The Offline Map Matching Problem and its Efficient Solution

Jörg Roth

Faculty of Computer Science
Nuremberg Institute of Technology
Nuremberg, Germany
Joerg.Roth@th-nuernberg.de

Abstract. In this paper we present an efficient solution to the offline map matching problem that occurs in the area of position measurements on a road network: given a set of measured positions, what was the most probable trip that led to these measurements? For this, we assume a user who moves according a certain degree of optimality across a road network. A solution has to face certain problems; most important: as a single measurement may be mapped to multiple positions on the road network, the total number of combination exceeds any reasonable limit.

Keywords. Route Planning, Optimal Paths, Map Matching, Geo Data

1 Introduction

For many applications and services it is useful to know where a user was in the past. All positioning systems to track a user's position have certain measurements error. If we need to know a position very precisely, we have to introduce further mechanisms. Certain approaches are useful for movements on a road network: if a position is measured when driving on a road, we can perform a projection on the road network. This approach is called *map matching* and incorporated into common car navigation systems. As the mapping is performed immediately whenever a new measurement appears, this is a typical *online* approach. However, we can take further advantage of a road network, if we considered an entire trip afterwards: a reasonable projection on the road network should produce a route that could have been driven by the driver regarding her or his driving behaviour and degree of optimality. As we only can provide such a mapping after the trip was finished, we call this problem the *offline map matching problem*. A solution of this problem is useful for several applications:

- statistical analyses of drivers and flows of traffic;
- collecting street charges;
- automatically record a driver's logbook;
- learn typical trips from users, to e.g. automatically generate traffic warnings.

The research about map matching was historically strongly related to the *Global Positioning System* (GPS), where the major goal was to improve GPS measurements with help of a road network [13]. As a first mechanism, a position is mapped to the geometrically nearest road position [11]. Further approaches took into account the road network's topology [3], [9]. [2] also tries to address incomplete topologies. Based on the *multiple hypothesis technique* they follow multiple possible tracks that may approximate a sequence of measurements. An abstract approach to iterate through possible routes that approximate measurements was presented in [12], however, without solving the problem of the combinatorial explosion of route variations. [4] suggests a genetic algorithm to solve it.

Existing work relies on a high density of measurements (e.g. every 10m), a generally high precision of the positioning system and measurement errors with nearly constant offset (of distance and direction) for longer time periods. The properties are fulfilled by satellite navigation systems. However, our approach in contrast should also work for sparse measurements (e.g. ever 500m) and for positioning systems based on signal strength fingerprinting currently used in smart phones. As a consequence, our approach has to fill gaps of unknown positions between measurements by a route planning approach. We strongly believe, a reasonable approach has to base on multi-routing capabilities to achieve a reasonable efficiency. I.e., we have to find optimal routes from multiple starts to multiple targets in a single call.

In this paper we often refer to the car driving scenario, however our approach is also suitable for other means of transportation (e.g. pedestrian, bicycle).

2 The Offline Map Matching Problem

2.1 Problem Statement

The basis of our considerations forms a *road network*: a topology of *crossings* and road segments between crossings, in the following called *links*. A routing function computes optimal paths (according to a cost function) between positions on the road network. In addition to the road topology, we need *link geometries* in order to map measurements to roads.

We can formulate the offline map matching as follows: given a sequence of measured positions that can be mapped to multiple possible positions on the road network; which combination of mapped positions represent the most probable driven route, if we connect all mapped positions by optimal paths?

Fig. 1 illustrates the problem. We have a sequence of measurements denoted by 1-6. Fig. 1a shows a simple map matching approach where we map each position to the nearest position that resides on the road network. If we now connect all mapped positions by optimal paths, we get a rather long path between 3 and 4. Moreover, we have U-turns at these positions. Fig. 1b shows a more probable mapping: even though the mapping distances are bigger on average, the resulting path is more probable than in Fig. 1a as the overall route is shorter and avoids U-turns.

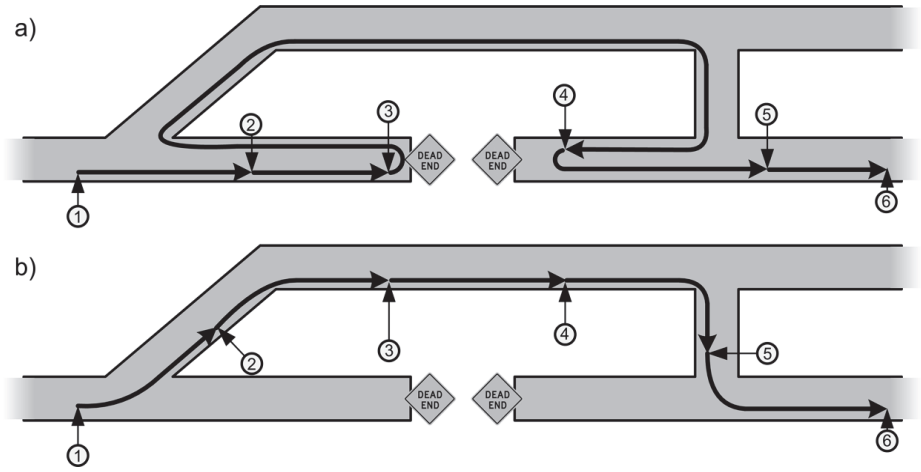

Fig. 1. General problem of mapping measurements to positions

Note that we still talk about probabilities as we cannot definitely say, how a route exactly was. The first variation is possible for a driver, who, e.g. looked for a certain address, drove into a dead-end, turned and searched again. The lack of measurements that represent the long part between 3 and 4 could be explained by temporary measurement problems. However, we would consider Fig. 1b as a more appropriate assumption of the driven route, as it suits more to the measured positions and represents typical driving. Later we want to formalize 'typical' driving by models.

