Westfälische
Wilhelms-Universität
Münster

Deutsches Zentrum
DLR für Luft- und Raumfahrt
German Aerospace Center

Strategies for Automated Updating of Road Networks for Driving Simulations

Master Thesis

submitted by:

Marius Runde

Matriculation Number: 382204

M.Sc. Geoinformatics

First Reviewer:

Prof. Dr. Judith Verstegen

Second Reviewer:

Dipl. Geoinf. Michael Scholz

Brunswick, 18th August, 2017

Abstract

This master thesis deals with the problem of updating complex road networks for driving simulations. These road networks differ a lot from the ones which are, for example, used in navigation services. So far there is no applicable solution for an automated or partially automated update of these road networks and in this master thesis strategies for this process are developed.

However, road networks for driving simulations are very important for the tests of new technologies before equipping them to real vehicles etc. and testing them in the real world. So it is crucial to have close to reality models of road environments in which these tests can be performed under safe and reproducible conditions.

To supply current versions of road networks for these tests, the road networks have to be updated. Therefore, alterations between an existing and a new road network have to be identified. Such alterations can be of three different types: geometrical, topological, and semantic. Furthermore, new features can also be added to an existing road network or existing features may be removed. This requires the matching of corresponding road features first. As altered features differ from their former state, an identification can sometimes only work by comparing several attributes of the features at once. When the alterations could be identified, they have to be merged in the third and last step of the updating process.

The development of these three steps is the major part of this master thesis. They are described and explained in theory but also implemented as a proof of concept. In two case studies the implementation is tested and their results are the basis for the final discussion and conclusion.

Contents

List of Algorithms

List of Figures

List of Tables

Source Code

1 Introduction

Self-driving cars are a highly debated topic in the media, in research, and in the industry. Already in the 1980s researchers began to develop concepts for automated driving with projects like the California PATH project[1], which is still ongoing [1]. However, today many more car manufacturers, research organisations, and information technology companies are working on solutions for full automation of cars. These solutions are tested for their reliability with computer simulations and in real world settings.

The level of automated driving is often described with SAE's J3016 standard [2], (see figure 1.3 and section 1.2.1). It describes the automation from level 0 (no automation) to level 5 (full automation). The so far developed self-driving cars are still far away from full automation (level 5). The relatively high number of disengagements from their autonomous mode [3] indicates this. And though Tesla, for example, already calls its advanced driver assistance systems (ADAS) an "autopilot", it has still not reached the full automation level but only level 1 and 2 (driver assistance and partial automation) [4]. Even the test routes for Google's Self Driving Car Project are mapped in every detail before navigated by one of their cars instead of letting them drive on "unexplored" routes [5].

And also in other countries like Japan the solutions for full automation are still far away from level 5. Mitsubishi Electric, for example, expects to reach a level 3 automation between 2019 and 2020 [6]. This is the same goal of the car manufacturer Honda. They target for a level 3 automation of their vehicles by 2020 and for a level 5 automation by 2025 [7].

Before a fully automated driving technology will be approved for deployment in a product, it has to fulfill standards, which are also still under development [8]. Therefore, the so far developed solutions can only assist the driver but cannot drive the vehicle fully automated yet [1]. This can also be seen in Tesla's Model S vehicle's manual which says in chapter "About Driver Assistance": "It is the driver's responsibility to stay alert, drive safely, and be in control of the vehicle at all times." [9]

It is much cheaper (and safer) to test technologies in simulations before testing them in the real world. Furthermore, simulations have an advantage over tests of self-driving cars in traffic of the real world because they are easily to reproduce. Therefore, it is very important to create suitable environments for these simulations [10] [11].

Driving simulators like the dynamic driving simulator (visualized in figure 1.1) of the German Aerospace Center (DLR) can increase the close-to-reality feeling in a driving simulation [12]. Such driving simulators consist of a cabin in which special simulator cars (with monitors as rear- and side-view mirrors) but also normal cars can be placed.

---

[1] `http://www.path.berkeley.edu/`

In front of the car is the simulation screen which is generated by numerous projectors. The idea is that the test person feels like driving a real car. The physical forces, which one normally experiences while driving a car in the real world, are simulated by the driving simulator via movements of the cabin into the required direction. This enables very realistic driving simulations in a safe and reproducible environment.

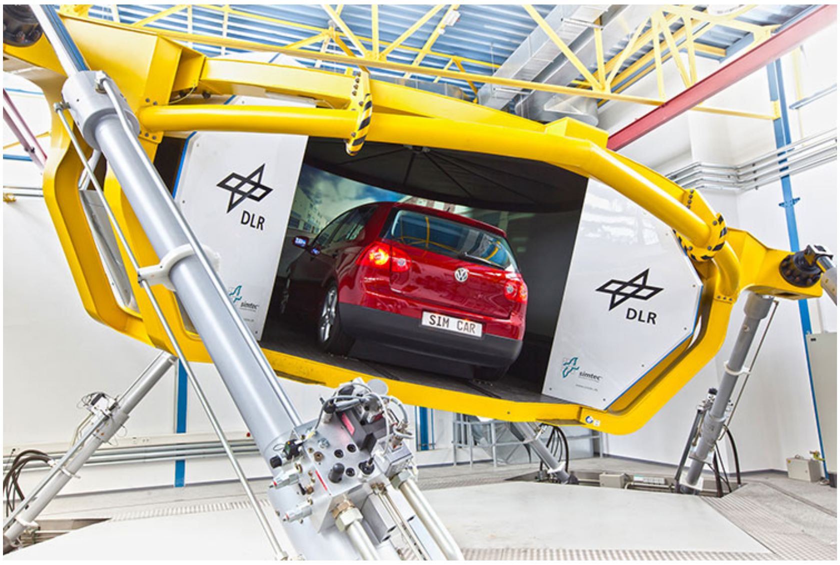


Figure 1.1: Dynamic driving simulator of the DLR [12]

The aim is to develop simulation scenarios representing real-world environments through processing of road networks with details at lane level. These road networks contain much more information than road networks for, for example, navigation (see section 1.2.2). The creation of complex road networks that can be used for driving simulations, is time consuming [11] [13].

Developing strategies to update such road networks for driving simulations is the goal of this work and is further described in the next section.

1.1 Research Question

The real world can change rapidly. And the creation of complex road networks which can be used for driving simulations is time consuming [11] [13]. Updating such road networks is a complex and expensive task as there is no homogeneous data source for the creation of road networks [10]. Until now navigation service providers use much less detailed road networks but also for them it is still a challenge to update their maps effectively.

Sometimes this can lead to analogue workarounds like in the German city of Stuttgart, where a traffic sign (which is visualized in figure 1.2) informed the drivers in front of a road construction on the highway A8 in 2016 about new road layouts. They were told by a traffic sign to not to follow their navigation service but the traffic signs instead [14].

Figure 1.2: Traffic sign in Stuttgart [14]

The overall goal of this work is the development of strategies for automated updating of road networks for driving simulations. As a proof of concept the developed strategies will be implemented in a prototypical application using the OpenDRIVE standard [15] in two case studies. One case study will update a road network by altering some of its features and the other case study will update a road network by merging it with another road network from a different source.

There are three main tasks which have to be carried out for a successful update of a road network:

1. First, the features of the new road network have to be matched with existing features of the original road network. This can be done via the features' unique IDs or via geometrical comparisons.
2. Then the features' alterations between the original road network and the new road network have to be identified. Such alterations can be geometrical (e.g. change of a road layout), topological (e.g. road marks indicating different connections between roads in junctions), or semantic (e.g. change of the lane type from *driving* to *biking*).
3. After a successful identification of the alterations they have to be merged with the original road network. How this can be done depends on various factors. Some of these factors are the coordinate reference system used by the road networks, the accuracy and currentness of the road networks. In this context it will be also important to define rules on how to handle missing or conflicting data.

From these objectives the following research question can be derived: Is it possible to update a road network for driving simulations automatically and if yes, what are the constraints?

1.2 Definition of Terms

In this master thesis some technical terms are used more often. To clarify the meaning of these terms, this section presents definitions for them.

1.2.1 Full Automation and the Different Levels of Automated Driving

Full automation of vehicles is defined as the highest level of automated driving, when the human driver does no longer have to interact with the vehicle while driving.

However, there is not a common standard for the levels below full automation. There are actually a number of standards for defining the level of automated driving, for example, from the Society of Automotive Engineers (SAE) [2], the US-American National Highway Traffic Safety Administration (NHTSA) [16], or the German Federal Highway Research Institute (BASt) [17].

However, the standards do not differ too much from each other [18] and the J3016 standard from the SAE [2] became the most often used one. Figure 1.3 describes the standard in more detail. To summarize it briefly, the J3016 standard describes automated driving in six levels. The first three levels (level 0-2) require the human driver to monitor his or her environment while driving. This is not mandatory for the last three levels (level 3-5), though in level 3 and 4 the human driver is still required to be a "fallback solution" in some situations. Only in level 5 full automation is reached and no interaction between the human driver and the car is necessary while driving.

