Matching River Datasets of Different Scales

Birgit Kieler[1], Wei Huang[2], Jan-Henrik Haunert[1], Jie Jiang[2]

[1]Institute of Cartography and Geoinformatics, Leibniz Universität Hannover, Appelstraße 9A, 30167 Hannover, [birgit.kieler, jan.haunert]@ikg.uni-hannover.de
[2]Department of Geo-spatial data product R & D, National Geomatics Center of China, 1 Baishengcun, Zizhuyuan, Beijing, 100044, [huangwei, jjie]@nsdi.gov.cn

Abstract. In order to ease the propagation of updates between geographic datasets of different scales and to support multi-scale analyses, different datasets need to be matched, that is, objects that represent the same entity in the physical world need to be identified. We propose a method for matching datasets of river systems that were acquired at different scales. This task is related to the problem of matching networks of lines, for example road networks. However, we also take into account that rivers may be represented by polygons. The geometric dimension of a river object may depend, for example, on the width of the river and the scale.

Our method comprises three steps. First, in order to cope with geometries of different dimensions, we collapse river polygons to centerlines by applying a skeletonization algorithm. We show how to preserve the topology of the river system in this step, which is an important requirement for the subsequent matching steps. Secondly, we perform a pre-matching of the arcs and nodes of the line network generated in the first step, that is, we detect candidate matches and define their quality. Thirdly, we perform the final matching by selecting a consistent set of good candidate matches.

We tested our method for two Chinese river datasets of the same areal extent, which were acquired at scales 1:50 000 and 1:250 000. The evaluation of our results allows us to conclude that our method seldom yields incorrect matches. The number of correct matches that are missed by our method is quite small.

Keywords: data matching, network, multi-scale representation, generalization, skeletonization

1 Introduction

In former times, the cartographer's job was to map the unexplored land. Today, however, we are rather faced with an excess than with a lack of data: for most parts of the Earth, digital geographic databases have been acquired multiple times, for multiple applications, and in multiple scales. This leads to two primary questions addressed by current cartographic research. First, how can we minimize the effort for keeping the databases up-to-date? Secondly, how can we combine the information given with different databases? An important prerequisite for answering these questions is to develop methods for database integration (Devogele et al. 1998). Subproblems of database integration are schema matching and data matching. Schema matching deals with the identification of corresponding concepts in data models (Volz 2005). Data matching aims to find corresponding objects in different datasets. In our paper we deal with data matching and focus on the matching of river datasets.

Data matching is useful for updating, since we can trigger an update from one object to a corresponding object in another dataset, once both datasets have been matched; for this purpose correspondences found by matching are stored as links in a database (Harrie and Hellström 1999; Dunkars 2004). Furthermore, we can combine the attribute sets given for both objects into a single detailed set, which, for example, allows users to perform complex analysis tasks. In this paper we assume that the schemas of both datasets were matched prior to the data matching process, for example, we can identify objects of river classes in both datasets and know that these classes represent similar concepts. Obviously, this knowledge is useful for data matching. Note, however, that data matching can also be applied to detect unknown correspondences between schemas (Kieler et al. 2007) and, when dealing with different scales, unknown generalization rules (Sester et al. 1998).

The identification of corresponding objects is often manually done or performed with semi-automatic procedures, which is expensive or even infeasible for large datasets. However, in recent years researchers have developed fully automatic methods for certain matching problems. Diez et al. (2008) consider the problem of matching road networks in datasets with different map projections; the transformation between the coordinate systems of both datasets is unknown. We, however, assume that the datasets are in the same coordinate system. The difficulty in our problem is not to find a global geometric map transformation but to deal with differences that are due to map generalization. Walter and Fritsch (1999) as well as Zhang and Meng (2007) developed methods for matching road datasets

that have similar scales but were captured for different thematic domains, that is, a topographic dataset and a dataset for car navigation. The problem becomes more involved if the difference in scale increases, since small-scale datasets contain geometrically simplified shapes. Moreover, objects may be eliminated or aggregated through generalization (Timpf 1998). Therefore, matching algorithms that rely on comparisons of geometric features may fail. In order to match datasets of different scales, additional criteria need to be considered. Most existing methods for matching networks of lines exploit topological relations between map objects (Lüscher et al. 2007; Mustière and Devogele 2008; Zhang and Meng 2007). These relations do not so much depend on the scale. For example, the geometry of lines representing roads may be simplified to a high degree, but the topology of the road network is mainly preserved during generalization. Uitermark et al. (1999) developed a method for matching road datasets of different scales, where all roads are represented by area objects; in order to derive a network of lines, a skeletonization method is applied. To conclude, the matching problem is most explored for objects of the same geometric dimension, also considering different scales. However, there are still open problems, especially when of objects with different geometric dimensions are to be matched. For that reason we did not use a standard matching tool, like RoadMatcher (Vivid Solutions 2005). This open source software only handles line networks and only finds one to one matches.

Generalization often reduces the geometric dimension of objects, for example, a river may be represented by a polygon in a large-scale map or by a line in a small-scale map (Haunert and Sester 2008). Furthermore, there may be river objects of different geometric dimensions in a single dataset, for example, wide rivers are represented by polygons and narrow rivers are represented by lines. In this paper we address the matching of river datasets of different scales, also allowing for different geometric dimensions. However, our approach is not restricted to rivers. Roads in datasets of very large scale, for example, in cadastral maps, are represented by polygons. Our method may also be applied to match such a dataset with a topographic dataset of smaller scale, where roads are represented by lines.

