Visual Interactive Map Matching

Robert Krüger, Georgi Simeonov, Fabian Beck, and Thomas Ertl

Abstract—Map matching is the process of assigning observed geographic positions of vehicles and their trajectories to the actual road links in a road network. In this paper, we present Visual Interactive Map Matching, a visual analytics approach to fine-tune the data preprocessing and matching process. It is based on ST-matching, a state-of-the-art and easy-to-understand map matching algorithm. Parameters of the preprocessing step and algorithm can be optimized with immediate visual feedback. Visualizations show current matching issues and performance metrics on a map and in diagrams. Manual and computer-supported editing of the road network model leads to a refined alignment of trajectories and roads. We demonstrate our approach with large-scale taxi trajectory data. We show that optimizing the matching on a subsample results in considerably improved matching quality, also when later scaled to the full dataset. An optimized matching ensures data faithfulness and prevents misinterpretation when the matched data might be investigated in follow-up analysis.

Index Terms—Map matching, data cleaning, data transformation and representation, geographic visualization.

1 Introduction

In geo analytics, large amounts of recorded trajectories need to be processed. These spatio-temporal paths might reflect the movement of vehicles or persons. Trajectories are usually recorded using GPS devices, which results in a high number of data points. The recording underlies uncertainties because of imprecise GPS localization and might contain gaps caused by temporarily losing the GPS signal. For reducing data complexity and making movement comparable, trajectories are mapped to a road network. *Map matching* algorithms assign each recorded trajectory to an individual geographic path in a transportation network. These heuristics try to compensate measurement uncertainties, but they only work if the algorithm's parameters are chosen wisely and fit the properties of the recorded data. Also, there might be matching problems because the road network model contains ambiguities or is outdated. When using the matching algorithm only in its default configuration, it might only provide results of poor quality. This could lead to misinterpreting the data in later analysis, for instance, investigating traffic flow for road planning, fleet management, or understanding mobility patterns.

This paper presents an approach for optimizing the map matching process and cleaning the input data. We focus on improving this preprocessing step only, whereas any further analysis of the trajectories is important and intended, but out of scope for this paper. We develop a visual analytics system that integrates a state-of-the-art map matching algorithm with interactive visualizations showing problems in the current matching results (Figure 1, B) as well as quality and performance measures (Figure 1, C). By refining the matching parameters (Figure 1, A), a data analyst can globally optimize the results and find an appropriate trade-off between quality and performance of the matching process. Also, the systems supports semi-automatic local corrections of problems in the road network, for instance, missing road links or imprecise road routing.

- *R. Krüger, G. Simeonov, and T. Ertl are with the Institute for Visualization and Interactive Systems, University of Stuttgart, Germany. E-mail: [firstname.lastname]@vis.uni-stuttgart.de*
- *F. Beck is with paluno, University of Duisburg-Essen, Germany. E-mail: fabian.beck@paluno.uni-due.de*

Manuscript received April 19, 2005; revised August 26, 2015.

The system allows analysts to reduce error rates and improve the matching result in an interactive process, guided by visual feedback and suggestions. In particular, this paper comprises the following key contributions:

- We discuss the challenges of an interactive computer-supported configuration of a map matching procedure and derive requirements for a visual analytics approach (Section 3).
- (a) We design an interactive visual analysis process to control and optimize a map matching process. (b) We develop an integrated system that supports this process with synchronized views containing controls for configuring the algorithm, map visualizations, and matching statistics (Section 4).
- We evaluate the system by demonstrating the optimization of the map matching in a case study (Section 5).

Finally, we conclude in Section 6 that our system not only provides a tool to improve the quality of a map matching process, but also makes the process more trustful for the analyst due to an increased transparency and understandability.

2 Related Work and Background

Our approach builds on foundations in visual trajectory analysis and map matching—we provide an overview on these two areas and identify most related approaches, which visualize map matching results. Since we target an improved preprocessing, we also review visual and interactive methods for data preprocessing or cleansing and highlight the lack of such approaches for the field of movement data processing.

2.1 Visual Analysis of Trajectory Data

Trajectory analysis is an established research field. Different approaches have been proposed to query, process, and visually explore the data [1], [2]. Chen et al. [3] survey recent publications and advances in the field. A common challenge of many approaches is to reduce the complexity that trajectories bring along and to find a clear representation that reduces clutter and overplotting to make visible the main movement patterns. Andrienko and Andrienko [4]

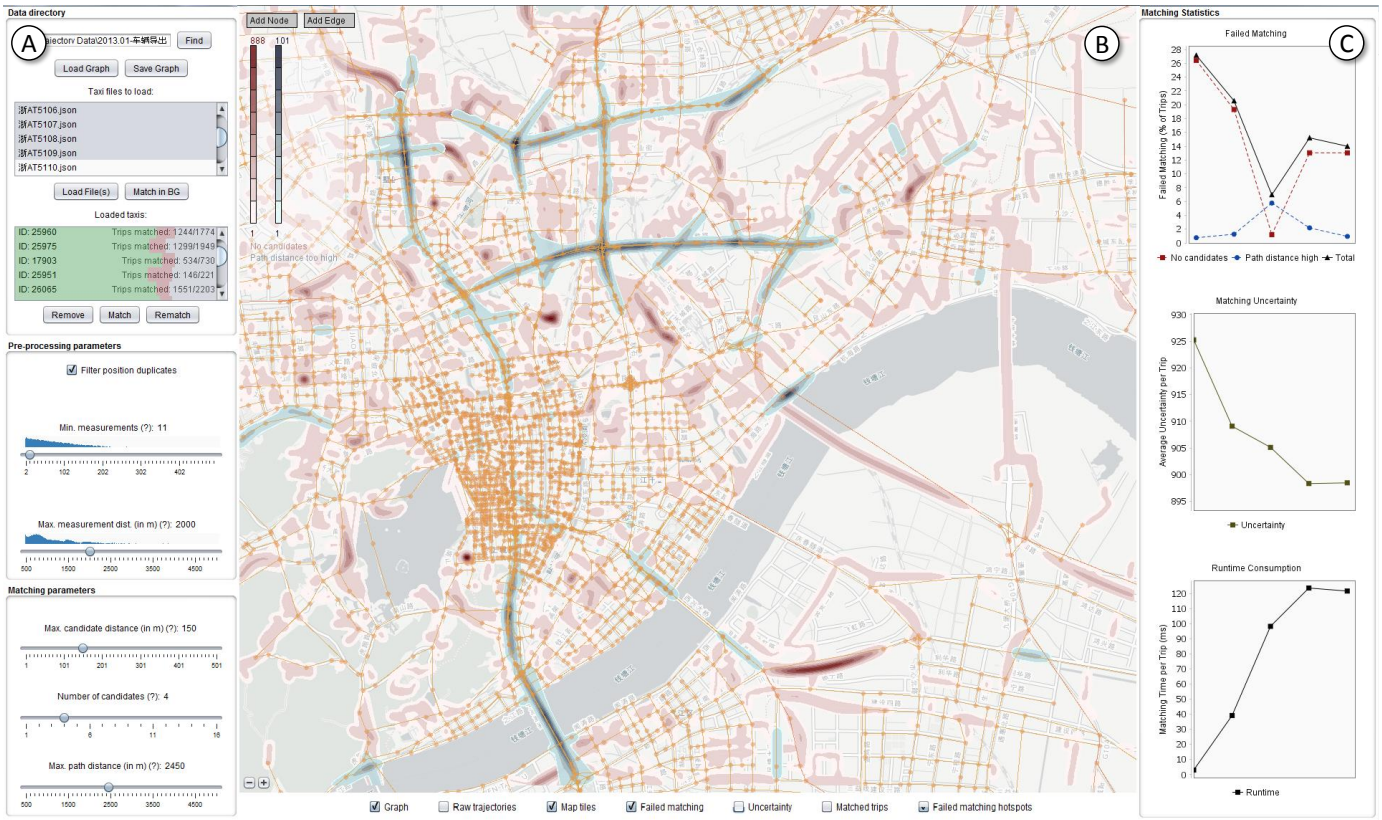

Fig. 1. The user interface for Visual Interactive Map Matching comprises multiple linked views: (A) A panel to load, select, and filter the data, as well as to configure map matching parameters. (B) A map to visualize the geographic scene (either generalized or with satellite imagery), the road network, and map matching results such as matched trips, errors, and uncertainty score. With visual information given, the user can explore and edit the road network in this view. (C) A statistics view showing the increase or decrease of errors and runtime during iterative refinements.

contribute a general framework for spatio-temporal trajectory aggregation and visual exploration. Spatial techniques include graph-based aggregation (e.g., to show main traffic flows [5]), density-based aggregation (e.g., visualized as heatmaps [6]), and vector field computation (which can be represented using flow visualization techniques [7], [8]). Also, interactive filtering approaches allow for visually exploring trajectory data and overcome clutter by drilling down to a subset of interest [9], [10].