We first want to describe the offline map matching problem more formally. Let M be the set of possible measurements, P be the set of possible road positions and R be the set of routes on the road network for any pairs of start and target. Let $\wp(X)$ denote the set of *finite* subsets of X *and* $\mathcal{L}(X)$ the set of *finite* sequences of X. Further let

- *map*: $M \to \wp(P)$, $m_i \mapsto \{p_{i1} \ldots p_{ik}\}$ be the function that maps a measured position to a *finite* set of possible route positions;
- *route*: $P \times P \to R$, $(p_a, p_b) \mapsto r_{ab}$ be the function that maps two road positions to the optimal route to connect them in the given order;
- *eval:* $R \to \mathbb{R}$, $r \mapsto e_r$ be the function that evaluates a route and produces a value e_r that is lower for 'better' routes;
- $| : R \times R \to R$, $(r_a, r_b) \mapsto r_{ab}$ be the concatenation of routes that is only defined for routes r_a, r_b, where the first position of r_b is identical to the last position of r_a.

Then, we can formulate the (offline) map matching function as follows:

$$\begin{aligned} &mapmatching: \mathcal{L}(M) \to \mathcal{L}(P), \\ &(m_1, \ldots, m_n) \mapsto \underset{p_{iv_i} \in map(m_i)}{\arg\min} \quad eval(route(p_{1v_1}, p_{2v_2}) \mid \\ &\qquad\qquad \ldots \\ &\qquad\qquad route(p_{n-1,v_{n-1}}, p_{nv_n})) \end{aligned} \tag{1}$$

Note, we require the output of *map* to be finite. This contradicts characteristics of real positioning systems as for each measurement an infinite number of possible positions may be the origin for this measurement. However, we can identify a *finite* set of representatives that are sufficient for our problem (see below).

Even though, we can easily denote the required function, its computation is difficult, if we have to consider execution time. A straight-forward approach would iterate through all permutations v_i to map a measurement m_i to a position p_{iv_i}. This, of course is virtually impossible for even small input sequences as the number of variations gets very large. Moreover, for every evaluation we have to call a time consuming *route* multiple times that executes optimal path planning on the road network.

2.2 Modelling Positioning System and Driver

The formalism above allows us to tailor the following approach to different scenarios that may differ in the used positioning system and characteristics of routes to detect.

Modelling the positioning system: Positioning systems are different according precision and distribution of measurement errors. A general model can be time-dependent; e.g. GPS errors depend on the satellite constellation. Usually a Gaussian distribution of measurements is assumed. For our approach the actual positioning model can be simplified as follows:

- The measurements to a position are distributed with a higher density closer to the actual position.
- There is a maximum error distance d_{max}.

The first point is obvious and is fulfilled by any Gaussian distributed positioning system. The second point is crucial: even precise positioning systems cannot state a maximum error – it is not intended to give a bound for any Gaussian distribution. However, as we have to keep the amount of possible mapped positions for a certain measurement finite, we have to state such a border. Even mathematically critical, this rule is not a problem in reality.

In our formalism, the positioning system model is reflected by two functions:

- *map*: only mapped positions up to the maximum distance d_{max} are computed;
- *eval*: the evaluation of routes takes into account the distances of measured and mapped positions.

Modelling the driver or driving goal: Drivers may have different opinions what a 'good' or 'typical' route is. Depending on the type of trip, the same driver can act different. E.g. the daily route to the office has a certain degree of optimality. In contrast the trip to watch some touristic sights or a travel to an unknown address may result in lower optimality measures and some sub-optimal bypasses. We can model this with the help of two functions:

- *route*: this function defines what generally is considered as 'good' or 'optimal'. One driver could consider the consumption of fuel as important, the other the driving time.
- *eval*: this function asserts a value that reflects the probability (in a wider meaning) that this route was actually driven. This function can reward certain attributes of the route (e.g. local optimality) to reflect the expected driving behaviour.

With these means we can model the system of roads and driver that affects the assumed route for a set of measurements.

2.3 Requirements, Assumptions

We want to declare some assumptions that we require for our approach later.

Assumption 1: We ignore the time of measurement. When collecting measurements, we could store the time stamp of the measured positions and we could consider these time stamps to evaluate of a route. The absolute values of time stamps are not of interest, but time differences of two subsequent measurements lead to the average speed in the driven section. We then could discuss about 'probable' speeds (e.g. in our *eval* function), however this is difficult. A speed can be arbitrarily low (e.g. red light or traffic jam). Thus, we can only identify unrealistic high speeds. This however would require to more carefully map a measurement to a position, in particular: an output of $map(m_i)$ depends on $map(m_{i-1})$ and $map(m_{i+1})$. This leads to an entirely different class of algorithms. As a starting point, in this paper we thus only discuss approaches that purely rely on the measured positions without a time stamp.

