

## Merging Route Data and Cartographic Data

Hanna Stigmar

Lund Institute of Technology, Lund, Sweden  
Swedish National Land Survey, Gävle, Sweden  
hanna.stigmar@lantm.lth.se

**Abstract.** Routing is an interesting application for Location Based Services. In order to provide this application on the National Mapping Agencies' cartographic data the routes must be provided by an external source. The integration of cartographic data and route data is somewhat problematic, why the problems and possible solutions are discussed.

### 1. Introduction

One of the interesting applications for Location Based Services is routing. In order to create a route several attributes of the road network are needed: road names or numbers, address ranges, turn restrictions, physical dividers, speed limits, relative road level, etc. These attributes enable the calculation of the fastest or shortest route.

The EU project GiMoDig (GiMoDig 2004) aims at establishing methods for providing maps to mobile users. The cartographic data used in the project is provided by the involved National Mapping Agencies (NMAs). However, as the NMAs' databases do not include all the attributes needed to create a fastest/shortest route, import and integration of route data is necessary in order to provide a service such as routing on these maps.

The simplest approach to the integration is performed by placing the geometric route data on top of the cartographic data. Ideally, their geometries should match perfectly. However, since the route data is computed from a different data source than the cartographic data, their geometries are different. In order to provide a user friendly presentation, the route geometry needs to be adjusted according to the cartographic data.

The major problem in this type of integration is to find the correct matching object pairs. The level of difficulty in this task depends on the similarities between the different datasets: if they have differing data models or attributes, similar level of detail, systematic geometric differences, if the data in datasets have been collected in a similar way, and if the datasets have relatively similar last dates of update.

The paper describes an ongoing study, i.e. that the merging process has not yet been fully implemented. It starts with a short review of previous work related to line network matching in section 2. Section 3 presents a system architecture for the implementation of the algorithm for real-time map applications. Section 4 describes the problems encountered in this type of conflation, and section 5 discusses how these problems should be solved.

## 2. Related work

The problem of finding the correct matching pairs has been dealt with within different research areas. Within the field of creating a multi-resolution database (MRDB) matching is often used to define links between objects in the different datasets. For example, Dunkars (2003) tries from a small-scale map to find matching objects in a large-scale map. Measures based on topology, geometry, semantics and inter-object relationships are used to compare the potential matching pairs. From the final potential corresponding object, the nearest is chosen. Jones et al. (1996) also use matching in their attempt to create an MRDB. To find the corresponding objects in the databases a weighted parameter matching is performed, where the parameters considered are geometry and attribute.

Walter and Fritch (1999) present a method for matching road networks from different datasets and data models. The matching pairs are found by buffering the referent, and from this buffer finding potential matching elements in the other dataset. For the potential pairs parameters such as angular difference, length, shape, and distance between elements are examined. The matching process is divided into five steps:

1. *Pre-processing* – transformation to eliminate systematic errors, verification of topology, and elimination of redundant nodes.
2. *Computing of potential matching pairs* – a buffer is used to find potential matching pairs.
3. *Use of geometric constraints* – excluding unlikely matching pairs by looking at constraints such as angular difference, length, shape, and distance between elements.
4. *Evaluations of matching pairs* – the potential matching pairs are evaluated using a merit function. The parameters used have been derived by statistical investigations of form, angle, length, position and topology.
5. *Calculation of the final matching* - An algorithm finds the optimal matching solution for the datasets, where the sum of the mutual information of the matching pairs leads to a maximum.

The approach is based on statistical investigations between the two datasets, which means that differences between the datasets have been studied manually in order to find correct values for the involved parameters.

von Goesseln and Sester (2003) use a different approach when matching in order to identify changes between different datasets, to make updates on one of them. The matching is performed by the help of an adjacency matrix showing the relations between the objects in the dataset. Forming clusters, the matrix can show every possible connection in the dataset, which will help to identify the corresponding objects in the other dataset. Following this matching von Goesseln and Sester perform an iterative closest point (ICP) algorithm (first developed by Besl and McKay, 1992) which has been implemented to find the best geometric correspondence between the objects in the two datasets. The original algorithm has been reduced in order to fit a two-dimensional space. In this form it requires the parameters position, scale, and orientation.

