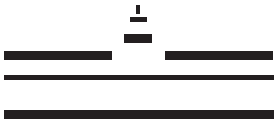

Westfälische
Wilhelms-Universität
Münster

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

External Semester

Final Report

submitted by:

Marius Runde

Matriculation Number: 382204

Course: M.Sc. Geoinformatics

First Supervisor:

Prof. Christian Kray

Second Supervisor:

Michael Scholz

Braunschweig, February 14, 2017

1 Introduction

As part of my master studies in Geoinformatics I did my external semester at the German Aerospace Center (DLR) in Braunschweig. I was working from the 1st September 2016 to the 14th February 2017 at the Institute of Transportation Systems. The weekly working hours were 39 hours. However, we had flexible working hours at the institute. Therefore, I was allowed to work longer or take a day off as long as I met the total number of working hours and fulfilled my work. As I had no vacation days, this was a welcoming option.

My supervisor from the Institute for Geoinformatics (ifgi) was Prof. Christian Kray to whom I reported on a two- to four-weekly period about my work.

At the DLR my supervisor was Michael Scholz. He had also studied Geoinformatics at the ifgi and now works at the DLR in the group for Spatial Data Processing and Engineering (GDV). Mr. Scholz introduced me to my work and also mentored and supervised me during the five and a half months at the DLR. Before I started my external semester in September 2016, I met with him in May 2016 to discuss the content of the learning agreement, which is described in the following chapter.

1.1 Institute of Transportation Systems

The Institute of Transportation Systems[1] is part of the DLR. It has two locations in Braunschweig and Berlin from which the one in Braunschweig is the greater one with more employees and the driving simulators etc. The task of the institute is the research and development of automotive systems, railway systems, traffic management, and public transport.

I was working in the GDV group and the project "Virtuelle Welt" (see section 2.1). This work was located in the area of automotive systems. And while I was also working with driving simulations and the Road2Simulation data format, most of my work concentrated on the development and implementations with the OpenDRIVE standard (see chapter 3).

[1] `http://www.dlr.de/ts/en/`

2 Learning Agreement

Next to the formal information about the involved persons and the time frame of the external semester, the learning agreement summarized the planned work, the project description in which I was involved, and the expected goals. The proposed topic was the development of geodata processing algorithms for driving and traffic simulations (*"Entwicklung von Algorithmen zur Geodatenverarbeitung für die Fahr- und Verkehrssimulation"*). The work was located in the project "Virtuelle Welt" (Virtual World), which is described in the next section.

2.1 Project "Virtuelle Welt"

The automotive branch of the Institute of Transportation Systems develops advanced driver assistance systems (ADAS) which are becoming increasingly important regarding traffic safety and driving behaviours. Such systems can be tested in driving simulations before being rolled out in the real world. The Institute of Transportation Systems has numerous simulators for various scenarios. Two of these simulators can be seen in the figures 2.1 and 2.2.

Figure 2.1: Dynamic Driving Simulator

Figure 2.2: Virtual Reality Lab

For such tests a close to reality environment is mandatory. The project "Virtuelle Welt" (Virtual World) develops such 3D models. They are based on spatial data and consist - besides models of the landscape and buildings - also of a detailed description of the road network. A great challenge hereby is to include logical information to the road network for the simulations.

Based on the previous projects SimWorld[2] and SimWorldURBAN a toolchain automatically generates digital landscapes and logical descriptions for the driving and traffic simulations. This toolchain and the data schema shall be extended to serve different stake holders. The result will be a digital atlas with multimodal metropolitan regions and a toolchain for generating virtual worlds automatically. The demo region is the city of Braunschweig. Figure 2.3 shows a screenshot of the 3D model of the research junction in Braunschweig.

Figure 2.3: Screenshot of the 3D Model from the "Virtuelle Welt" Project

[2] http://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-5462/10466_read-23036/

3 OpenDRIVE

OpenDRIVE is an open industry standard. It can be used by anybody free of charge and its specification [5] is publicly available. There are numerous users and contributors from research and industry. OpenDRIVE is considered as the de facto standard for logical road description [2].

