PFG 2015 / 1, 0095–00104
Stuttgart, February 2015 **Article**

An Algorithm to Generate a Simplified Railway Network through Generalization

Paul Czioska, Frank Thiemann, Monika Sester, Robin Giese & Hermann Vogt, Hannover

Keywords: generalization, line merging, railway, skeletonisation

Summary: Maps play a major role in communicating information in the context of public railway transportation, for instance as route maps. In addition, the current position of the trains can be shown on such a map to enrich the information content (*Live Map*). For the positioning and routing of the trains on the track, a graph structure of the traffic network is necessary. Since railway tracks are mostly arranged in a parallel manner, it makes sense to merge the track lines into representative (centre-)lines, which reduces significantly the amount of edges and helps to identify possible topological errors in the input data. This work presents an algorithm which merges track data based on topological properties, so that branches and crossings can be distinguished. The implementation uses generalization techniques like an area collapse operator and methods from computational geometry like polygon triangulation. An evaluation shows that the output graph is capable for routing tasks.

Zusammenfassung: *Ein Algorithmus zur Erstellung eines vereinfachten Bahnnetzes durch Generalisierung.* Karten spielen eine wichtige Rolle bei der Informationskommunikation im Rahmen des öffentlichen Verkehrs, etwa als Liniennetzplan, auf dem zusätzlich die aktuelle Position des Verkehrsmittels dargestellt wird (*Live Map*). Um solche Karten zu erstellen, ist eine routingfähige Graphstruktur der Gleisdaten nötig. Da Gleise meist parallel verlaufen, ist es sinnvoll, die Gleise zu repräsentativen Linien zu aggregieren, um die Redundanz zu verringern und mögliche Topologiefehler der Eingangsdaten zu beheben. In dieser Arbeit wird ein Algorithmus vorgestellt, der einen Gleisdatensatz nach topologischen Eigenschaften zusammenfasst. Der Implementierungsansatz nutzt hierzu Techniken aus dem Bereich der Skelettierung von Polygonen durch Triangulation. Eine abschließende Evaluation zeigt, dass das Ergebnis für Routinganwendungen geeignet ist.

1 Introduction

The sector of public transportation is of outstanding importance for the infrastructure and thereby for the wealth of a country. According to Destatis (2013), approximately 30 million people per day use the public transportation system in Germany. It is obvious that such a huge and complex network needs an efficient way to communicate traffic information to its users.

Maps play a major role in this task. They are able to illustrate connections between different means of transport in a way that every user is able to understand quickly. In most cases, the connections are shown in a schematic manner, like the popular route map of the London Tube. Since this type of map lacks a lot of additional cartographic information, it has become also common to show the geometry of the route as an overlay on a real map, so that the user is able to locate stations and the travelled route in between.

A dynamic enhancement of this type of map is the so called *Live Map*, where, in addition to the tracks, the positions of the moving vehicles are drawn in real-time. The position information can be gathered either directly from GPS or indirectly calculated through spatiotemporal interpolation on the given track, like it can be seen for example at the German "*DB Zugradar*", www.bahn.de/zugradar (Fig. 1).

A basic requirement for the routing of the trains is a graph structure of the track network.

DOI: 10.1127/pfg/2015/0255 1432-8364/15/0255 $ 2.50

The nodes of this graph thus have real world coordinates, so that the correct geometry of the tracks can be shown. Such a graph structure is necessary for different tasks: the edges can be used to interpolate an approximation of the current position of the train when position signals are available only at discrete survey stations along the track, and a routing on the graph can be used to determine the complete route of a train. In some Live Map applications, this route can be highlighted when desired by the user (like the red line in the example of Fig. 1 shows).

A trivial solution to obtain a track graph is to use raw track data without any modification, that is, the line data of the tracks simply converted into a graph structure without any further processing. In fact, many Live Map applications at present use such an unprocessed data basis.

Track datasets for this purpose can be obtained either from official sources (like rail operators or transport associations) or derived from volunteered geographical information (VGI), accessible through portals like OpenStreetMap (OSM). Unfortunately, depending on the quality of the input data, one cannot be sure that the obtained track data is topologically correct, i.e. that line endings are connected properly. If this basic condition is not fulfilled, a routing will lead to unexpected errors, such as confusing detours or no possible connection at all. Especially when VGI is used as input data the quality cannot be ensured, so that an additional check is recommended.