2.2 Map Matching

Map matching approaches *"integrate positioning data with spatial road network data (roadway center lines) to identify the correct link on which a vehicle is traveling and to determine the location of a vehicle on a link"* [11]. They can be categorized into global and incremental approaches [12]. Global approaches take the object's whole route into consideration, while incremental approaches iteratively map parts of the trajectory. Global approaches often achieve more accurate results, whereas incremental approaches have lower computational costs. Quddus et al. [11] present a different taxonomy by categorizing map matching algorithms into geometry, topological, and probabilistic approaches. Geometry-based approaches usually only search the network for nearby edges with a minimal distance to the original trajectory segment, while topological approaches also consider connections between roads. Lastly, probabilistic approaches also make use of additional information such as travel speed, directions, and route types and lead to better results in case this additional information is available.

Many map matching approaches are made publicly available as web services. Common API examples comprise Google Roads[1], Mapbox Map Matching API[2], and the graphhopper routing[3]. Users can query an endpoint that maps raw trajectory data to a road network. The response delivers a node- or edge-based path description of the movement. Some applications, e.g., by graphhopper[4], also provide visual interfaces that allow the user to upload raw trajectory data and inspect the matched results. However, to the best of our knowledge, there exists no visual map matching system that allows the analyst to visually validate map matching results, to interactively configure matching parameters, or to edit the network representation.

To realize our Visual Interactive Map Matching approach, we employ ST-matching [13], a global and probability-based matching algorithm. In addition, this algorithm has the advantage of having understandable and user-friendly parameter configuration options.

2.3 Visualizing Map Matching Results

Several visual analysis approaches leverage map matching as a preprocessing measure to cope with large and complex movement datasets and to reduce signal dithering as well as other inaccuracies. Some visual analysis techniques [14], [15], [16], [17] apply map matching, but their visual analysis approaches do not focus

1. developers.google.com/maps/documentation/roads
2. www.mapbox.com
3. www.graphhopper.com
4. https://github.com/graphhopper/map-matching

on preprocessing but on partitioning and clustering of the map matching results. Lu et al. [18] present an approach to analyze route choice behavior. Similarly, they first map the trajectories to an underlying road network. This results in faster processing, data filtering, and a clearer visualization. Instead of overplotting resulting from similar routes, trajectories are aggregated, colored accordingly, and visualized side by side.

2.4 Visual Analytics for Data Preprocessing

While we found no visual analytics technique that allows for an interactive and visual map matching, there exist visual analytics means for other data preprocessing tasks, often addressing related challenges. Kandel et al. [19] present a general table-based approach for data transformation and manipulation. More specific approaches, e.g., for time-oriented data, have been proposed by Bernard et al. [20] and Gschwandtner et al. [21]. In the domain of mobility data, Krueger et al. [22] present a prototype to align and annotate geographic data. Maps can be visually overlaid with additional imagery, e.g., proving POI information. Analysts can manually annotate maps by defining areas of interests. Subsequently, trajectory data is enriched based on this additional content. Wang et al. [23] propose an interactive method for trajectory data cleansing and quality checks. They extract features and train a supervised model with poor quality examples to detect similar cases. While the aforementioned approaches help to clean data, the registration of trajectories to an underlying transportation network (map matching) is not yet addressed.

3 Design Considerations

Designing the visual analytics system we present in this paper, our main goal was to facilitate data analysts to efficiently deal with map matching algorithms. While they might be experts in general data analysis and visualization, their knowledge of map matching typically is limited. Hence, without tool support, they likely choose a non-optimal configuration of the algorithm. Problems in the raw trajectories or in the road network might go unnoticed. Besides increasing final matching results, we believe that it is also important that the analyst develops a better understanding of what the algorithm is doing. This finally makes the results more trustful, both regarding objective and subjective measures. With these goals in mind, we discuss the challenges of a visually supported interactive map matching approach and derive specific requirements that our visual analytics system is intended to fulfill.

3.1 Challenges

Most algorithms for data processing have a number of parameters that users or developers need to configure. With an appropriate configuration, an algorithms efficiently computes results of high quality. But when misconfigured, the algorithms could slow down or outputs questionable results.

For global map matching algorithms (like the applied ST-matching), a considerable slow down could be caused by selecting a set of candidate points in each matching step that is too large. On the other hand, when only searching for candidates in a small radius, the right candidates might be missed, which results in errors of the matching. While some generalizable recommendations or useful default configurations might be possible to provide, an optimal configuration necessarily depends on the characteristics of the input data, for instance, the density of the road network or the quality of the GPS measurements. Hence, the parameter configuration of the map matching algorithm and the preprocessing of the input data is the general challenge that our approach addresses. In detail, we consider the following challenges:

- The quality of a matching results is determined by the correctness of the matched trajectories (i.e., *precision*) and also the number of the trajectories the algorithm matches (i.e., *recall*). A first specific challenge is to *find a trade-off between precision and recall* of matches when configuring the algorithm's parameters.
- The results might also be flawed because some of the raw trajectories are too imprecise to be assigned with sufficient confidence to a path through the network. It is important to *identify and remove unreliable measurements* to not affect the matching results.
- Similarly, the second input source for the matching algorithm, the road network model, might be imprecise, outdated, or contain errors. Hence, to *correct and adapt the road network model* could lead to an improved matching result. Other map data, satellite images, or the recorded trajectories give hints to where road links are missing or need to be refined.
- Finally, the algorithm's configuration might need to be applied to a large set of trajectories. When only focusing on the quality of the results, however, the algorithm might run too slow. Reducing the amount of computations per trajectory will speed up the process, but could affect quality as well. Hence, the analyst also needs to *find a trade-off between quality and performance*.

3.2 Requirements

In order to address the above challenges, we formulate a set of requirements that our system shall implement. They translate the abstract challenges into specific needs. The requirements are independent of the actual algorithms (e.g., the map matching algorithm) and visualizations (e.g., the specific representations for errors and matching statistics) later selected in our approach, and hence, could also serve as a basis for developing alternative solutions.

The basis for working with trajectories is a map visualization. This visualization needs to show the raw data, which would support the analyst in spotting unreliable GPS measures and other issues. Not least important is a visualization of the matching result, to check the success of the algorithm. But just displaying the successful matches, which is usually done for further analysis of the data, is not sufficient. Also, displaying the matching errors, failed matches, and uncertainties is relevant for optimizing the matching process.

> **RQ 1: Map Visualization**
> Show the results of the map matching algorithm together with the input data (raw trajectories and road network) on a map and visualize errors as well as uncertainties of the algorithm.

The optimization process to find a good parameter selection needs to be interactive, with quick visual feedback of the results. Hence, analysts adapt the configuration of the map matching and all preprocessing steps from the user interface and the visualization updates automatically. To fulfill this, the algorithm and interface

need to respond quickly. To track the progress of the optimization, an explicit representation of the evolution of matching statistics is required (e.g., measures of precision and recall as well and runtime performance). It should make transparent how the values developed across the past iterations of running the algorithm.