In contrast to GIS formats OpenDRIVE is more complex. It describes the world mathematically and uses a track coordinate system for the road description and the features along it. This is visualized in the figures 3.1 and 3.2. The file format of OpenDRIVE is based on XML.

Various features can be stored, e.g. junctions, roads, signals, or connectors. These are all logically connected. Junctions contain roads, roads have signals and so on. In driving simulations precise maps are necessary. OpenDRIVE can model roads extremely precisely. Each lane can have its own geometry and signals like traffic lights or holding lines are valid for specified lanes only.

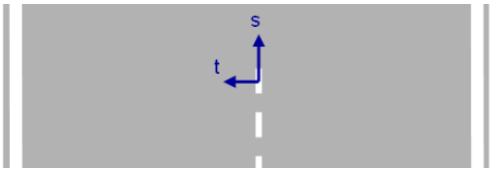

Figure 3.1: (s,t)-Coordinates

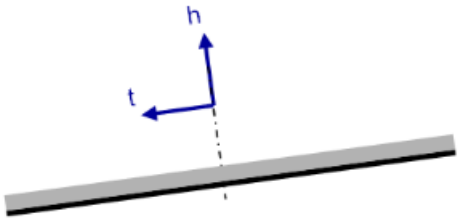

Figure 3.2: t-Coordinate and Height

4 My Contribution

During my external semester I worked on numerous tasks of the project "Virtuelle Welt". All these tasks included the development and implementation of software modules. The preferred programming language was Java. However, for some of the implementations I also used JavaScript and Python. Further details about the used technologies are described in the specific sections. It is difficult to say how much time I spent exactly on each task as I was very often working on multiple tasks in parallel. Nevertheless, the development and implementation of the signal connector for OpenDRIVE road networks was the major task (see section 4.2).

After my computer was set up in the first week and I got some information about the DLR and its institute, I read up on the OpenDRIVE specification [5]. Then I could start with my work which is described in more detail in the following sections.

4.1 Update of the OpenDRIVE Persistence Library

It is of vital importance to make tests reproducible. Therefore, the OpenDRIVE road networks are stored in a database to be reused. Storing these data is done by the OpenDRIVE persistence library. The software library is implemented in Java and uses the JPA Hibernate[3] and JAXB[4] frameworks. The database the data is stored in is a PostgreSQL database.

As the OpenDRIVE specification [5] gets regular updates, so does the software library. It was my task to update the implementation to the newest version (1.4).

Furthermore, I also extended the existing software library by the possibility to store spatial data. My supervisor Mr. Scholz already installed the PostGIS[5] extension in the database for this purpose. Then I attached the OpenDRIVE entries, which are spatially referenced, with a PostGIS geometry attribute. The calculations for the transformation from the track coordinate system to the real world coordinate system were based on the work of Orozco [4].

With the PostGIS extension it was then possible to load the geometries of OpenDRIVE road networks into other spatial applications, too. Even if they did not support the OpenDRIVE standard. Figure 4.1 shows a screenshot of Braunschweig's city ring road loaded into QGIS[6]. Next to it, figure 4.2 shows the city center of Braunschweig in

[3] `http://hibernate.org/orm/`
[4] `https://jaxb.java.net/`
[5] `http://www.postgis.net/`
[6] `https://www.qgis.org/en/site/`

OpenStreetMap[7] to give an impression of the location and environment of the city ring road.

Though it is not directly possible to work with the other OpenDRIVE elements this way, nevertheless it is a helpful way to get an impression of the stored data. Furthermore, it can be also used to make spatial selections, as described in more detail in section 4.3.2.

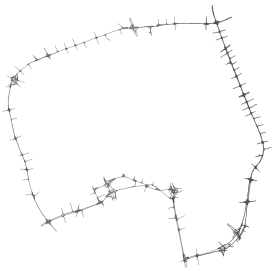

Figure 4.1: Screenshot of Braunschweig's city ring road loaded in QGIS

Figure 4.2: Screenshot of Braunschweig in OpenStreetMap

4.2 Signal Connector for OpenDRIVE Road Networks

The existing OpenDRIVE road networks did not contain signal information. However, this information is crucial for driving and traffic simulations. Therefore, my main objective during the external semester was to develop and implement a software module (signal connector) to include and connect signals from heterogeneous data sources to OpenDRIVE road networks. The signals I was working with first came from cadastral data in form of shapefiles and in another scenario as Road2Simulation [1] data.

