An off-line map-matching algorithm for incomplete map databases

**Francisco Câmara Pereira · Hugo Costa ·
Nuno Martinho Pereira**

Abstract The task of map-matching consists of finding a correspondence between a geographical point or sequence of points (e.g. obtained from GPS) and a given map. Due to many reasons, namely the noisy input data and incomplete or inaccurate maps, such a 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 us to handle incomplete map databases. We test and compare this with other approaches and ultimately provide guidelines for 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 geographical system to associate information to specific geo-referenced locations. Thus, 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 negatively affect their usefulness. For example, for car navigation, for example, extreme care must be taken to continually locate the real position of the driver as opposed to what the GPS receiver estimates. 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 traffic situations [1]. These 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 stronger example could be the 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, 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 under a more careful examination this shows that there are two different approaches to two entirely 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 to determine a correspondence between sequences of geo-referenced

F. C. Pereira (✉) · H. Costa · N. M. Pereira
Centro de Informática e Sistemas da Universidade de Coimbra
(CISUC), Departamento de Engenharia Informática,
Universidade de Coimbra,
Pólo II, Pinhal de Marrocos,
3030 Coimbra, Portugal
e-mail: camara@dei.uc.pt

H. Costa
e-mail: hecosta@student.dei.uc.pt

N. M. Pereira
e-mail: nfmart@student.dei.uc.pt

points previously obtained (e.g. from GPS) and a given map. 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 presently widespread and available. With noticeable less accuracy, other low cost localization approaches are becoming common, such as cell-phone based localization (e.g. [4, 5]). For these, accurate Map Matching becomes a quite complex and determinant task.

Another aspect 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. In roads that are absent on maps, the Map Matching algorithms typically take some time to become aware of it. They stay "glued" to existing road links until they become too distant, and then typically enter into an "initialization mode" that starts promoting a new match when sufficiently close to a recognized map link. However, the "new road" segment becomes blurred in this process. For applications that demand some accuracy this may affect results. There is at least one application on 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 these two approaches and is 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, which is 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 following section will provide an overview of the YouTrace project in order to provide some context to M-GEMMA.

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

The experiments and a comparative analysis are shown in Section 7 concluded with a consensus of our thoughts of strengths and weaknesses about the algorithm.

2 Giving some context: the YouTrace project

The YouTrace project intends to be a social platform that allows users to collaborate with the construction of a map-of-the-world [7] (Fig. 1). A key element is the Map Generation Engine that is responsible for aggregating the users' traces into a single map. A YouTrace user can upload their traces while contributing to the construction of a joint map of the world. The users can then receive an updated map that will allow, for example, a more efficient car navigational application. An innovative characteristic of collaborative mapping is the strength of its dynamics, as opposed to the current static maps, as we are well aware: Roads are constantly being updated and aggregated as new traces are introduced. The collected traces can then provide information for 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, as they 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 in turn, will be added to the map.

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, which thus

---

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

improves road precision. The non-matched points of the trace are aggregated on to the map, creating new roads (or trajectories). Two databases are then generated from this process; the map database and the statistics database. These databases serve to provide data for external services such as route planning and traffic analysis. For more information on the YouTrace project, please refer to [7]. As can be understood, the Map Matching is a key element for YouTrace, which is entirely responsible for finding the parts of the trace that already exist on the map. This allows for the distinction between the parts that should be aggregated and those that should be updated, therefore the quality of the final map is 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 are noticeably 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. 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. Online algorithms are less effective. Offline algorithms can take the advantage of not only matching each point according to past data but also based on the following of a "future" point, which helps the algorithm to select the correct road when 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 is higher than in post-processing ones. In fact, only one offline algorithm is presented [6]. These algorithms use GPS coordinates as input source to perform the matching, but most 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 incorrectly estimated positions increases drastically as more readings are made since a new position is calculated based on previous readings from inaccurate sensors [10]. In [8], the author classifies Map-Matching algorithms into four groups, depending on the techniques applied by each 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 road segments, with preference to the closest segment to the point. These tend to 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 will match a point to the closest point belonging to the road network. The second one will prefer the closest map link. The last is based on the point-to-point match where it selects a set of candidates and then the final curve is chosen as the closest composed by the current matched points. 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, this 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.