Assumption 2: We consider a finite number of variations per measurement. Fig. 2 provides a motivation why the output of *map* is finite. If we look at Fig. 2a, a measurement can be mapped to an infinite number of positions that all reside on the same link. For better clearness, we only show the projection to one side of the road – our final implementation has to consider both driving directions.

In this example, the infinite set does not carry additional information: a route starting from left can be continued to the right with any of these positions, thus all positions can be represented by a single position, e.g., the closest to the measurement. More formally: all positions that reside on a certain link form an *equivalence relation*, and any representative is a useful mapping. In Fig. 2b we see that more than one equivalence class may result from a mapping, if possible positions reside on different links. However, we have to keep in mind:

- The considerations above do not apply for the very first and very last mapping of our measurements.
- If sets of possible mappings of subsequent measurements overlap, we cannot choose *arbitrary* representatives. If we chose positions inside the circle in Fig. 2c, resulting routes may include two U-turns, thus get significantly different.

We called the latter problem the *back-driving problem* and discuss it in more detail later. In simple words: if two measurements are too close, poorly selected representatives may result in routes of (partly) wrong direction.

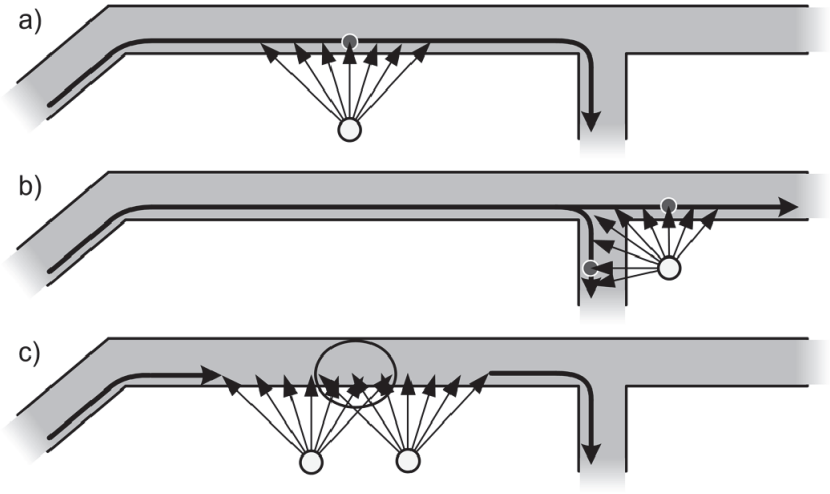

Fig. 2. Motivation to map measurements to finite sets of positions

Assumption 3: We use multi-routing as a building block. Even though our formalism only requires a *route* function for a single route, we strongly believe an efficient implementation requires a function

multiroute: $\wp(P) \times \wp(P) \rightarrow \wp(R)$

that computes *all* optimal routes between sets of start and target positions. Even tough such a function could call *route* for every permutation, we assume a more efficient approach as presented in [8].

Assumption 4: We expect a locality of the eval function. Until now, *eval* could have any characteristics. To benefit from some mechanisms to reduce the overall computation time, we have to demand a *locality* of route evaluation. This means: a local change of a route only affects input of the *eval* function that is *nearby* these changes. More formally: if we have two routes $r_a \mid r_x$ and $r_b \mid r_x$ with the same trailing r_x then

$$eval(r_a \mid r_x) < eval(r_b \mid r_x) \Rightarrow eval(r_a \mid r_x \mid r_y) < eval(r_b \mid r_x \mid r_y) \quad (2)$$

The corresponding rule for same leadings:

$$eval(r_w \mid r_x \mid r_a) < eval(r_w \mid r_x \mid r_b) \Rightarrow eval(r_x \mid r_a) < eval(r_x \mid r_b) \quad (3)$$

We require these rules for 'sufficiently long' r_x, whereas the minimum length depends on the respective *eval* function.

Assumption 5: We expect pre-segmented routes. We expect a mechanism that segments all measurements according to trips. I.e. a set of measurements passed to our algorithm represent, e.g. trip *from home to office*, another set represents *from office to shop* etc. This is required as stopping a trip for e.g., parking invalidates all assumptions about routes, e.g. regarding optimality. In particular, any new route goes to any direction regardless from prior driving. The basis of a segmentation mechanism is to detect halts of steady positions; however, such a mechanism in not part of this paper.

Strongly related to this issue are U-turns. We define U-turns as a 180° turnaround on the *same* road. If we expect finite mappings per measurements (assumption 2), we have a certain exception for start and end of routes. As U-turns can be viewed as end of route and start of a new route, we would thus produce this exception at every occurrence. Even though we could access minus points for U-turns in the *eval* function, we decided to directly filter out variations with U-turns to reduce the overall number of variations. To deal with U-turns in real routes, we have to rely on a segmentation mechanism to cut the sequence of measurements. Note that more complex manoeuvres to turn the direction (e.g. 'three left turns around the block') are not crucial.

2.4 Problems, Failures

The offline map matching problem turned out to be surprisingly hard to solve. The actual research was executed over two years whereas several ideas were realized that turned out not to be successful. We identified two major problems.