The signal connector operates in the following order:

1. Read OpenDRIVE road network and extract the roads' geometries
2. Read signals and objects and transform them into a common data format
3. Iterate through the signals and find a matching road they can be added to
4. Create dependencies for dynamic signals
5. Create controllers for dynamic signals
6. Add objects to the nearest road

[7] `http://www.openstreetmap.org/`

In the first step the OpenDRIVE road network is loaded from a local file or the database. To make it easier to find the nearest road in a later step, all the roads' geometries are extracted. The list of extracted geometries can then be attached with a spatial index to increase the performance of spatial operations on it.

After the OpenDRIVE road network has been loaded, the signals are loaded. As the signals can come from different sources, I developed algorithms to read signals from many kinds of data formats. The signals are then transformed into a data format which I developed for the signal connector. The format is already close to the OpenDRIVE standard but contains some more data about parent objects. Adding new objects to the OpenDRIVE road network (e.g. traffic light poles) is a very similar task as adding signals to it. Therefore, the signal connector can also load other objects from the given data sources.

When all the required data is loaded by the signal connector, a search algorithm iterates for each signal through all roads within a given distance. This already excludes signals with a greater distance and improves the performance as the list of roads to work with stays relatively small for each signal. From the work of Kopf et al. [3] I got the idea of using a radial visibility sweep algorithm to find the nearest roads for a signal. So the algorithm first filters the roads by their distance and then by their visibility from the signal. If other roads are in between a road and a signal, the road further away will be omitted. Then the signal connector calculates the headings of the filtered roads and the signal. If any road's heading matches the signal's heading and also has valid lane types (usually of the type *driving*), the signal will be added to the road. This requires a transformation of the real world coordinates of the signal into the track coordinate system of the OpenDRIVE road network. If no road matches with the signal, then next roads which were omitted by the sweep algorithm will be compared to it. Signals which do not match with any of the roads must be skipped by the signal connector. This can happen for example when there is an error in the data source of the signals.

As the OpenDRIVE specification [5] highly relies on the logical connections of its elements, it is important to connect the signals with other elements, too. Especially traffic simulations require information about connections between the signals. The signal connector creates dependencies between traffic lights and holding lines. This enables vehicles in traffic simulations to know whether they have to stop at a holding line when the corresponding traffic light shows red or whether they can continue their drive.

Another aspect which is important for traffic and driving simulations is the synchronicity of the traffic lights. In OpenDRIVE this can be easily managed as dynamic signals like traffic lights are included in controllers. These controllers can handle traffic lights with the same or opposite directions and are created by the signal connector in the next to last step.

Finally the objects are added to their nearest road which works very similar as the way the signals are added to them. However, this algorithm works a bit simpler as the heading is of no importance here and objects do not have to be connected with any other elements than their parent road. After the signal connector have finished, the result can be stored in the database to be used in other applications. Figure 4.3 shows a screenshot of the research junction in Braunschweig's city road ring with connected signals.

4.2.1 Preparing OpenDRIVE Road Networks for Driving Simulations

When the implementation of the signal connector was mostly finished, I started working together with a colleague from the driving simulation group to export a demo route for the AAET 2017[8]. By preparing the demo route for a driving simulation I was able to test my signal connector. It was also challenging to communicate errors in the algorithms with my colleague who used another data format for the driving simulation. Nevertheless, this was a great success and showed that my algorithms worked. Table 4.1 shows the complexity of the OpenDRIVE data format. Though the test route had a length of only 3.6 km, it contained hundreds of road entries with signals, controllers, etc.