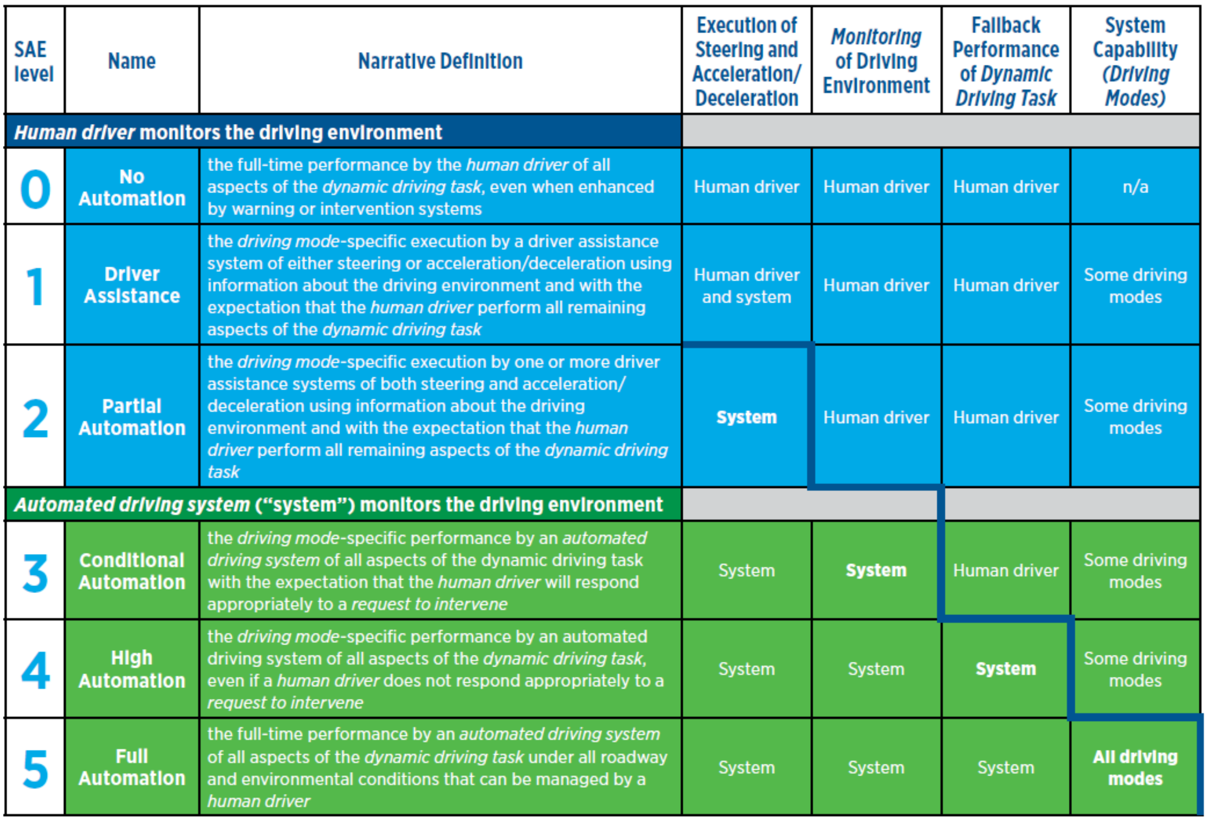
1.2.2 Road Networks for Driving Simulations

In contrast to most road networks for navigation services, road networks for driving simulations are more complex. Many navigation services already integrate environmental features in their maps, for example, parking spaces as in figure 1.4, or information about the turning directions of road lanes as in figure 1.5.

However, for a driving simulation the complete environment has to be modeled (figure 1.6). Road networks for driving simulations also contain features like trees, benches, buildings, etc. And the road geometries have to be modeled highly detailed, too. Road lanes get their own geometry and are not simply part of a graph network as in many navigation services.

There is some ongoing process to enhance road networks for navigation services with environmental features, for example, in the Navigation Data Standard (NDS) which is developed by some of the leading vehicle manufacturers and navigation service developers [20]. Unfortunately this is a proprietary standard and is still in development.

This work uses the OpenDRIVE standard [15] for the description of road networks. The standard is described in the next section and gives further, detailed information about road networks for driving simulations.



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1.3: SAE's J3016 standard for automated driving [2]

1.2.3 OpenDRIVE

The OpenDRIVE standard is an open industry standard. It can be used by anybody free of charge and its specification is publicly available [15]. There are numerous users and contributors from research and industry and it is considered as a well-established standard for road networks for driving simulations [21].

This standard is more complex than most GIS and navigation data formats. It describes the world mathematically and uses a track coordinate system for the road description and the features along it. This track coordinate system is visualized in figure 1.7. Each road follows a reference line which begins at a location with (x,y)-coordinates. These coordinates use a normal geographical reference system as it is used in many other GIS applications. The reference line's track is then described in form of mathematical functions. Other spatial features can be localized by the road's reference line with (s,t,h)-coordinates. The s-coordinate follows along the road's reference line while the t-coordinate indicates the distance to the left or right from the reference line. In most GIS software libraries this system is known as linear referencing. The height is measured relative to the reference line, too. Usually these distances are measured in meters of a projected coordinate reference system.

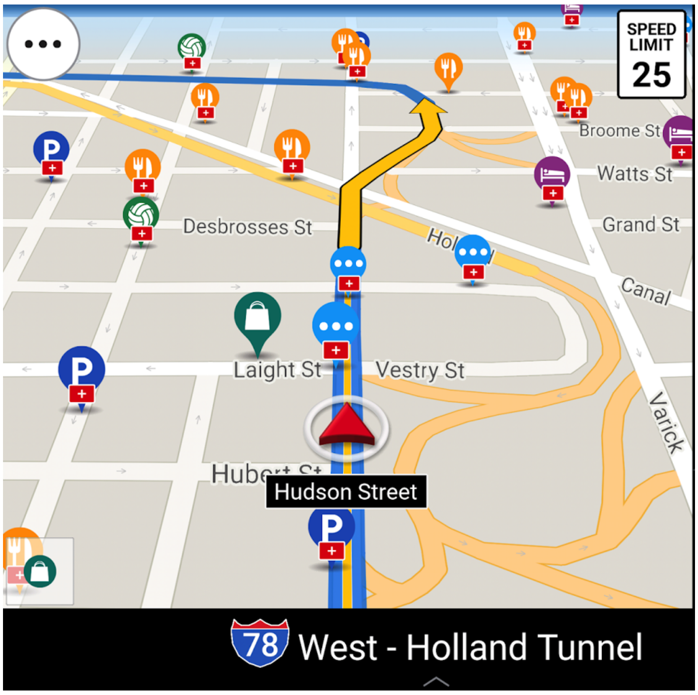


Figure 1.4: Environmental features in navigation services (here: CoPilot GPS) [19]

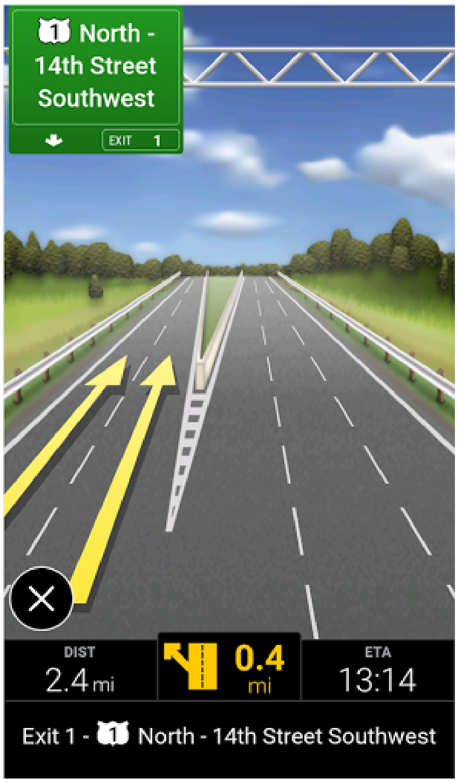


Figure 1.5: Road lanes in navigation services (here: CoPilot GPS) [19]

In figure 1.8 an example of a road in the OpenDRIVE standard [15] is visualized. It includes some of the most relevant features, for example, road lanes (1), objects (2), and signals (3). The dotted, blue lines represent the road lanes' marks, which can either be a solid line, dotted line, or no line.

These roads are then linked to each other in form of a graph-like network. This is visualized in figure 1.9 where each line represents a road. Each road can have one predecessor and successor. As in junctions one road can be followed by more than one successor road, the road's successor is then set to the relevant junction. This enables 1:n relationships in the graph via the junction as an associative entity. Roads in junctions (the red lines in figure 1.9) are named connecting roads to highlight their function. The possible complexity of junctions is further shown in figure 1.10. It visualizes how one junction can handle multiple incoming roads and their relations via numerous connecting roads.

The OpenDRIVE standard [15] is based on an XML file format and stores numerous features to describe the road network including not only the roads but also features like trees or buildings next to them. Figure 1.11 displays the hierarchy of the different features.

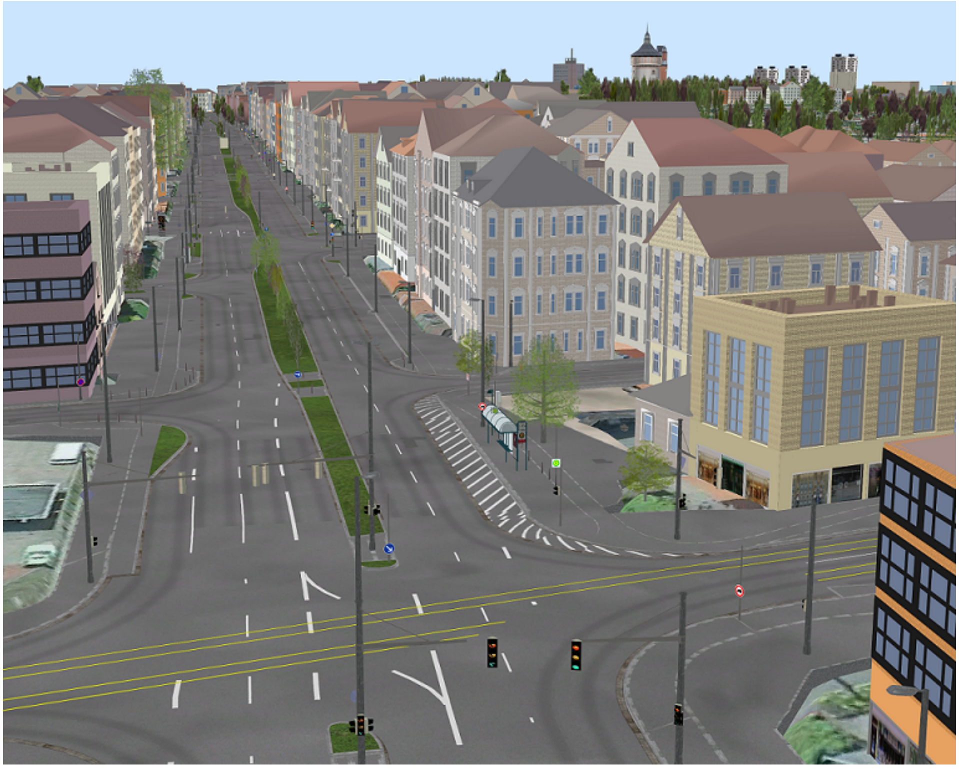


Figure 1.6: Road network for a driving simulation with environmental features [13]

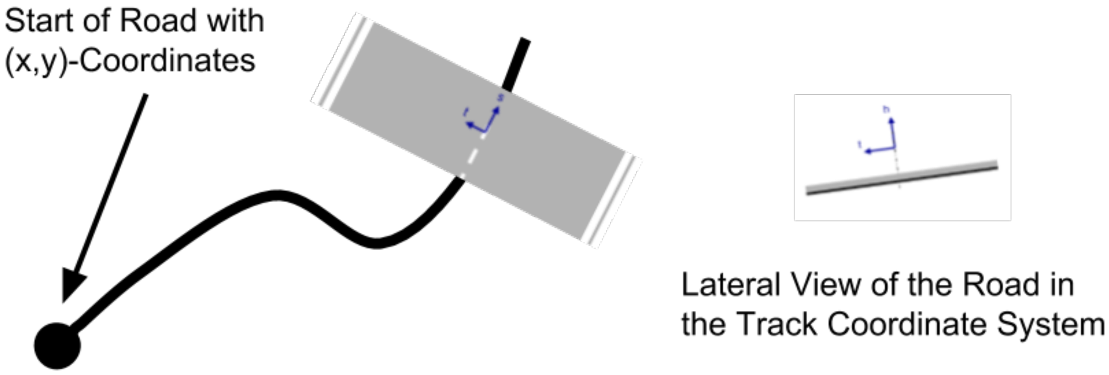


Figure 1.7: Track coordinate system of the OpenDRIVE standard [15]

There are three main features in the road network which are: road features and its child elements, object features and signal features. These three features are all geographical objects, which is further explained in chapter 2. All the other features of a road network are only "logical" features. They do not have their own geometry etc. and usually link towards other features only, for example, connections are used to define the predecessor and successor road features in a junction whereas junctions are just a collection for these connections.