3.2 Topological algorithms

Since maps are usually represented as graphs, topological algorithms tend to preserve continuity in the matching, avoiding frequent errors. However, they do generally ignore
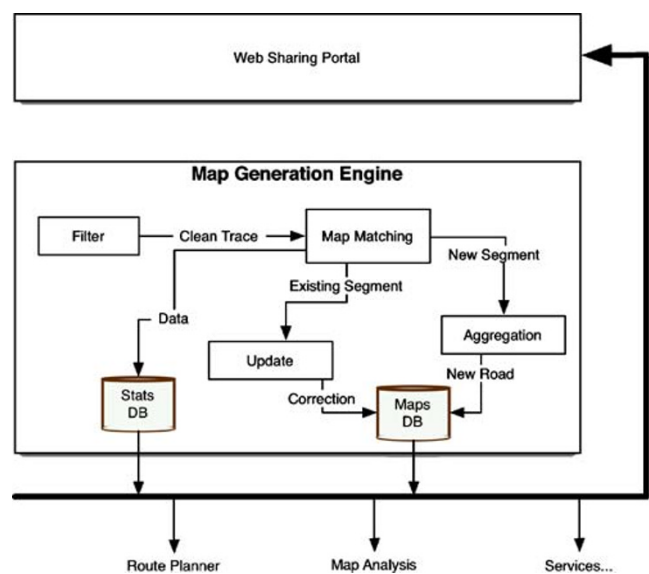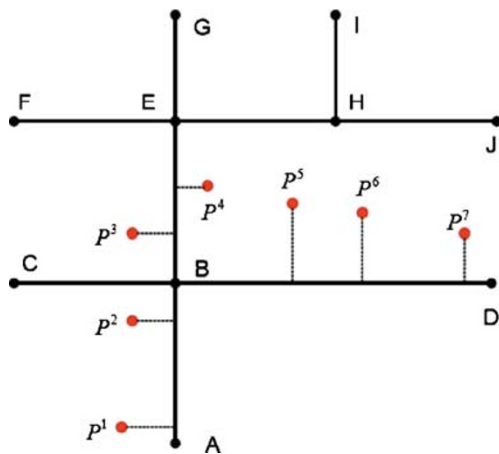

Fig. 1 YouTrace Map Generation Engine architecture

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

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

additional readings from certain GPS readable 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 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 while keeping the network topology in consideration. In [8], the author also adds that these kinds of algorithms have some problems at certain 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, an "error region" which is 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] however only create these near to junctions, which improves the performance of the algorithm and also avoids mismatches in case of having other road segments nearby. Figure 3 shows an example of an error region.

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
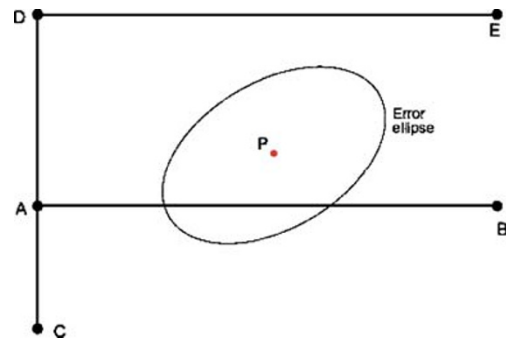
from GPS coordinates, these algorithms are often aided with extra information such as speed, heading, connectivity of the roadmap, quality of the input data or even using correction errors from third party systems (e.g. Differential GPS). The approaches most used here are: fuzzy logic models, Dempster–Shafer's 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 matching. 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 matching completed. However, this is often an incorrect assumption and algorithms might show unexpected behaviour where 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, yet 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 then based on the sum of the least Euclidean distance between each trace

Eur. Transp. Res. Rev. (2009) 1:107–124

111

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 longer 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 in their opinion tends to give fair results. New candidates are created, they are 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 m for the distance and 30 s 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 and more obvious reason is that the addition of a mechanism to detect whenever a new set of points correspond to a non-existent road in the map. For this reason, every matched distance (between GPS point and matched map position) should fall below a given threshold (fixed at 20 m, after several experiments). This condition must be verified in the initial matching process and during subsequent matches, for the best candidate path. This confirms that every set of unmatched points will be then 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 with the matched segments and the other with the unmatched segments. These two sets correspond to the output of our implemented Map-Matching algorithm module (Fig. 4).