The matching method that we propose comprises three steps. First, river polygons are collapsed to centerlines by applying a skeletonization algorithm. We show how to preserve the topology of the river system in this step, which is an important requirement for the subsequent matching steps. Secondly, a pre-matching of arcs and nodes is performed. In this step we detect candidate matches and define their quality. Thirdly, the final matching is performed by selecting a consistent set of good candidate matches.

The paper is structured as follows. We briefly sketch the context of our work, that is, we present the data we are dealing with and how they are captured and used in applications (Section 2). Then we present our matching method of three steps in Section 3, which is the main part of our paper. We evaluate and discuss the results of our experimental tests in Section 4 and conclude the paper in Section 5.

2 The Use Case: Chinese River Datasets

The national mapping agency of China manages topographic databases of four different scales, namely 1:50 000, 1:250 000, 1:1 000 000, and 1:4 000 000. Until now, these databases are collected and maintained independently, but for the future it is aimed to apply automated generalization and matching methods in order to ease the updating process. The databases are used to derive analog maps but also to directly support offices in their planning activities and decision-making procedures. Each database contains information on river systems; the geometric detail and the number of attributes reflect the particular scale. The lines representing rivers constitute so-called digital line graphs (DLGs). We use this term for the datasets we are dealing with, but we explicitly include polygons representing rivers. From now on, we refer to the river datasets of scales 1:50 000 and 1:250 000 as DLG 50 and DLG 250, respectively. We exclude the two datasets of smallest scales from our investigations. Figure 1 shows our test area, which has an extent of approximately 54 km² and is located in a rural area close to Shanghai. The Chinese datasets have attributes that allow for a distinction of natural and man-made waterways. There is also an attribute reflecting the name of the river. However, we do not consider these attributes in our approach, since this information has not been captured completely.

When we started to develop the matching strategy for rivers, we expected that the river network would have a structure similar to a tree, that is, we expected many confluences of rivers but only a few bifurcations. Our idea was to exploit this pattern for the matching procedure. However, we did not follow this idea, because the actual structure of the river network is not similar to a tree, see Fig. 1. The encountered structure with many bifurcations results from the extensive canal and dam system.

Finally it is remarkable that dataset DLG 250 is more up-to-date and contains a lot of new canals, which are not reflected in dataset DLG 50.

DLG 50 DLG 250

Fig. 1. Two river datasets of different scales in the test area: DLG 50 (large-scale dataset) and DLG 250 (small-scale dataset); lines are displayed in grey and polygons in black

3 Matching Procedure

Our matching method, which is illustrated as a process chart in Figure 2, is a three-steps procedure. First, in order to cope with geometries of different dimensions, we construct a network of lines, which is described in detail in Section 3.1. For this purpose, we collapse river polygons to centerlines by applying a skeletonization algorithm. The used basic method is presented in Section 3.1.1. Afterwards, in Section 3.1.2, we show how to preserve the topology of the river system in this step, which is an important requirement for the subsequent matching steps. The second step (Section 3.2) includes the pre-matching process, where we separately detect matching candidates from the arcs and nodes of the line network which we generate in the first step. Therefore we use distance criteria and angle difference criteria, in order to assess the quality of the matching candidates. In Section 3.3, we present the last step of our method. Based on the results of the pre-matching step of Section 3.2, we perform the final matching by selecting a consistent set of good candidate matches.

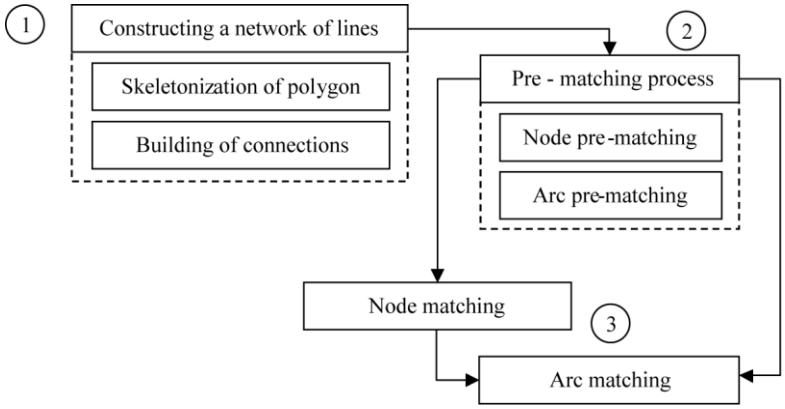


Fig.2. The process chart of our matching method deals with geometries of different dimensions and datasets of different scales.

3.1 Constructing a Network of Lines

In this section we present our method for automatically constructing a network of lines from the input data, which includes lines and polygons representing rivers. We first present a basic method for deriving centerlines for single polygons (Section 3.1.1) and then discuss how to preserve the topology of the river system (Section 3.1.2).

3.1.1 Creating Centerlines for a Single River Polygon

Haunert and Sester (2008) compare different types of skeletons that are commonly used in geographic information systems for deriving polygon centerlines. This includes the *medial axis*, which comprises straight lines and second-order lines. We do not select the medial axis, since handling second-order lines would cause computational overhead. An alternative skeleton is the *straight skeleton*, which only comprises straight lines. However, the existing algorithms for constructing the straight skeleton are too slow to handle large datasets. Therefore, we select a simple skeleton that is based on a constrained Delaunay triangulation of the polygon, see Figure 3. Penninga et al. (2005) discuss this method in detail. We give an outline of this method.