Furthermore, it is apparent that railway tracks are mostly arranged in a parallel manner. If the whole track network is used for a routing application, a significant redundancy of track information is present that slows down the algorithm, while the single tracks are of no importance for a macroscopic routing application. In addition, users could be irritated when the Live Map shows a train pulling into a station on a specific platform, but in reality this platform was only chosen because it represents the shortest way in the graph and not the platform defined in the timetable. Hence, a transformation into a representative geometry is desirable in many cases.

In order to handle those issues, the task is therefore to generalize the redundant tracks while preserving the topologic relationship. In the context of generalization, this procedure can be classified as *merging* (Shea & McMaster 1989). As a by-product, the use of such an operator has the possibility to identify and repair incorrect input data. In detail, the following requirements are necessary:

- Bundle parallel tracks that belong to the same transport route and create a graph with real world coordinates that represents the approximate centre of the route (Fig. 2)
- Preserve the topology of the input data. In other words: insert a node at track switches (Fig. 3), but do not insert a node at crossings, e.g. bridges or tunnels (Fig. 4)
- Repair topological errors in the input data, so that the network is routable (Fig. 5).

In this paper we present an algorithm that fulfils these requirements and generates a simplified rail network graph. It combines different existing computational geometry methods such as buffering and skeletonisation and uses in addition a new technique called *Track Tracing*.

The outline of this paper is as follows. Section 2 gives a brief overview over other topics that are related to the current problem. In

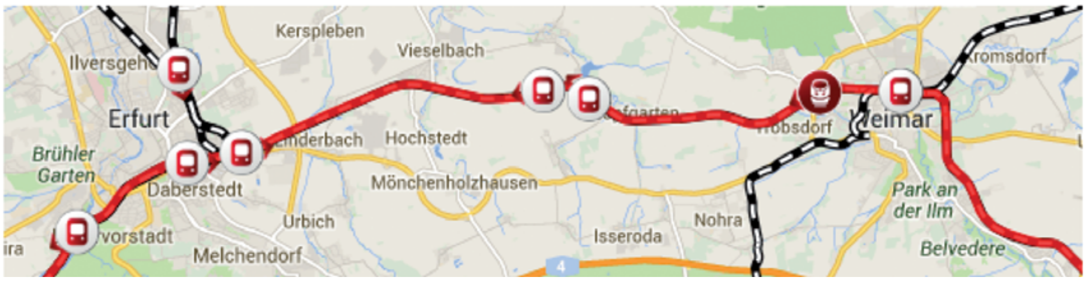

Fig. 1: Screenshot of the Live Map "*DB Zugradar*". Source: www.bahn.de/zugradar (24.6.2014)

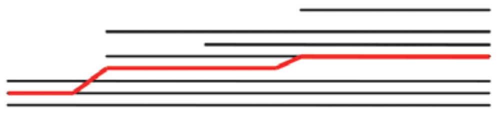
Fig. 2: Task: Bundle parallel tracks.

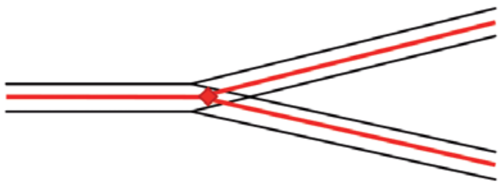
Fig. 3: Task: Insert a node at track route branches.

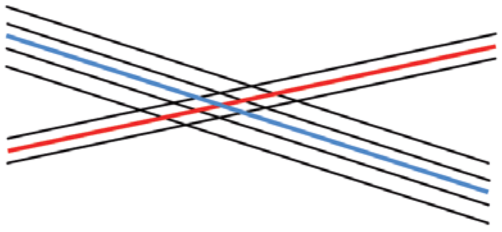
Fig. 4: Task: No node insertion at crossings.

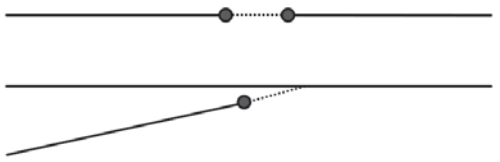
Fig. 5: Task: Handle topologic errors in the input dataset.

section 3, the algorithm is presented. Furthermore, in section 4 results of the process are shown. Section 5 discusses the evaluation of the results with a test routing. Finally, section 6 provides the conclusion and directions for future work.

2 Related Work

The generation of a routable graph network from line data without further simplification is a fairly simple task since it is sufficient to treat all vertices as nodes and add the corresponding line connections as edges to the graph. Problems may only arise when the input data is somehow flawed. Neis et al. (2011) discovered that the amount of topological errors at street connections in OpenStreetMap has declined rapidly in the past several years.