Length	3.6 km
Junctions	24 entries
Roads	278 entries
Signals	298 entries
Controllers	163 entries

Table 4.1: OpenDRIVE Demo Route

4.3 RESTful Web Services to Export and Filter OpenDRIVE Road Networks

As the OpenDRIVE road networks are also used by others, a function to export the data out of the database is helpful. Therefore, I developed and implemented some RESTful web services to do this. The web services are implemented in Java and also use the JPA Hibernate and GeoTools frameworks. I developed the following functions:

- Exporting complete OpenDRIVE road networks
- Exporting filtered OpenDRIVE road networks, which are filtered by...
  - A bounding box
  - A selection of roads

The OpenDRIVE standard does not require a geometry attribute for all kinds of elements. For example, junctions are only logically connected with roads which in turn have a geometry attribute. So it is only possible to filter elements like junctions spatially by filtering the dataset logically via other elements, too. Filtering an OpenDRIVE road network by a selection of roads works in a similar way except that the roads are already

[8] `http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-3930/20125_read-47764/`

specified and do not have to be filtered via the bounding box. However, in both cases it is important that only the elements are exported which are included by the filter. Roads are linked to their predecessor and successor roads and junctions have entries about all roads which meet or are contained by the junction. However, when a preceeding road for example does not meet the filter requirements, it shall neither be in the exported OpenDRIVE road network as an element nor be referenced to by any other element. Therefore, the filter algorithm also has to remove such null pointer references of which there are many more than the here mentioned ones. Otherwise the exported dataset would have incorrect entries which may cause errors in other applications.

I tested the implementation successfully on an Apache Tomcat[9] and a GlassFish[10] server, running on my local machine. The web services were already used in production for the preparation of the demo route, described in section 4.2.1.

4.3.1 Web Frontend

To enable an easier usage of the web services instead of requesting the datasets via REST requests programmatically, I also developed and implemented a web frontend for it. The frontend is implemented in JavaScript and uses Bootstrap[11] for the design and Leaflet[12] for the map representation. A WMS layer shows the OpenDRIVE road networks from the database so that the user can easily select the desired area with a bounding box. Figure 4.4 shows a screenshot of the web frontend.

4.3.2 QGIS Plugin

In addition to the web frontend, I also developed and implemented a QGIS plugin to use the web services. The plugin is implemented in Python with the PyQt[13] library. It opens a dialog (shown in figure 4.5), which lets the user enter the OpenDRIVE database ID (*OpenDRIVE-ID*) and either a bounding box or a selection of roads. The values for the bounding box must be entered manually. However, the road selection can also be loaded into the plugin via standard selection functions in QGIS. The user only has to load the OpenDRIVE road network into QGIS via a PostGIS layer and can then just select the roads he or she wants to export. Alternatively, it is also possible to enter the roads' IDs manually. The filtered OpenDRIVE road network will then be exported into an .xodr file which is the file type of the OpenDRIVE standard.

[9] `http://tomcat.apache.org/`
[10] `https://glassfish.java.net/`
[11] `http://getbootstrap.com/`
[12] `http://leafletjs.com/`
[13] `https://sourceforge.net/projects/pyqt/`