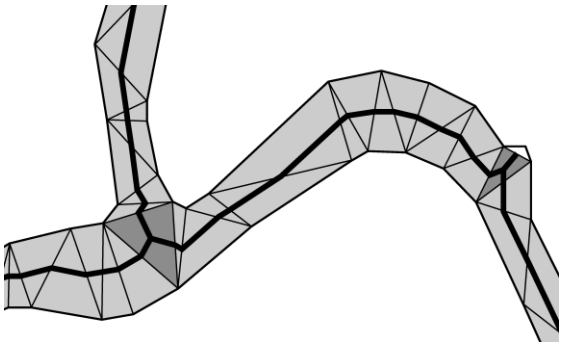


Fig. 3. Skeletonization of a river polygon. The constructed skeleton is bold black. There are two 0-triangles (dark shaded) and one 2-triangle (white); all other triangles are 1-triangles (light shaded).

The constrained Delaunay triangulation of a polygon is an exhaustive partition of the polygon into non-overlapping triangles. Most existing triangulation algorithms yield additional triangles in the exterior of the polygon; however, only the triangles in the interior of the polygon are used for constructing the skeleton. After constructing the triangulation, the triangles are handled independently. For each triangle, a piece of the skeleton is added. This procedure differs for different types of triangles:

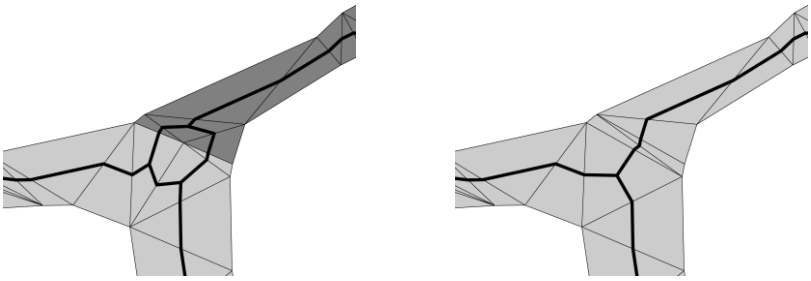
- For each triangle that shares *two* edges with the polygon (that is, a *2-triangle*), no skeleton edge is added.
- For each triangle that shares *one* edge with the polygon (that is, a *1-triangle*), one skeleton edge is added. This edge is defined by connecting the midpoints of both other triangle edges.
- For each triangle that shares *no* edge with the polygon (that is, a 0-*triangle*), three skeleton edges are added. Each such edge is defined by connecting the midpoint of a triangle edge with the triangle's centroid, that is, the point $((x_1+x_2+x_3)/3, (y_1+y_2+y_3)/3)$, where (x_1, y_1), (x_2, y_2), and (x_3, y_3) are the triangle vertices.

The main disadvantage of this skeleton compared to the medial axis is that it is not smooth. Often the centerline is zigzagging. However, we will define a matching method that is quite robust against these geometric distortions. We are mainly concerned with keeping the topology of the river network correct. We accept the disadvantage of a less smooth shape, since the described skeleton can be applied to construct a topologically correct river network, which we show in the next section.

3.1.2 Preserving the Topology of the River System

In the previous section we presented a method for constructing a skeleton of a single polygon. We now discuss how to preserve the topology of a river system that is represented by multiple lines and polygons. In order to accomplish this task, we perform a pre-processing prior to the skeleton construction and a post-processing after the skeleton construction. Both the pre-processing and the post-processing comprise two steps.

In the first pre-processing step, we amalgamate all mutually adjacent river polygons. Without this step we would obtain incorrect results at river junctions. Figure 4 shows the skeletonization result when calculating the triangulation for each polygon independently (Figure 4(a)) and when amalgamating the adjacent polygons before calculating the triangulation (Figure 4(b)). In the left figure there is a node on the boundary shared by both polygons; this causes the artifact during the skeleton construction. The latter result is better, since it represents the topology of the river network correctly.



(a) without amalgamation of polygons (b) with amalgamation of polygons

Fig. 4. The skeleton for two adjacent river polygons (different shades).

In the second pre-processing step, we deal with the case that the endpoint of a line lies on the polygon boundary, for example, a narrow river (the line l) flows into a wide river (the polygon p). In this case we need to ensure that the shapes of the two rivers remain connected. We could try to solve this problem after the skeleton construction without any pre-processing, for example, by extending the line l in its original direction, until it touches the constructed skeleton. This approach, however, is too naive, since we would possibly create intersecting lines (see Figure 5). In order to avoid new intersections, we propose to define the connection of l and the skeleton of p based on the triangulation of p. Our approach requires that, if an endpoint v of l lies on the polygon boundary, the same point is a vertex of the triangulation. We can ensure this requirement simp-

ly by introducing *v* as an additional polygon vertex. This is done in the pre-processing, that is, before constructing the skeleton.

Fig. 5. When collapsing a river polygon, the connectivity of the river system can be affected ((a) and (b)). Extending lines to meet the skeleton line can result in unwanted intersections (c). Therefore, we suggest another approach, see Figure 6.