Other major modifications were recently made owing to a few map representation differences, which affect the entire matching process. For the authors' purpose, 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]. Our map is also represented as a directed graph, yet it consists of two layers. The upper layer, contains 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 used in [6] and corresponds to the notion of the "road network" where each link connects two junctions (as opposed to being connected to another link). The lower layer presents super-nodes which are shown as nodes but super-links are replaced by a set of nodes connected by directed links that represent the pattern of the super-link. These connections help to keep the road geometry as close to the originally obtained traces as possible while maintaining a flexible representation in terms of geometry corrections. Since the matching process utilizes 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 fair results 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) which are very common in curves (e.g. Figs. 5 and 6). Another reason has to do with the map representation. Since we are working with the lowest layer on the map, the function can be triggered on every link and not only in junctions. This may lead us to have a considerable set of candidates, however similar which makes the matching break unnecessarily near some junctions. This break occurs because the pattern of the trace and the correct path candidate are different, which causes the algorithm to remove such candidates from the set since they had weak scoring when compared to the majority of the candidates. Then, two or three points ahead, matching would need to 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 the link's length are not homogeneous, we introduced a new concept named *tolerance link* (see Fig. 7). The idea is to provide the possibility of accurately matching a point to a link without the need of the previous link being

---

[2] To understand this reasoning more clearly, the reader should be aware 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 likely that one of the intersections was reached.

112

Eur. Transp. Res. Rev. (2009) 1:107–124

Fig. 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

matched with any point. This allows a reduced number of non-matched links to be 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 has passed over this area). This also helps to avoid the need of the matching process to stop and restart unnecessarily. At this moment, the number of *tolerance links* is fixed to 2.

Due to all these new situations, we decided not to include the condition of testing if a jump to the following link is performed or not. Instead of this, at each trace point we decided to create a set of candidates per current candidate. Each new path candidate has a distinct link to perform the matching. The links are as follows: the link that matched the previous point of the past candidate and, all reachable links to 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 drastically reduced and we guarantee to have the best possible candidates available. On the second restriction, the distance between the last trace point and its matched link must be lower than a given threshold (fixed to 45 m). 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

We have created a new genetic algorithm since we did not find any reference in the Map-Matching literature to algorithms that use evolutionary approaches and which would meet our expectations. We knew that in terms of computational performance it would be less efficient than other types of algorithm, especially Marchal's. The main concern was on improving the quality of the matches. The goal was ultimately to design an algorithm that would not

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

Fig. 7 Concept of *tolerance link*. There is one link without being matched by any trace point between two consecutive matched links
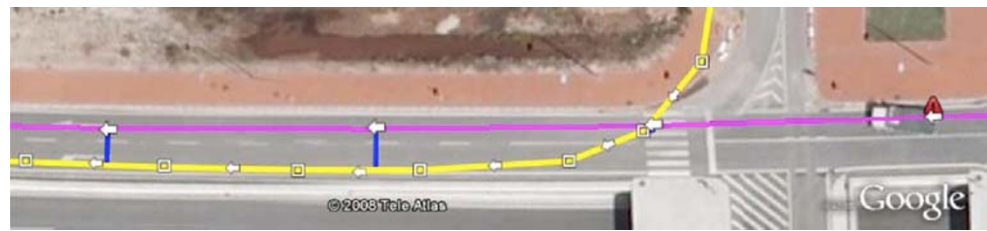
have the same problems commonly seen in other algorithms, as described in the state-of-the-art section (e.g. matching errors due to topological situations or outliers) and also one that could perform smooth transitions between a matched segment to an unmatched and vice-versa, and in transitions between two matched segments that are not yet interconnected. 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. A special value is also inserted to give the opportunity not to perform any match for the given point. After an initial population that is 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. Afterwards, 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 doing so, better individuals are obtained in less time. The break points are then selected based on a score function that prefers less crowded areas, noticeably away from junctions.

5.2 Segmentation

As previously mentioned, the segmentation was inspired on [12] in order to speed up the algorithm and to have 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 represent 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, using the previous sets as segment borders, the algorithm will try to form the widest segments available, yet these are restricted to a maximum number of points per segment (currently fixed at 50). 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 m) and the heading variation in the neighbourhood of the trace point—the closer the trace curves are the larger the heading variation is.

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 an unmatched point to be performed. At this time, these link candidates can be collected according to two distinct methods. The first comes 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 set at 20 m. Links with opposite heading to the trace point are discarded in order to avoid matches with roads running in the opposite direction. The second method, Marchal's algorithm is first run for each 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 m, opposite heading and one link per super-link). This second approach improved results especially close to junctions 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 each given trace. Each gene of the individual corresponds to a trace point and everyone has their own set of alleles that are unchangeable between them. 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 matched part and vice-versa. The transitions between road segments that are not interconnected are also evaluated. The fitness function is constituted by the sum of eight parameters that are normalized

---

[3] It is far beyond the scope of this paper to describe how Genetic Algorithms function. Further information on this subject may easily be sourced; one such example is shown in [13].