> **RQ 2: Matching Process**
> Support an iterative configuration of the matching algorithm in an interactive (hence, scalable) process, while providing a monitor for quality and performance measures.

Finally, the analysts need to adapt the road network model, for instance, to correct errors like missing road connections, refine the route of a street, or the position and connectivity of intersections. Since such edit operations could be costly in terms of interaction effort, algorithmic support is desirable. This support, however, should not be fully automatic to keep the analyst in full control.

> **RQ 3: Semi-Automatic Editing**
> Provide suggestions for an interactive improvement of the road network model to improve the matching results.

These requirements cover all main features of the system we present in this paper. They also reflect the challenges discussed above: The interplay of the map visualization (RQ 1) and the interactive matching process (RQ 2) addresses the trade-off between precision and recall, the removal of unreliable measures, and the trade-off between quality and performance. Further, we translate the editing of the road network model to RQ 3.

4 Approach

Based on the requirements, we designed and implemented a visual analytics process (see Figure 2). The process starts with trajectory data and road network selection (Section 4.1). It allows for data inspection and cleaning (4.2), before the map matching is applied (4.3) and results can be visually explored (4.4). The matching process is iteratively optimized (4.5) based on matching quantities and qualities gained through interactive visualizations. When dealing with large datasets, it is often advisable to start the map matching with a smaller data sample. Once matching results are satisfying, the algorithm can be applied (generalized) to the remainder of the data (4.6).

4.1 Data Selection

The map matching process starts by loading a movement dataset into the system. Movement data usually contain a set of moving objects O, where each object $o_j \in O, 1 \leq j \leq w, w = \|O\|$ holds a set of trajectories $TR = \{tr_1, \ldots, tr_m\}$. A trajectory is a path through space as a function of time. Although movement is continuous, movement recording technology usually samples it into a set of spatio-temporal measurements. With the recorded measurements of length n, a trajectory tr is a sequence, where p_i is the spatial location of a measurement at time t_i with $i = 1, \ldots, n$ (see Equation 1).

$$tr = \big((p_1,t_1),(p_2,t_2),\ldots,(p_n,t_n)\big) \text{ with } t_1 < t_2 < \ldots < t_n \quad (1)$$

In our system, every moving object appears in a list that the user can choose from. Also, the user needs to load a road network that the trajectories will be mapped to. The prototype implementation can handle graphs in the Open Street Maps graph format (*.osm* files). These road networks can be configured and extracted from open street map instances and contain, e.g., only major highways or detailed information including even bicycle paths. When the data is loaded, the user inspects the road network and the raw trajectories in the map view (see Figure 3) and might activate and deactivate different map overlays (see Figure 1, bottom). In the background of the main view, map tiles provide geographic context, either generalized to road segments, buildings, etc. or with satellite imagery data. This context will later serve as a means to control the correctness of the raw trajectory data and the network structure. With this feature, we fulfill a first part of RQ 1.

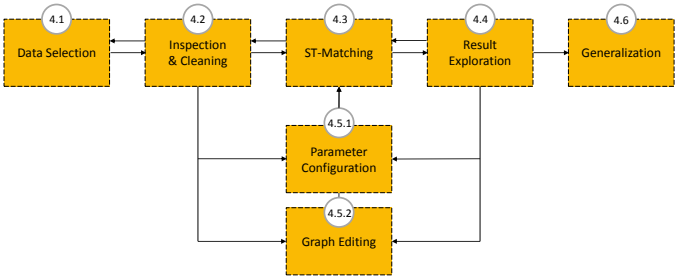

Fig. 2. The interactive visual map matching pipeline (labeled by sections). Raw trajectory data is first loaded, inspected, and cleaned. Thereafter, the data is registered to the road network using ST-matching. The user can explore results and iteratively refine map matching parameters and graph structures. Finally, the generalization step applies the optimized setup to a larger dataset.

4.2 Raw Data Inspection and Cleansing

To inspect the covered area and data quality, the user might investigate the raw trajectories (further also referred to as trips) before matching. In case data cleansing is necessary, the system provides filtering options. In the trajectories shown in Figure 4 (a, c), for example, there are different recorded trajectories with severe signal dithering (imprecise trajectories at bends) and recording anomalies (crosscutting trajectories). To overcome these issues as shown in Figure 4 (b, d), the user can control two cutting thresholds.

The first threshold ($dist_{\max}$) defines the maximal distance between two measurements allowed. Considering a recording frequency of, e.g., 30 seconds and an upper speed limit of 70 km/h,

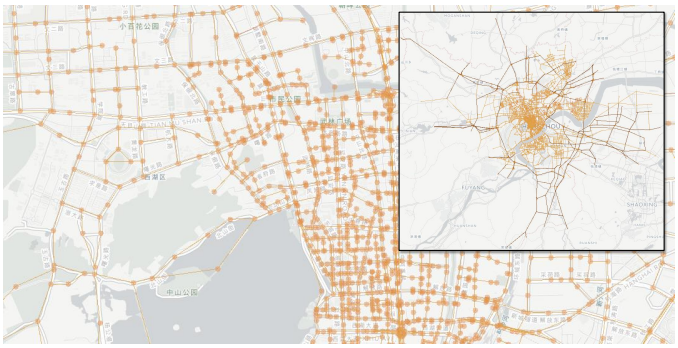

Fig. 3. The road topology of the inner city of Hangzhou (in orange) overlays rendered map tiles. Nodes represent road intersections or bending points and are marked with filled circles. Edges are colored depending on their road types and speed limitations. Darker edges (in overview, right) represent highways.

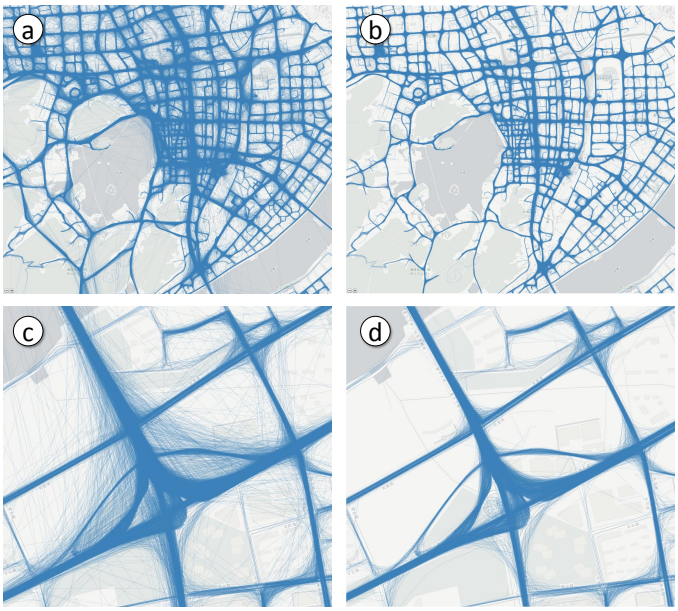

Fig. 4. Parts (a) and (c) show the raw trajectory data, parts (b) and (d) show the same regions with trajectories filtered with thresholds $dist_{\max} = 580$ m, $n_{\min} = 20$.

the distance should not be larger than 580 m. Movement trajectories might exceed this restriction due to signal dithering and wrong records (outliers) as discussed above. Another reason is that individual trips may not be clearly separated in the recording and an unrecorded change of position between two recorded trips can lead to abrupt and unexpected change of position.

The second threshold ($n_{\min}$) defines the minimal number of measurements within a trip. For example, for car trajectories, some trips only comprise starting and stopping the engine at a parking lot. They later cannot be matched to the road network. By increasing this threshold, the user removes insignificant recordings.

We discard all trajectories conflicting with either of the two criteria from further consideration in the map matching process. Supported by immediate visual feedback, the iterative refinement of the parameters allows for a stepwise improvement of the data quality of input data and will improve the later map matching accuracy.