In the first post-processing step, that is, after applying the skeletonization method from Section 3.1.1, we construct the connections between the endpoints of the original river lines and the derived skeleton. Let *v* be a polygon vertex that is equal to the endpoint of a line. Since *v* is a vertex of the polygon, it is also a vertex of at least one triangle of the triangulation. We define *T* as the counter-clockwise ordered sequence of triangles that share the vertex *v*. We select a subsequence $T' = (t_1, t_2, ..., t_n)$ of *T* such that *T'* has the maximum number of elements among all subsequences of *T* that have the following property: each two subsequent triangles share a common edge. Note that the sequence *T'* is usually equal to *T*. However, the definition of *T'* is needed, since we can also construct special cases where two subsequent triangles in *T* do not share a common boundary. We now handle two different cases separately:

- If the number *n* of triangles in *T'* is *even*, we select the triangle edge *e* that separates $t_{n/2}$ and $t_{n/2+1}$. We insert a skeleton edge by connecting the vertex *v* and the midpoint of *e*, see Figure 6(a).
- If the number *n* of triangles is *odd*, we select the triangle $t = t_{(n+1)/2}$. Again, we consider different cases:
    - if *t* is a 2-triangle we insert a skeleton edge by connecting the vertex *v* and the midpoint of its opposite triangle edge as shown in Figure 6(b).
    - if *t* is a 1-triangle we insert a skeleton edge by connecting the vertex *v* and the midpoint of the skeleton edge that we added for *t* as shown in Figure 6(c).
    - if *t* is a 0-triangle we insert a skeleton edge by connecting the vertex *v* and the triangle's centroid as shown in Figure 6(d).

This approach indeed ensures that the topology of the river system is preserved. Intersecting connections as shown in Figure 5(c) are not possible. This is because, for each triangle, there is only a finite set of potential connections. These potential connections do not intersect.

In the second and final post-processing step we remove some 'dangling' arcs. More precisely, we discard any arc that terminates at a vertex of degree one and is shorter than the river width.

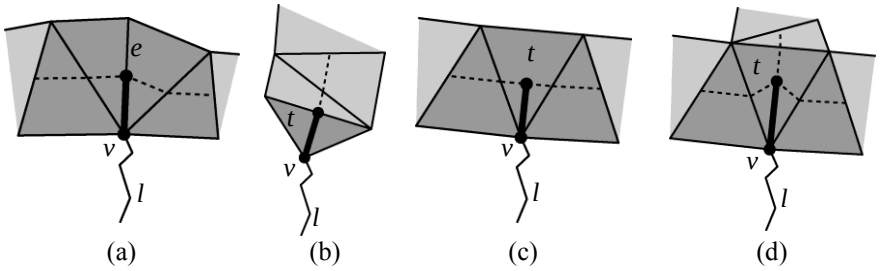


Fig. 6. A narrow river (line l) enters a wider river (dark and light shaded regions). To preserve the connectivity, we add a connection (bold black) between l and the constructed skeleton (dashed lines). The triangles in T' are shaded dark grey.

3.2 Pre-Matching Process

In this section we present our method for the selection of matching candidates, separately for nodes (Section 3.2.1) and arcs (Section 3.2.2) in order to reduce the input data for the final matching step. Furthermore, we define the quality of matching candidates.

3.2.1 Pre-Matching of Nodes

In the final matching step we will search, for each node n of the small-scale dataset, a corresponding node n' of the large-scale dataset. In this paper, we use the small-scale dataset as the reference dataset and the large-scale dataset as the target dataset. It is likely that a node corresponding to n exists, because the less detailed dataset usually does not contain more information than the detailed one. In the pre-matching step we search a set N'_n of nodes that *possibly* correspond to n. We should keep N'_n as small as possible while ensuring $n' \in N'_n$. The set N'_n may contain more than one node or could also be empty.

We perform the pre-matching of nodes based on a distance threshold δ. First, we calculate a buffer for each node n of the small-scale dataset, that is, a circle of radius δ with centre n. We define the candidate set N'_n as the

set of nodes in the large-scale dataset that are contained in this buffer. In order to define an appropriate buffer size, we need to discuss two cases. First, if the chosen buffer size is too small, correct matches may be lost. Secondly, if the buffer size is too large, we need to resolve many ambiguous cases in the further process. Since lost matches cannot be recovered in the final matching step, an over-selection of candidates is better than an under-selection.

We store the results of the pre-matching of nodes in a table. Each row contains the identifiers of two nodes that form a potential match. The table also has a column that represents the distance of both nodes. This distance can be seen as a quality measure. Candidates with a small distance to the reference node have a high quality and candidates with a large distance have a low quality. All node candidates are analyzed concerning their suitability further in the node matching process of Section 3.3.1.

3.2.2 Pre-Matching of Arcs

Beside the detection of node candidates, we also perform a pre-matching of arcs based on another distance threshold ε. The pre-matching of arcs is similar to the pre-matching of nodes. We also store the results in a table. For each arc a of the small-scale dataset, the pre-matching yields a set A'_a of arcs of the large-scale dataset. Again, we need to ensure that A'_a is small and contains the actual match a'. To select an appropriate set of candidates for a, we calculate a buffer polygon β_a, which contains all points at distance ε from a or closer. We define the candidate set A'_a as the set of arcs of the large-scale dataset that are completely contained in this buffer polygon or cross its boundary.