The JCS Conflation Suite (JUMP 2004) is an open source Java library containing interactive tools for performing conflation on spatial datasets. It addresses such conflation problems as coverage cleaning, coverage alignment, and road network matching. The road network matching intends to match road edges between two different representations of the same road network. Following this, attributes can be transferred between the matching pair, and missing road sections can be added from one network to the other. The road network matching provides methods for automatic and human-assisted matching. The user can choose whether

to perform a manual match or an automated match followed by a completing manual match. In the manual match the user has the possibility to choose which segments from the different datasets should be matched, split segments to enable better matching, adjust endpoints, exclude segments from the result network etc. In the automatic match a maximum distance is set for the distance between segments to be matched. The matching algorithm starts from the nodes in the referent dataset. Within the maximum matching distance the best matching node from the other datasets is computed based on the distance between the two nodes and the angular difference between the node edges. For the node edges the best match is also saved. The matching is then continued with focus on the edges. By stepping through the matched edges the algorithm is able to find differences between segment length in the two datasets and to split the segments where necessary in order to create more similar geometries in the two datasets. The new segments can then be matched more easily. The result of the conflation process is given as one conflated network with the full coverage of the original datasets.

### **3. System architecture for implementation of merging procedure**

The merge algorithm should be implemented in system architecture for real-time map services. The implementation requires cartographic data and route data in vector format. In theory, the computations could be carried out by the client application, but since the process capability of the client is often low, for instance in the case of mobile devices, the merging should preferably be done before the map is sent to the client.

The study described in this paper is part of the EU project GiMoDig (Geospatial info-mobility service by real-time data integration and generalisation; GiMoDig 2004); which is a project intended to establish methods of distributing cartographic data from core databases at national mapping agencies to mobile devices (mainly following the Open GIS Consortium standards). Figure 1 is a simplified overview of the GiMoDig system architecture (see Lehto (2003) or Sarjakoski and Lehto (2003) for details).

According to the GiMoDig service architecture, the client will access the service on the *Value-Added Service* (VAS) layer. The VAS will acquire route data from a third-party source and include it in the query to be sent to the *GiMoDig Portal Service*. The route data will be transformed (by an XSL Transformation) from the original format to the *OpenGIS Location Services* (OpenGIS 2004) standard (OLS). The Portal Service translates the query and forwards it to the *Data Processing Service* (DPS). The DPS creates a *Web Feature Service* (WFS 2003) query to obtain the cartographic data in the form of a *Geography Markup Language* (GML) dataset (GML 2003). The main task of the DPS is spatial data generalisation by simplification or removal of detailed geometries. After the real-time generalisation, the results are still expressed as GML-encoded spatial data, also including the route as a GML feature. After receiving the generalised dataset from the DPS, the Portal Service translates the data into a map image (e.g *Scalable Vector Graphics*, SVG, image). Finally, the VAS layer will receive the resulting map and return it to the client.