Problem 1: Combinatorial explosion of variations. If we mapped a single measurement to multiple possible positions and connect them, the resulting set of variations get huge very quickly. An established approach to deal with such a problem is *Viterbi* [10]. It tries to detect hidden states (real positions) from observed events (our measurements). It implicitly deals with a huge number of variations as it only considers the most probable last state for a new hidden state, thus the number of variations does not increase. However, this is a significant problem, as a decision will not be revised later, when we know about the subsequent states. Thus, the results were usually poor.

We also tried *Simulated Annealing* [1] – an approach to find a global optimum of a function (here *eval*). This also was a failure. It expects for small changes in the input also small changes in the output. However, changing a mapping of a measurement can produce arbitrary big changes in the evaluation.

Another experiment was a *Divide-and-Conquer* approach: we first check optimal routes from all start to target measurement mappings. If one of these routes approximate all intermediate measurements, we have a solution. If not, we look for an intermediate measurement to split and recursively try to find an appropriate route. This approach again was a failure, whenever the route was not driven according to optimality measures assumed by our *route* function. As a result, the recursive segmentation goes very deep and produces a combinatorial explosion we tried to avoid. Moreover, the approach to independently search for appropriate routes failed as the route parts are not actually independent – at the intermediate points, the values depend on both parts of connected routes.

As a final direction to solve this problem we tried to directly check certain properties of possible routes. If not fulfilled, we exclude them without asking the *eval* function. This significantly reduced the set of possible route variations. We conducted experiments with the property of *local target orientation* [6]. This property means: for a sufficiently small part of every trip we can find an optimal route, even if the overall route is not optimal. This approach also was a failure due to two reasons: first, the computation costs to check the local target orientation were considerable high. Sec-

ond and most important: the degree of locality depends on the current driving situation. If we used a fix value, very often the set of possible routes gets empty, as not any route fulfils the given property.

Problem 2: The back-driving problem. This occurs as measurements can be very close and suggest a 'driving in the opposite direction' (Fig. 3). In our example the route goes from left to right (Fig. 3a). The route positions Pos_{10} and Pos_{11} were mapped to measurements M_{10} and M_{11}. Even though they are in range of the maximum error d_{max}, they change their ordering according to the route direction. If we now map the measurement to the nearest positions on the road network (p_{10j}, p_{11k}), a possible route extension has to drive a '*route around the block*' (Fig. 3b; the numbers ①, ② and ③ indicate the driving sequence of the assumed route). Obviously the assumed route significantly differs from the real route, even though the mapped positions are very close to the real positions.

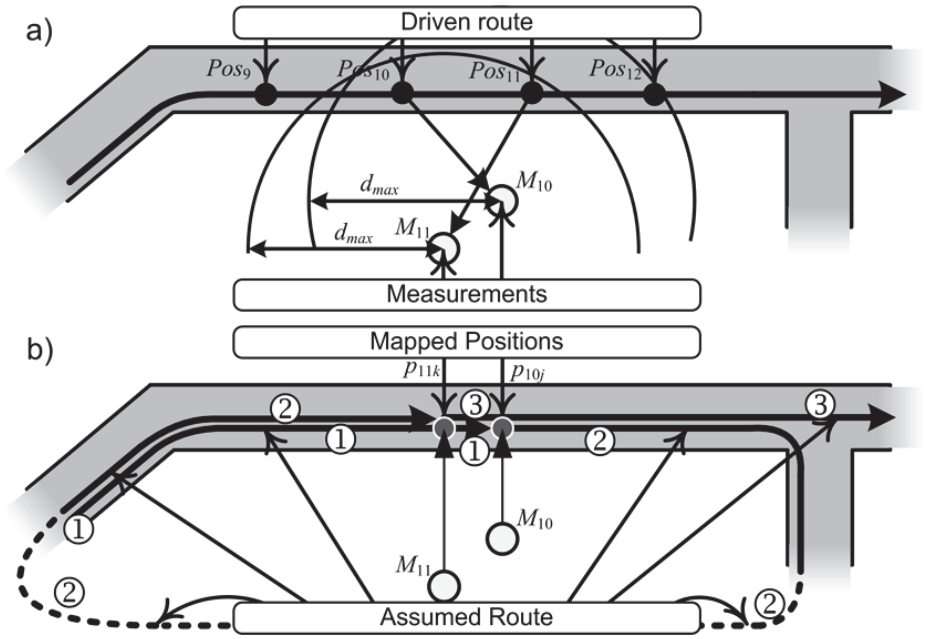

Fig. 3. The back-driving problem

2.5 An Efficient Solution

Considering the problems above, we finally developed a solution for the offline map matching problem. It iterates through the measurements from trip start to termination. For each iteration we extend all conceivable route variations by new variations that result from possible mappings of a measurement.