We measure the quality of a potential arc match by comparing the direction of both arcs. The direction of an arc can be defined as the orientation angle of the straight line connecting both endpoints of the arc. However, an arc in the detailed dataset may correspond only to a part of an arc in the less detailed dataset. Therefore, the directions of a and a' can be very dissimilar when measured for the whole arcs. To define an appropriate quality measure, we perform a local comparison. For each arc b of the large-scale dataset, we define a buffer polygon β_b, which we also define based on the threshold ε. In order to assess the quality of the match (a,b), we calculate the intersection of both buffers, that is, we define $\beta_{ab} = \beta_a \cap \beta_b$, see Fig. 7. The part of a that lies in β_{ab} may correspond to the part of b that lies in β_{ab}. For both parts we can construct a straight line by connecting the start and endpoint. Comparing the orientation angles of these lines we can infer about the quality of the match (a,b).

If the difference of the angles is small, then the arcs have almost the same orientation. In this way the quality of the possible correspondence is high. However, if the angle difference is high, then the likelihood that the arcs have the similar orientation is quite low. Figure 8 displays all matching candidates that reach a certain quality; from left (Fig. 8 (1)) to right (Fig. 8 (3)) the quality increases. In Fig. 8 (1) all segments of the large-scale dataset are displayed in bold grey; these are possible matching candidates, because they are located in the buffer polygon of the investigated reference arc. In Fig. 8 (2) the segments with an angle difference of $\Delta\alpha \leq$ 45 degrees are displayed in bold grey. Obviously the angle difference is sufficient, since all orthogonal river parts of the investigated reference part, which are apparent improper matching candidates, will get a low quality. In Fig. 8 (3) we show the matching candidates which fulfill the angle difference of $\Delta\alpha \leq$ 25 degrees. In this case some correct matching candidates are missing, because the orientation of the arcs are too different. However, the remaining arcs get a high quality.

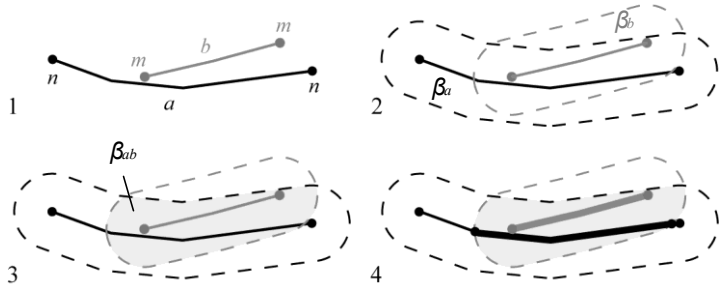


Fig. 7. Method for identification of arc segments for the comparison of orientation angles. (1) Arc *a* (black) of small-scale dataset and arc *b* (grey) of large-scale dataset; (2) Buffer polygons β_a and β_b; (3) Intersection area β_{ab} (light grey polygon); (4) Comparable parts of an arc: reference (bold black) and target (bold grey) dataset.

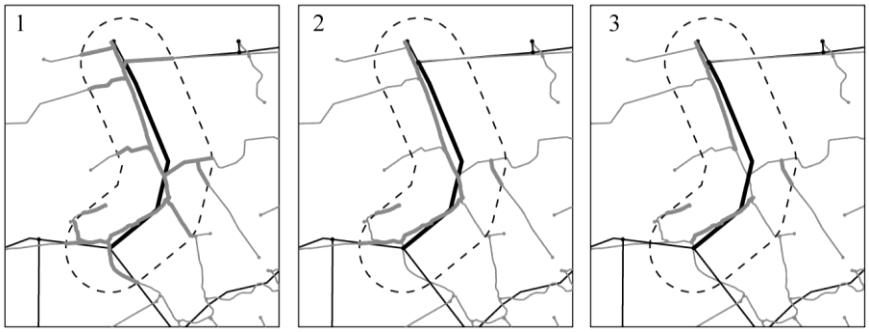


Fig. 8. (1) All matching candidates (bold grey) for an arc *a* (bold black) of the small-scale dataset; (2) all matching candidates whose orientation angle α is similar to the orientation angle of *a* ($\Delta\alpha \leq$ 45 degrees); (3) all matching candidates whose orientation angle α is *very* similar to the orientation angle of *a* ($\Delta\alpha \leq$ 25 degrees).

3.3 Final-Matching Process

The pre-matching results of the previous section are used as input for the final matching process, which comprises the matching process of nodes (Section 3.3.1) and, thereafter, the matching of arcs (Section 3.3.2).

3.3.1 Matching of Nodes

In this section we identify, for the nodes of the small-scale dataset, the nodes of the large-scale dataset that correspond best. For this we exploit the pre-matching results from Section 3.2.1 and analyze the node-arc topology. Therefore we have to extend the analysis to the arcs that are connected to the nodes.

First, we detect a set of node matches that are quite obvious – we call them *certain* matches. For this, we determine the node degree *deg(n)* for each node of the small-scale dataset and for all its node matching candidates. Normally, the less detailed dataset does not contain additional arcs. Therefore, we define that a node match *(n,m)* cannot be certain if the degree of the reference node *n* is greater than the degree of the candidate node *m*, that is, if *(deg(n) > deg(m))*. For example, the match in Fig. 9 (1) cannot be certain, but the matches in Fig. 9 (2) can be certain.