4.3 ST-Matching

After the data is loaded, cleaned, and prefiltered it can be registered to the road system using the ST-matching algorithm [13]. The algorithmic procedure can be described in three steps. First, the *candidate preparation* the measurements are projected to the road network, resulting in a list of candidates. In the *spatial and temporal feature computation* step, candidate features are derived based on distance measures and constraints such as speed. This results in a candidate graph with candidate points as nodes and edges reflecting the shortest paths in the road network between neighboring candidate nodes. Lastly, the *result matching* step matches the trajectory to the highest rated path in the candidate graph by evaluating the computed spatial and temporal features. In the following, we summarize these steps as introduced by Lou et al. [13].

4.3.1 Candidate Preparation

For every point p_i, we compute the k closest candidate points $\{c_i^1, c_i^2, \ldots, c_i^k\}$, where a candidate c_i^j represents the point on an edge e_i^j that is geographically closest to p_i with dist $dist(p_i, c_i^j) \leq candDist$ (Figure 5, left). All road network edges are indexed with a uniform rectangular grid structure to speed up candidate querying.

4.3.2 Spatial and Temporal Feature Computation

This step can be split into a spatial and a temporal feature computation. Two spatial features are computed and multiplied:

$$F_s(c_{i-1}^s, c_i^t) = N(c_i^t) \cdot V(c_{i-1}^s, c_i^t),\ 2 \leq i \leq n \tag{2}$$

The observation probability $N(c_i^j)$ for each candidate is based on the distance between measurements and candidates and is modeled as normal distribution $N(c_i^j) = f(|p_i - c_i^j|, \mu, \sigma)$. The transmission probability $V(c_{i-1}^t, c_i^s)$ (Equation 3) compares the distance $d(p_{i-1}, p_i)$ from a measurement p_{i-1} to p_i (numerator) with the distance $w(c_{i-1}^s, c_i^t)$ that follows the shortest path from candidate c_{i-1}^s to c_i^t through the road network (denominator). The path length $w(c_{i-1}^s, c_i^t)$ is computed using the Dijkstra algorithm [24] to find the shortest path. If it exceeds a certain threshold $w_{\max}$, this connection is discarded. An example where transmission probability becomes necessary is shown in Figure 5, center: c_i^1 is closer to p_i, but would require taking a longer path.

$$V(c_{i-1}^s, c_i^t) = \frac{d(p_{i-1}, p_i)}{w(c_{i-1}^s, c_i^t)} \tag{3}$$

The temporal feature computation takes into account traveling speed. The average speed is computed with respect of the shortest path in the road network between the candidate points.

$$\overline{v}(c_{i-1}^s, c_i^t) = \frac{w(c_{i-1}^s, c_i^t)}{\Delta t(p_{i-1}, p_i)} \tag{4}$$

Then, the algorithm calculates $F_T(c_{i-1}^s, c_i^t)$ as a variant of the cosine similarity between the speed constraints vector $(v_1, v_2, ..., v_m)$ along the path $(e_1, e_2, ..., e_m)$ from c_{i-1}^s to c_i^t and a vector of same length with m constant values $\overline{v}(c_{i-1}^s, c_i^t)$ (for details see Lou et al. [13]). In case the graph structure has no speed information assigned, we assume an average speed of 40 km/h for inner city and 70 km/h for outer city street types.

4.3.3 Result Matching

Finally, the algorithm decides which path to take through the n sets of candidates. We compute a candidate graph that describes all possible transitions between candidates of different sets (see Figure 5, right). Using the temporal and spatial feature computation, the ST-function is defined as follows.

$$F(c_{i-1}^s, c_i^t) = F_S(c_{i-1}^s, c_i^t) \cdot F_T(c_{i-1}^s, c_i^t),\ 2 \leq i \leq n \tag{5}$$

The matching algorithm by Lou et al. [13] searches the best matching path P, which maximizes the sum of ST-values with $1 \leq s \leq k$ and $1 \leq t \leq k$.

$$F(P) = \sum_{i=2}^{n} F(c_{i-1}^s, c_i^t) \tag{6}$$

To evaluate the quality of successful matchings, we compute the arithmetic mean over each found shortest path P_i, with $i = 1, 2, \ldots, m$ (normalized by the length of the path) as a certainty measure S.

$$S = \frac{1}{m} \sum_{i=1}^{m} \frac{F(P_i))}{|P_i| - 1} \tag{7}$$

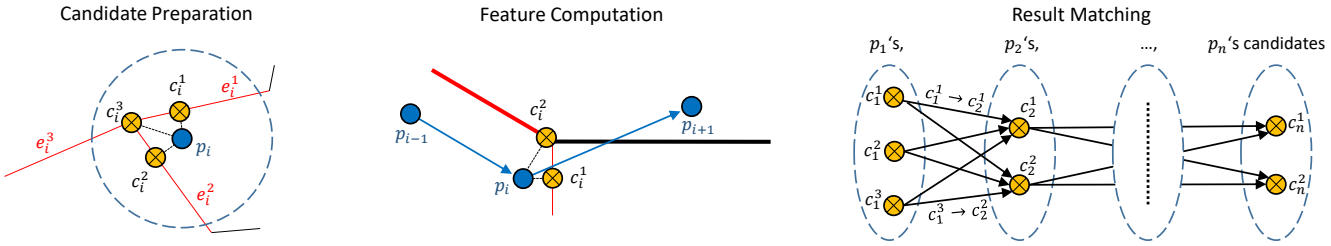

Fig. 5. Illustrations for the three steps of ST-matching, according to Lou et al. [13]. (Left) Candidate points c_i^1, c_i^2, c_i^3 on edges e_i^1, e_i^2, e_i^3 for sampling point p are selected. (Center) An example where transmission probability becomes necessary to decide whether c_i^1 or c_i^2 is more likely. (Right) A candidate graph for a trajectory.

Fig. 6. The matched trips overlay gives information where and how many trips were matched. This is double-encoded by edge thickness and color. Because of the highly varying amounts of matched movements a log scale is used.

As this usually leads to very small values, we compute uncertainty score $U = 1{,}000 \cdot (1 - S)$ for further evaluation of the matching results.

In our implementation, we parallelize the processing to speed up the matching process. The system automatically detect the number of available CPU cores and handles each movement object in an own thread. During the algorithm is executed, a progress bar shows for each movement object how many trips are already processed, how many trips were matched, and how many trips failed (see Figure 1, A).

4.4 Result Exploration

After the raw trajectory data is registered (matched) to the road network, users visually explore the results in the interactive map view that shows the relative amounts of matched trips per road segment (see Figure 6, edge thickness). In addition to the road graph and raw trajectory visualizations described in Section 4.2, users can investigate failed matchings and matching uncertainty (Figure 7). This information is getting available already while the matching algorithm is running. The map iteratively updates, allowing the user to monitor results *online* and to interrupt and restart the procedure if necessary. These features meet the goals expressed in RQ 1.

For some of the trajectories, the matching process might fail because (a) no candidates are found within distance *candDist* or (b) the path length $w(c_{i-1}^s, c_i^t)$ between two found candidates c_{i-1}^s and c_i^t exceeds threshold $w_{\max}$. In both cases, the whole trajectory is discarded; matching only parts of the trajectory would introduce artifacts by artificial start and stop points of trips. An overlay of the map view shows trajectory segments for which no candidates are found in red. These errors appear often when the GPS tracking error is high, the road network is incomplete, or bending roads are sparsely sampled (see Figure 7, left). Trajectory segments for which the path length between candidates is too high are displayed in blue. These errors appear, e.g., when there are multiple edges found on crossroads and multi-lane roads, or intersections points are not correctly modeled (Figure 7, left). To support the user in spotting the most significant errors, a table can be opened that lists hotspots in descending order of severity for both error types. The computation of these hotspots is based on the uniform grid structure we use to rasterize the map and the number of trajectory segments in each grid cell that cannot be matched (Section 4.3.1). When selecting a hotspot in the table, the respective cell is shown and highlighted in the map view (see Figure 8).