Fig. 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
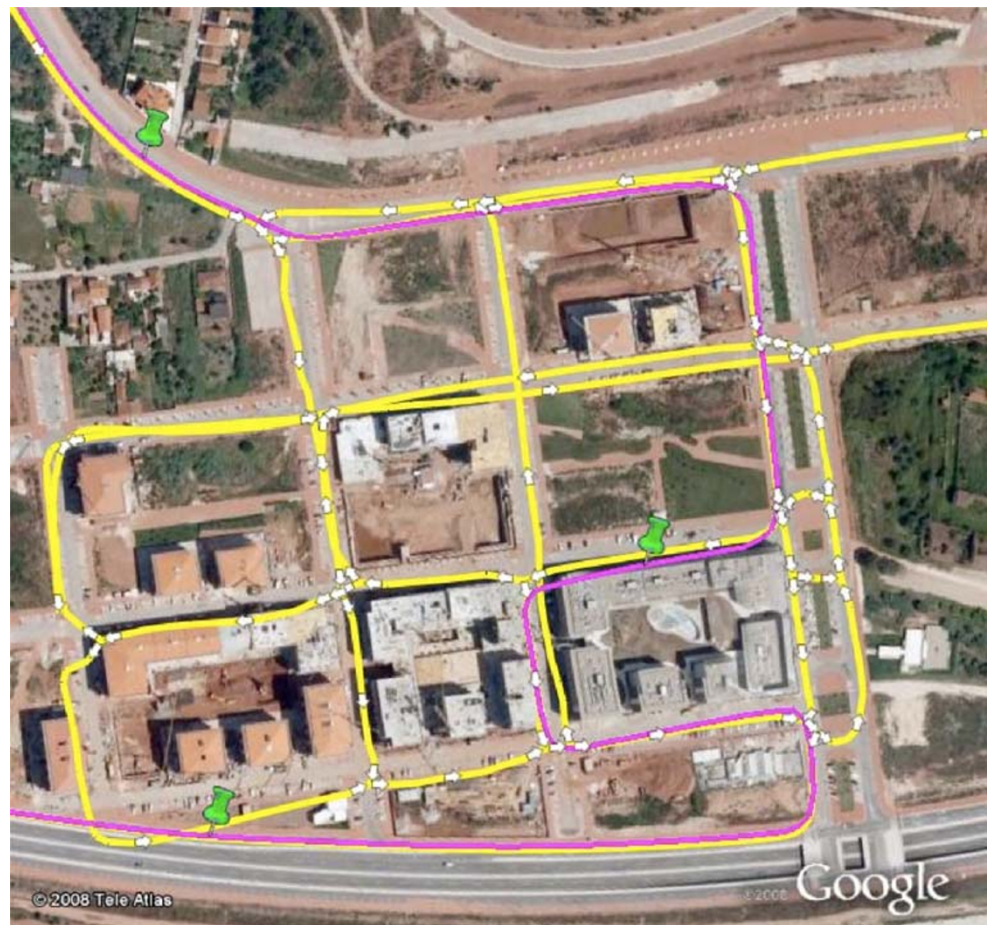
according to their dimensions. Since this is a minimization problem, best individuals who have the lowest scores are always greater or equal to zero. One of these parameters is the average distance between each trace point and its matched link. Unmatched points are also considered with a default value of 20 m. The maximum distance previously obtained is another parameter, and the other is the maximum 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 non-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 Fig. 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 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

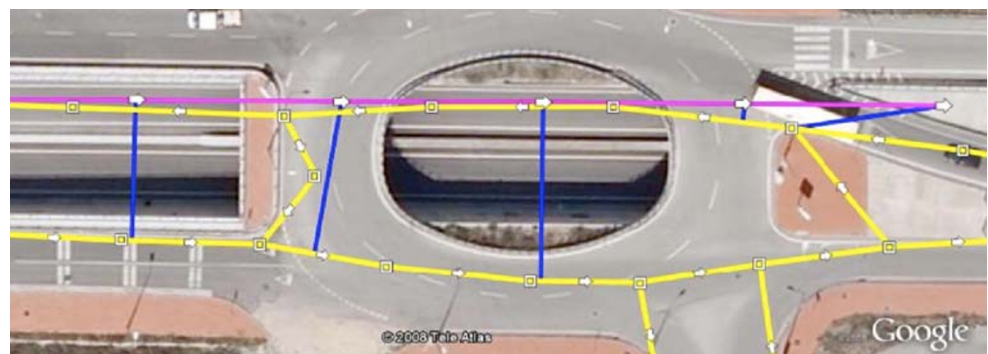**Fig. 9** Final points wrongly matched by Marchal based algorithm

Fig. 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)

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 un-

matched lengths, the distances between unreachable links and the square of the continued matched lengths have the highest weights (Fig. 8).