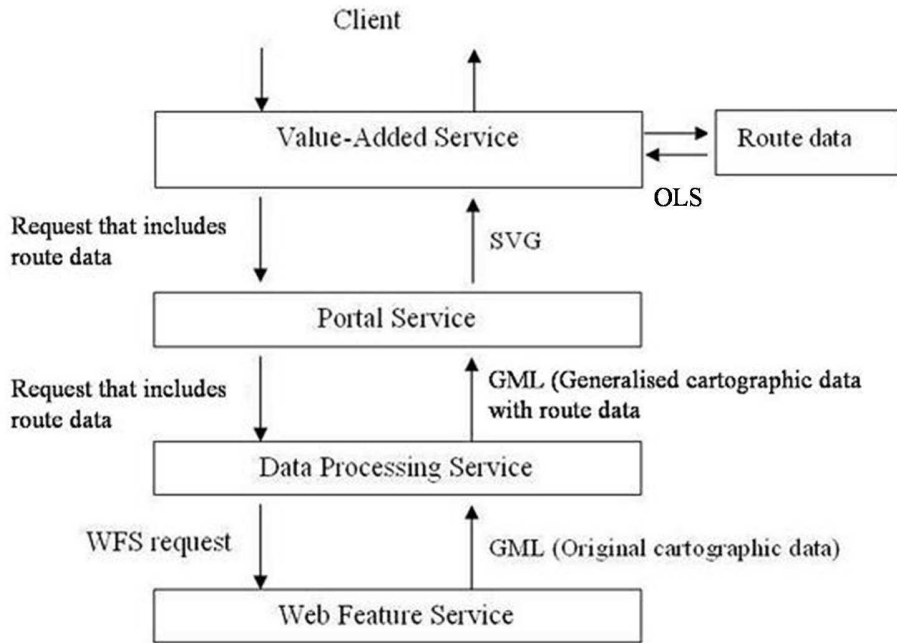


Figure 1. A simplified version of the GiMoDig system architecture for route merging.

#### 4. Problem description

The problems with matching routes to cartographic data in the GiMoDig system are due to the fact that the cartographic data and the route data come from different data sources. The cartographic data comes from the National Mapping Agencies' (NMA's) databases, while the route data comes from an external navigation service provider, which in their turn gets the navigational cartographic data from another data provider. However, the navigational cartographic data can be left out in this discussion. We will only deal with the integration of the (precalculated) geometric route and the NMA's cartographic data. Figure 2 shows an example of a route on top of the cartographic data. It is clear that the geometry of the route does not fully agree with the geometry of the road network.



Figure 2. An example of the geometrical differences between route data and cartographic data from different sources.

Some issues that need to be considered before deciding on a specific integration method are:

- Since the original databases cover large areas and originate from different sources the geometrical differences between the cartographic data and the route data can differ geographically (due to e.g. different collection methods or transformation accuracies). The matching method must be able to handle these potential differences in the integration process.
- Semantics can be a major help when trying to match different geometries, but they will not always be compatible. One example is road names, which will not always be present in one of the datasets, or in either of the datasets. However, if road names are present in both datasets, they will help in the matching process.
- The matching will be performed in real-time. It is therefore of major importance that the process is time efficient.
- The cartographic data will undergo generalisation as well. It is important that the route matches the cartographic data even after the generalisation transformation is performed.

## 5. Methodology

### 5.1 General integration approaches

The first issue to consider is whether the merging process should be performed before or after the generalisation process. The generalisation process simplifies or removes detailed geometries in the road network geometry, in order to make the road network suited for presentation on a small screen. The generalisation can therefore both facilitate and complicate the matching process depending of the nature of the route. By generalising the road network prior to matching some roads that should be matched to might be removed and the matching can not be done correctly. On the other hand, by removing roads of minor importance it might be easier (and faster) to find the correct matching major road. To be able to determine the order for generalisation and matching the nature of the route (i.e. the possible ways a route can be generated) needs to be considered.

The integration itself can be performed in different ways. One major concern is to determine how the route should be represented in the resulting dataset. The route can either be an object of its own or simply an attribute of the road. The representation of the route is especially important when the generalisation process is to follow the matching. As the generalisation process might alter the geometries of the road and the route, it is important to make certain that they share geometries also after this process.

If the route is to be a road attribute, the generalisation will not be a problem. The route will automatically have the same geometry as the road. Here the road object should have some sort of boolean value *isRoute*. This attribute marks whether the road object has a matching route or not. However, the route is not necessarily identical to the road geometry in the sense that the start and end points of the matching segments are identical. The end nodes of the route can be situated somewhere along the matching road segment, and therefore the road object will also need an attribute giving the position of the route node. This position can be given as point coordinates or as a length value of how far along the road segment the route node is situated.