To decide whether a node match *(n,m)* with *deg(n)* $\leq$ *deg(m)* is certain, we analyze the set S_n of arcs that are incident to *n* and the set S_m of arcs that are incident to *m*. By querying the table that contains the pre-matching results for arcs, we select the set *P* containing all potential arc matches where one arc is in S_n and the other one is in S_m. Our approach is to define

the node match *(n,m)* as certain only if there is a sufficiently good subset *Q* of *P* that has the following properties:

- Each arc in S_n belongs to exactly one match in *Q*.
- Each arc in S_m belongs to maximally one match in *Q*.
- The quality of each arc match *(a,b)* in *Q* is not worse than a certain threshold θ. As defined in Section 3.2.2, the quality is measured according to the angle difference of the arcs *a* and *b*.

In order to find such a subset, we propose a simple iterative approach. Initially, we set *Q* empty. We order the potential arc matches in *P* according to their quality, into a list. We iterate through this list. First, we select the arc match of highest quality and then we proceed with the next arc match. Let *(a,b)* be the arc match selected in a certain iteration. We add the match *(a,b)* to *Q* if its quality is high enough and if *Q* does not contain another match with *a* or *b*. After we reach the end of the list, we test whether each arc in S_n belongs to exactly one match in *Q*. In this case we define the node match *(n,m)* as certain.

After selecting the certain matches, we select additional node matches. We iteratively assess the potential node matches, ordered by decreasing quality. We call a node *free* if it has not yet been matched to another node. In each iteration we apply the following rule:

- we accept a potential node match *(n,m)* if both *n* and *m* are free and if there is no other potential match *(n,o)* such that *o* is free.

After assessing this first rule for all potential node matches, we apply a second rule:

- we accept a potential node match *(n,m)* if both *n* and *m* are free and if *m* is the free node closest to *n*.

Note that there may still be reference nodes that are unmatched.

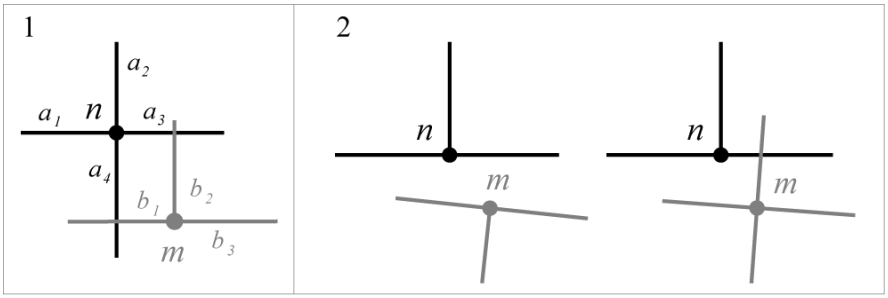


Fig. 9. (1) Exclusion of such candidate nodes *m* of the large-scale dataset, because *deg(n)>deg(m)*; (2) Further investigation of such kind of matches, because the defined conditions *deg(n)=deg(m)* or *deg(n)<deg(m)* are fulfilled.

3.3.2 Matching of Arcs

We perform the final matching of arcs based on the pre-matching of arcs and the matching of nodes. We select the arcs one by one from the small-scale dataset and search for the corresponding set of arcs in the large-scale dataset. In doing so, three cases need to be considered: (1) both endpoints of the arc of the small-scale dataset are matched with nodes in the large-scale dataset, as shown in Fig. 12a; (2) only one endpoint is matched with a node in the large-scale dataset; (3) none of the endpoints is matched with a node in the large-scale dataset, as shown in Fig. 12b. Different methods will be applied to the three cases.

Case (1): In this case we search a path that comprises pre-matched arcs of the large-scale dataset and connects the nodes that are matched to the endpoints of the selected small-scale arc. In order to find a good path, we minimise a cost function. This approach for matching arcs has been proposed by Mustière & Devogele (2008). For each arc a included in the path we charge a cost equal to the product of its length and the angle difference to the small-scale arc as defined in Sect. 3.2.2. This minimisation problem can be solved with a shortest-path algorithm, for example, the algorithm of Dijkstra (1959).

Case (2): In this case we first select all pre-matched arcs that have an endpoint that is matched with an endpoint of the small-scale arc and whose angle difference with the small-scale arc is smaller than a certain threshold μ. We match the small-scale arc with one arc of this selection, more precisely, with the arc that is closest to the unmatched endpoint of the small-scale arc.

Case (3): The small-scale arc remains unmatched.

4 Experimental Results

Our proposed matching process was tested in the test area (see Fig. 1) for the DLG 250 as the small-scale dataset and the DLG 50 as the large-scale dataset. In the first step we constructed a network of lines and ensured that the topology of the river network is complete. The result is shown in Fig. 10. For the distance criterion in the pre-matching of nodes we applied a threshold of $\delta = 440$ m, which was empirically determined. For the following pre-matching of arcs we applied the buffer size of $\varepsilon = 330$ m in order to get, for each arc a of the small-scale dataset, the set A'_a of arcs of the large-scale dataset.