5.6 Running the algorithm

For each segment of the trace, the algorithm generates a random population. Each gene has a roulette wheel with 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%. The population in each generation has the opportunity to be recombined and mutated.

Fig. 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. 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

116

Eur. Transp. Res. Rev. (2009) 1:107–124

Fig. 12 Marchal's original version: another situation with incomplete maps

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 next, 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). One stop condition only exists 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. These are built from the best individuals of each trace segment. As it is in its nature, the genetic algorithm itself has been suffering some evolution through time, especially in regard 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 as frequently changed as in the fitness function, and it is less likely that they need new modifications. Adjustments were made after running several tests showing that these could improve the quality of the matching process. We currently have a set of traces with a total of 526,728 points that corresponds to an approximate length of 11,486 km throughout Portugal (essentially the central area).

Pairs of two individuals are then selected using the tournament selection method and have a probability to be recombined, fixed at 75%. This recombination method uses one point crossover, with a randomly selected point. Afterwards, each gene of the individual in the population has a slight probability

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

Eur. Transp. Res. Rev. (2009) 1:107–124

117

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 to carry any of these out separately. Since GEMMA is extremely time consuming, we needed to restrict its usage to ambiguous areas where Marchal's has performed some difficulties. Firstly, we run Marchal's algorithm in both directions (processing the trace forwards and then backwards). This gives us two possible path matches, which can be different or exactly the same, depending on trace and map complexity. If the results are exactly the same then there is no ambiguity in the match and thus, 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 Fig. 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.

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 leads the algorithm to stop and some weights in the fitness function. Since the new segments are commonly very small, the population size remained fixed to fifty individuals and the number of stabilized generations necessary for stopping is set at seventy-five. This helped the algorithm find the best individual in the first few generations. Processing time became longer than simply running Marchal's alone since we have to run Marchal's twice per trace and GEMMA on some segments. Despite that, results show a gain in quality which justifies a loss of performance and as the map becomes more complete, fewer ambiguous segments appear to run on the genetic algorithm, thus further reducing 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 Open-StreetMap.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 Open-StreetMap is globally less complete and more imprecise than those other professional databases, which becomes more of a challenge for our purposes.

Fig. 16 Marchal's improved version: transition between un-matched to matched area

We made two main experiments, one in the large area of Coimbra (Portugal) to assess general performance issues; the other in a smaller area, near to our department, which contains many junctions, buildings, areas under construction and new roads (see Fig. 10). With the latter we intended to find 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 15,901 links (1,264 super-links). 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 s to determine the entire match. M-GEMMA took 0.874 s 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 needed nearly five times more processing effort than Marchal's approach in areas with high density of intersections.

More experiments would 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, however, is a naive analysis, since no attention is given to aspects such as distribution of segmentation, sensitive areas or other parameters on any of the algorithms.

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 Fig. 10. Within this scenario, we tested a set of ten 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.

Although Marchal's algorithm behaves reasonably well with traces with regular samples (in space and time) and with a complete map, it becomes inaccurate when either of these

Fig. 17 U-turn matching by Marchal based algorithm

Fig. 18 Small matches being ignored by Marchal based algorithm

premises fail (Figs. 11 and 12). Aiming to solve some of those weaknesses we added the improvements described in Section 4.2. As can be seen in Fig. 13 we did achieve some better results, particularly in unmatched areas. However, it still had mistakes in transitions where the algorithm tends to insist in making matches (where it is already in a "new road"). See Figs. 14, 15 and 16.

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

The last problem found in Marchal's approaches has to do with small matched segments. Generally, paths that have only matched up to three trace points correspond to incorrect 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 does not always hold true. Figure 18 shows an example where an accurate match could have been made.

In general, the changes we made to Marchal's algorithm allowed for an accurate detection of unmatched areas. 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, higher accuracy is necessary. The multi-objective properties of the Genetic Algorithm allow the fitness function to be tuned for 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 Figs. 19, 20 and 21.