Similar to the error overlays, the uncertainty overlay represents aggregated uncertainty values (see Figure 7, right) using a dichromatic gradient. High uncertainty values appear when the trajectories can still be matched (candidates are found and connectable) but measurements have a high spatial offset to these candidates (Figure 7, right). There are multiple possible reasons such as signal dithering or imprecise and oversimplified road network representations. For example, uncertainty increases along multi-lane roads when they are represented by a single edge in the graph.

Visualizing the errors and uncertainties is not trivial because we have to deal with hundreds of overlapping trajectory segments. To address this issue, we apply kernel density estimation as proposed by Lampe and Hauser [25], a variant of edge splatting [26]. Since our approach is designed to work with a data sample (as discussed in the introduction of Section 4), we assume that this sample follows an underlying density distribution that reflects the ground truth. We use a Gaussian kernel for each splat, with $\mu = 21$ and $\sigma = 3$ (in pixels). The resulting two-dimensional array contains a density value for each pixel, which we finally map to color. Using a fixed pixel size for the kernel is results in a stronger aggregation when zoomed out and allows for a drill down by zooming in (see Figure 9). The legends (see Figure 1, B, top-left) provide information about the color mapping. We use a non-linear color mapping (cube root function) to increase the color resolution for lower to mid values.

4.5 Iterative Optimization

The panel to the right (see Figure 1, C) provides aggregated matching statistics. It shows the percentage of trajectories for which the matching failed split by error type, the average uncertainty per trip (i.e., trajectory), and the average runtime time consumed per trip in milliseconds. These values are encoded in line plots. They are updated by adding new data points to the right when parameters

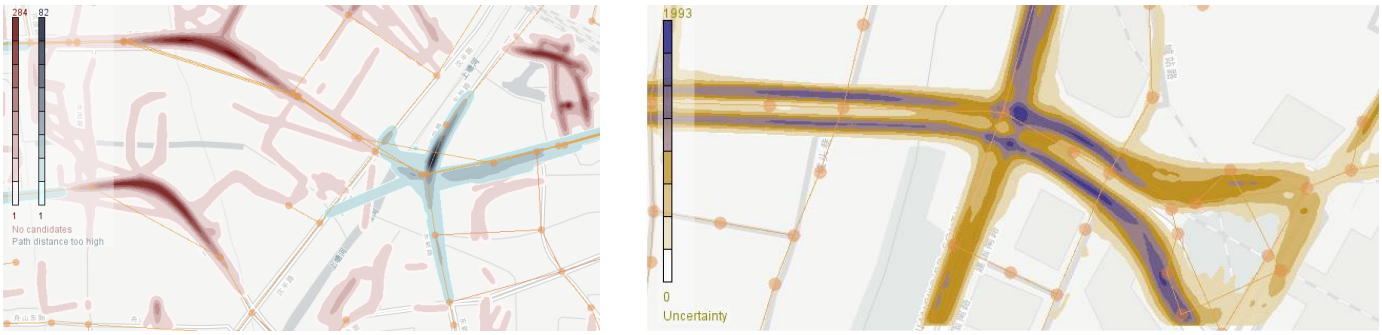

Fig. 7. Kernel density estimation is used to compute a smooth density field for map matching errors and uncertainties. (Left) No-candidate errors in red, path-length-exceeded errors in blue. (Right) A dichromatic gradient shows high uncertainty values in purple and moderate ones in ocher.

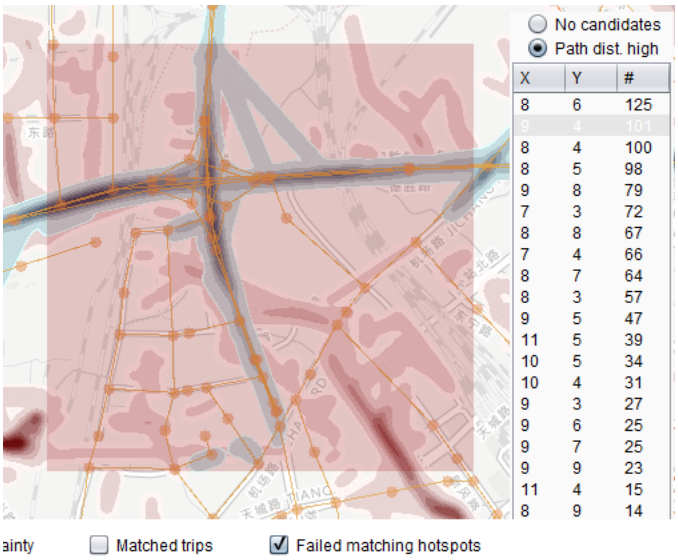

Fig. 8. A table lists the errors in descending order for a selectable error type and allows quickly focusing on grid cells with error hotspots.

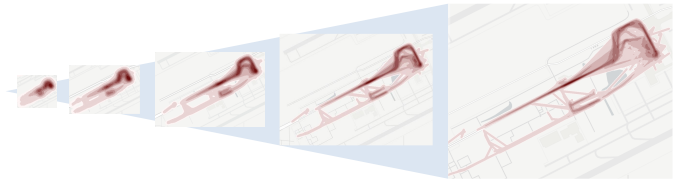

Fig. 9. Increasing detail with increasing zoom level is achieved by invariant splatting kernel size (in pixels). This series shows an airport, where many trajectories start and end (here taxi trips). The airports' roads are not fully modeled in the graph structure, leading to many unmatched trips.

are altered and the matching is rerun. Hence, the view presents a history of matching runs and allows for tracking the success in improving these quality metrics.

Datasets might differ in temporal recording resolution, signal dithering, geographic area and road network, and objects' movement behavior. Hence, the first run of ST-matching might not produce satisfying results and the error and uncertainty overlays might show several hotspots that reflect insufficient matchings. To adjust the map matching to these specifics, the user has three main measures: (i) preprocessing the raw trajectory data, (ii) configuring the ST-matching input parameters, and (iii) editing the road network (graph) structure. The first measure is discussed in Section 4.2. In this section, the remaining two measures are presented. With these features, we finally support the interactive matching process and the semi-automatic editing requested in RQ 2 and RQ 3.

4.5.1 Parameter Configuration

ST-matching has a set of input parameters, which the user can iteratively adjust using different threshold sliders (see Figure 1, A).

As introduced in Section 4.3.1, *candDist* defines the geographic range in which the algorithm searches for potential candidates. With a larger radius for the query, more candidates are found, but there is also a higher chance to incorrectly map the trajectory to a road. A small candidate distance, on the other hand, results in fewer options. In cases where signal dithering is high and the trajectory has a high offset to the road, the correct edge segment might be missed. There is a trade-off between precision (matchings are correct) and recall (amount of trajectories that are matched). Figure 10 shows an example of a crossing (a) and the raw trajectory data (b). When *candDist* is high, trajectories can be matched, but this comes at the price of high uncertainty values (c). Reducing *candDist* decreases uncertainty but introduces candidate errors and hence leads to a lower number of matchings (d).

The number of considered candidates k can be adjusted, too. For example, with $k = 5$ the five nearest neighbors from the candidate query with radius *candDist* will be taken into consideration. As a drawback, increasing *candDist* may indirectly increase k. Both adjustments may significantly affect the runtime of the map matching procedure. According to Lou et al. [13], the runtime complexity of the ST-matching is

$$\mathcal{O}(nk^2 \cdot m \cdot \log m + nk^2) \tag{8}$$

where n is the length of a trajectory and m is the number of edges in the graph. A good combination of the two thresholds highly depends on the underlying road network and data. For example, in inner cities, when the road network is fine-grained, it could be effective to consider a high number of candidates but the distance can be moderate. In contrast, in remote areas where roads are sparse, an opposite balance could work. Currently, our implementation only allows for a global configuration. However, we plan to extend this to local reconfiguration in the future, e.g., using the grid structure.