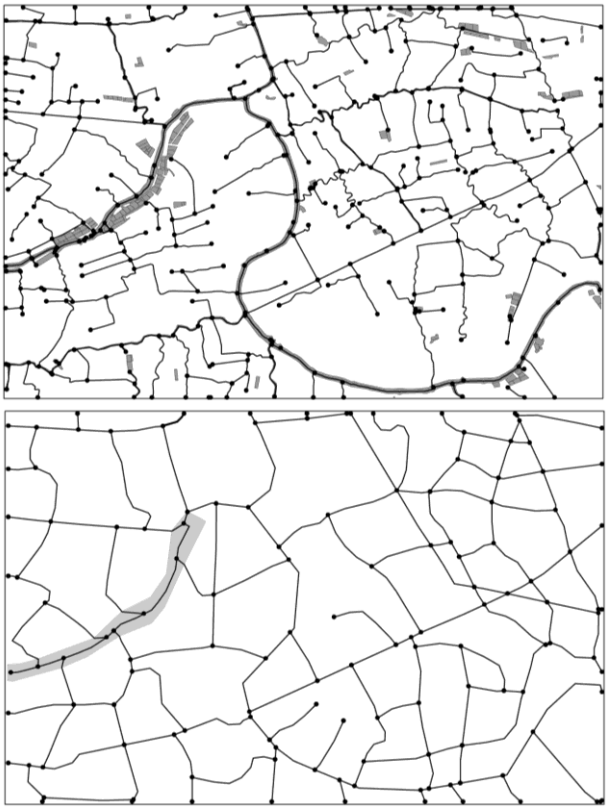


Fig. 10. The original datasets (grey) overlaid by the result of the construction of the topological correct line networks (black): (top) large-scale dataset DLG 50 and (bottom) small-scale dataset DLG 250.

Based on the results of the pre-matching step, we perform the final matching step. First, in the node matching step, we define the threshold $\theta = 60$ degrees in order to decide whether a node match is certain. Secondly, we define $\mu = 12$ degrees in the arc matching step in order to compare the direction of arcs. Figures 11 and 12 show our matching results.

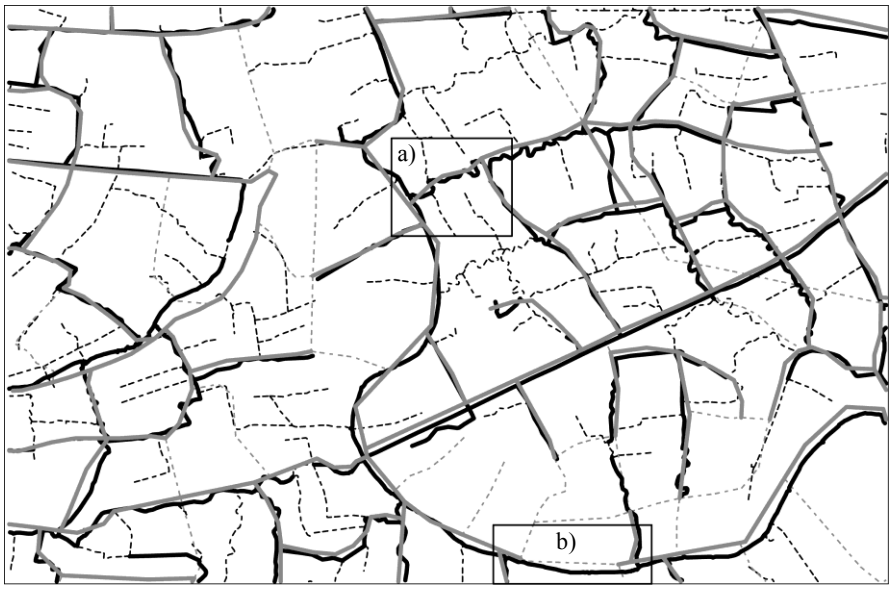


Fig. 11. Matches found by our method for the sample in Figure 10; line networks of the small-scale dataset are displayed dashed in grey and of the large-scale dataset dashed in black; matched arcs are displayed bold.

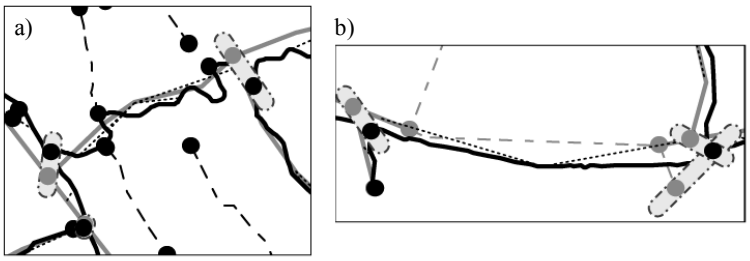


Fig. 12. Two magnified parts of figure 11; a) Relation one to many: the small-scale arc (grey) that was matched to four arcs of the large-scale dataset (black) marked by dotted black lines; b) the small-scale arc displayed in dashed grey is not matched, because none of the endpoints is matched.

We evaluated our matching results by comparing them with results that were obtained manually. In order to decide whether two objects should be manually matched, we compared their location, shape and orientation. Furthermore we took the network topology into account. Table 1 summarizes our results. For each arc match found by our method we assessed whether it is correct or not. We compare the total length of all correctly matched arcs and the total length of all arcs that were incorrectly matched (first line

of Table 1). We see that only a small part of the found matches is wrong (2.3%). Furthermore, for each small-scale arc that was not matched by our method, we assess, whether there is indeed no corresponding large-scale object or whether an existing correspondence was missed. Again, we aggregate our results by summing up the lengths of the involved arcs (second line of Table 3). The human expert was able to match arcs of a total length of 85057.13 m + 8983.76 m = 94040.89 m. Compared to this value, our method failed to find 9.5% of the matches.