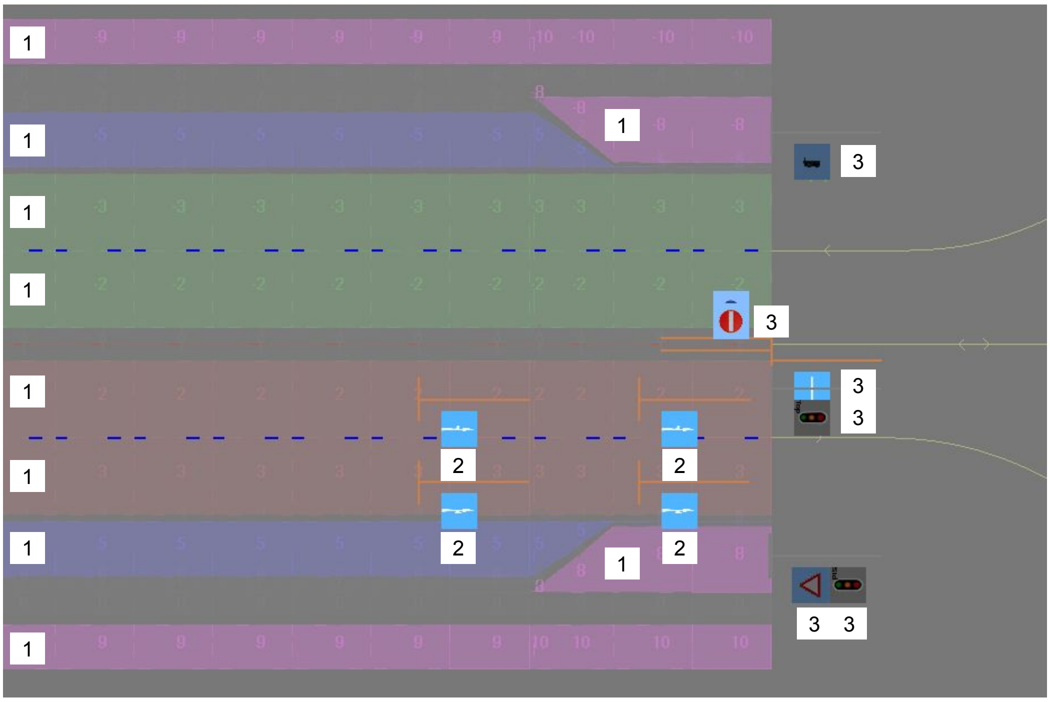
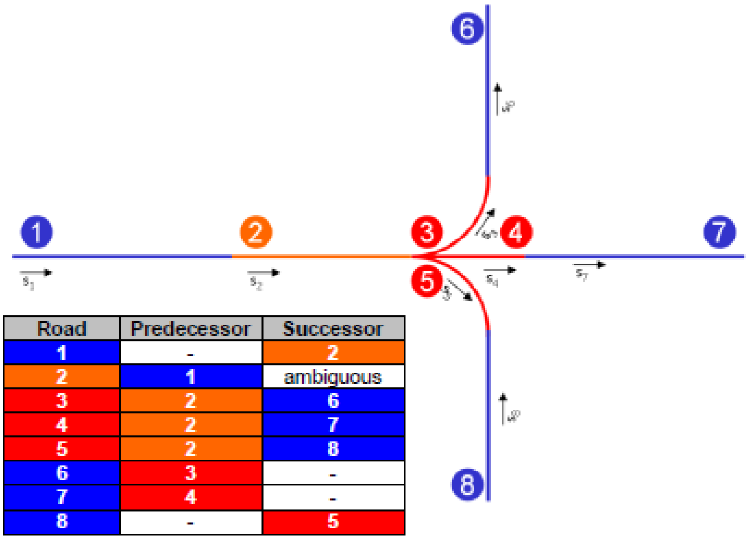


Figure 1.8: Example of a road in the OpenDRIVE standard [15] (1: road lanes, 2: objects, 3: signals)



Road	Predecessor	Successor
1	-	2
2	1	ambiguous
3	2	6
4	2	7
5	2	8
6	3	-
7	4	-
8	-	5

Figure 1.9: Road linkage in the OpenDRIVE standard [15] (in different colours for a better readability)

Only lanes are an exception here. They have their own geometry but as they are one of the most important elements of a road feature, they are seen as a composite aggregation of the road feature and not as individual features in a road network.

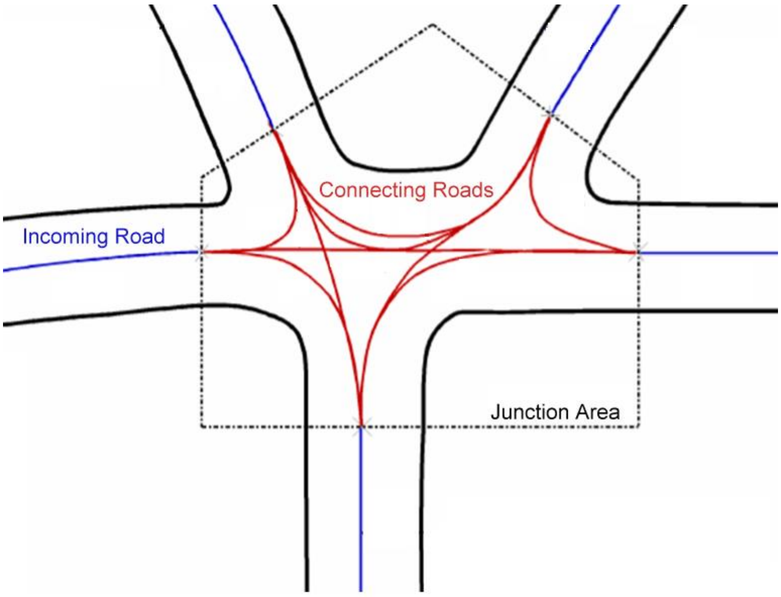


Figure 1.10: Junction with incoming and connecting roads in the OpenDRIVE standard [15]

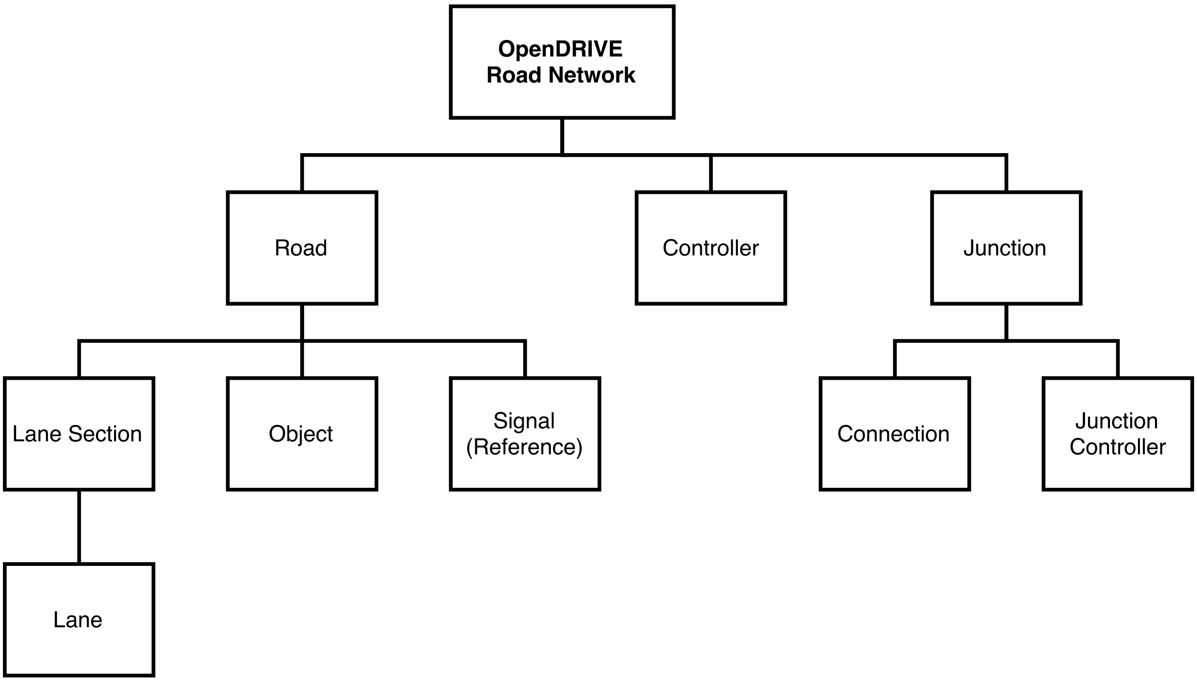


Figure 1.11: Hierarchy of the features of the OpenDRIVE standard [15]

The hierarchy and the different features of the OpenDRIVE standard [15] are further described in the following list, which summarizes the child entries of an OpenDRIVE road network.