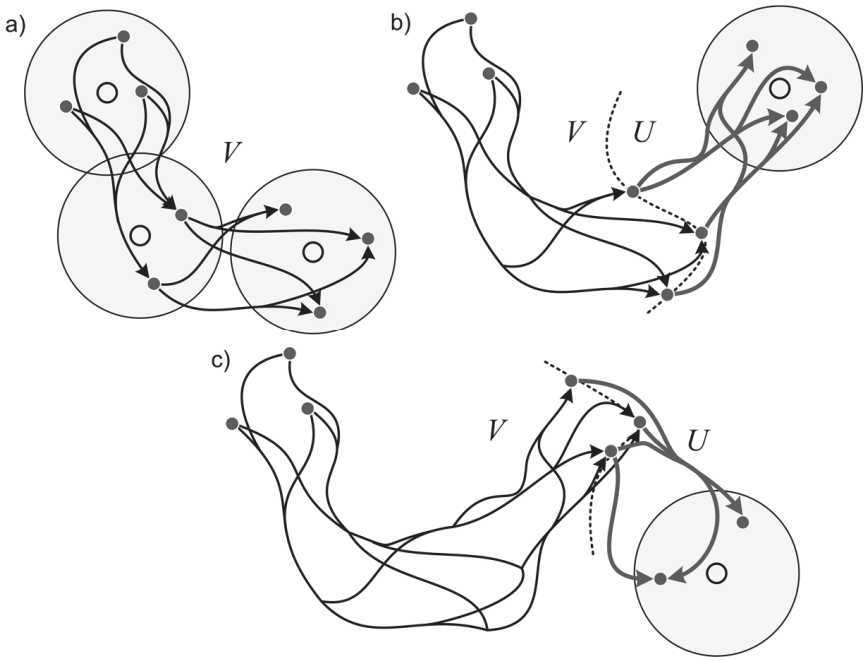

Fig. 4. Basic idea of our map matching approach

Fig. 4a shows the situation after three steps. We manage a set *V* that holds all conceivable route variations through the mapped positions. If we take the next measurement (Fig. 4b and c), we create all optimal routes (*U*) from current route endpoints to mapped positions of the new measurement. For the next iteration, we now have to create a new set *V*, built from all extensions of old *V* extended with all routes of *U*.

Due to the high number of route variations, we have to incorporate mechanisms to reduce computing costs to a manageable level.

1. Mechanisms to reduce the number of mappings

According to assumption 2, we map a measurement to the nearest position of all links in range. The easiest way to reduce the number of route variations that result from different mappings is to reduce the mappings themselves. Our mechanism (1a) is not to map all measurements but only a reasonable subset. The motivation: the number of measurements is pre-defined by the position measurement system, e.g. by the smart phone app that logs the trip positions. This could, e.g., measure one position per second. This set up is not under control by the map matching system, but *given*. Our algorithm should thus have the chance to take into account a lower number of measurements to produce route variations. As we thus can keep a certain minimum distance between measurements, this is an effective mechanism against the back-driving problem. In our approach, we do not ignore the other measurements but always evaluate *all* measurements in the *eval* function.

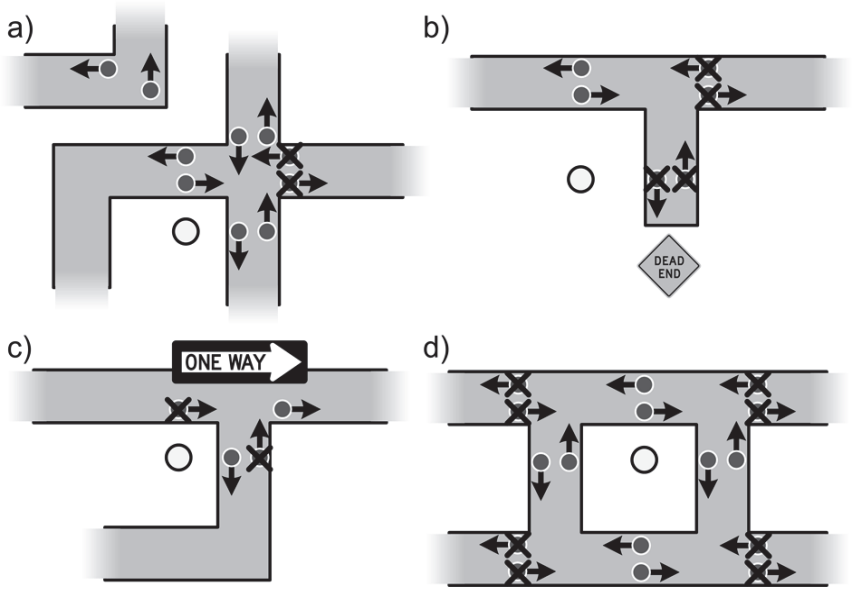

Fig. 5. Idea of the projection compression rule

Our second mechanism (1b) is called the *projection compression rule*. Fig. 5a shows a scenario where a measurement produces 10 mapped positions. There exist 14 routes to drive through the range of this measurement (2 through the upper left corner, 12=4·3 through the crossing). However, we could remove two of the 10 mappings (marked with 'X') without loosing any of the 14 route variations.

Also dead ends and one way roads may reduce the mappings without loosing variations (Fig. 5b and c). Fig. 5d shows a complex situation with 24 variations to drive through (without counting loops). Here, we can remove 8 of the 16 mappings without loosing variations.

To identify removable mappings we proceed as follows:

- We remove all mappings in dead ends.
- For each remaining mapping, we assign the distance of measurement and mapping. If for a mapping p_i all links to drive *to* or drive *from* p_i contain a nearer mapping p_j, we can remove mapping p_i.

Note that the projection compression rule does not avoid all double variations. E.g. in Fig. 5a, the routes through the 4-way-crossing from top to bottom are still presented by two measurements.

As a last mechanism (1c) to reduce the number of mappings: we can remove all mappings that are not endpoint of any currently considered route variation.