In terms of accuracy, the major drawback of GEMMA happens when two (parallel) matches consistently compete along a large width (Fig. 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 m) this may become common. For example, when main and secondary roads parallel each other through entire avenues. It is clear that GEMMA fails in regard to their topology, conversely this is the main strength of Marchal's algorithm.

Quite naturally, M-GEMMA takes advantage of the complements of the two approaches mentioned above. It not only assures topological continuity but also finds

Fig. 19 GEMMA: the same scenario as in Fig. 17. The U-turn with the genetic algorithm

Fig. 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
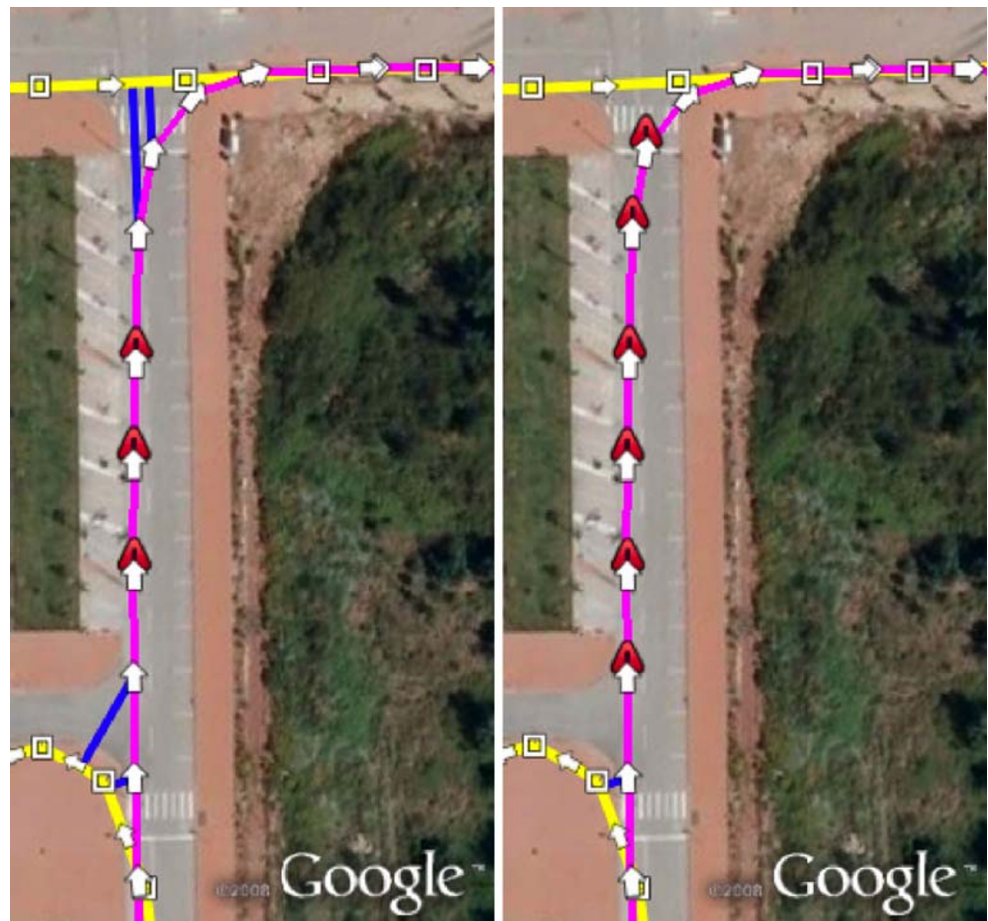
smoother transitions between matched and unmatched areas. Figure 23 shows an example of a trace that includes a variety of transitions, matched areas, unmatched areas and irregular sized samples. The journey starts on the right and

then goes around two blocks, finally moving out of the area through the speedway at the bottom.

In terms of map-matching accuracy, M-GEMMA still has its limitations. For example, the problem of parallel roads is

