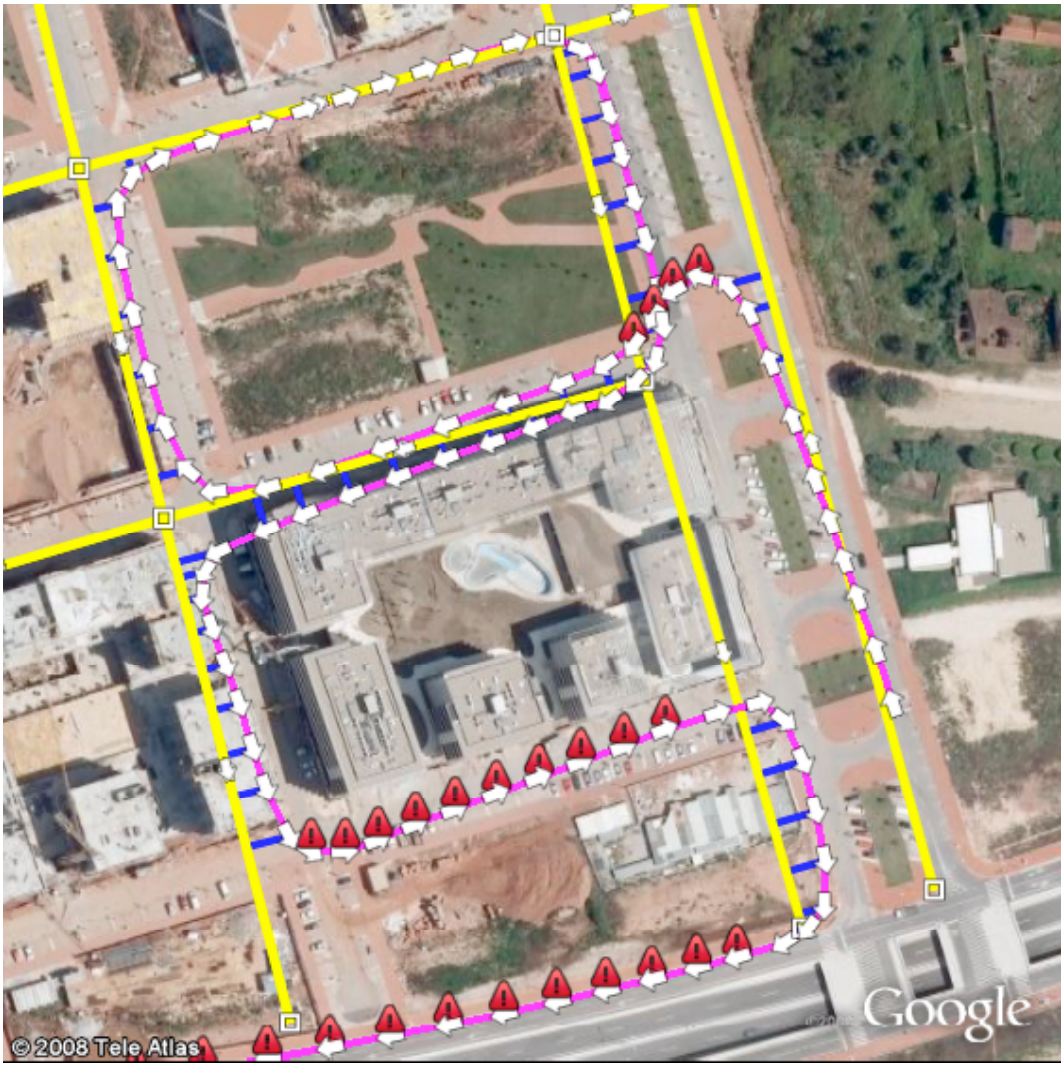

Figure 23 – M-GEMMA: An example of the hybrid approach.