- **Roads:** List of road features
    - **Road:** A road holds information about its topological connections to other roads (predecessor and successor), its geometry and elevation, the outline of the road lanes, and all objects and signals which have the road as their parent feature. The geometry of a road is described via a starting point with (x,y)-coordinates and from that starting point onward via mathematical functions which can describe straight lines, spirals, arcs, or cubic polynomials. This geometry is the reference line to which the child entries' geometries refer to. An example of how the geometries of a road are stored is shown in the following source code from the OpenDRIVE standard [15]:

      Source Code 1.1: Example of how road geometries are stored

      ```
      <planView>
         <geometry s="0.0" x="-7.0710678117841717"
            y="7.0710678119660715" hdg="5.4977871437752235"
            length="0.48660000000023864">
            <line/>
         </geometry>
         <geometry s="0.48660000000023864" x="-6.7269896520425938"
            y="6.7269896522231525" hdg="5.4977871437736381"
            length="3.1746031746031744">
            <arc curvature="-0.12698412698412698"/>
         </geometry>
      </planView>
      ```

        - **Lanes (Lane Sections):** List of lane sections which are containers for lane features in a specific spatial range. Lane sections hold lane features in three different attributes: left, center, and right. The center lane is just the reference line for all other lanes and the objects and signals. The left and right lanes describe the lanes on the left (positive IDs) and right (negative IDs) of the center lane.
            - **Lane:** A lane feature does not only contain information about its shape and connections to other lanes but also about attributes including road marks, material, visibility, speed, access, and height.
        - **Objects:** List of object features
            - **Object:** An object is a feature on or next to the road. It is solely connected to its parent road and to no other component in the road network. A typical object is, for example, a tree next to the road. However, an object can also be a road mark which describes the allowed turning directions of the road lane. This can then have an indirect effect on other components. This is further described in section 2.3. The geometry of an object is described relative to its parent road via (s,t,h)-coordinates. If numerous objects of the same

        type exist along a road, one object entry with a ”repeat” attribute is sufficient.

    - **Signals:** List of signal features and signal reference features
        - **Signal:** The signal feature is very similar to the object feature. However, it can be directly connected to other signals via a controller. A typical signal feature can be a traffic light. The geometry of a signal is described similarly as the object’s geometry relative to its parent road via (s,t,h)-coordinates.
        - **Signal Reference:** References a signal of another road which is also valid for this road. Its geometry is described in the same way via (s,t,h)-coordinates.

- **Controllers:** List of controllers
    - **Controller:** Controllers are used to logically connect dynamic signals (usually traffic lights) which shall be able to be controlled commonly in a driving simulation.
- **Junctions:** List of junctions
    - **Junction:** Junctions contain connections of roads and their lanes to handle the transition between them.
        - **Connections:** List of connections
            - **Connection:** Two roads are always connected via a junction. The connection describes which lane of a road is followed by which lane of another road.
        - **Junction Controllers:** List of junction controllers
            - **Junction Controller:** Similar to the signal reference a junction controller references a controller which is valid for this junction.

These features are all logically connected, for example, junctions contain roads which in turn contain signals. Driving simulations depend on highly precise maps which can be modeled with the OpenDRIVE standard [15] very good. For example, each lane has its own geometry and signals like traffic lights or holding lines are valid for specified lanes only.

1.3 Related Work

This chapter provides an overview of existing work in this field of research. It is distinguished into two sections. The first section (1.3.1) is about the generation of road networks for driving simulations. This is followed by the second section which is about updating road networks for driving simulations (1.3.2).

1.3.1 Generation of Road Networks for Driving Simulations

Most of the related work on generating road networks for driving simulations has been published by the DLR and the Swedish National Road and Transport Research Institute (VTI). For example, Richter et al. from the DLR [13] describe the difficulty of creating a road network for driving simulations from heterogeneous data sources. Using the project Virtual World as an example, they describe how they use the OpenDRIVE standard [15] to transform the data into a common format. The merging of heterogeneous data sources is an important part of this master thesis, too. As the authors describe, there is no single data source for all the required data and the existing data sources use various data formats.

A very similar approach has been done by researchers of the VTI [22]. They also generated a road network for driving simulations from heterogeneous data sources. Their methodology does not differ a lot from the one of the DLR, which has been previously described. However, both works go beyond the scope of this master thesis as they also deal with, for example, the generation of terrain and building models. This master thesis only deals with the actual road network as it is described with the OpenDRIVE standard [15].

There are also two master theses from the VTI about the generation of road networks for driving simulations. The work of Kurteanu et al [23] describes how road networks for driving simulations can be manually generated. Their approach does not use external data for the generation. Nevertheless, many of the algorithms and calculations are a general requirement for the generation of road networks independent of the type of generation (manual or automated).

The other thesis from Shi [24], which should be mentioned for completeness only as it lacks in reliable sources, tries to generate OpenDRIVE [15] roads from OpenStreetMap (OSM)[2] data. Unfortunately the result is not at all a complete road network. It shows that it is possible to generate single roads from OSM data but without connecting them.

Using OSM data for the generation of road networks for driving simulations has also been tested in an approach by the DLR [25]. The result of the work are that OSM can be used as a data source but does not fulfill all the requirements of a road network for driving simulations.

---

[2] `www.openstreetmap.org/`

1.3.2 Updating Road Networks for Driving Simulations

While there is already some work on the generation of road networks for driving simulations, there is nearly no work on updating them. Mitsubishi Electric [6] presented in March 2017 a product for updating their road networks automatically. However, this is a proprietary solution and no details about the exact solution have been publicly published yet.

Publicly published research usually deals with the update of road networks for navigation services and not the update of road networks for driving simulations [26]. Unfortunately, the road networks for driving simulations are much more complex (as described in section 1.2.2) which makes it impossible to update them in the same way. Road networks for driving simulations contain, for example, high-precision data of the road lane geometries, which is not necessarily stored in road networks for navigation services [27].

Updating Road Features

However, some of these works can be used to update some features in road networks for driving simulations, for example, road geometries. Wu et al. [26] give a very good overview of the different ways of creating and updating road networks for navigation services. Their approach is to use trajectories from GPS-based mobile devices. They developed an algorithm to compare the geometries of these trajectories with the geometries of the road network. The algorithm is then able to identify differences between them and update the road network's geometries accordingly.

This master thesis will use the OpenDRIVE standard [15] to describe road networks for driving simulations. However, this standard is not compatible with GIS formats and therefore approaches from, for example, Wu et al. [26] cannot be applied to it. With the work of Orozco [28] it is possible to transform the roads of the OpenDRIVE standard [15] into GIS-readable formats. Her work describes a way to transform spatial OpenDRIVE components like roads into any GDAL compatible data format. This enables the usage of approaches from Wu et al. [26] and others, for example, Mustière et al. [29].

The work of Mustière et al. [29] describes how two road networks with different levels of detail can be matched. They compare the road features by their geometry, topology, and semantic to identify alterations. These are also the three types of alterations which are used in this work. To find corresponding road features in the two datasets, a process that is also performed in this work in section 2.1, the authors compare at first the geometries of the road features. Even though sometimes the street name already provides the information one is looking for, they primarily base their methods on the geometrical matching. This is extremely relevant for road networks for driving simulations which do not always contain information about the street name of a road feature. Especially in junctions it may not be clear which street name should be used for a road lane describing a left- or right-turn. Therefore, the usage of geometrical matching is extremely relevant for this work. After the geometrical identification of possible corresponding road features the topological information can validate these matches. The semantic information is used as an optional, additional validation source. The overall methodology of the work of Mustière et al. is very similar to the one in this work. However, their work only handles road features in a road network. Handling all the other features of a road network for driving simulations, too, is more complex. Therefore, their work will only be a small part of this work.

Updating Signal Features and Others

Other road network components like road signals require other techniques to be updated. The work from Kopf et al. [30] uses a so called radial visibility sweep algorithm to find the nearest road segments to a given location. Finding the nearest road segments for a road signal is an important task in connecting them with the roads topologically. The algorithm works in a way that it draws straight lines (sight lines) from a given location to all road segments around it. Whenever a sight line intersects another road segment, the road segment at the end of the sight line will be removed from the resulting set (unless it is a higher classified road). The OpenDRIVE standard [15] uses a track-based coordinate reference system to map the signals. Therefore, the nearest road segment has to be found for each signal to connect it to the road.

1.4 Structure of this Work

This is the introduction to the research question with an overview of the definition of terms and a literature review of related work. In chapter 2 the developed algorithms are then described as well as other theoretical work for this master thesis. While the algorithms are shown as pseudo code in chapter 2, chapter 3 describes how they have been implemented in a prototype as a proof of concept. This prototype is then tested in two case studies. Thereafter, the results of this work are presented in chapter 4 and discussed in chapter 5. The work ends with a conclusion and a brief discussion of possible future work on this topic in chapter 6.

2 Methodology

In this chapter the methodology of this work is described. It is divided into three main sections in order of their application, which are also shown in figure 2.1 as a flow chart. The first section deals with finding corresponding features in road networks (section 2.1). Only when a pair of corresponding features could be found in both datasets (the original and the new one), the next two steps are possible. These are the identification of alterations of such features (section 2.2) and the merging of these alterations in road networks (section 2.3).

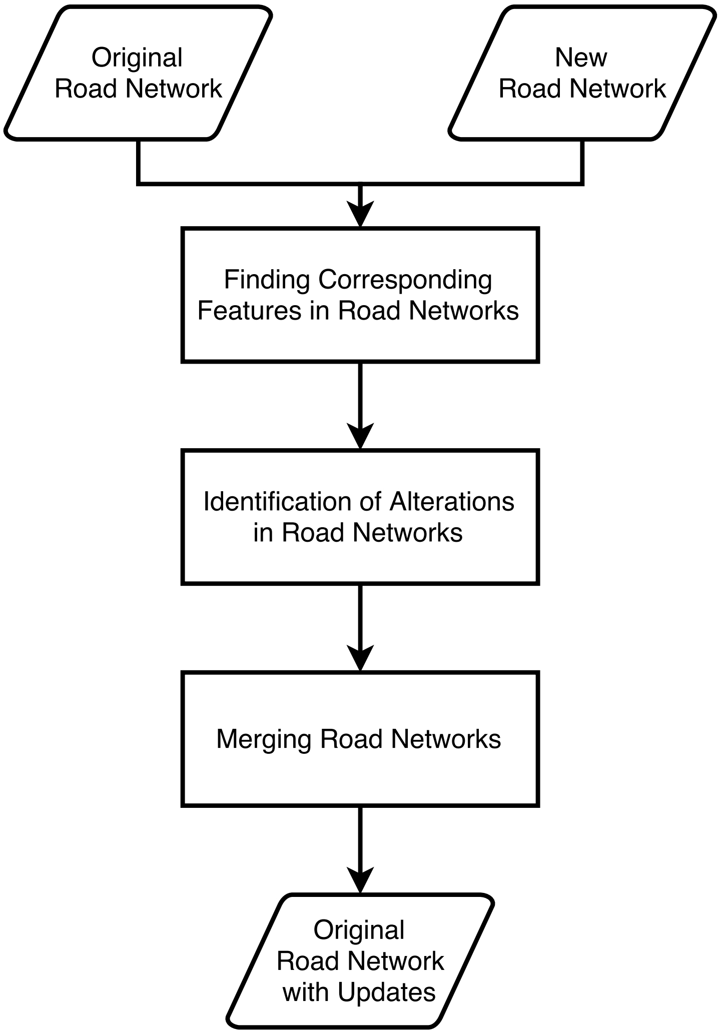

Original Road Network
New Road Network
Finding Corresponding Features in Road Networks
Identification of Alterations in Road Networks
Merging Road Networks
Original Road Network with Updates

Figure 2.1: Methodology of automated road network updating

Geographical objects are generally defined as locations with a geometry, topology, semantics, and dynamic [31] [32]. The features of a road network are geographical objects with the exception that they are static and not dynamic. There are only three feature types in a road network (which uses the OpenDRIVE standard [15]) which can be altered by the new road network: objects, roads, and signals. Road lanes are here seen as an essential part of roads while objects and signals can be seen as independent features, because they can be moved to another road easily. Controllers, junctions, and signal dependencies are only virtual and not physical features. They can only be altered by alterations of the physical features they are connected to.

