Mohammed Alhessi

Development and Implementation of the OpenLR Map Interface for Shapefiles

Duration of the Thesis: 6 months
Completion: May 2013
Tutor: Rainer Schütze, MSc
Examiner: Prof. Dr.-Ing. Volker Schwieger

Background

One of the goals of the transport Telematics is to reduce the traffic jams. That requires providing the driver with information about the current status of the traffic, so that the congested roads and road hazards can be avoided. The information is associated with a location on the network. To transmit the location with the associated information, the location is first identified in the sender map and encoded, then transmitted to the terminal destination, and finally decoded in the receiver map and presented to the user. The process of encoding and decoding the location is called Location Referencing. The main challenge in the location referencing workflows is the differences and deviations between the maps on both sides of communication. Many methods are developed to deal with this challenge, notably AGORA-C and OpenLR.

OpenLR method is the subject of the thesis. It is developed by TomTom and implemented using Java environment. The input to OpenLR encoder or decoder should be a map with specific requirements so that OpenLR method can work as intended. The requirements are: (1) the input map should be navigable consisting of lines and nodes, and (2) it should contain the following data: coordinates in WGS84, line (road) lengths in meter, valid geometry, Function Road Class (FRC) and Form of Way (FOW) of the lines. The map which satisfies the previous requirements is called OpenLR Map.

TomTom implemented OpenLR to deal with its maps in certain formats, such as SQLite format. However, it provides the possibility to use maps with other formats (e.g. shapefile) by providing a number of Java interfaces. The interfaces are: (1) OpenLR Map (OM) interface which is a collective term for other three interfaces (MapDatabase, Line and Node), and (2) OpenLR Map Loader (OML) interface. If OM interface is implemented for a certain format, then the internal data structure of that format is translated into one which satisfies the
the aforementioned map requirements, and hence it can be used as an input to OpenLR encoder or decoder. Furthermore, if OML interface is implemented, it would be possible to load the shapefile map into OpenLR applications. One of those applications is Map Viewer which is also developed by TomTom for viewing the maps and encoding and decoding the locations.

The main objective of the thesis is to use the maps with shapefile format as an input to OpenLR encoder or decoder, and hence OM and OML interfaces should be implemented. In addition, some of auxiliary tools are developed to complete the workflow of the adopted methodology. The maps used in the thesis are TomTom and OpenStreetMap (OSM) maps in shapefile format.

The benefits of using shapefiles in the context of location referencing are: (1) The shapefile is widely-used format (2) Integrating the GIS functions provided by ArcGIS with the location referencing workflows, and (3) Some formats (e.g. OSM-XML ) requires a lot of time for parsing XML data.

Methodology

1. Implementation of OM interface

Using shapefiles as input to OpenLR encoder or decoder requires translating it into OpenLR Map. The differences between the shapefile and OpenLR Map are shown in the following table:

<table>
<thead>
<tr>
<th>Shapefile map</th>
<th>OpenLR Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapefile contains only geometrical and attribute information</td>
<td>OM should contain geometrical, topological and attribute information</td>
</tr>
<tr>
<td>It contains many attributes (e.g. road name, width, FRC, pavement...etc)</td>
<td>It should contain only few certain attributes (e.g. FRC,FOW,length, valid geometry, coordinates in WGS84)</td>
</tr>
<tr>
<td>FRC and FOW values may exceed the OpenLR values</td>
<td>FRC and FOW values should be limited to OpenLR values (i.e. 8 values)</td>
</tr>
</tbody>
</table>

The translation process can be accomplished by implementing OM interface. Based on the above table, a general approach is developed for any shapefile. That approach is then applied for TomTom and OSM maps. The general overview is shown in the following figure:
The general workflow can be divided into three general steps: (A) reading the shapefile, (B) creating the OpenLR lines and nodes, and (C) building OpenLR Map.

**Step-A** is accessing the shapefile and reading its features. This can be technically accomplished by using the third-party library GeoTools. The shapefile should also be validated in this step so that it stores a road network (polyline features) and contains the necessary data (e.g. FRC, FOW…etc).

**Step-B** is creating OpenLR lines and nodes from the corresponding shapefile features. This step is divided into 4 steps as shown in Figure 2.

![Figure 1: General Conceptual view of translation process](image)
Step-B1, Step-B2 and Step-B3 are preliminary steps for Step-B4. They prepare the data necessary for creating OpenLR lines and nodes.

In step-B1, the features are validated such that each feature represents a valid road. For example, OSM shapefile map contains some features with null geometry which implies invalid road. In addition, TomTom and OSM maps contain features which are assigned only for pedestrians; those roads are invalid for car navigation and hence are invalid for the application of OpenLR. All invalid features should be excluded.