Nevertheless, the street topology is not flawless, so that an additional data preparation is still necessary.

However, most articles focus on street or pedestrian routing, whereas a routing on railway tracks or the quality of its geometry is not examined. Due to the lack of such measurements, we checked the topological integrity of railway track data from OpenStreetMap by a simple check of the distance between every node and its nearest neighbour. If the distance is greater than zero and above a certain threshold (we used an empirical value of 0.75 m), a node pair is marked as erroneous. The result of this systematic analysis showed that the raw railway track data in the region of Germany seems very suitable for the purpose of routing: in 175,000 nodes only 44 errors have been detected, corresponding to an error probability of ~ 0.03%. Certainly, for more accurate results a more sophisticated analysis is necessary since not every error type is covered in this test.

In terms of line network generalization, most previous work focuses also on road networks. A common approach is the principle of *good continuation* proposed by Thomson & Richardson (1999). In this work, a road network is grouped into linear elements called *strokes*, which represents a chain of road sections which have a good continuation. Based on this technique, Thom (2005) presents a method to collapse dual carriageways into a single centre line. Unfortunately, since railway vehicles have a huge turning radius, nearly all rail tracks have a relatively good continuation which makes this principle not applicable.

A topic that also deals with line merging is the map inference from movement trajectories. One of the main research topics in this field is the construction of a road map by processing vehicle tracking data. Due to the uncertainty of GPS measurements, trajectories are mostly scattered around the true movement path. A reasonable way to infer a road network is therefore the creation of a centreline of all trajectories belonging to a common street segment, which is also a form of a merging operation.

One approach to achieve this goal using k-means clustering is presented by Edelkamp & Schrödl (2003). In the work of Lee et al. (2007), similar trajectories are detected and grouped in order to derive a representative trajectory for common parts, which is close to the problem investigated in this paper. Another inspiring idea comes from Cao & Krumm (2009), who simulate physical attraction between the trajectories. Zhang et al. (2010) describe a method to extract a centreline out of GPS traces with perpendicular lines using fuzzy c-means clustering in order to separate close roads. Biagioni & Eriksson (2012) present a map inference pipeline similar to the workflow described in this paper. First, a kernel density estimation (KDE) is applied on the raw GPS traces, which produces a density raster. Subsequently, a gray-scale skeletonisation is used to derive the road centrelines.

However, car trajectories differ in some characteristics from railway track data. Trajectories have measurement errors, while a track derived from a map is assumed to have the correct coordinates. Also, single lines are often treated as outliers in a car trajectory analysis and are thus neglected, but a single railway track has to be preserved. These differences reveal the requirement of a new algorithm.

3 Merging Operator

The basic concept of the proposed simplification operator is the merging of tracks which are located close to each other while preserving the underlying topological information of the traffic routes. In order to carry out this task, a kind of morphological operation is performed: At first, the tracks are buffered (Dilation), and subsequently the buffered area is transformed back to a line (Erosion), which corresponds to the Closing operation. The major challenge – which is also different from conventional skeleton approaches – is the preservation of the topological relations.

In detail, the algorithm consists of the following four steps, which are explained more in detail in the course of this section:

1) buffering of the tracks and amalgamation of all buffer polygons,
2) removal of holes,
3) triangulation of the buffer polygon,
4) creating the skeleton in consideration of the underlying tracks.

Optionally, point information, e.g. stations, can be added in a fifth step by a map matching operation to enable station-to-station routing. Due to the limited space, this step is not covered in this article.

3.1 *Buffering of the Tracks*

In the first step, a buffer operation is performed on every track segment of the input data. The buffer width has to be defined manually by the user – the smaller the distance value is chosen, the more single tracks will be created in the output dataset. A big value corresponds thereby to a more generalised result. In the experiments, a typical buffer distance of 12 m was selected in order to span all neighbouring tracks, which corresponds to the tripled value of the normal track distance of 4 m. Subsequently, all buffer polygons are amalgamated into one large polygon, respectively multipolygon in the case of an incoherent track network, with holes.

3.2 *Removal of Holes*

In areas with a high track density and many parallel tracks, e.g. at marshalling yards or big stations, some holes may occur in the total polygon when the gap between two tracks is slightly higher than the chosen buffer distance in step one. Those holes lead to disruptive splits in the resulting graph, because an output line will be created on both sides of the hole. This effect can be avoided by a removal of holes which have a size smaller than a defined threshold. The threshold size has to be chosen carefully – too big values may destroy useful information while small values lead to many forks in the graph. In practice, values between 4000 m^2 and 10000 m^2 have lead to good results, depending on the desired level of detail. For the results shown in this work a threshold of 7500 m^2 was used.