Table 1. Statistics of our matching results

	Total Length (m)	Correct (m)	Wrong (m)	Precision
Matched	87020.19	85057.13	963.06	97.7%
Not-matched	33573.85	24590.09	8983.76	73.2%
Sum	120594.04	109647.22	9946.82	90.9%

When both endpoints of an arc were matched, also the arc match is usually correct. However, if there is only one or even no matched endpoint, our method usually matches the arc only with some corresponding parts, but not with all.

We also compared our test results for parts of the river system that were originally represented by lines and parts that were originally represented by polygons. In this case the matching was done based on the derived skeleton lines. In conclusion, we did not observe a difference in the performance for both parts of the river system. Therefore, we assume that our skeletonization method is suited to support the matching method.

5 Conclusion and Outlook to Future Work

We have presented a new method for matching river datasets of different scales. In particular, we have shown how to cope with different geometric dimensions. For this purpose we have proposed a skeletonization method that constructs a topologically correct network of lines. The actual matching of the networks is based on a pre-matching of arcs and nodes and the final matching step. Our method was tested for different Chinese datasets. The results were compared with those obtained by a human expert.

We conclude that our method finds 90.5% of the actual correspondences. Only 2.3% of the matches found are wrong. These numbers are satisfactory, in particular, since large parts of the river system were represented by polygons in the large-scale dataset and by lines in the small-scale dataset. For these parts of the river system, we observe, on the whole, a simi-

lar performance compared to parts that are represented by lines in both scales. Therefore, we assume that our skeletonization method is suitable for matching tasks.

A possibility to improve our method is to also consider semantic attributes that are given for the rivers objects, for example, attributes that reflect the name or expressing whether a river is natural or man-made. Future research should also consider that generalization operators like aggregation and typification influence how objects are represented in different scales. For example, two rivers running parallel to each other may be represented by a single river line.

Acknowledgements

This research is funded by the German Science Foundation (DFG) and the National Nature Science Foundation of China (NSFC): 40620130438. The support is gratefully acknowledged.

References

Devogele, T., Parent, C., and Spaccapietra, S. (1998): On spatial database integration. International Journal of Geographical Information Science 12(4), pp. 335-352.

Diez, Y., Lopez, M.A., and Sellarès, J.A. (2008): Noisy Road Network Matching. T.J. Cova et al. (Eds.): GIScience 2008, LNCS 5266, pp. 38–54, 2008.

Dijkstra, E. W. (1959): A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269-271.

Dunkars, M. (2004): Multiple Representation Databases for Topographic Information. Ph.D. thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.

Harrie, L., and Hellström, A.-K. (1999): A prototype system for propagating updates between cartographic data sets. The Cartographic Journal 36(2), pp. 133-140.

Haunert, J-H., and Sester, M. (2008): Area Collapse and Road Centerlines based on Straight Skeletons. GeoInformatica 12(2), pp. 169-191.

Kieler, B., Sester, M., Wang, H., and Jiang, J. (2007): Semantic Data Integration: Data of Similar and Different Scales. Photogrammetrie Fernerkundung Geoinformation (PFG), vol. 6, pp. 447-457.

Lüscher, P., Burghardt, D., and Weibel, R. (2007): Matching road data of scales with an order of magnitude difference. Proc. XXIII International Cartographic Conference, Moscow, Russia, August 3–10, 2007.

Mustière, S. and Devogele, T. (2008): Matching Networks with Different Levels of Detail. GeoInformatica 12(4), pp. 435-453.

Penninga, F., Verbree, E., Quak, W., and van Oesterom, P. (2005): Construction of the planar partition postal code map based on cadastral registration. GeoInformatica 9(2), pp. 181-204.

Sester, M., Anders, K.-H., and Walter, V. (1998): Linking Objects of Different Spatial Data Sets by Integration and Aggregation. GeoInformatica 2(4), pp. 335-358.

Timpf, S. (1998): Hierarchical Structures in Map Series. Ph.D. thesis, Technical University Vienna, Austria.

Uitermark, H., Vogels, A., and van Oosterom, P. (1999): Semantic and Geometric Aspects of Integrating Road Networks. A. Vckovski, K.E. Brassel, and H.-J. Schek (Eds.): INTEROP'99, LNCS 1580, pp. 177-188, 1999.

Vivid Solutions (2005): RoadMatcher User Guide - RoadMatcher Version 1.4. http://www.vividsolutions.com/products.asp?catg=spaapp&code=roadmatcher (accessed 2009/02/10)

Volz, S. (2005): Data-Driven Matching of Geospatial Schemas. A.G. Cohn and D.M. Mark (Eds.): COSIT 2005, LNCS 3693, pp. 115-132, 2005.

Walter, V. and Fritsch, D. (1999): Matching spatial data sets: a statistical approach. International Journal of Geographical Information Science 13(5), pp. 445-473.

Zhang, M., and Meng, L. (2007): An iterative road-matching approach for the integration of postal data. Computers, Environment and Urban Systems 31(5), pp. 597- 615.