Fig. 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
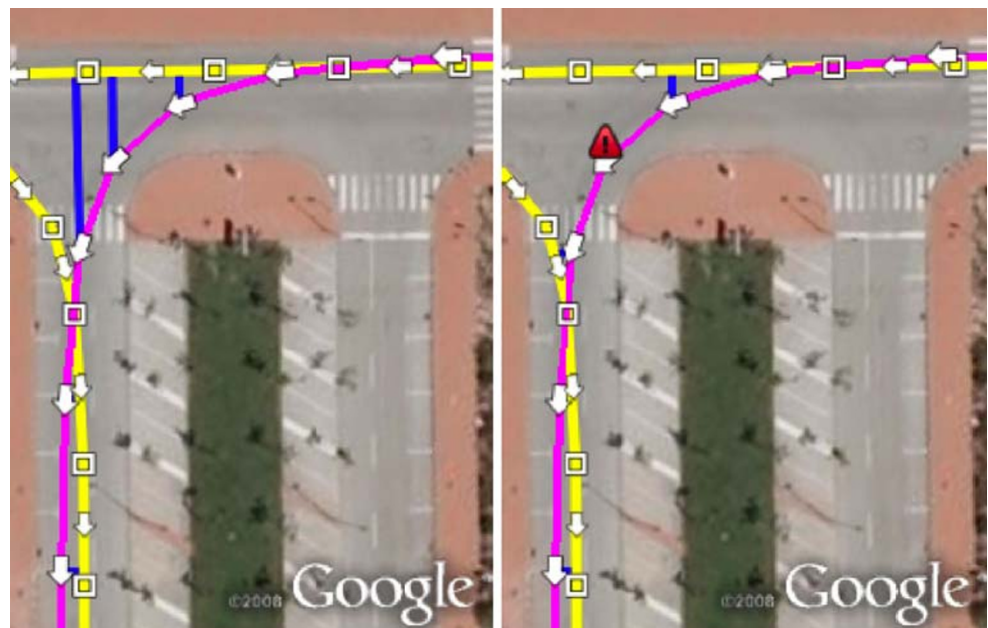
Fig. 22 GEMMA—parallel road ambiguity

only partially solved. The Marchal's part of the algorithm does prevent the bouncing between two roads, however the proximity between the roads leads M-GEMMA to choose between two options: to make the entire match (incorrect option); or not make any match (correct option). The fine-tuning of this system is thus complicated: if we make it too restrictive (slight 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 gives opposite result. 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 with respect to the quality of the trace. For the same DOP or SNR

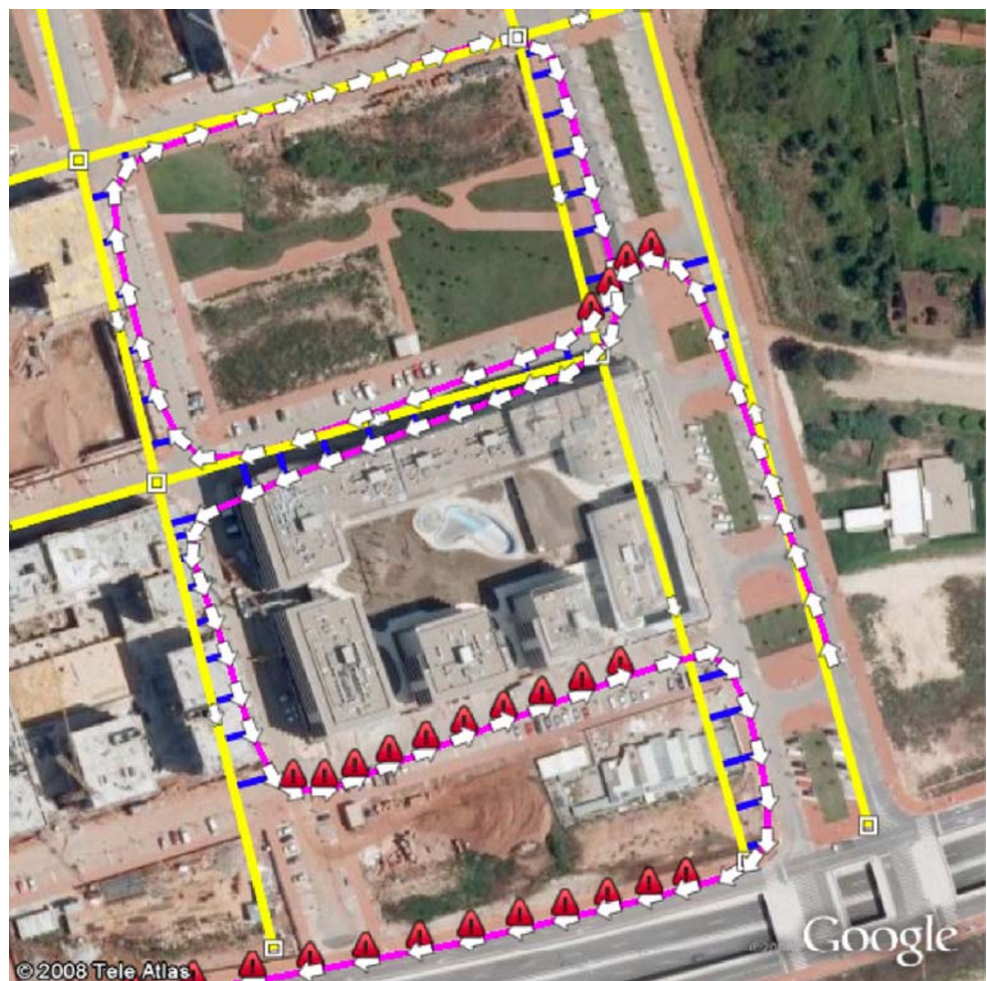**Fig. 23** M-GEMMA: an example of the hybrid approach