In Step-B2, the data (e.g. FRC,FOW…etc) required for creating OpenLR Map should be extracted. This step also includes mapping the attributes in the shapefile into the corresponding OpenLR values. For example, TomTom shapefile map contains 22 values for FOW attribute while OpenLR Map should contain only 8 values. That requires mapping the 22 values into the 8 values of OpenLR. The mapping process should be accomplished carefully because it affects in some way the encoding and decoding process.

In Step-B3, the features should be split at nodes where there are real road intersections or the attributes of the road change. This step is necessary to ensure that the map is navigable. This step is required for OSM maps but not for TomTom ones which are originally prepared for navigation purposes. OSM map models the road junctions by a vertex. Therefore, the OSM features should be split at shared vertices. This step can be accomplished internally (within the implementation code) or externally by an external tool. In this thesis, it was implemented...
by an external tool which has the advantage of splitting features only once, and not each time when the map is loaded in OpenLR applications.

In Step-B4, the OpenLR lines and nodes are created by utilizing the data prepared in the previous steps (cf. Figure 2).

If the feature represents a two-way road and it is to be split into 4 features, then the number of lines to be created is 8 lines. OpenLR requires creating a line per direction. In addition, two nodes (start and end) should be created for each line. The node can be created by identifying its longitude and latitude values and ID. On the other hand, the line can be created from the corresponding split geometry (cf. Figure 2) and some other attributes such as FRC,FOW, name, length and ID.

Technically speaking, a Java class is created to represent the line and it should implement Line interface. Likewise, another class is created to represent the node and it should implement Node interface. The implementation can be accomplished easily because the data necessary for implementing the methods defined on Line and Node interfaces are prepared in the previous steps.

**Step-C** is building the OpenLR Map from the previously created lines and nodes. The lines and nodes should be stored in a hash map data structure which is very efficient in storing and retrieving the data. Technically speaking, a Java class is created to represent the OpenLR Map and hold its lines and nodes. This class should implement MapDatabase interface. By doing that, we can say that the shapefile is translated into OpenLR Map and the OM interface is implemented.

### 2. Implementation of OML interface

The OML should be implemented if the shapefile needs to be loaded into OpenLR applications, such as Map Viewer. The general steps of implementing OML are shown in the following figure:
The first two processes are managed by OML. Implementing OML includes the implementation of OM which is described in the previous section. Technically speaking, a Java class is created to represent the OpenLR map loader and it should implement OpenLRMapLoader (OML) interface.

3. Developing auxiliary tools

A number of auxiliary tools are developed to assist in completing the whole methodology. The tools are:

1) OSM-XML into Shapefile converter: this tool is developed as an ArcGIS model tool to convert OSM map from OSM-XML format into shapefile format.

2) Location Viewer Tool: this tool is developed as a python script to visualize the encoded or decoded locations in ArcMap.

3) Splitter tool: this tool is developed as an ArcGIS add-in to split features at shared vertices.

Testing and Evaluation

As mentioned previously, the main objective is to use shapefile as input to OpenLR encoder or decoder. Therefore, the testing procedure first includes loading the shapefile in Map Viewer and secondly testing the encoding and decoding functions on the loaded shapefile.

The loading process is tested for OSM and TomTom shapefiles. Figure 4 and Figure 6 proves the successful loading of OSM and TomTom maps in Map Viewer. The loading process also includes translating the internal data structure of shapefile into OpenLR Map. To test the translation process and hence the attributes mapping between the original shapefile and the loaded one, the attributes for a road in both maps are compared as shown in Figure 4 and Figure 5. The results prove the success of the translation process.
The attributes are also compared between the TomTom and OSM maps after loading. It is found that for the location shown in Figure 4 and Figure 6:

- In OSM map, the line is represented as a single segment but represented as two segments in TomTom map. That marks one of the differences encountered in maps from different vendors.

- The line is classified as a single carriageway road with FRC-3 in OSM map but classified as a multiple carriageway road with FRC-4 in TomTom map. That can be referred to two reasons: (1) the different interpretation of map vendors and (2) the mapping process made for the OSM map.

Despite the differences in TomTom and OSM maps, the selected location in the following figures is encoded on TomTom map and decoded successfully on OSM as shown in Figure 7. That proves that the main objective is achieved by using the shapfile map for encoding and decoding functions.

![Figure 4: Loading the OSM shapefile in Map Viewer](image-url)
Figure 5: The original OSM shapefile as it appears in ArcMap

Figure 6: Loading TomTom map in Map Viewer
Conclusion

- The main objective has been achieved
- The approach developed has shown that it can be applicable for different datasets
- The underlying algorithms has shown a good performance
- The auxiliary tools have facilitated the workflow
- It is now possible to integrate GIS functions with the location referencing workflows

Recommendations

- It is recommended to transfer the whole OpenLR APIs to work within ArcGIS environment. That is possible by the employing ArcGIS add-ins.
- Mapping the attributes into OpenLR values require inspecting each map individually. It is recommended to automate this process by providing extra GUIs.
- Future works are recommended to investigate the possibility of incorporating the ArcGIS network analyst into the workflow.