Road networks for driving simulations can be updated either from complete or partial road networks which can describe the same road network in the same geographical area or another one which is connected to the original road network. The here described methods require the usage of the OpenDRIVE standard [15] for the update with a complete (or partial) road network. For object- or signal-only updates this is not necessary. However, the signal features have to follow the same guideline of naming signal features, for example, in Germany the StVO [33], as the original road network.

2.1 Finding Corresponding Features in Road Networks

This section describes the process of finding corresponding features in the original road network. Then the alterations can be identified (section 2.2) and updated on the original feature (section 2.3).

The process of this method is visualized in figure 2.2 as a flow chart. Each feature of a road network using the OpenDRIVE standard [15] has a unique identification attribute (*ID*). Therefore, the most straightforward method to find a corresponding feature in the original road network is to look for the feature of the same type and with the same unique ID. However, it is not always possible to find a corresponding feature in the original dataset this way. It can happen that the new road network uses other unique IDs or it contains further features. In the latter case these features appear to be additions to the original road network.

As already explained before, there are only three different types of features in a road network (which uses the OpenDRIVE standard [15]) which can alter the road network: objects, roads, and signals. Finding objects and signals can be achieved with a similar method. However, there are some differences though. Therefore, this section is divided into three subsections, describing how the three feature types can be found in a road network, if no corresponding feature can be found via its unique ID. As objects and signals can be also found via their parent road feature, it is best practice to begin with the road features first. Therefore, the order of finding corresponding features in road networks is: road features, then signal features, and finally object features.

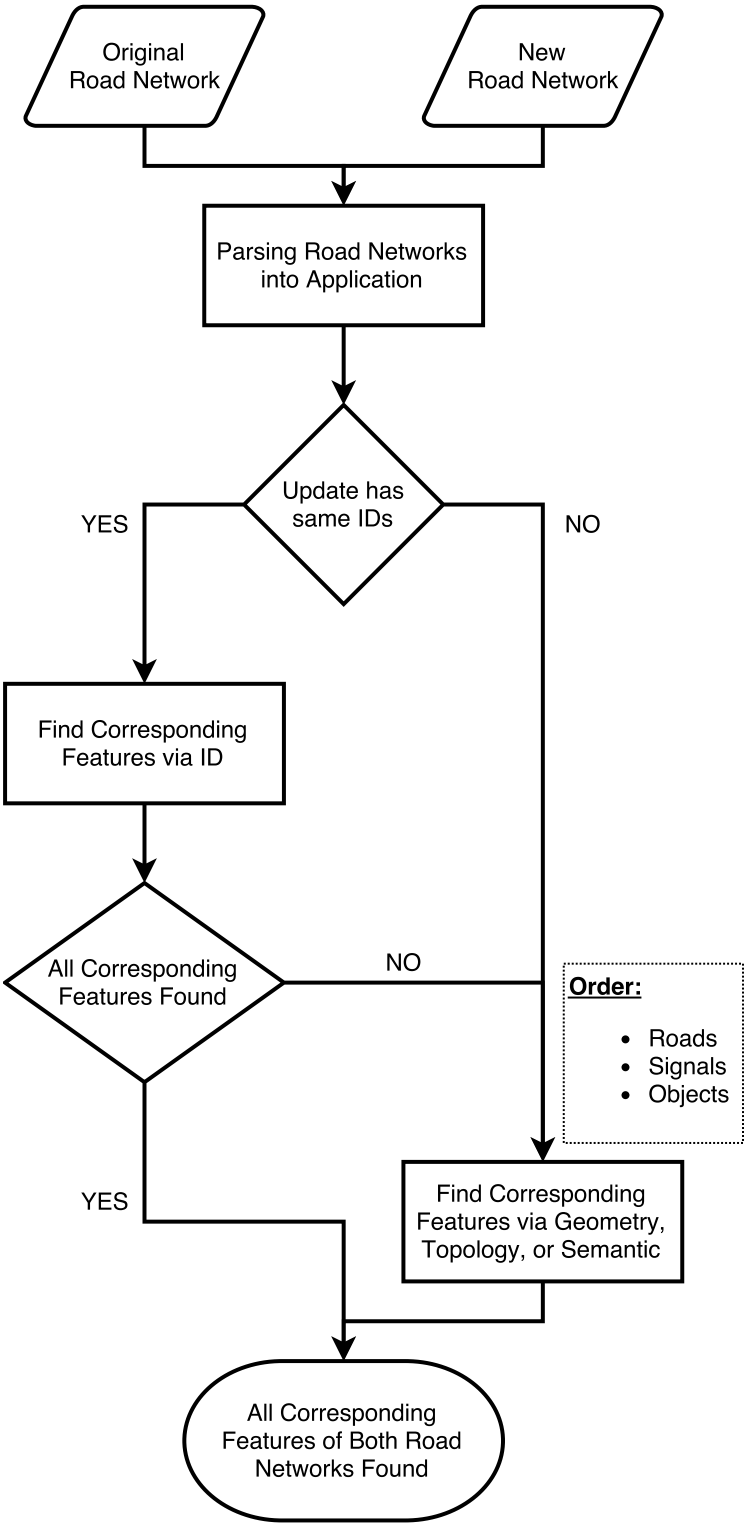

Original Road Network
New Road Network
Parsing Road Networks into Application
Update has same IDs
YES
NO
Find Corresponding Features via ID
All Corresponding Features Found
NO
Order:
- Roads
- Signals
- Objects
YES
Find Corresponding Features via Geometry, Topology, or Semantic
All Corresponding Features of Both Road Networks Found

Figure 2.2: Process of finding corresponding features as a flow chart

2.1.1 Road Features

Finding a corresponding road feature in a road network is a bit more complex than finding a corresponding object or signal feature. There are two ways to do that: either by comparing the road's geometry or its topology with the other roads of the original road network.

The first method, which uses the geometry approach and is also shown in algorithm 1, compares the start and end points of the new road feature's geometry with the start and end points of the road features' geometries of the original road network. To consider the two features as corresponding, the two start and the two end points should have a distance of less than 3 meters to each other, respectively. This value is generated from the maximum width of 3.75m for driving lanes on highways in Germany [34] minus a buffer of 0.75m towards other features. To ensure that the nearest suitable road feature is chosen, the algorithm only returns the nearest of these road features as the corresponding road feature of the original road network. These locations are sufficient to be sure about the correspondence of the two road features as each road feature has a unique start and end point.

Algorithm 1 Finding a corresponding road feature by its geometry (pseudo code)

```
Require: Road feature from the new road network ($rf$) and list of road features from the original road network ($list$)
Ensure: $list$ contains $rf$
$nearest = null$
$distanceStart = -1$
$distanceEnd = -1$
for $i = 0, \dots, list.length$ do
    if $rf.geometry.start == list[i].geometry.start$ and $rf.geometry.end == list[i].geometry.end$ then
        return $list[i]$
    end if
    $currentDistanceStart = distance(rf.geometry.start, list[i].geometry.start)$
    $currentDistanceEnd = distance(rf.geometry.end, list[i].geometry.end)$
    if $currentDistanceStart < 3m$ and $currentDistanceEnd < 3m$ then
        if $nearest == null$ or ($distanceStart > currentDistanceStart$ and $distanceEnd > currentDistanceEnd$) then
            $nearest = list[i]$
            $distanceStart = currentDistanceStart$
            $distanceEnd = currentDistanceEnd$
        end if
    end if
end for
return $nearest$
```

The other method to find a corresponding road feature is by using its topology. This is shown in algorithm 2. Each road feature usually has links to at least one preceding and one succeeding road feature. It can happen that a road feature does not have either a predecessor or successor, for example, at the end of the modelled road network. However, in junctions it can also be the case that a road feature has more than one predecessor or successor, for example, to connect the roads via left and right turning lanes to the other road features at the junction.

Algorithm 2 Finding a corresponding road feature by its topology (pseudo code)

```
Require: Road feature from the new road network ($rf$) and list of road features from
    the original road network ($list$)
Ensure: $list$ contains $rf$
for $i = 0, \ldots, list.length$ do
    $con1 = rf.connections$
    $con2 = list[i].connections$
    if $con1.pre == con2.pre$ and $con1.suc == con2.suc$ then
        return $list[i]$
    end if
end for
return $null$
```

The here described algorithm simplifies the problem by respectively setting the road feature's predecessor and successor as a single connection. The algorithm iterates through all road features of the original road network and compares each of their predecessors and successors with the connections of the new road feature. If a road feature with the same connections can be found, it will be returned as the result.

Furthermore, it is also mandatory that the different feature IDs of the two road networks are linked to each other. Therefore, this method tries to find the corresponding road features by their geometry first and then the corresponding road features of the remaining road features by their topology.

2.1.2 Signal Features

Though the method for finding the corresponding object feature is similar to the method for finding the corresponding signal feature in the original dataset, this subsection describes the method for signals first. After that a brief description of the minor changes for object features is then given.

To find the corresponding signal feature in the original road network the geometry of the new signal feature has to be compared with the geometries of the signal features in the original road network. This is also shown as pseudo code in algorithm 3. The algorithm iterates through the list of all signal features from the original road network and compares each of the signal features' signal type, heading, and geometry with the equivalent attributes of the new signal feature.