Lastly, the user can configure the maximal path length $w_{\max}$. It defines a threshold to connect found candidates in the graph. If two candidates are found, but they are not connectable through graph

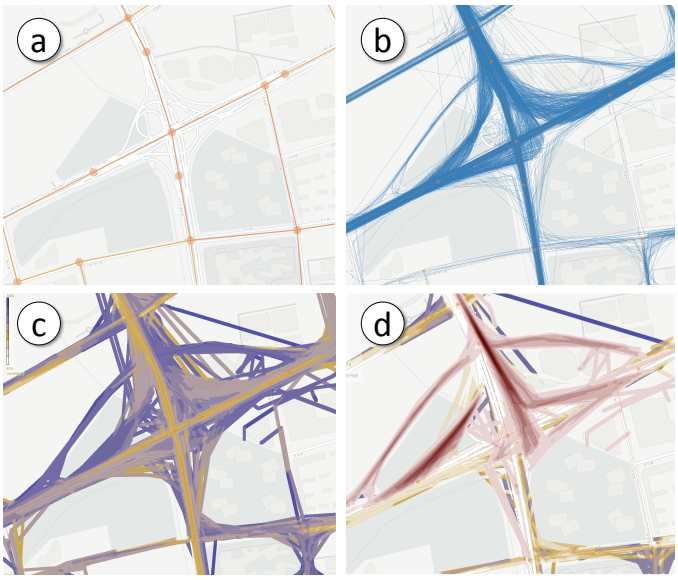

Fig. 10. Matching uncertainty at a crossroad. Image (a) shows the road network, which simplifies the crossroad. In part (b), the raw trajectory data is displayed. Part (c) shows the matching uncertainty, which is higher for trajectories farther away from the graph edges. When reducing the candidate distance in part (d), average uncertainty decreases but matching errors appear.

edges within $w_{\max}$, then they cannot be joined and the matching for this setup is not successful.

It depends on the application case and goals which preprocessing and matching settings are suitable. A carefully chosen parameter setting may reduce both, errors and uncertainty.

4.5.2 Graph Editing

In addition to the parameter configuration, the map matching results highly depend on the accuracy and resolution of the underlying graph structure. When choosing the level of detail, the user again faces a trade-off. With a more accurate graph, the chance to match to certain roads is higher. However, with more edges more options for the mapping appear. For imprecise trajectory data, this might increase runtime but also the ratio of wrong mappings. Hence, graph editing has to be done with care. It highly depends on the application case if, e.g, side roads are necessary to include, or if it is sufficient to map the data to main roads.

When the graph is loaded into the system, the users can visually investigate and edit the graph. They might add new nodes and edges, replace and reconnect them, or delete them. By showing the rendered map tiles (satellite or generalized tiles), users are supported to compare the roads in the graph with the underlying scenery and thereby detect missing or misplaced nodes and edges.

The visual interface contains two editing buttons (see Figure 1, top). By activating the *add node* and *add edge* button, the user can insert nodes and edges as shown in Figure 11. When a node is placed in proximity to an existing edge, the edge is split and the node is inserted, respectively. To alter existing connections, the user can also click on a node and relocate it, or select an edge and delete/bend/reconnect it to another node. By default, the new road segment gets assigned the road type and speed per majority vote of the connected edges.

After the map matching is completed, the errors and uncertainty visualization guide the user to hotspot areas. Often, errors might be reduced by altering the graph structure. Users can either maintain

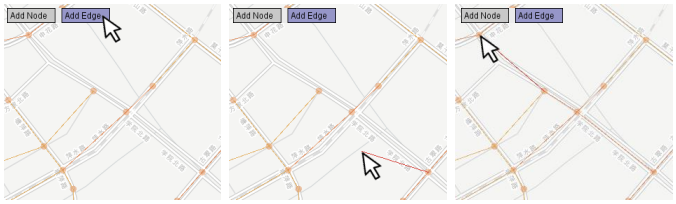

Fig. 11. In this example, the map tiles are used to identify missing edges. The user can add an edge by activating the edge button and selecting the nodes to connect.

manually correction or make use of automatic means. By clicking on the error region (see Figure 12, 3), the system then gives further explanation about the error and suggests fixes.

For errors of type *no candidates found*, an algorithm searches the neighborhood of the respecting measurements and can automatically insert an additional node. In Figure 12, the error occurred on a bending road (see 1, 2). By making use of the automatic node insertion, the nearby edge is split as follows. For all trajectories involved in the error, the algorithm picks the measurement with the maximal distance to the nearby edge, except the measurement is closer to another edge. Next, the algorithm averages the identified positions across all involved trajectories. After reviewing, the user can choose to accept the suggestion or decline it (3, 4, 5). When the map matching is re-executed, the error disappears (6).

An alternative option suggested to the user is to globally increase the candidate distance threshold (3). In this case, again the nearest edge is found and the distance from measurements to this edge is computed. This distance is then used as new threshold.

4.5.3 Local Re-matching

After the graph structure has been edited, the trajectories have to be re-matched. However, re-matching has to be carried out for only those trips that intersect the edited region. The runtime of a full re-execution of the ST-matching would reduce the interactivity of the visual map matching process. The uniform grid structure applied (Section 4.3.1) enables local re-matching. Each cell of the grid indexes a geographical region that holds indexes of all passing trajectories. When the graph is edited, the system finds the intersecting cell. ST-matching is then re-calculated only for associated trajectories.

4.6 Generalization

Usually, the iterative trajectory cleansing, the adjustment of the graph structure, and the steering of matching parameters is carried out with a subset of the available data. Having a dataset with millions of trajectories, the interactivity would be low in our prototype implementation and even difficult to achieve with a highly optimized version. Users would have to wait long for the matching process to finish. The approach thus follows the idea of working with a subset that represents the whole dataset, optimizing the settings accordingly. When the user is satisfied with the results (depending on the application case and data quality), the remainder of the dataset can be matched in the background. Although the user will only see the visual results of matched roads after the process is fully executed, global statistics are still updated online in a progress tracking panel. After the process has finished, the user investigates the results using the matched trips overlay and might export the matched data for further analysis.

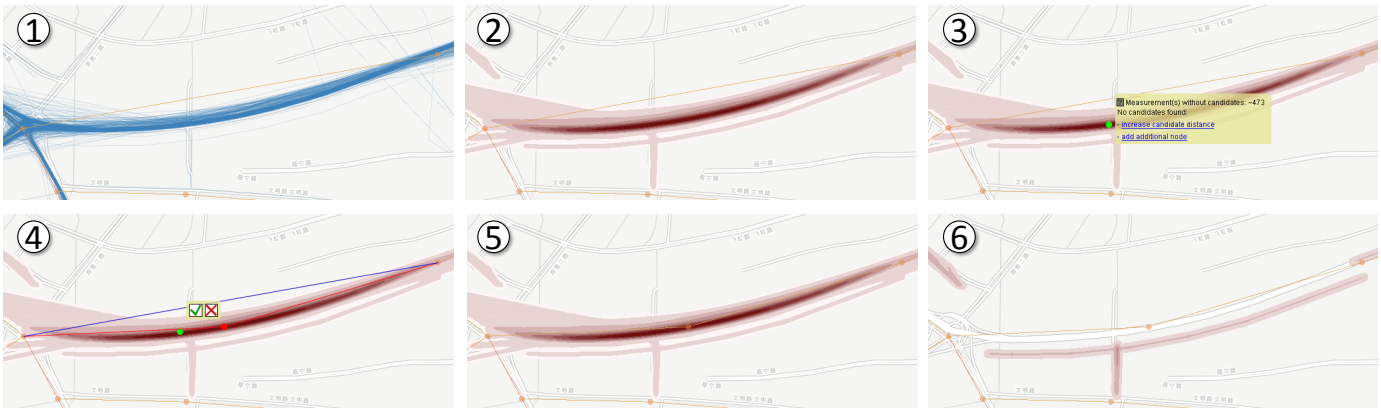

Fig. 12. Labels 1-6 illustrate the editing process based on candidate errors. (1) The raw trajectory visualization shows a clear offset between trajectories and the edge that represents the road. (2) The "no candidate found" error guides the user to these errors. (3) The user clicks on the error hotspot (green dot). The tooltip shows the amount of errors (matching of 473 trajectories failed) at this position and suggests to add a node. (4) The user can choose whether to accept the suggested node (red dot). (5) The node is automatically added. (6) Errors disappeared after local re-matching.