In terms of map-matching accuracy, M-GEMMA still does leave many open issues. For example, the parallel roads problem mentioned above is only partially solved. The Marchal's part of the algorithm does prevent the bouncing from the two roads, but the proximity between the roads leads M-GEMMA to choose between two options: make the entire match (wrong option); not make any match (right option). The fine-tuning of the system in this regard is complicated: if we make it too restrictive (small distance of tolerance), it becomes resistant to "parallel roads", but then it will often wrongly report unmatched segments that could easily be properly processed. Making it too loose drives to the opposite. The GPS NMEA protocol allows for (Dilution Of) Precision estimates (HDOP, VDOP) or SNR (Signal-to-Noise Ratio), but curiously these values are not consistent among different receivers in which respects to the quality of the trace. For the same DOP or SNR values, we have observed very different qualities of traces along the 4 GPS receivers tested. For the case of YouTrace, we rely on the statistics to distinguish between an error and a parallel road (with many traces, there should be two centrelines gradually emerging out of the "statistical evidence"). The parallel road problem increases even more when speaking of several road platforms on top of each other, as happens in many highway entrances and exit ways (although, normally, geometry helps distinguish correct solutions).

Regarding time performance and complexity of the algorithms, we knew from the beginning that, in terms of speed, GEMMA would have a poor performance due to its nature. Some modifications were made such as the creation of the upper layer of the map in order to speed up some searches in the map, which benefitted Marshal's algorithm as well. Despite these modifications and other minor ones, GEMMA alone remains slow. Moreover, adding some more modifications to the Marchal's algorithm, this solution becomes even faster. The difference of time when running both algorithms with the same set of data is notable.

8. Applying M-GEMMA to YouTrace

Regarding the inclusion of these algorithms in YouTrace, the large base map from above (225 km, 730 intersections) needs 186 seconds to be generated, while with Improved Marchal 125 seconds are necessary. The results are different, however, since Marchal's algorithm fails in some areas. On a different test, with a set with 87 005 points which corresponds approximately to 1 392 kilometres of trace length, Marchal's algorithm (either version) took 20 seconds in the matching process, for GEMMA it took approximately one hour and 79 seconds for M-GEMMA.

To allow the reader to have some clearer insight on the quality of the results, we now show and describe some snapshots of the map in the same area used for the tests. A map was generated from scratch (starting with an empty map) using a small set of 24 traces (2926 points). It took a total of 15 seconds to generate the entire map, with 33% of GEMMA matched points (and 67% of Marchal's improved version, obviously). In Figure 24, we can see the overall picture of the final map. We can observe that YouTrace could gather many of the involved roads and crossings. There are, however, some issues. It remains to be observed whether the addition of a large amount of new traces solves partially or totally those issues. In Figure 25, we can see two inferred crossings. Topologically and in terms of correspondence to the original trajectories, both are correct, although geometrically the one on the left seems smoother. This is due, obviously, to the quality of the traces. Figure 26 shows a zoom on the right side of the map. The system inferred all the road segments (some of which did not even exist in the base map of the tests of section 7).

Figure 24 – Map Generated from scratch out of 24 traces (2926 points)

Figure 25 – Two generated crossings (zoom of figure 24).

Figure 26 – Inferred map (zoom of figure 24)

9. Conclusions

In this article, we presented an off-line Map Matching algorithm that shows to be reliable and robust in which regards to the potential incompleteness of the base map at hand. This algorithm is the result of an iterative process in which the authors implemented and tested previous work and added their own new implementations. The result is the integration of two algorithms: Marchal's algorithm [6] and GEMMA. Marchal's algorithm is used primarily for using topological continuity to infer matches. When ambiguities arise, the portion of the ambiguous segment is isolated and GEMMA is called.

M-GEMMA showed to be slower than Marchal's original algorithm, but its performance is more than acceptable when running for a single user. The scalability to multiple simultaneous users (as is expected in YouTrace) remains to be tested and may demand improvements. Despite this issue, it is preferable to use the integration of both algorithms because of the improvements on having smoother transitions – which have a direct impact on map's quality.