If the route is to be an object there are different possibilities. The route can either be an object with full geometry or an object with limited geometry. If the route has full geometry, i.e. the route is a regular line object, the generalisation process might change the agreement between

the route and road geometries. Another solution is if the route segments have links to the matching road segments. Then, after the generalisation has been performed, the route can be adjusted according to the matching road objects with the help of these links. In this case there is no need to store the full geometry of the route (since it is linked to the road), but rather a start and end point geometry. As in the previous example, the route and the corresponding road object do not necessarily share the same start and end points why the route object need to store its node positions.

The representation of the route is also important considering the amount of information that needs to be stored with it. This is dependent on the purpose of the final result. If the aim is to give the user thorough navigational information including turn descriptions, distances, speed limits, etc, the route should probably be represented as an object. If, on the other hand, the aim is to give the user a simple geometrical description of the route, a limited amount of information needs to be stored and the attribute representation of the route will be enough.

## **5.2 Our integration approach**

The routes used in this project (and presumably from other route providers as well) can be generated as either a fastest or a shortest route. This means that the route can not be presumed to navigate only one or a few different road classes, but all (or most) of the represented road classes. Therefore we can not make the assumption that removal of the minor roads in the road network will not affect the matching process. In our study we have found that the geometrical differences between the route and the road network are relatively small, i.e that there should only be few potential matches for each route segment. We therefore assume that the matching process will be easier the more detailed the geometries of the datasets, and will perform the matching before the generalisation. Doing this it is important that we mark the matched road segments in some way to make certain that they are not removed in the generalisation process.

Since we do not want to limit the potential usage of the final product, we choose to assume that it will present as much navigational information as possible. Therefore the routes will be represented as objects in the resulting dataset. In this way it is possible to keep all navigational attributes necessary. In order to simplify the procedure following the generalisation process we choose to represent the route objects with the limited start and end node geometry, but with links to their corresponding road segments. In this way the route segments will automatically inherit the generalised geometry of their matches.

Figure 3 shows an overview of the matching and generalisation approach to take place in the Data Processing Service (DPS) from figure 1. The route will enter the DPS directly from the Portal Service together with the request for cartographic data. While the cartographic data request is sent to the database the route remains in the DPS waiting for the cartographic data. When the cartographic data is returned from the database the matching and generalisation can take place. The resulting integrated and generalised dataset is sent to the Portal Service as one GML file.