Fig. 13. (Left) Without cleansing and parameter adjustment the map matching underlies many candidate errors (red) as well as path length errors (blue). (Right) After the iterative adjustment (Iteration 9), the map view shows much fewer errors.

When working in the interactive mode, there is a trade-off between reliability and fast response times mitigated by the number of sampled trajectories. To avoid overfitting the parameters of the preprocessing and algorithm to a small sample of the data, we recommend to work iteratively within the visual analytics approach before finally applying the generalization step. Global preprocessing and matching parameters can be estimated with a few trajectories only. To test if the quality metrics remain stable, the user can load additional trajectories step by step. If the quality metrics vary significantly, the parameters need to be re-adjusted and the success can be checked in another iteration of loading more or different samples. Editing the road network might be done with relatively many samples because the local re-matching speeds up the update process.

5 Case Study: Matching Taxi Fleet Data

To demonstrate our approach in a realistic usage scenario, we report a case study matching taxi trajectories to the road network of a city. We use a dataset from a large taxi fleet, comprising more than 8,400 vehicles, in Hangzhou, China. These vehicles are equipped with GPS sensors and tracking devices which record the position and timestamp of the vehicle every 30 seconds. The dataset we study was recorded in January, 2013 and covers one month of movement information. This leads up to a massive dataset, containing about 24 million trajectories. In addition to the trajectory data, a prefiltered OSM graph covers roads in the Hangzhou city area (see Figure 3). This graph also contains speed information for different road types such as inner city roads and highways. In the following, we report on the iterative parameter and graph refinements, and on error and uncertainty score reductions achieved.

5.1 Import, Inspection, and Initial Matching

We first load the meta data in our systems, giving us a list of all 8,400 taxis. From this list, we select a small sample of 20 taxis, containing to around 60,000 trips. We then initially run the map matching process with initial values as listed in Table 1 (Iteration 1). While the matching process is proceeding, we can live-monitor the increasing number of matched trips as well as the matching success. For most taxis, around 74 % cannot be registered to the road network and cause errors (see Table 1, Iteration 1).

TABLE 1
Iterative parameter adjustment and error rates of the case study. k defines the number of candidates, *candDist* the maximal distance to a candidate, *NC* the percentage of failed trips caused by *no candidate* errors, and U the uncertainty score.

Iteration	k	*candDist*	CE	U	Comment
1	3	20	74	920	Initial matching
2	3	20	63	916	After cleansing
3	3	500	1	902	Large *candDist*
4	3	150	16	904	Reasonable *candDist*
5	6	150	16	893	Increasing #candidates
6–8	6	150	7–12	893/4	Graph adjustments
9	6	150	7	894	Path length increase

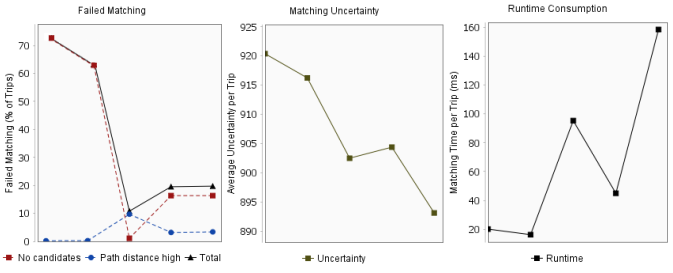

Fig. 14. The statistics panel shows the trade-off between error rates, uncertainty, and runtime. The values are based on the iterative global parameter adjustments listed in Table 1 (iterations 1–5). Uncertainty and errors are decreasing at the costs of higher runtime consumption.

5.2 Filtering out Recording Inaccuracies

Taking a first glance at the raw trajectory data in the map view shows us that there are many inaccurate measurements. This either results from the low recording rate, gaps on the recording, or signal dithering, which significantly reduces the matching success (see blue and red errors in Figure 13, left). To clean the data, we iteratively adapt the preprocessing thresholds. We first set $dist_{\max}$ between two trips to 580 meters. Additionally, in consideration of the large dataset, we choose to accept a loss of some data and set $n_{\min}$ to 10. This greatly reduces the amount of outliers and filters out insignificant recordings, e.g., on parking lots. The result of the data cleansing can be seen in Figure 4. Subsequently, we re-executing the map-matching process. Error rates and uncertainty both slightly decrease (see Table 1, Iteration 2).

5.3 Adjusting Global Parameters

The map view reveals the error hotspots. However, before we start to locally fix major errors, we first test if a global parameter changes can reduce error rates.

We first increase *candDist* from 20 to 500 meters and re-execute the matching. The parameter changes lead to a significantly higher runtime consumption, but we can already see that this eliminates candidate errors almost completely (Table 1, Iteration 3). It, however, can also lead to many wrong matchings as the road grid in the inner city is fine-grained. This results in an increase of path length errors as wrong matchings between candidates cannot be connected. Hence, we correct *candDist* to a reasonable value of 150 meters. This leads to a higher *no candidate* error, but also reduces the path length errors. Lastly, we set the number of candidates k from 3 to 6, which does not affect the error rate, but improves the result quality, as can be seen in a lower uncertainty score (Figure 14, Iteration 5). Additionally, we choose to increase the maximum path length $w_{\max}$. This results in a reduced amount of path length errors (see Figure 16, Iteration 9).

5.4 Visual Investigation and Local Error Correction

The overview of the whole city gives us a first glance about where candidate and path length errors appear. It seems that there are multiple areas where the graph is not fully reflecting the road network (Figure 7, red) as well as multi-lane streets where individual matchings cannot be connected (blue). To investigate the main cases, we activate the failed matching hotspot table. The first candidate error leads us to a road with a strong bend (Figure 12). By activating the graph overlay, we find out that this road is modeled with only a few nodes and edges, leading to a severe offset of the raw movement recordings and the graph edges. To control and validate this situation, we activate the map tiles. The rendered roads confirm our assumption. To correct the error, we use the interactive graph editing capabilities of the system. Clicking on the error visualization on the map, a tooltip appears and suggests to add an additional node. We accept the suggestion and rerun the map matching. The processing panel shows that only some of the loaded trips have to be re-matched. After the re-matching is executed (within 15 seconds), the error disappears in the map and hotspot table accordingly.

The second highest error appears at an area where the graph does not contain nearby roads. Taking a closer look, we cannot see any road segments on the rendered road tiles either. We therefore activate the satellite imagery, which reveals a parking areas and some private roads. Accordingly, we adapt the road network by inserting additional nodes and edges, which removes the error after re-executing the matching process. Figure 15 shows areas with high candidate error counts.

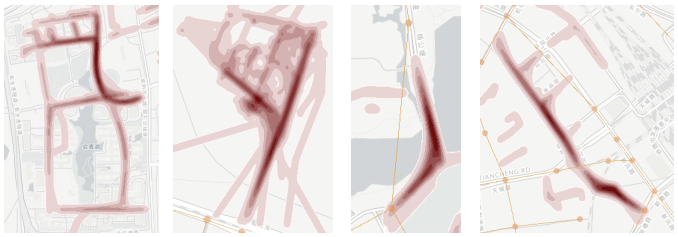

Fig. 15. Typical candidate errors. From left to right: private property, private property, sparsely sampled road, missing road connection.

After iteratively investigating errors, validating the causes, and editing the road network, we are able to decrease candidate errors from 16 % to 6 % (see Figure 16). Lastly, we investigate an error caused by a too high path length. The system proposes to increase the path length to 2,600. We choose to accept this suggestion and further decreased the errors, while the uncertainty score is nearly maintained. Figure 13 (right) shows the relatively small amount of remaining errors.