2. Mechanisms to reduce the number of route variations

This first mechanism (2a) is to directly avoid U-turns and not to wait for a bad evaluation by the *eval* function later (see assumption 5). Our approach iterates through

measurements and tries to extend existing route variations by a new part of the route. Thus, the only chance to produce a U-turn is when an existing route is extended. As a simple mechanism, we only accept extensions without U-turns.

A second mechanism (2b) 'compresses' the set of existing variations. The idea: if an existing route variation is not able to be part of the final route with the best evaluation, we can remove it. We apply assumption 4, formula (2) for this. In more detail: if we identify two route variations with the same trailing r_x, the one with the worse evaluation cannot be the final 'winner' ($r_b \mid r_x \mid r_y$ according to formula (2)). We thus only have to check all variations for same trailings, evaluate them and keep only the best.

3. Mechanism to speed up the evaluation

For each iteration and variation, *eval* has to be executed again. This is because old results of eval are not useful anymore as routes are extended (by a route of U). Even though the number of variations stays constant (due to mechanisms 2a and b), the length of each route gets longer as more measurements have to be taken into account. As a result, for a naïve realization, the execution time of *eval* would significantly increase for each iteration.

As our final mechanism we apply assumption 4 and formula (3). It turned out that after a certain time, all route variations in V have a same leading route ($r_w \mid r_x$ in formula (3)). Whenever this happens, we can ignore r_w for new evaluations as it does not affect the ordering.

We now can put our considerations together to our map matching algorithm below (references to the mechanisms are in brackets). It contains two main loops: one to compute all mappings and one to subsequently extend route variations.

Note that in contrast to the formal definition (1), this algorithm returns a route, not a list of mapped positions. However positions and a route that connects these positions are equivalent information.

mapmatching: input: measurements m_i output: most probable route v	
select a subset $\mu=\{\mu_1,\ldots,\mu_m\}\subseteq\{1,\ldots,n\}$, with $\{1, n\}\subseteq\mu$	(1a)
for each $\mu_i\in\mu$ { compute $P_i = map(m_{\mu_i})$	
compress P_i according *projection compression rule*	(1b)

}
$V=multiroute(P_1, P_2)$
for i=3 to m {	

$P'_{i-1}=\{p\in P_{i-1} \mid p$ *is endpoint of a* $v\in V\}$	(1c)
$U=multiroute(P'_{i-1}, P_i)$	
$V=\bigcup_{v\in V,u\in U} v \mid u$ if u extends v without producing a U-turn	(2a)
compress V	(2b)
} return $\arg\min_{v\in V} eval(v)$	(3)

2.6 Again the Back-driving Problem

According to Fig. 2c and Fig. 3 the back-driving problem can occur, if two subsequent measurements have overlapping circles (with radii d_{max}) and at least one link resides in the overlapping area. With the selection of the subset μ we are able to control this: we can either select measurements that have at least a distance of $2d_{max}$ (no overlapping areas at all), or we can geometrically check for links in the overlapping areas. The latter approach requests some complex geometric computations, but tolerates closer measurements.

However, for positioning systems with large positioning errors, this selection causes a low measurement density. The problem: if two selected subsequent measurements are too far, the route in-between is not represented adequately.

To solve this, we introduced another approach that allows to select arbitrary close measurements with the costs of additional computation. We modify the *map* function: a position is not mapped to the nearest position on a link but on a fix *relative* position, e.g. always on the centre of the link's running length. As a result: even close measurements do not produce back-driving, as subsequent mapped positions stay constant on a link. This is possible due to assumptions 1 and 2. Two crucial points:

- The very first and very last measurement should additionally be mapped to nearest route points.
- We have to deactivate the projection compression rule, otherwise we still may produce the back-driving problem. This may occur, if nearby a crossing, the rule removes projections in the wrong driving order.

If we consider more measurements, we have a longer runtime due to more multiroute calls. In addition, the number of mappings per measurement is increased on average, thus this modification may dramatically increase the overall runtime.

A last consideration: selecting two measurements that are closer than $2d_{max}$ would allow us to detect small route changes between measurements that were otherwise undiscovered. On the other hand, these route changes may be below the measurement precision, thus remain undiscovered even though respective measurements are selected. As a best practice, we thus suggest at least $2d_{max}$ distance between selected measurements.

2.7 A Best-practice eval Function

We currently only request assumption 4 to be fulfilled by the *eval* function. In this section we describe an *eval* function that turned out to be useful for a driving scenario.

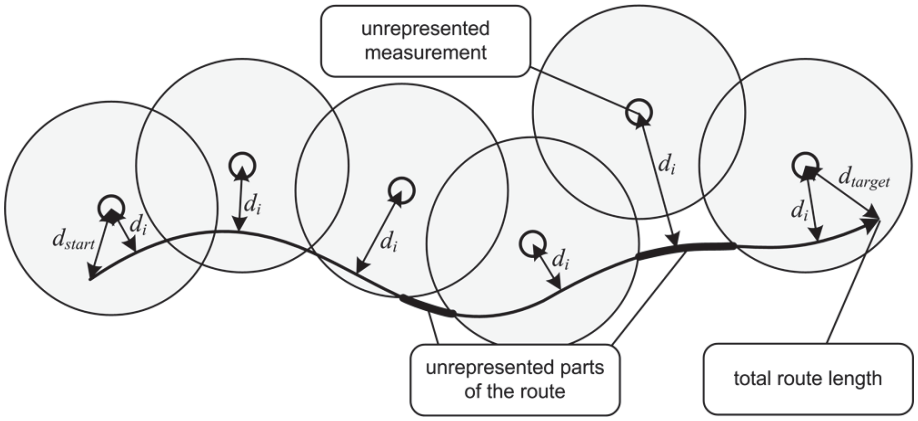

Fig. 6. Properties considered by the evaluation

Our function considers the following properties (Fig. 6):