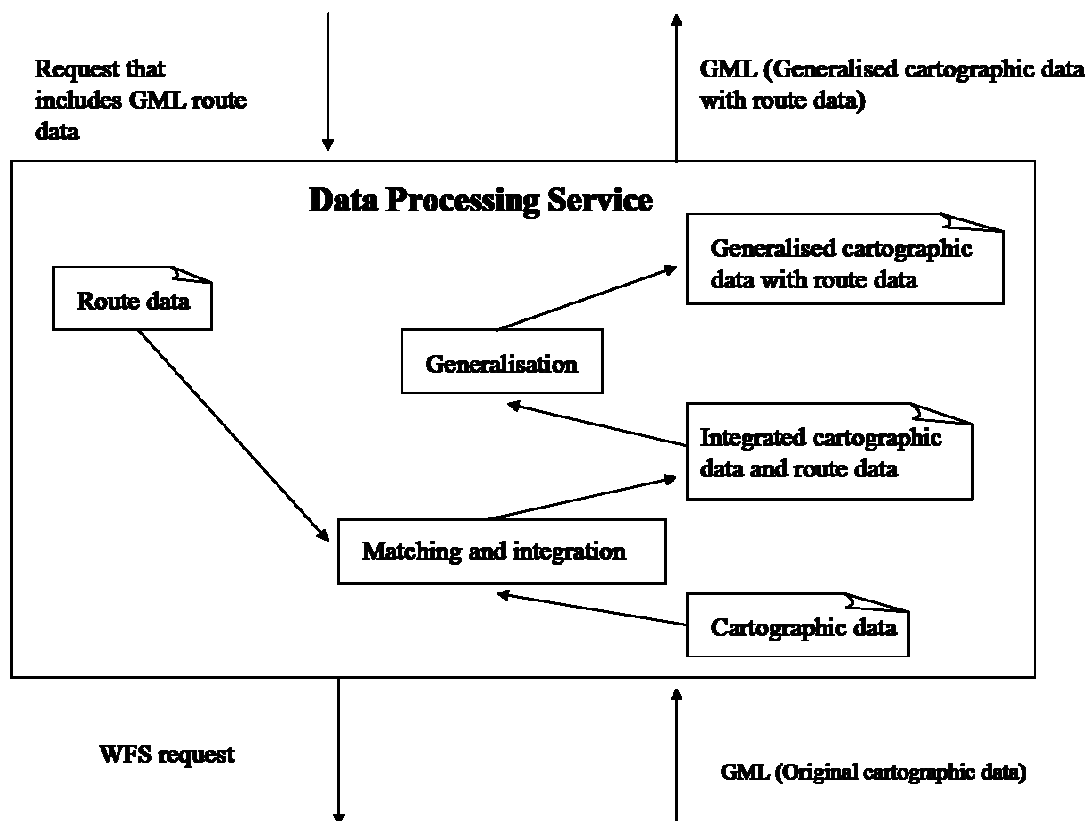


Figure 3. A schematic view of the integration and generalisation processes from the Data Processing Service.

### 5.3 Matching

As a first step in the matching process, the cartographic data must be filtered, to ensure that the matching process is only performed on relevant objects. This filtering can be performed by a semantic processing step which selects only the road objects. Since the semantic attributes, e.g. road names, may be somewhat inconsistent in the datasets the matching must be performed based on the geometries of the objects. However, the semantic attributes can be used as a help later in the process if more than one potential objects are found.

For some datasets a systematic geometrical difference may be present. In order to find this geometrical difference the datasets need to be compared and evaluated beforehand. E.g. Walter and Fritch (1999) and von Gösseln (2003) use a local transformation to find a better geometric correspondence between the objects in the datasets. By studying the geometrical differences between the datasets, transformation parameters will be found that can later be used in the process (if necessary).

Our main matching problem is due to the different representation of segments in the two datasets. Not only have the two datasets different geometrical appearance due to different collection methods and level of generalisation. The subdivision of roads into segments also differs. The segments in the cartographic road network are limited by the intersections in the road network. The route segments, on the other hand, are limited by the intersections where a turn is made, or anywhere the road changes its name or another attribute. Since the matching is to take place in a real-time environment it is important that the process functions automatically. The reviewed matching methods all rely on manual intervention to some degree and therefore need to be further implemented.

The JCS Conflation Suite (JCS) and JUMP provide a good environment for working with the matching algorithm. It is here possible to integrate the user interface with the Java code while implementing. Figure 4 shows an example of the user interface JUMP Workbench with a road network dataset and a route dataset.

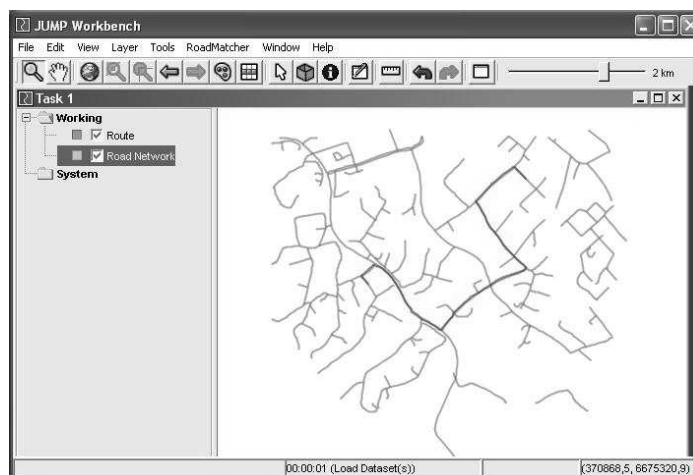


Figure 4. The JUMP Workbench user interface with road network data and route data.