Algorithm 3 Finding a corresponding signal feature (pseudo code)

```
Require: Signal feature from the new road network ($sf$) and list of signal features from the original road network ($list$)
Ensure: $list$ contains $sf$
$nearest = null$
$distance = -1$
for $i = 0, \dots, list.length$ do
    if $sf.type == list[i].type$ and $diff(sf.heading, list[i].heading) < 20$ then
        $currentDistance = distance(sf.geometry, list[i].geometry)$
        if $currentDistance < 3m$ then
            if $nearest == null$ or $distance > currentDistance$ then
                $nearest = list[i]$
                $distance = currentDistance$
            end if
        end if
    end if
end for
return $nearest$
```

The signal feature's heading is not an attribute from the OpenDRIVE standard [15]. It uses the attribute "validity" which indicates whether the signal feature is headed towards ("-") or against ("+") the road feature's direction. However, by taking the road feature's geometry at the location of the signal feature into account it, is possible to calculate the absolute heading of the signal feature in degrees.

To make a preselection of possible corresponding signal features, the signal types have to be equal and the headings must not differ by more than 20 degrees. The value of 20 degrees has been found as a suitable limit from my experience with the work of road networks using the OpenDRIVE standard [15]. Greater differences in the headings can be ignored (at least for German roads) as the German traffic regulation (StVO) demands an approximately right angle for traffic signs towards the related road (VwV-StVO §§ 39 to 43 [35]). Therefore, a greater difference between the two signal feature headings strongly indicates that the signal features must be valid for two different road features.

In the next step the geometries of the two signal features are compared. It can happen, that the new road network has a minor difference to the original road network's geometries. Therefore, signals whose geometry differs by less than 3 meters can be regarded as corresponding signal features. This is the same buffer used also for the road features above. To ensure that the nearest suitable signal feature is chosen, the algorithm only returns the nearest of these signal features as the corresponding signal feature of the original road network.

2.1.3 Object Features

Finding a corresponding object feature works similar to the above described method and is shown in algorithm 4. However, it does not compare the types of the object features, as there is no attribute called "type" for them. The heading can also be ignored, as it is not of importance in the context of driving simulations.

Algorithm 4 Finding a corresponding object feature (pseudo code)

```
Require: Object feature from the new road network ($of$) and list of object features
    from the original road network ($list$)
Ensure: $list$ contains $of$
nearest = null
distance = -1
for $i = 0, \ldots, list.length$ do
    $currentDistance = distance(of.geometry, list[i].geometry)$
    if $currentDistance < 5m$ then
        if $nearest == null$ or $distance > currentDistance$ then
            $nearest = list[i]$
            $distance = currentDistance$
        end if
    end if
end for
return $nearest$
```

Polygonal Objects

A special case are polygonal object features. In contrast to single-point features like signal features and most other object features, they require more than only one location to be compared. Nevertheless, the algorithm does not differ too much from algorithm 4 except that it compares two polygons instead of two points.

2.2 Identification of Alterations in Road Networks

After the corresponding features have been found, the feature pairs from the original and new road network can be compared. In case no corresponding feature has been found, this is considered as either a removal of the original feature or an addition of a new feature. This can also be the case for features which exist in both road networks but have been altered so much, that they cannot be matched to one another via their attributes any more.

The following sections describe how alterations of a feature's geometry, topology, and semantics can be identified.

2.2.1 Geometrical Alterations

Algorithm 5 describes the method of how geometrical alterations can be identified.

Algorithm 5 Identification of geometrical alterations (pseudo code)

```
Require: Geometries ($geom_1, geom_2$) of a pair of features of the same type from the
    original and new road network
Ensure: $geom_1 == geom_2$
if $geom_1.type \neq geom_2.type$ then
    return true
else
    if $geom_1.type == POINT$ then
        if $geom_1 \neq geom_2$ then
            return true
        end if
    else
        if $geom_1.geometries.length \neq geom_2.geometries.length$ then
            return true
        end if
        for $i = 0, \ldots, geom_1.geometries.length$ do
            if $geom_1.geometries[i] \neq geom_2.geometries[i]$ then
                return true
            end if
        end for
    end if
end if
return false
```

At first the algorithm compares the geometry types of the two features. If they already differ, it will return *true* as this is a geometrical alteration. Otherwise it continues and compares the geometries of the two features depending on whether their geometry type is a point or something else, for example, a collection of geometries as they are used for road features.

Points can be compared directly. However, their geometry should first be converted from the relative track coordinate system into a projected spatial reference system in case the geometry of the parent road has been altered, too.

To identify geometrical alterations of road features, the algorithm compares the number of available geometries for each feature. If there is a difference, this is a geometrical alteration, too. Otherwise the algorithm iterates through the geometries and compares them pairwise. And identified difference there is a geometrical alteration. In every other case the algorithm returns *false*.

2.2.2 Topological Alterations

If the geometry of a feature is altered, often its topology is altered, too. However, as the identification of geometrical alterations is already handled in section 2.2.1, this section describes the identification of solely topological alterations independent of any geometrical alteration.

Similarly to the section before, the two kinds of features (road features and point features) have to be handled separately. The topology of road features is described by connection attributes, which define the predecessor and successor of a road feature. Point features (object features and signal features) do not have any attribute to describe their topology. They are children of a parent road feature. Therefore, their topology is defined by their parent road feature only.

Due to this difference of the topological description of the different feature types, this section separates the methods for the identification of topological alterations into a section for road features and for point features, respectively.

Road Features

Road features are topologically connected to other road features. Furthermore, they also connect the road feature lanes to one another. Each road feature can have multiple incoming and outgoing connecting roads. In the OpenDRIVE standard [15] road features are usually not directly connected to each other but via connection road features in junctions, which is also described in section 1.2.3 above. The connections to these connection road features are then handled by the junction.

For a more simplified description of the algorithm, it is here assumed that each road feature can point directly to another road feature without any other instance in between. Each road feature shall have a list of connections to other road features. In algorithm 6 it is described in pseudo code how the connections of a pair of road features from the original and new road network are compared.

Initially the lengths of the lists of connections are compared. Unequal lengths are already an evidence for a topological alteration. For lists of equal length the algorithm iterates through the connections and compares them pairwise to find any difference. Hereby the connection's type is compared beforehand to make sure that the road has the same direction and not inverted predecessor and successor road features. The algorithm finally returns **true**, if a topological alteration has been found. Otherwise it returns **false**.

Point Features

Point features in a road network always have a parent road feature. Therefore, the identification of topological alterations of point features is much simpler than for road features. Algorithm 7 shows how only the parent attribute of two point features has to be compared for equality. If they are unequal, this will be treated as a topological alteration.

Algorithm 6 Identification of topological alterations of road features (pseudo code)

```
Require: List of connections ($conn_1, conn_2$) of a pair of road features from the original
    and new road network
Ensure: $conn_1 == conn_2$
if $conn_1.length \neq conn_2.length$ then
    return true
end if
for $i = 0, \ldots, conn_1.length$ do
    for $j = 0, \ldots, conn_2.length$ do
        if $conn_1[i].type == conn_2[j].type$ then
            if $conn_1[i].connectsWith \neq conn_2[j].connectsWith$ then
                return true
            end if
        end if
    end for
end for
return false
```

Algorithm 7 Identification of topological alterations of point features (pseudo code)

```
Require: Pair of point features ($f_1, f_2$) from the original and new road network
Ensure: $f_1.parent == f_2.parent$
if $f_1.parent \neq f_2.parent$ then
    return true
end if
return false
```

2.2.3 Semantic Alterations

Apart from geographical and topological alterations, there can be also semantic alterations of road networks. An identification of these alterations is relatively simple as long as the original and new road networks, which are compared against, use the same naming convention for the semantic data of their features. For example, signal features usually have a unique identification number which is in Germany defined by the road traffic regulations (StVO) [33]. If they do not have the same naming convention, the naming convention of one road network has to be transformed into the other one. Hereby it completely depends on the kind of update whether the naming convention of the original road network or the naming convention of the new road network should be transformed, for example, an update caused by a reform of the StVO.

The algorithm for the identification of semantic alterations is shown as pseudo code in algorithm 8. The algorithm first compares the number of attributes per feature and then (if the number is equal) iterates through all pairs of attributes from the two features. An unequal number of attributes indicates that there are new or missing attributes. The algorithm then returns *true* as a semantic alteration has been found. Otherwise it iterates through the attributes and compares them pairwise.

The result (*true* for an identified semantic alteration and *false* for no semantic alteration) is only an indicator whether an alteration exists. In the merging phase (section 2.3) the features' attributes are further inspected.

Algorithm 8 Identification of semantic alterations (pseudo code)

```
Require: List of attributes ($attr_1, attr_2$) of a pair of features of the same type from the
    original and new road network
Ensure: $attr_1 == attr_2$
if $attr_1.length \neq attr_2.length$ then
    return true
end if
for $i = 0, \ldots, attr_1.length$ do
    for $j = 0, \ldots, attr_2.length$ do
        if $attr_1[i].name == attr_2[j].name$ then
            if $attr_1[i].value \neq attr_2[j].value$ then
                return true
            end if
        end if
    end for
end for
return false
```

2.3 Merging Road Networks

After the alterations between the original and new road networks have been found, the road networks have to be merged. This chapter describes how the merging process works. It is also separated into three sections for geometrical, topological, and semantic alterations.

Before the merging algorithms are described, it has to be mentioned that many alterations of road networks also have direct effects on other features while other alterations do not. For example, object features are independent from any other feature except their parent road feature they belong to. Therefore, alterations of object features do not affect any other feature type in a road network. On the other hand, alterations of road features affect nearly all other feature types as most of them are directly connected to road features. As in section 2.2 already explained, there are only three feature types (object features, road features, and signal features) which can be directly altered by the new road network.

It is important to first take a look at the effects of the different kinds of alterations before thinking about how to merge them. Some alterations cause a lot of required updates on other elements in a road network while other alterations only solely demand for an update of the altered element. Using this knowledge can make the algorithms much more efficient as they do not have to perform unnecessary comparisons between the altered elements in a road network.

An exception from this are the additions of new elements and removals of existing elements. They can always cause updates on all those elements in a road network which are affected by geometrical, topological, and semantic alterations combined.

The following three sections 2.3.1, 2.3.2, and 2.3.3 describe the effects of geometrical, topological, and semantic alterations on the elements of a road network which are based on the OpenDRIVE standard [15]. Table 2.1, 2.2, and 2.3 also show an overview of the possible effects respectively.