- The total route length L in meters.
- The sum of squares of distances d_i between measurements and nearest route positions scaled by $1/d_{max}^2$.
- The square of first route point to first measurement d_s and last route point to last measurement d_t scaled by $1/d_{max}^2$.
- The route meters not represented by any measurement L_{unrep}, and the number of measurements not covering a route position cnt_{unrep}.

We build the weighted sum of these values (4).

$$eval(r)=\begin{pmatrix} L \\ \sum d_i^2 / d_{\max}^2 \\ (d_s^2 + d_t^2) / d_{\max}^2 \\ L_{unrep} \\ cnt_{unrep} \end{pmatrix} \cdot \begin{pmatrix} w_{len} \\ w_{dist} \\ w_{st} \\ w_{lunrep} \\ w_{cntunrep} \end{pmatrix} \tag{4}$$

We achieved good results with the settings: w_{len}=1/3, w_{dist}=50, w_{st}=100, w_{lunrep}=1, $w_{cntunrep}$=10.

3 Evaluation

We fully implemented the presented offline map matching approach inside our routing platform *donavio* [6] that is part of the *HomeRun* environment [5], [7]. We conducted a number of experiments to show the efficiency of our computations and the quality of the output. For all routes we set d_{max}=200m.

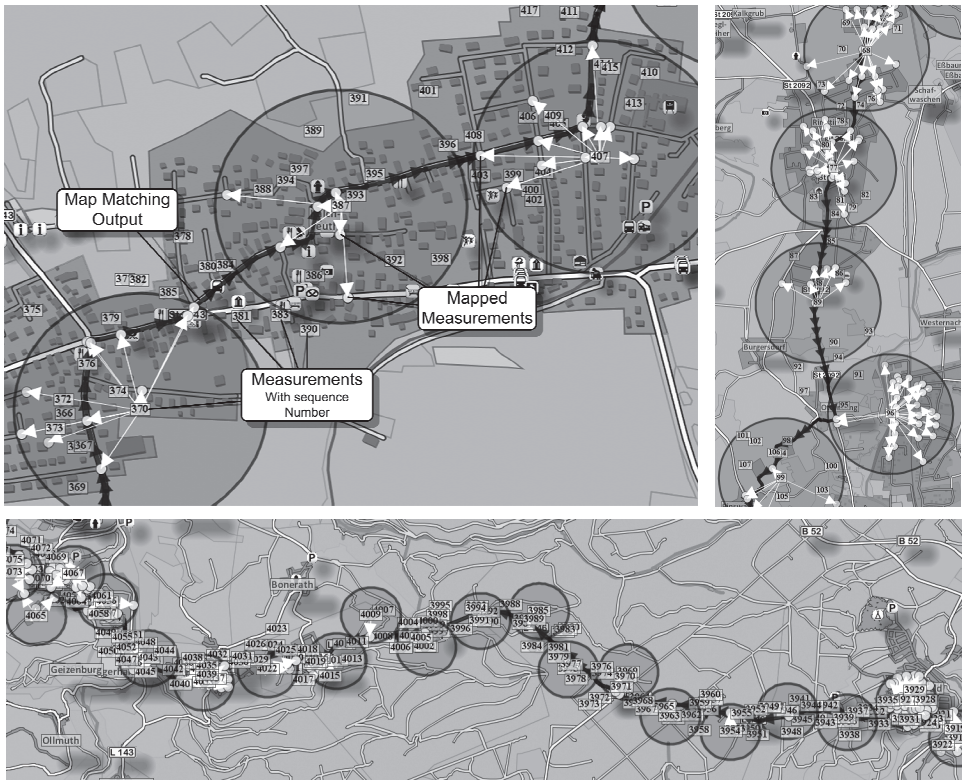

Fig. 7. Example measurements and result routes

Table 1. Summarized Results of Conducted Experiments

Type	Driven km	Driving Time (hh:mm)	Number of Measurements	Mappings per Measurement	Totally Checked Route Variations	Avg. $\|V\|$ per Iteration	Exec. Time/ Mapped Mes. (ms)	Time for *multiroute* (%)	Time for *eval* (%)	Org. Positions on Result Route (%)	Result Route km/ Driven km (%)
urban	2.3	0:07	69	13.7	510	23	108.7	8.7	40.1	100.0	105.9
urban	4.8	0:07	151	17.5	1891	90	315.2	4.8	81.8	95.4	97.7
urban	5.4	0:08	129	16.3	2306	105	445.1	3.4	86.5	96.9	99.8
urban	16.1	0:32	458	6.9	2229	294	78.7	2.2	54.1	97.2	100.5
urban	18.5	0:22	186	12.8	4335	290	177.4	5.5	73.7	98.4	99.7
suburban	11.2	0:14	240	10.1	2037	139	113.9	3.5	66.0	95.4	102.3
suburban	16.1	0:21	301	10.5	3108	238	272.9	2.3	84.6	99.7	99.1
suburban	19.2	0:22	457	6.9	1776	188	94.6	2.9	61.3	97.4	99.3
interurban	412.3	3:39	4124	9.5	38377	3492	454.7	2.0	76.8	98.0	100.0
interurban	977.8	8:42	9650	9.8	112458	8425	550.0	1.2	59.5	99.0	98.7

We selected measurements with a distance of $2d_{max}$. For execution time measurements we used a PC with i7-4790 CPU, 3.6 GHz. Table 1 summarizes the results; some routes are illustrated in Fig. 7.