Since a road network conflation process already exists in JCS it will be advantageous to use the existing code. JCS also has the advantage (for us) of splitting segments where necessary in order to enable matching between the two datasets. However, using the existing code in its current state is not sufficient. As JCS starts the matching by finding matching nodes in the two datasets, not enough nodes will be found due to the datasets' different segment representations. The current JCS algorithm must therefore be extended in order to fill our needs.

In our study we have decided to test whether it is possible to implement a satisfactory process by extending the JCS road network conflation algorithm with iterative functionality where the matching will continue until there are no more segments to be matched, or if another type of algorithm is needed. The article by Walter and Fritch (1999), described in section 2, describes the process of automatic matching between GDF data from TeleAtlas and the German topological database. Since the route data in our project also originates from TeleAtlas GDF data and the cartographic data from the NMAs' databases, this method is likely to be adaptable to our project as well. As a first test of an iterative solution we will therefore try the buffer idea of Walter and Fritch, where a buffer is created for the object to be matched and is then step-wise enlarged until a potential matching object is found

In the following process steps different constraints need to be used to determine a final matching pair. In order to determine which constraints to use and their weighting, the datasets and the outcome of the first matching attempts need to be thoroughly examined.

### **Acknowledgements**

The research described in this paper is part of the GiMoDig project, IST-2000-30090, which is funded from the European Union via the Information Society Technologies (IST) programme (GiMoDig 2004). We would like to thank our colleagues in the GiMoDig-project.

## References

- Besl, P. and McKay, N., 1992. *A Method for Registration of 3-D Shapes*, Trans. PAMI, Vol. 14, No. 2, pp. 239-256.
- Dunkars, M., 2003. Matching of Datasets. In: *Proceedings of the 9th Scandinavian Research Conference on Geographical Information Science*, Espoo, Finland.
- GiMoDig, 2004. *Geospatial info-mobility service by real-time data-integration and generalization*, <http://gimodig.fgi.fi/> (accessed 2004-05-24).
- GML, 2003. *Geographic Markup Language*, <http://www.opengis.org/techno/documents/02-023r4.pdf> (accessed 2003-09-12).
- Jones, C. B., Kidner, D. B., Luo, L. Q., Bundy, G. L., and Ware, J. M., 1996. Database Design for Multi-Scale Spatial Information System. *International Journal of Geographical Information Systems*, Vol. 10, No. 8, pp. 901-920.
- JUMP, 2004. *The JUMP Project*, <http://www.jump-project.org> (accessed 2004-05-24).
- Lehto, L., 2003. *GiMoDig system architecture*, Available at <http://gimodig.fgi.fi/deliverables.php> (accessed 2003-09-10).
- OpenGIS, 2004. OpenGIS Consortium, <http://www.opengis.org> (accessed 2004-05-24).
- Peuquet, D. J., 1992. An algorithm for Calculating Minimum Euclidean Distance Between Two Geographic Features. *Computers & Geosciences*, Vol. 18, No. 8, pp. 989-1001.
- Sarjakoski, T., and Lehto, L., 2003, Mobile Map Services Based on an Open System Architecture. In *Proceedings of the 21st International Cartographic Conference (ICC)*, 10 - 16 August 2003, Durban, South Africa, pp.1107-1113.
- von Goesseln, G. and Sester, M., 2003. *Change Detection and Integration of Topographic Updates from ATKIS to Geoscientific Data Sets*.
- Walter, V. and Fritsch, D., 1999. Matching Spatial Data Sets: A Statistical Approach, *International Journal of Geographical Information Science*, Vol. 13, No. 5, pp. 445-473.
- WFS, 2003. *Web Feature Service Implementation Specification*, <http://www.opengis.org/techno/specs/02-058.pdf> (accessed 2003-09-10).