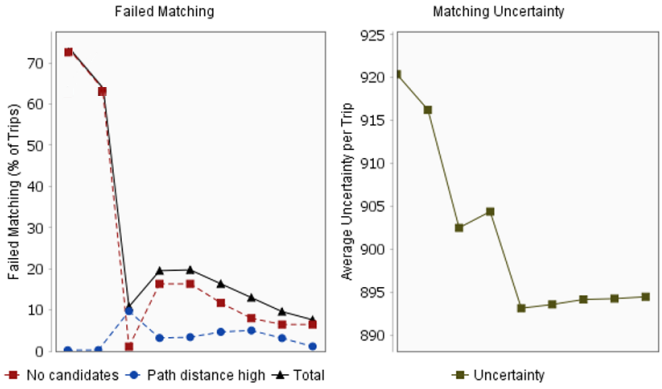

Fig. 16. Manual editing (starting at Iteration 6) led to a decrease of 10 % of the candidate errors and path length errors (from 3 % to 1 %). Uncertainty score is nearly maintained (893 → 894).

5.5 Upscaling to 24 mio. Taxi Trips

As already mentioned, the Hangzhou dataset consists of around 8,400 taxis. Our 20 taxi sample thus only reflects a 0.24 % sample of the taxis. To map the rest of the data, we activate the *match in background* mode. An additional panel gives online updates about the matching process. After the matching is finished, we investigate the matched trajectories and export the edge-weight information using the export function of the system for further analysis.

6 Conclusion

We presented a visual and interactive approach for map matching. It allows users to interactively and visually clean trajectory data, adjust matching parameters, and edit the underlying road network structure. We showed the applicability of the approach in a case study where we significantly reduced matching errors and matching uncertainty. By visual exploration and adjustment of the road network and data, the analyst can gain an increased understanding of structures, data quality, and trade-offs. While the presented techniques are already effective, there are still some limitations and open challenges. For instance, the graph editing process still happens case by case. These manual corrections are time

intensive. As an extension, more automated support would be helpful. This could be done by integrating supervised machine learning. Algorithms may detect similar cases to an edited instance and make additional suggestions that the user accepts or declines.

ACKNOWLEDGMENTS

This work was funded by the European project Cimplex (grant agreement no. 641191), by the joint project Data-Driven Intelligent Transportation between China and Europe announced by the Ministry of Science and Technology of China, Zhejiang Provincial Natural Science Foundation (No. LR14F020002), and by the DFG SPP 1894 project Volunteered Geographic Information (VGI). We also thank our collaborators for providing the data.

REFERENCES

[1] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel, *Visual analytics of movement*. Springer Science & Business Media, 2013.

[2] F. Giannotti and D. Pedreschi, *Mobility, data mining and privacy: Geographic knowledge discovery*. Springer Science & Business Media, 2008.

[3] W. Chen, F. Guo, and F.-Y. Wang, "A survey of traffic data visualization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 2970–2984, 2015.

[4] G. Andrienko and N. Andrienko, "A general framework for using aggregation in visual exploration of movement data," *The Cartographic Journal*, vol. 47, no. 1, pp. 22–40, 2010.

[5] N. Adrienko and G. Adrienko, "Spatial generalization and aggregation of massive movement data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 2, pp. 205–219, 2011.

[6] N. Willems, H. Van De Wetering, and J. J. Van Wijk, "Visualization of vessel movements," *Computer Graphics Forum*, vol. 28, no. 3, pp. 959–966, 2009.

[7] N. Ferreira, J. T. Klosowski, C. E. Scheidegger, and C. T. Silva, "Vector field k-means: Clustering trajectories by fitting multiple vector fields," *Computer Graphics Forum*, vol. 32, no. 3pt2, pp. 201–210, 2013.

[8] S. Kim, S. Jeong, I. Woo, Y. Jang, R. Maciejewski, and D. Ebert, "Data flow analysis and visualization for spatiotemporal statistical data without trajectory information," *IEEE Transactions on Visualization and Computer Graphics*, 2017, online first.

[9] R. Krüger, D. Thom, M. Wörner, H. Bosch, and T. Ertl, "TrajectoryLenses—a set-based filtering and exploration technique for long-term trajectory data," *Computer Graphics Forum*, vol. 32, no. 3pt4, pp. 451–460, 2013.

[10] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva, "Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2149–2158, 2013.

[11] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312–328, 2007.

[12] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 853–864.

[13] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '09. ACM, 2009, pp. 352–361.

[14] D. Guo, S. Liu, and H. Jin, "A graph-based approach to vehicle trajectory analysis," *Journal of Location Based Services*, vol. 4, no. 3-4, pp. 183–199, 2010.

[15] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. Van De Wetering, "Visual traffic jam analysis based on trajectory data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2159–2168, 2013.

[16] X. Huang, Y. Zhao, C. Ma, J. Yang, X. Ye, and C. Zhang, "TrajGraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 160–169, 2016.

[17] X. Wang, S. Zhao, and L. Dong, "Research and application of traffic visualization based on vehicle GPS big data," in *Proceedings of the Second International Conference on Intelligent Transportation*. Springer, 2017, pp. 293–302.

[18] M. Lu, C. Lai, T. Ye, J. Liang, and X. Yuan, "Visual analysis of multiple route choices based on general GPS trajectories," *IEEE Transactions on Big Data*, vol. 3, no. 2, pp. 234–247, 2017.

[19] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive visual specification of data transformation scripts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. ACM, 2011, pp. 3363–3372.

[20] J. Bernard, T. Ruppert, O. Goroll, T. May, and J. Kohlhammer, "Visual-interactive preprocessing of time series data," in *Proceedings of SIGRAD 2012; Interactive Visual Analysis of Data*, no. 081. Linköping University Electronic Press, 2012, pp. 39–48.

[21] T. Gschwandtner, W. Aigner, S. Miksch, J. Gärtner, S. Kriglstein, M. Pohl, and N. Suchy, "TimeCleanser: A visual analytics approach for data cleansing of time-oriented data," in *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business*, ser. i-KNOW '14. ACM, 2014, p. 18.

[22] R. Krüger, D. Herr, F. Haag, and T. Ertl, "Inspector Gadget: Integrating data preprocessing and orchestration in the visual analysis loop," in *Proceedings of the EuroVis Workshop on Visual Analytics*, ser. EuroVA '15. The Eurographics Association, 2015, pp. 7–11.

[23] Z. Wang, X. Yuan, T. Ye, S. Chen, J. Liangk, Q. Li, H. Wang, and Y. Wu, "Visual data quality analysis for taxi GPS data," in *Proceedings of the 2015 IEEE Conference on Visual Analytics Science and Technology*, ser. VAST '15. IEEE, 2015, pp. 223–224.

[24] S. Skiena, "Dijkstra's algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, pp. 225–227, 1990.

[25] O. D. Lampe and H. Hauser, "Interactive visualization of streaming data with kernel density estimation," in *Proceedings of the 2011 IEEE Pacific Visualization Symposium*, ser. PacificVis '11. IEEE, 2011, pp. 171–178.

[26] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf, "Parallel Edge Splatting for scalable dynamic graph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2344–2353, 2011.

Robert Krüger is a postdoctoral researcher at the Institute for Visualization and Interactive Systems (VIS), University of Stuttgart.His research interests are on visual analytics with a special focus on movement and social media data. He is a member of the IEEE.

Georgi Simeonov is a professional software developer. He received the BSc degree in computer science from the University of Stuttgart.

Fabian Beck is an assistant professor at the University of Duisburg-Essen. He received a Diplom degree in computer science and a Dr. rer. nat. (PhD) degree in computer science from the University of Trier. His research focuses on information visualization, visual analytics, and software visualization. He is a member of the IEEE Computer Society.

Thomas Ertl is a full professor of computer science at the University of Stuttgart at the Institute for Visualization and Interactive Systems (VIS) and director of the Visualization Research Center of the University of Stuttgart (VISUS). He received a MSc in computer science from the University of Colorado at Boulder and a PhD in theoretical astrophysics from the University of Tuebingen. His research interests include visualization, computer graphics, and human computer interaction. He is a life fellow of the IEEE.