3.3 *Triangulation of the Polygon*

In order to obtain a line dataset based on the created polygon, an area collapse mechanism is needed which returns the *skeleton* of the polygon. Several different approaches for a skeletonisation exist in the literature. A good overview is given for example by HAUNERT & SESTER (2008).

For the algorithm presented in this paper it is required to use the method of skeletonisation by polygon triangulation, as proposed by CHITHAMBARAM et al. (1991). A brief description of the procedure is given in section 3.4.

The triangulation itself is based on a constrained Delaunay triangulation of the polygon, where triangles outside the polygon are removed. The more homogeneous the triangles are shaped, the smoother the skeleton gets. Therefore, the algorithm uses a *Conforming Delaunay Triangulation,* where additional *Steiner Points* are inserted into the polygon edges. For further details we refer to BERN & EPPSTEIN (1992).

3.4 *Creation of the Skeleton*

A conventional skeleton based on a triangulated polygon is created by an analysis of the triangles and distinguishes two types of triangles: Normal triangles (1) have at least one common edge with the polygon boundary, while interior triangles (2) have no edge coincident with the polygon and occur on junctions. The skeleton edge is then created for every triangle by linking the midpoints of the interior triangle edges. When triangles of type (2) are processed, a centre point of the triangle is inserted and linked with all three midpoints of the triangle edges.

In contrast to a conventional skeleton operation, the proposed algorithm uses also the topologic relations from the original tracks. All triangles are processed iteratively and the following steps are executed:
1) track search,
2) repairing topology errors,
3) analysis of the track connectivity,
4) inserting connection lines,
5) at interior triangles: track tracing.

Track search

In order to detect the underlying topology of the tracks, the first step is the retrieval of the affected track data in the area of the currently processed triangle. To avoid a wrong connectivity analysis, it is necessary to cut the tracks at the triangle edges.

Repairing topology errors

Depending on the quality of the input data, topological errors like not correctly connected track segments may occur in the input data and can lead to gross errors in the output data. They can be detected by measuring the distance and angle difference between track line endings and/or vertices – if the distance is very small and the track segments have an approximately similar direction, an error may be present. It can be repaired by connecting the separated nodes.

Analysis of the connectivity

The topologically clean tracks are then investigated with regard to the connectivity. Fig. 6 shows exemplarily three possible track constellations in a triangle:
a) two tracks simply pass the triangle,
b) a track merges (Branch),
c) a trac splits (Branch).

The connectivity analysis contains mainly the partition of the tracks into distinct track sets. Distinct means that a track set has no connection to another track set.

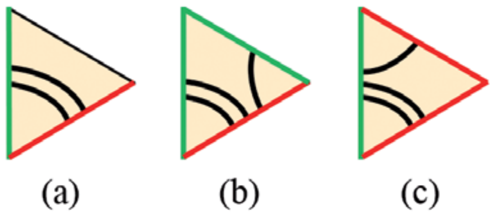

Fig. 6: Examples of different track constellations in a triangle, green: input edge, red: output edge.

Inserting connection lines

This information is now used to determine the necessary connection lines in the output graph. In most cases like in Fig. 6a, this is a trivial task where just the centres of track intersections at two triangle edges are connected. The only exception occurs at interior triangles, which marks the initial situation for a track tracing.

Track tracing

Since it is not possible to distinguish branches from crossings in just one single triangle, it is necessary to trace the tracks further until a proof is found that the tracks that are coming from different triangle edges merge (Branch) or divide without any common node in between (Crossing). For such a tracing, the triangle structure provides a good frame condition since the calculations can thus be locally limited.

An interior triangle of type (2) always marks the start of a tracing function if all three edges are crossed by tracks. An example can be seen in Figs. 7 and 8: The green interior triangle labelled with 1 has different tracks crossing each triangle edge, thus a track tracing over the triangles labelled with 2 is necessary until the decision between branch and crossing can be made. Fig. 7 shows the case of a branch where the tracks meet a few triangles further (labelled with 3). In contrast, Fig. 8 illustrates a crossing where the four tracks of the route from top left to the bottom right do not have a common node with the two tracks of the other route until they diverge again in the triangle labelled with 3.