2.3.1 Effects of Geometrical Alterations

Can have an effect on other...	Geometrical Alterations of...		
	Object Feat.	Road Feat.	Signal Feat.
Controllers	NO	NO	NO
Junctions	NO	NO	NO
Object Feat.	NO	YES	NO
Road Feat. (Lanes)	NO	NO	NO
Signal Feat. (Ref.)	NO	YES	NO

Table 2.1: Relation of geometrical alterations of road network features and their possible effects on other features

...of Object Features

Object features are independent to most other elements in a road network (except their parent road feature) and geometrical alterations do not have an effect of any of the other elements. To alter any other element, the geometrical alteration would have to be so great that it would already be considered as a completely new object feature.

...of Road Features

As road features are the parent elements of object features and signal features, geometrical alterations of a road feature always cause an update of its child elements' geometries.

...of Signal Features

What applies for geometrical alterations of object features is also valid for signal features. They are less independent from other elements in a road network but geometrical alterations would have to be too great for them too, to alter any other element. Such signal features would also be regarded as additional features in the new road network.

2.3.2 Effects of Topological Alterations

Can have an effect on other...	Topological Alterations of...		
	Object Feat.	Road Feat.	Signal Feat.
Controllers	NO	INDIRECT	YES
Junctions	NO	YES	INDIRECT
Object Feat.	NO	YES	NO
Road Feat. (Lanes)	NO	YES	NO
Signal Feat. (Ref.)	NO	YES	YES

Table 2.2: Relation of topological alterations of road network features and their possible effects on other features

...of Object Features

Similar to geometrical alterations topological alterations of object features do not alter any other element in a road network.

...of Road Features

Topological alterations of road features can affect all other elements in a road network. Very often topological alterations happen together with geometrical alterations. Therefore, they share nearly the same effects which are already described before in section 2.3.1. In addition to these effects controllers and junctions can be affected, too. Controllers are not directly affected by topological alterations of road features but are indirectly connected to them via the signal features of the altered road features. And as junctions describe the topological connections between road features, the direct effects on them are natural.

...of Signal Features

The effects of topological alterations of signal features on controllers and junctions is vice versa to the effects of topological alterations of road features. Controllers are directly connected to signal features while junctions only link to the controllers which are valid for them.

Object features and road features are not affected but signal feature references can be. If they refer to a signal feature whose parent feature road is altered, the reference is no longer valid and has to be removed.

2.3.3 Effects of Semantic Alterations

Can have an effect on other...	Semantic Alterations of...		
	Object Feat.	Road Feat.	Signal Feat.
Controllers	NO	NO	YES
Junctions	NO	NO	INDIRECT
Object Feat.	YES	NO	YES
Road Feat. (Lanes)	YES	NO	YES
Signal Feat. (Ref.)	YES	NO	YES

Table 2.3: Relation of semantic alterations of road network features and their possible effects on other features

...of Object Features

Only semantic alterations of objects features can have an effect on other elements in a road network, too. Road features and road feature lanes can be topologically altered, for example, if the object feature is a pictogram which describes the allowed turning directions of a road feature lane in a junction (see figure 2.3). This may require an update of the road features (or road feature lanes) in the junction, if the road mark changes from, for example, straight-right to right-only. This example would eliminate a connecting road feature (lane) in the junction which connects the incoming road feature lane with the straight ahead located road feature.

Other object features, signal features and their references can be also affected. Any of these elements that describe the topology of the road network as explained before. Such alterations also cause an update of the other elements which are valid for the same road feature and road feature lanes.

...of Road Features

Semantic alterations of road features do not have any effect on other elements in a road network. Alterable attributes are, for example, the road name and other meta data which may be important for the driving simulation but is not connected to other elements in any way.

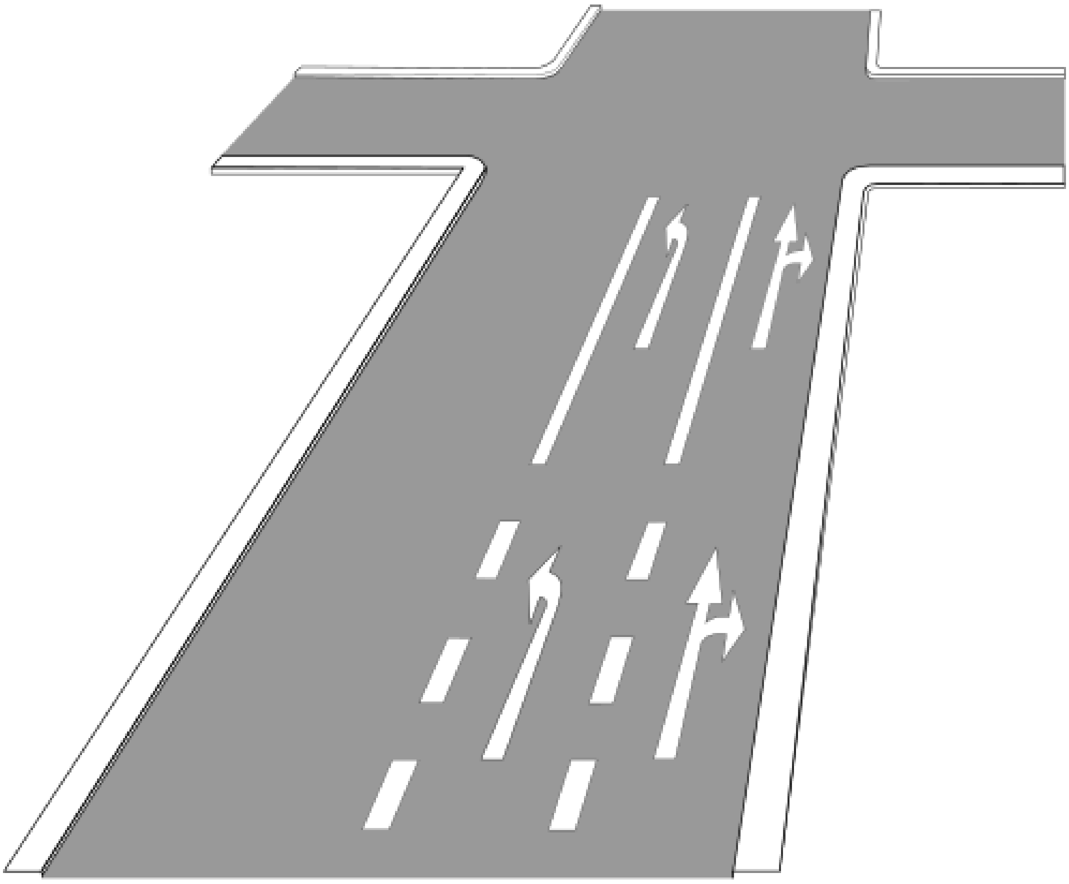


Figure 2.3: Pictograms on the road to describe the allowed driving directions [36]

...of Signal Features

Just like semantic alterations of object features such alterations of signal features have a much greater effect on other elements in a road network than geometrical and topological alterations. They can affect not only the controllers and junctions as for topological alterations but also other object features, road features, and signal features for the same reason object features do, which is described before.

2.3.4 Order of Merging

As road features and signal features are the only features which can affect all other types of features in a road network, it is the best practice to begin the merging process with them. However, a signal feature is always a child entry of a road feature and should be accordingly handled after the road feature. Figure 2.4 visualizes this order in more detail based on the results of sections 2.3.1, 2.3.2, and 2.3.3.

Thereby, it is probable that updates on other features may be redundant. An example for that could be a new road feature in a junction, describing a left turn which did not exist before. New object and signal features, for example, pictograms and traffic lights,

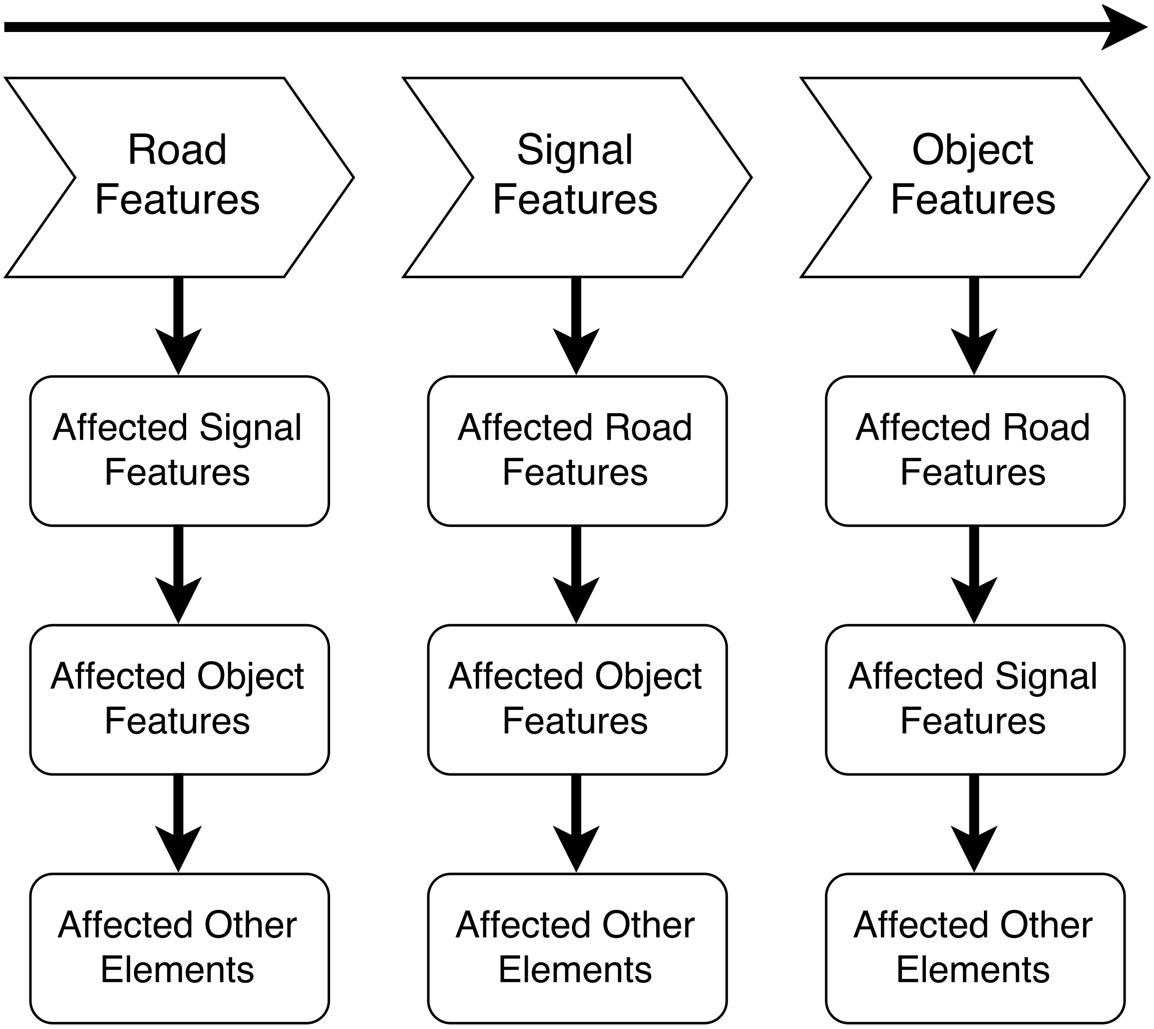


Figure 2.4: Detailed overview of the order of merging

may lead to the same update: that a new left turn must be created. On the other hand, the update of the road feature may require the update of the previous road mark to match with the new possibility to turn left on that road lane.