Fig. 24 Map generated from scratch out of 24 traces (2,926 points)

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 centre-lines gradually emerging out of the "statistical evidence"). The problem with parallel roads increases drastically when speaking of several road platforms on top of each other, as so happens at the entrance and exit of highways (although, normally, geometry helps distinguish this 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, with adding some more modifications to the Marchal's algorithm, this solution was speeded up. The difference of time when running both algorithms with the same set of data is noticeable.

8 Applying M-GEMMA to YouTrace

Regarding the inclusion of these algorithms in YouTrace, the large base map from above (225 km, 730 intersections) needed 186 s to be generated, while with Improved Marchal 125 s were necessary. The results were different, however,

Fig. 25 Two generated crossings (zoom of Fig. 24)

Marchal's algorithm failed in some areas. On a different test, with a set with 87,005 points which corresponded approximately to 1,392 km of trace length, Marchal's algorithm (either version) took 20 s in the matching process, for GEMMA it took approximately one hour and 79 s for M-GEMMA.

To allow the reader to have a clearer insight on the quality of the results, we then 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 (2,926 points). It took a total of 15 s to generate the entire map, with 33% of GEMMA matched points (and obviously 67% of Marchal's improved version). In Fig. 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 this partially or totally. In Fig. 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 obviously due 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).

9 Conclusions

In this article, we presented an off-line Map Matching algorithm that showed reliability and robustness in regard to the potential incompleteness of the base map at hand.

Fig. 26 Inferred map (zoom of Fig. 24)

Fig. 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 m

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 used.

M-GEMMA is visibly 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 (as previously mentioned, set at 20 m). We fixed this value because, after various observations, it could fit 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 when one of the roads already exists on the map and the other one is absent, all the presented algorithms will assume that this is always the same road. Even when reducing the maximum matched distance threshold, this could not solve all the cases. Figure 27 describes a typical scenario.

In terms of the integration into YouTrace, M-GEMMA is presenting satisfying results during the preliminary experiments. Testing this whole system thoroughly 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 this at will.

References

1. NCHRP Project 70–01 (2005) 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
2. Ben-Akiva M, Bierlaire M, Koutsopoulos HN, Mishalani R (1998) DynaMIT: a simulation-based system for traffic prediction. Proceedings of the DACCORD Short-term forecasting workshop (DACCORD) February, 1998
3. Logi F, Ullrich M, Keller H (2001) Traffic estimation in Munich: practical problems and pragmatic solutions. 2001 IEEE Intelligent Transportation Systems Conference Proceedings—Oakland (CA), USA—August 25-29
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, Hackney J, Axhausen KW (2005) Efficient map-matching of large global positioning system data sets: tests on speed monitoring experiment in ZŸrich. Transp Res Rec 1935:93–100
7. Edelkamp S, Pereira FC, Sulewski D, Costa H (2008) Collaborative map generation—survey and architecture proposal. In: Hoeven F, Shaick J, Speck SC, Smith MJ (eds) Urbanism on track—application of tracking technologies in urbanism. IOS Press (Research in Urbanism Series, Vol. 1), Amsterdam
8. Quddus MA, Ochieng WY, Noland RB (2007) Current map-matching algorithms for transport applications: state-of-the art and future research directions. Transp Res, Part C Emerg Technol 15 (5):312–328, ISSN 0968-090X
9. Greenfeld JS (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 MA (2006) High integrity map-matching algorithms for advanced transport telematics applications, PhD Thesis. Centre for Transport Studies, Imperial College London, UK
11. Quddus MA, Noland RB, Ochieng WY (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, Boston
14. Open Street Map platform. URL: http://www.openstreetmap.org