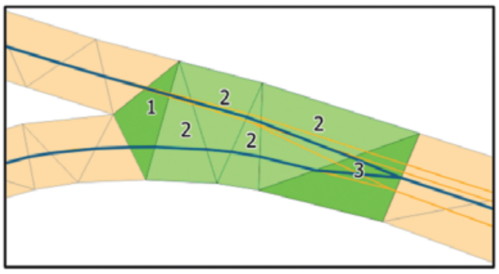

Fig. 7: Branch of two rail routes.

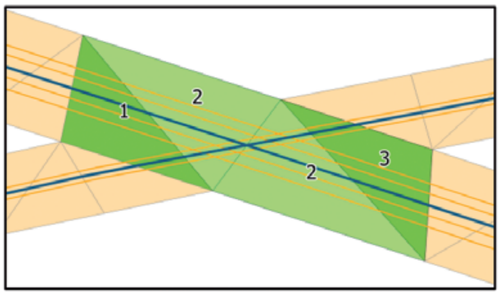

Fig. 8: Crossing of two rail routes.

The proceeding of the track tracing is as follows. At first, all three edges of the starting triangle are classified as *input* or *output edge*. Usually, a starting triangle has two input edges and one output edge where all tracks from both input edges converge, as shown in Fig. 6b. In rare cases it can happen that this assumption is not fulfilled, e.g. when there are direct connections between all three edges. In this case, a tracing is not possible and the algorithm only inserts output edges between the corresponding intersection points.

In all other cases, the following loop is executed:

1) <u>Label distinct track sets</u> All distinct track sets that intersect the same input edge are grouped into a *route track set (RTS)*. This holds also if new tracks accrue during the loop.
2) <u>Go to the neighbouring triangle(s)</u> at the output edge(s) and group the underlying tracks to the RTS.
3) <u>Check for connections</u> All RTS are checked among each other if a connection exists. If so, a branch is present and the RTS are united. If desired by the user, RTS can also be united due to a certain similarity, e.g. if the angle difference and/or distance between the RTS are below a certain threshold.
4) <u>Insert output edges</u> between the intersection centres of the RTS at the triangle edges. Old output edges in the current triangle are removed and overwritten unless a *Collision exit* is present, see below.
5) <u>Check the stop criteria</u>
   - *Branch exit*: Only one RTS remains due to a branch
   - *Crossing exit*: Two RTS leave the triangle through two different output edges
   - *Collision exit*: The current triangle is the end of a tracing from the opposite direction.

Otherwise, if two or more RTS leave through an output edge, continue with step 1.

The track tracing is in principle able to handle an arbitrary number of traced RTS simultaneously. Note that care is necessary when a new tracing overwrites an old tracing but stops at an earlier triangle compared to the old tracing – in this case, the output edges may not match at the triangle edge.

Since the output edges only span the based triangle and are thus relatively short, it is useful to connect those fragments in a final step to create longer polylines.

4 Results

The following examples show the result of processing a German-wide railway dataset obtained from OpenStreetMap. For the calculation, only standard tracks have been used excluding tracks in marshalling and maintenance yards. In Fig. 12 the result network is visualized as a red line on top of an OpenStreetMap background in the area of the main station in Hamburg. It can be seen that the parallel running railway tracks in OSM have been replaced by one single line which follows approximately the medial axis of the tracks.

Fig. 9 shows exemplarily the output of processing a branch with several tracks. Here, the original railway tracks are drawn in orange and the output line is overlaid in blue. Fig. 10 shows a more complex track junction near Kreiensen, Germany, where the original topology is also preserved correctly. In this special case, the stop criterion was a *Collision exit* where the central part was processed from the left side as well as from the right side.

In general, however, it is not always possible to generate a visually attractive appearance in addition to topological correctness. Such visual drawbacks mostly occur at big marshalling yards or junctions with more than three track routes converging, as the complex example of Saarbrücken (Fig. 11) shows. Although the topology of the blue output graph is correct, the visual appearance is not satisfying due to many successive track tracings with overwriting the existing output.

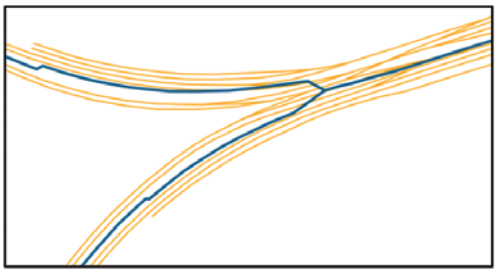
Fig. 9: Example of a simple branch.