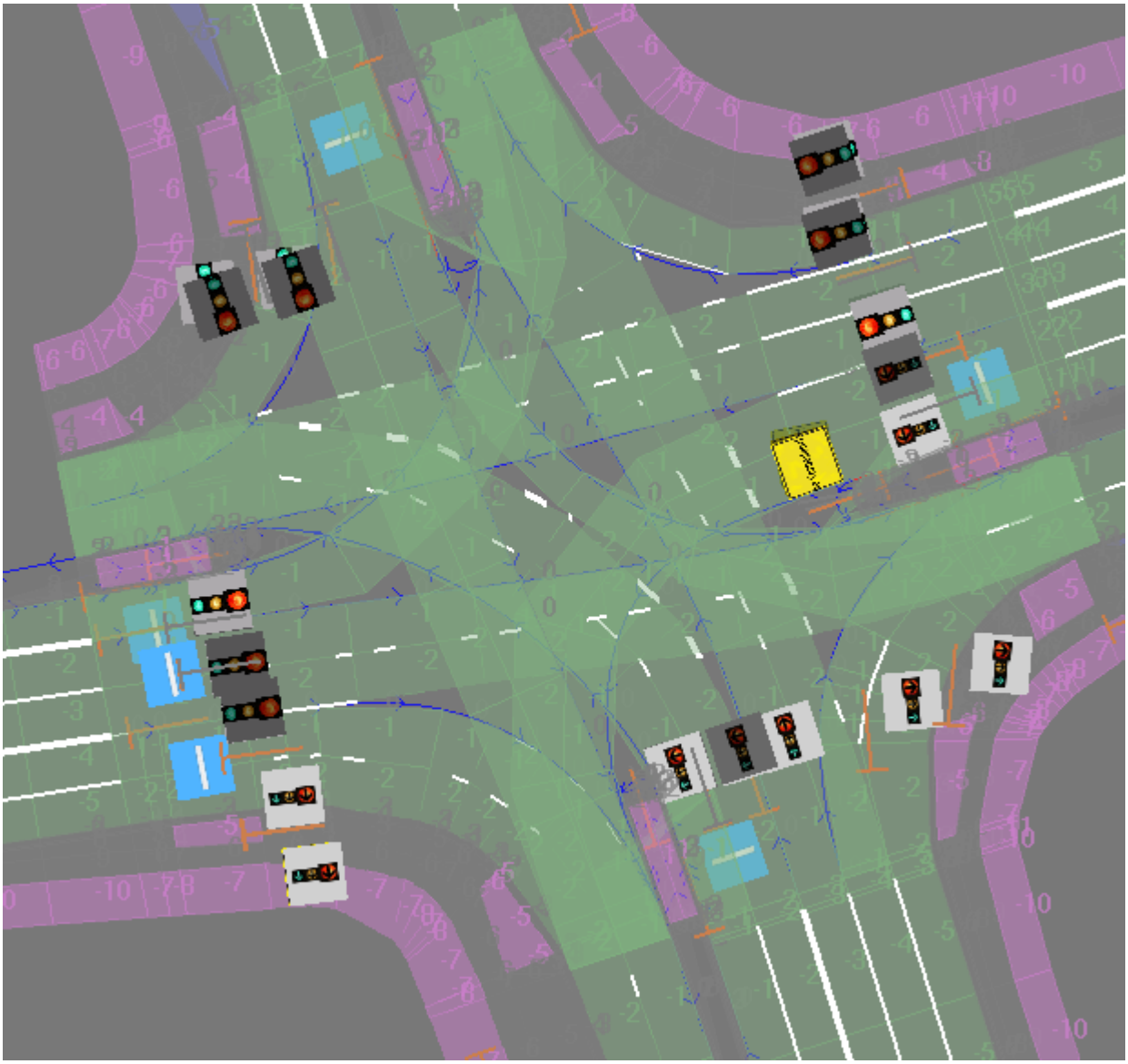

Figure 4.3: Screenshot of an OpenDRIVE road network with connected signals

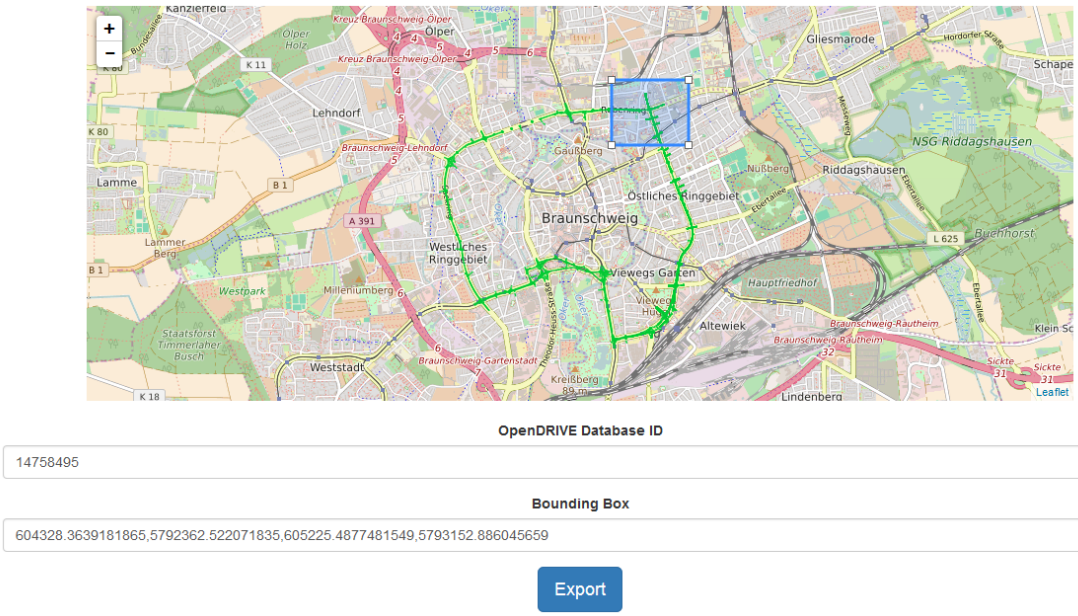

Figure 4.4: Screenshot of the Web Frontend for Exporting OpenDRIVE Road Networks(WMS coloured in green, editable bounding box in blue)