To assess the similarity of the assumed route by our algorithm and driven route, we measured the amount of original positions that reside on the result route. As the result route may also contain additional tracks, we also compared the route lengths. In summary, in all our experiments, the result routes were always very similar to the original driven route.

We also measured the execution time. Even though it is not primarily required for an offline approach to perform execution in time, we got approx. 260ms per mapped measurement. Thus, this approach could in principle also process measurements in realtime.

We tried to identify the component that needed most of the execution time. We thus analysed the CPU usage for the *multiroute* and *eval* functions. The multiroute function only requires 4% on average. This supports assumption 3 to heavily rely on the multi-routing function. We also conducted experiments where we replaced our multi-routing approach by permutations with single routing (not shown in the table). The result: our multi-routing is factor 1022 times faster (max. 2355 times) compared to a single-routing approach.

The most time-consuming function in our implementation is the *eval* function. This is due to complex geometric computations, in particular to identify the unrepresented parts of the route. For time-critical execution scenarios we could consider to simplify the *eval* function.

As a last set of experiments, we also checked the alternative mapping for measurements closer than $2d_{max}$ (section 2.6). We thus also select measurements with a distance of d_{max}. However, disabling the projection compression rule increases the number of mappings per measurement by factor 2.5 on average; as expected this significantly affects the overall computation time: it increased by factor 4.2.

4 Conclusions

In this paper we presented an approach for offline map matching. The main idea was to map a measurement to all potential road positions and to check, how a sequence of mapped positions can be connected by a driven route. It heavily relies on a multi-routing mechanism to immediate check connections between set of start and target positions. To address the problem of combinatorial explosion, we suggest a number of mechanisms. An evaluation shows the effectiveness of these mechanisms.

References

1. Bertsimas, D., Tsitsiklis, J., 1993: Simulated Annealing, Statistical Science, Vol. 8, No 1 (1993), 10-15
2. Haunert, J.-H., Budig, B., 2012: An algorithm for map matching given incomplete road data, Proc. of the 20th Intern. Conf. on Advances in Geographic Information Systems, 2012, 510-513
3. Marchal, F., Hackney, J., Axhausen, K. W, 2005: Efficient map-matching of large GPS data sets – Tests on a speed monitoring experiment in Zurich, Transportation Research Record: Journal of the Transportation Research Board, Vol. 1935, 2005, 93-100
4. Pereira F. C, Costa, H., Pereira, N. M., 2009: An off-line map-matching algorithm for incomplete map databases, European Transport Research Review, Oct. 2009, 107-124
5. Roth, J., 2013: Combining Symbolic and Spatial Exploratory Search – the Homerun Explorer, Innovative Internet Computing Systems (I2CS), Hagen, June 19-21 2013, Fortschritt-Berichte VDI, Reihe 10, Nr. 826, 94-108
6. Roth, J., 2014: Predicting Route Targets Based on Optimality Considerations, International Conference on Innovations for Community Services (I4CS), Reims (France) June 4-6, 2014, IEEE xplore, 61-68
7. Roth, J., 2015, Generating Meaningful Location Descriptions, International Conference on Innovations for Community Services (I4CS), July 8-10, 2015, Nuremberg (Germany), IEEE xplore, 30-37
8. Roth, J., 2016: Efficient Many-to-Many Path Planning and the Traveling Salesman Problem on Road Networks, KES Journal: Innovation in Knowledge-Based and Intelligent Engineering Systems, to appear
9. Schuessler, N., Axhausen, K. W, 2009: Map-matching of GPS traces on high-resolution navigation networks using the Multiple Hypothesis Technique (MHT), Working Paper 568, Institute for Transport Planning and System (IVT), ETH Zurich, 2009
10. Viterbi, A., 1967: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In: IEEE Transactions on Information Theory. 13, Nr. 2, 1967, 260–269
11. White, C. E., Bernstein, D., Kornhauser, A, 2000: Some Map Matching Algorithms for Personal Navigation Assistants, Transportation Research Part C: Emerging Technologies, Vol. 8, No 1–6, Febr.-Dec. 2000, 91–108
12. Yanagisawa, H., 2010: An Offline Map Matching via Integer Programming, Proc. of the 20th, Intern. Conf. on Pattern Recognition (ICPR), IEEE, 2010, 4206-4209
13. Zhou, J, Golledge, R., 2006: A Three-step General Map Matching Method in the GIS Environment: Travel/Transportation Study Perspective, Intern. Journal of Geographical Information Systems, Vol. 8, No 3, 2006, 243-260