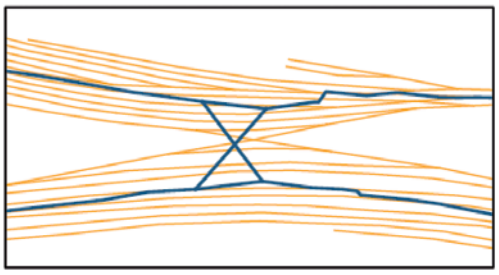
Fig. 10: Example of a more complex branch in Kreiensen, Germany.

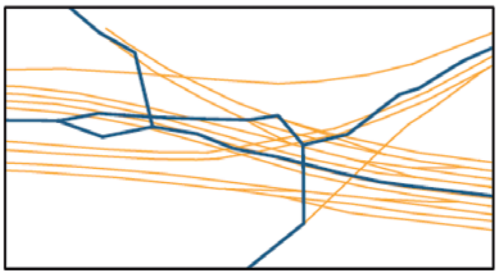
Fig. 11: Example of a very complex track field near Saarbrücken, Germany.

5 Evaluation

Besides the examination of the visual appearance, the results were evaluated concerning the topological correctness and routing ability. The topological correctness of a processed German-wide track network obtained from OSM was estimated by an automatic check for node connections (see Related Work for more information about the test configuration). It yielded only two errors out of 28.528 tested node pairs, which is a rather good result that indicates a highly reliable quality of the topology. On the contrary, it shows also that the algorithm does not work faultlessly since errors may still appear in rare cases. In various other experiments with different parameter settings, such errors occurred particularly at helical tunnels or huge marshalling yards.

In order to verify the result with respect to the routing ability, it is not sufficient to check only node connections, because other topological errors can as well significantly influence the routing and may result in large detours of the routed trains. Hence, a test routing was carried out. The basis for the test routing was:

- Railway track data from OSM
- Station coordinates from OSM
- Timetable information, accessed at
  www.bahn.de
  www.der-metronom.de

The routing itself was executed by the routing algorithm of the HAFAS journey planner from HaCon. A total of 4 relations in northern Germany have been tested and analyzed (Fig. 13). The evaluation was carried out in two ways: on the one hand manually by visual inspection of the route between the start and end nodes, and on the other hand by calculating the needed train speed between the stations. If the calculated speed of a train rises to unrealistic values, e.g. above 200 km/h for local trains, it can be assumed that this error comes from a topological issue in the routing dataset. The train has to take a detour and thus a longer distance, but the travel time keeps the same, which results in a higher speed.

The result: both the visual inspection of the route and the speed check yielded no problems – all trains could take the assumed route with an appropriate speed.

When processing large datasets, time and memory complexity is an important aspect as it describes how much the time- and memory usage grows with increasing input size. The theoretical complexity mainly determined by the buffering and union steps is $O(n \log n)$. The memory and time usage was determined experimentally. Thereby a near-linear behaviour of both time and memory usage was observed. In our case, the calculation of small datasets (~ 14.000 line segments) took approximately 3 minutes, whereas the whole German railway network (~ 130.000 line segments) required approximately 40 minutes (CPU: Xeon E5 processor with 2,4 GHz).

A drawback in terms of calculation time is the dependence of the processing direction of the tracks; that is, different input datasets result in a different geometry of the output data, even in regions where the original data is congruent. Thus, a tile based partition of the input data in order to parallelize the computation is not directly possible.

6 Conclusion

In this work we presented a new algorithm to derive a routable rail network from detailed railway track data in consideration of the to-

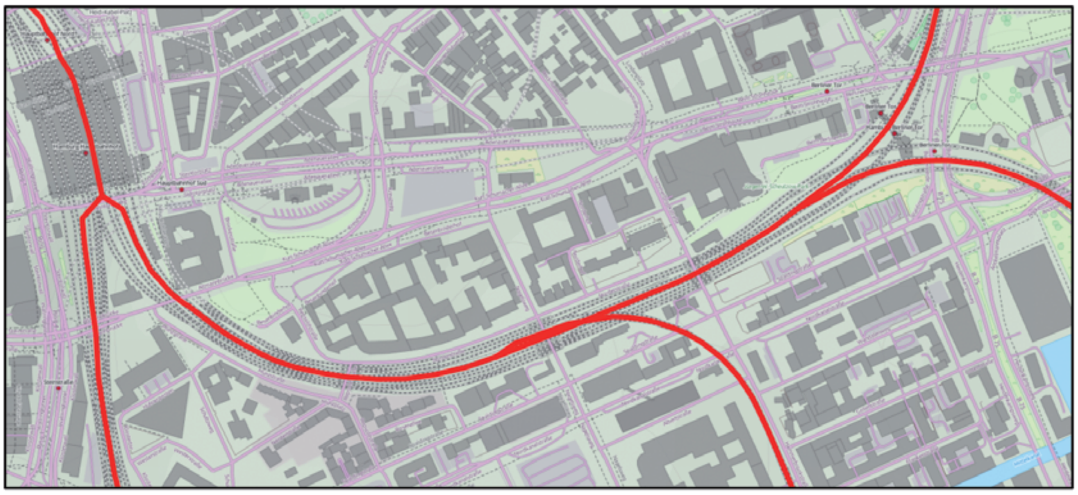

Fig. 12: Output railway network in the area of Hamburg, Germany (background: OpenStreetMap).