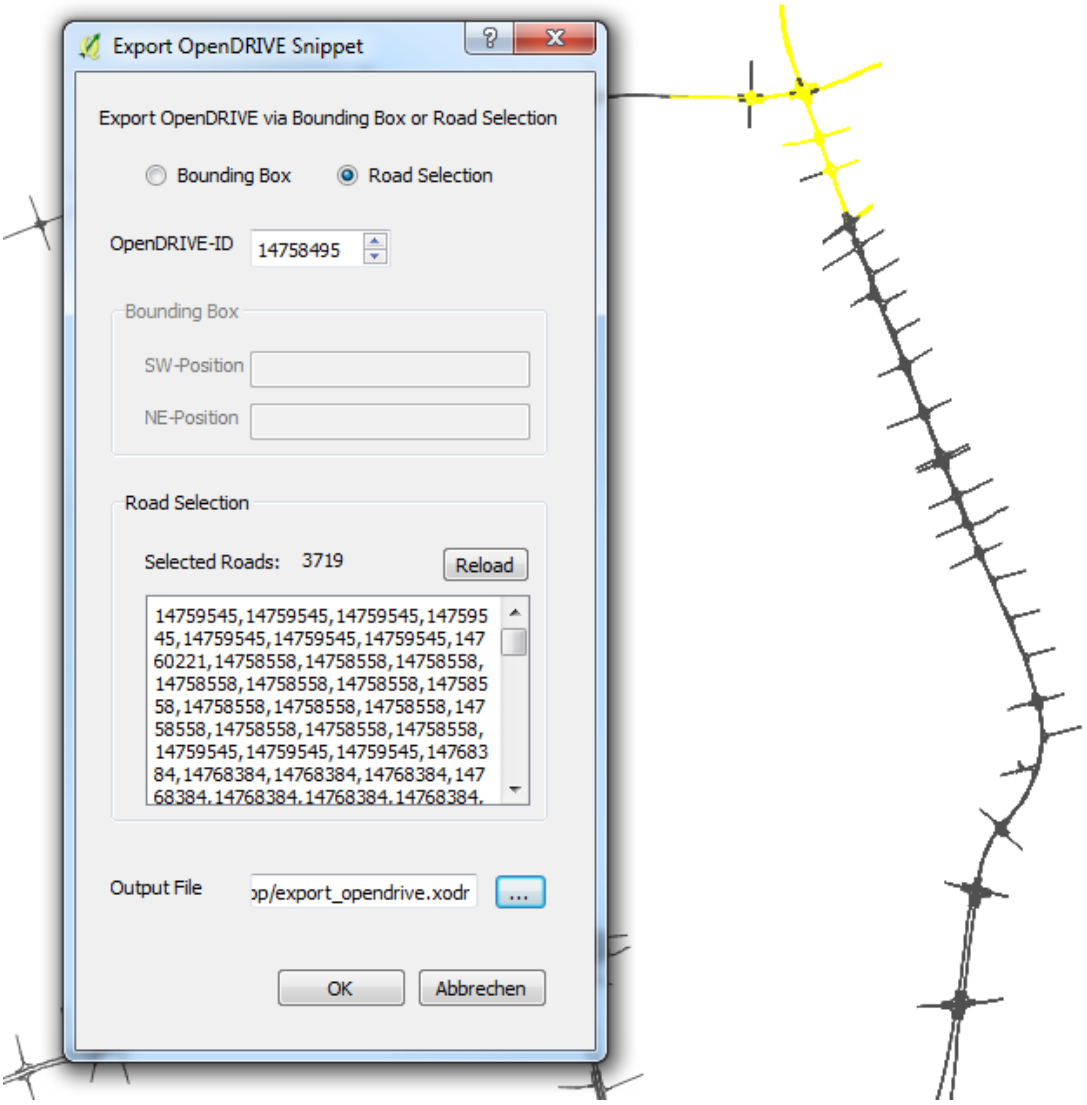

Figure 4.5: Screenshot of the QGIS Plugin with a loaded OpenDRIVE Road Network (background)

5 Conclusion and Self Assessment

In the five and a half months at the DLR I learnt a lot of new technologies and working methods. The team I was working with consisted of excellent colleagues who always tried to help me. I was included in discussions and never had the feeling of being "only an intern who should make coffee". Every week we had a meeting with the project's team and another meeting with the GDV group. This was very welcoming as it kept everybody up to date of what each other was currently working on. The project's work plan was therefore very clear to me and motivated me to work on additional tasks instead of only the ones which were actually planned for my external semester.

However, it was something new to me to have a fixed work plan to follow. In my previous jobs and study projects I usually had agile working environments. However, I still do not know whether I would prefer this fixed way of working as agile methods can sometimes bring some more freedom with it.

Before my external semester I had no knowledge of the OpenDRIVE standard but worked myself quickly into it. I found it very interesting though first also challenging to work with such a complex data format. When I started my work, I was only given a broad introduction to the project. Then I got the OpenDRIVE specification [5] and the work from Orozco [4] as reading material and the source code of all its related software libraries. In other projects and companies I have already seen worse when it comes to the documentation of source code. However, in the project "Virtuelle Welt" the work was well documented. Even all the team meetings got their own protocol archived. Therefore, I could start with my actual work relatively quickly. I do not think that this would have been the case (at least in that short time) if the documentation were less detailed. I learnt from this that I can easily work myself into some new technologies with the right motivation - and a well-documented source code.