Although the results represent clear improvements to the state of the art, offline Map-Matching algorithms continue to have problems that none of our solutions could solve. One has to do with the threshold for the maximum matched distances (fixed in 20 meters, as said before). We fixed

this value because, after various observations, it could fit the most common situations. However, there are some exceptions where this is not true. For situations where we have parallel roads that do not interchange, and one of the roads already exists in the map and the other one is absent, all the presented algorithms will assume that it is always the same road. Even when reducing the maximum matched distance threshold could not solve all the cases. Figure 27 describes a typical scenario

Figure 27 – Matching wrongly a parallel road. Part of the trace matches a parallel road segment instead of leaving it unmatched. The matching distance is about 18 meters.

In terms of the integration into YouTrace, M-GEMMA is presenting satisfying results during the preliminary experiments. Testing thoroughly this whole system demands considerably larger chunks of traces and it is clearly beyond the scope of this paper. Future publications will focus on this task.

The code of M-GEMMA is written in C++ and is available as open source at http://eden.dei.uc.pt/~camara/files/mgemma.zip. The reader is invited to download and use it at will.

References

[1] NCHRP Project 70-01. Private-Sector Provision of Congestion Data. Probe-based Traffic Monitoring. State-of-the-Practice Report . University of Virginia Center for Transportation Studies. Virginia Transportation Research Council. November 21, 2005

[2] Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H. N., and Mishalani, R. (1998). DynaMIT: a simulation-based system for traffic prediction. Proceedings of the DACCORD Short-term forecasting workshop (DACCORD) February, 1998.

[3] Filippo Logi, Marc Ullrich, Hartmut Keller. Traffic Estimation in Munich: Practical Problems and Pragmatic Solutions. 2001 IEEE Intelligent Transportation Systems Conference Proceedings - Oakland (CA), USA - August 25-29, 2001

[4] Travel Time Estimation Using Cell Phones (TTECP) for Highways and Roadways. Department of Electrical and Computer Engineering Florida International University

[5] Performance and Limitations of Cellular-Based Traffic Monitoring Systems. CellINT Traffic Solutions. 2007. Ohio Transport Engineering Conference.

[6] Marchal, F. and Hackney, J. and Axhausen, K.W., 2005. “Efficient map-matching of large global positioning system data sets: Tests on speed monitoring experiment in ZŸrich”. Transportation Research Record 1935, 93–100.

[7] Edelkamp, S. and Pereira, F. C. and Sulewski, D. and Costa, H., 2008. “Collaborative Map Generation – Survey and Architecture Proposal”, in Urbanism on Track – Application of Tracking Technologies in Urbanism, Hoeven, F. and Shaick, J. and Speck, S. C. and Smith, M. J.

[8] Quddus, M. A. and Ochieng, W. Y. and Noland, R.B., 2007. “Current map-matching algorithms for transport applications: State-of-the art and future research directions”, Transportation Research C: Emerging Technologies, 15(5), pp 312 - 328, ISSN 0968-090X.

[9] Greenfeld, J.S., 2002. “Matching GPS observations to locations on a digital map”. In proceedings of the 81st Annual Meeting of the Transportation Research Board, January, Washington D.C.

[10] Quddus, M. A., 2006. “High integrity map-matching algorithms for advanced transport telematics applications”, PhD Thesis. Centre for Transport Studies, Imperial College London, UK.

[11] Quddus, M. A. and Noland, R. B. and Ochieng, W. Y., 2006. “The effects of navigation sensors and digital map quality on the performance of map-matching algorithms.” Presented at the Transportation Research Board (TRB) Annual Meeting of the Transportation Research Board, Washington D.C., January 2006.

[12] Chawathe, S., 2007. “Segment-Based Map Matching.” Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istanbul, Turkey.

[13] Goldberg, D., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Kluwer Academic Publishers, Boston, MA. 1989.

[14] Open Street Map platform. URL: http://www.openstreetmap.org