pology of the input tracks. The algorithm uses different methods of geometry type change (buffer operation and dimensional collapse) in order to create a representative line of the track routes. The core of the operator is a track tracing mechanism based on triangles in order to distinguish branches from crossings and thus to preserve the topology of the original track network.

Different evaluation approaches like a test routing indicate a very good quality in terms of topological correctness and routing ability of the output graph. The visual appearance of the resulting representative lines is rather good in most cases, although some complex

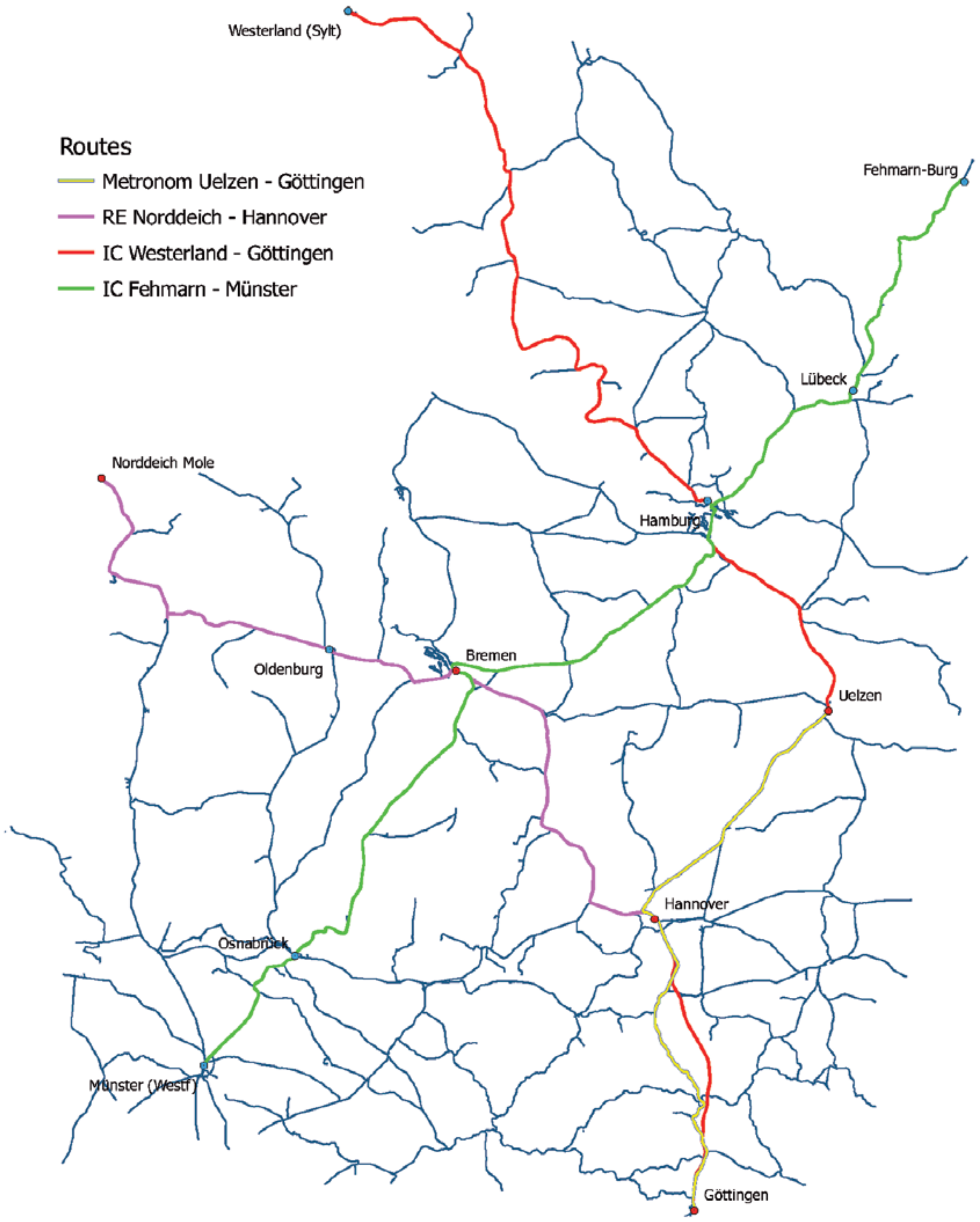

Fig. 13: Routed relations in northern Germany, based on a processed OSM dataset.

junctions may result in a confusing output line constellation. At this point, the visual appearance could benefit from an additional post processing which preserves the topology.