Other new technologies (apart from the OpenDRIVE standard) were the JAXB and JPA Hibernate frameworks. I have not had to use the JAXB framework a lot but instead the JPA Hibernate framework quite often. I have never worked with such persistence libraries before and only slowly got into it. However, when I was given the task to update the existing OpenDRIVE persistence library, I had the chance to spend some more time with it and then quicker got aware of its usage.

A nice add-on to my external semester was the business trip to the INTERGEO[14] conference in Hamburg. There I had the chance to take a look at solutions of other companies in this field of research and to talk with the developers there about it.

It was a delight to work at the DLR and the project "Virtuelle Welt" in particular. And I am very happy to have the chance of writing my master thesis here, too.

[14] `http://www.intergeo.de/`

Bibliography

[1] DLR. (2016). Road2simulation guidelines, [Online]. Available: `http://www.dlr.de/ts/Portaldata/16/Resources/projekte/road2simulation/Road2Simulation_Guideline.pdf` (visited on 02/09/2017).

[2] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks", in *Proceedings of the Driving Simulation Conference DSC Europe*, 2010.

[3] J. Kopf, M. Agrawala, D. Bargeron, D. Salesin, and M. Cohen, "Automatic generation of destination maps", in *ACM Transactions on Graphics (TOG)*, ACM, vol. 29, 2010, p. 158.

[4] A. M. Orozco Idrobo, "Extension of the geospatial data abstraction library (gdal/ogr) for opendrive support in gis applications for visualisation and data accumulation for driving simulators", MSc Thesis, Technische Universität München, 2015.

[5] VIRES. (2016). Opendrive specification, [Online]. Available: `http://opendrive.org/` (visited on 02/08/2017).