A drawback of the presented method is though the dependence of the processing direction, which complicates a parallelization of the computation. For this purpose, approaches have to be used that deal with overlapping parts (see e.g. THIEMANN et al. 2013).

References

BERN, M. & EPPSTEIN, D., 1992: Mesh generation and optimal triangulation. – Computing in Euclidean geometry **1:** 23–90.

BIAGIONI, J. & ERIKSSON, J., 2012: Map inference in the face of noise and disparity. – 20th International Conference on Advances in Geographic Information Systems ACM: 79–88.

CAO, L. & KRUMM, J., 2009: From GPS traces to a routable road map. – 17th ACM SIGSPATIAL **2009:** 3–12.

CHITHAMBARAM, R., BEARD, K. & BARRERA, R., 1991: Skeletonizing polygons for map generalization. – Technical Papers ACSM-ASPRS Convention, Cartography and GIS/LIS **2:** 44–54.

DESTATIS (DEUTSCHES STATISTISCHES BUNDESAMT), 2013: Busse und Bahnen mit neuem Fahrgastrekord. https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2014/04/PD14_127_461.pdf (15.9.2014).

EDELKAMP, S. & SCHRÖDL, S., 2003: Route planning and map inference with global positioning traces. – Computer Science in Perspective, Springer: 128–151.

HAUNERT, J.H. & SESTER, M., 2008: Area collapse and road centerlines based on straight skeletons. – Geoinformatica **12** (2): 169–191.

LEE, J., HAN, J. & WHANG, K., 2007: Trajectory clustering: a partition-and-group framework. – ACM SIGMOD International Conference on Management of Data.

NEIS, P., ZIELSTRA, D. & ZIPF, A., 2011: The Street network evolution of crowdsourced maps: OpenStreetMap in Germany 2007–2011. – Future Internet **4** (1): 1–21.

SHEA, K.S. & MCMASTER, R.B., 1989: Cartographic generalization in a digital environment: When and how to generalize. – AutoCarto **1989:** 56–67.

THIEMANN, F., WERDER, S., GLOBIG, T. & SESTER, M., 2013: Investigations into partitioning of generalization processes in a distributed processing framework. – 26. International Cartographic Conference, Dresden, http://www.icc2013.org/_contxt/_medien/_upload/_proceeding/395_proceeding.pdf (15.9.2014).

THOM, S., 2005: A Strategy for collapsing OS Integrated Transport Network (ITN) dual carriageways. – 9th ICA Workshop on Generalisation and Multiple Representation, http://generalisation.icaci.org/images/files/workshop/workshop2005/Thom.pdf (15.9.2014).

THOMSON, R.C., & RICHARDSON, D.E., 1999: The 'good continuation' principle of perceptual organization applied to the generalization of road networks. – ICA 19th International Cartographic Conference.

ZHANG, L., THIEMANN, F. & SESTER, M., 2010: Integration of GPS Traces with Road Map. – Workshop on Computational Transportation Science in conjunction with ACM SIGSPATIAL, San Jose, CA, USA.

Address of the Authors:

M.Sc. PAUL CZIOSKA, Dipl.-Ing. FRANK THIEMANN, Prof. Dr.-Ing. MONIKA SESTER, Leibniz Universität Hannover, Institut für Kartographie und Geoinformatik (IKG), Appelstr. 9A, D-30167 Hannover, Tel.: +49-511-762-5422, Fax: +49-511-762-2780, e-mail: {czioska} {thiemann} {sester} @ikg.uni-hannover.de

ROBIN GIESE, Dr. HERMANN VOGT, HaCon Ingenieurgesellschaft mbH, Lister Str. 15, D-30163 Hannover, Tel.: +49-511-33699-0, Fax: +49-511-33699-99, e-mail: {robin.giese} {hermann.vogt} @hacon.de

Manuskript eingereicht: Juni 2014
Angenommen: Oktober 2014