To avoid redundant updates, it is helpful to compare the required updates with the identified alterations from section 2.2. In the here described example there could be identified alterations for the road marks and traffic lights already. The merging algorithm (which is schematically visualized as a flow chart in figure 2.5) can then take these alterations into account for the updates on the different features. This also increases the accuracy of the update as it is possible to compare the update, which the algorithm would do on its own with the external update of the updating road network. If the merged alteration is a topological alteration, the preceding road feature will be also updated as signal features and object features usually indicate the further course of the road feature lanes.

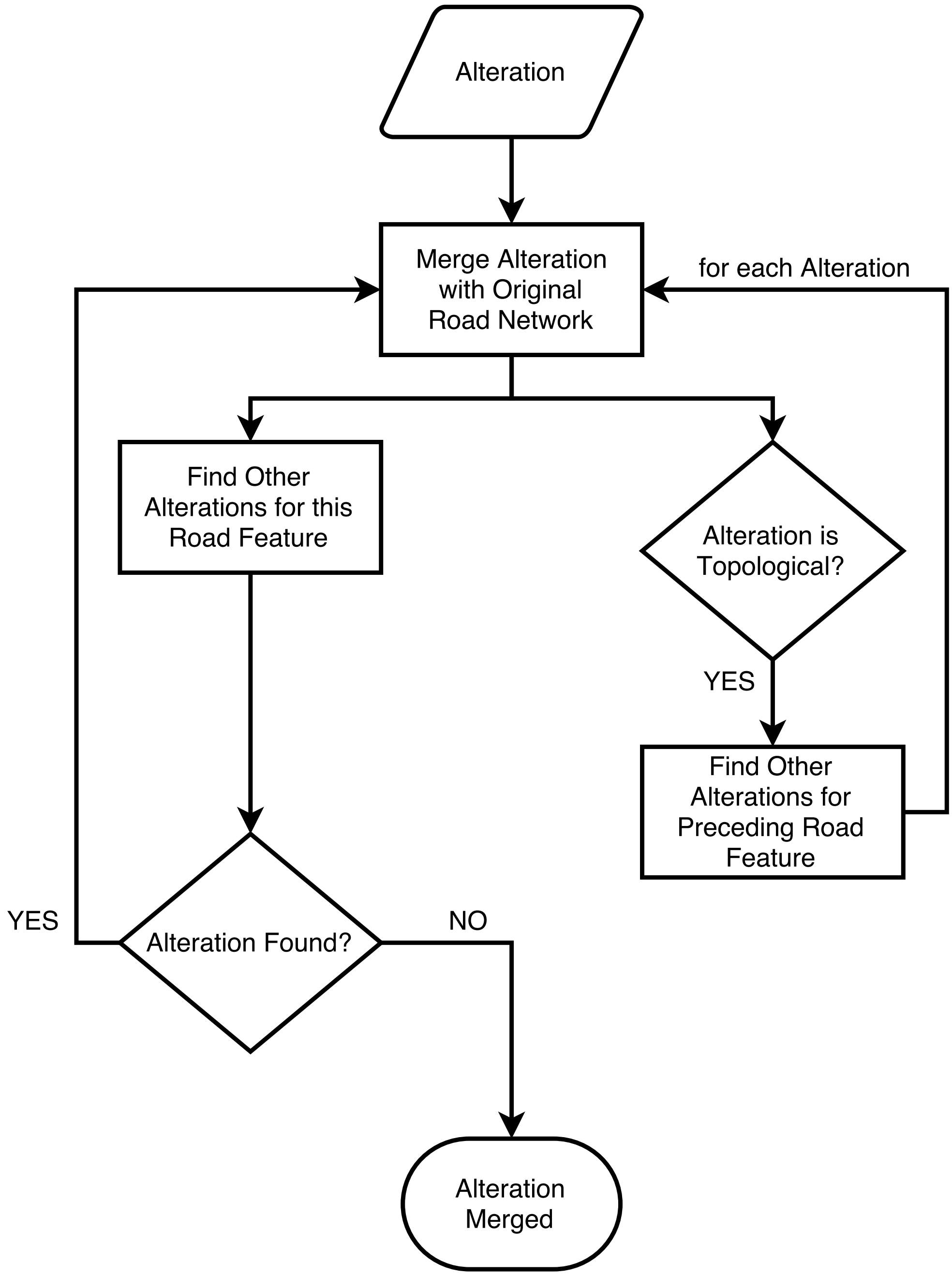

Alteration
Merge Alteration with Original Road Network
for each Alteration
Find Other Alterations for this Road Feature
Alteration is Topological?
YES
Find Other Alterations for Preceding Road Feature
YES
Alteration Found?
NO
Alteration Merged

Figure 2.5: Overview of the process of merging alterations by taking related alterations from the current road feature and its predecessor road feature into account

After merging the road features the process continues with merging the signal features next. Then the object features which affect only a few other feature types in a road network and whose alterations' effects on other feature types are in the best case already handled before by merging alterations of road features and signal features.

2.3.5 Merging Road Features

This section describes the methods for merging road features. As road features can affect all other elements in a road network and are parent elements of object features and signal features, this section is the most important part of the merging process.

Topological Alterations

Topological alterations of a road feature can affect all other kinds of elements in a road network. There are two types of topological alterations: connections to new road features / road feature lanes and removal of such connections. These two types often appear together when a connection to one road feature is removed as another road feature has replaced it.

In the first case the road features' topological attributes (of the affected road features) have to be updated as well as the connections which are described in the junctions. The signal features and object features which describe the road feature's topology have to apply the new topology, too. Alterations of these features can be skipped later as they are already handled here. Last but not least the controllers and junction controllers have to be checked whether they are still compatible to the altered topology. More about that will be discussed in the section about additions of road features later.

The second case deals with the removal of connections. This can be caused by a removed road feature in a junction. For example, if it is no longer allowed to turn left at a junction coming from the altered road feature, then the following connecting road feature, which describes the left turn, must be removed. Therefore, in this case the following road features must be compared with the new road feature's topology. In case the affected connecting road feature still exist, it will be removed with all its child elements as long as these are not specifically valid for this road feature only. For example, a signal feature with a speed limit may be valid for the other connecting road features, too. The signal feature will then be handled as an addition while the rest, including also the junction and controller entries for this road feature, is removed.

Geometrical Alterations

To merge geometrical alterations of a road feature, other road features next to the altered road feature have to be inspected. First of all it can be that the geometrical alteration affects the start or end point of the road feature. If the road feature has a preceding or succeeding road feature at this point, it has to be ensured that the geometries of the road features form a continuous function. In case the road feature's topology has been altered too, the geometrical alteration may have just been altered to match the altered predecessor or successor road feature.

However, in every case it has to be assured that the road feature's altered geometry does not intersect another road feature's geometry. Naturally road features in junctions can intersect each other if, for example, a left-turning road feature intersects a straight road feature from the opposite side of the junction, but there are also cases in which an intersection of the road features' geometries must not happen. For example, neighbouring road features with the same predecessor road feature must not intersect each other as this could cause traffic accidents when two vehicles have to use the same trajectories. Therefore, the altered road feature geometry has to be compared with its neighbouring road features' geometries, which is also shown in algorithm 9.

Algorithm 9 Comparing road features' geometries for intersections before merging (pseudo code)

```
Require: Altered road feature ($rfAltered$) and the list of road features of the new road
    network ($rfList$)
Ensure: The geometry of $rfAltered$ does not intersect neighbouring road features'
    geometries from $rfList$
for $i = 0, \ldots, rfList.length$ do
    if $rfAltered.predecessor == rfList[i].predecessor$ and $rfAltered \neq rfList[i]$
    then
        if intersects($rfAltered.geom, rfList[i].geom$) then
            return true
        end if
    end if
end for
return false
```

To compare the road features' geometries the algorithm first iterates through the list of road features of the new road network. By using the new road network instead of the original road network it can be ensured that updates on neighbouring road features are already taken into account for the comparison. When the algorithm found another road feature with the same predecessor road feature as the altered road feature, the geometries of the two road features are compared for any intersections.

In case that an intersection has been found, the algorithm stops and returns *true.* This is considered as a fatal error in the new road network and will cause the application to stop the merging process, informing the user that the new road network has to be corrected before an update of the original road network can be accomplished.

If no intersection has been found, the algorithm returns *false* and the next step, the comparison of the geometrical connections to the predecessor and successor road features from the new road network are checked. They should describe a continuous line in the best case. However, it can happen that there are minor differences in the geometries. In that case the road feature cannot be merged as the new road network obviously contains severe errors in its road connectivity. The application then stops and informs the user that the road features are not connected properly.

So if no unwanted intersection or gap in the road network has been found for this geometrical alteration, the road feature's geometry can be updated. As this also affects the object features and signal features of the altered road feature, their geometries have to be updated, too. And to ensure that no redundant updates are made, the algorithm 10 will also search for identified geometrical alterations of these features.

Algorithm 10 Updating object features' and signal features' geometries of an altered road feature (pseudo code)

```
Require: Altered road feature ($rfAltered$) and the list of identified alterations of object features ($listAlteredObjects$) and signal features ($listAlteredSignals$)
for $i = 0, \ldots, rfAltered.objects.length$ do
    $alteredObject = null$
    if $listAlteredObjects.contains(rfAltered.objects[i])$ then
        $alteredObject = listAlteredObjects.$get$(rfAltered.objects[i])$
        if $alteredObject.$isGeometricalAlteration() then
            $rfAltered.objects[i].geometry = alteredObject.geometry$
            $alteredObject.$setGeometricalAlteration(false)
        else
            recalculateGeometry($rfAltered.objects[i]$)
        end if
    else
        recalculateGeometry($rfAltered.objects[i]$)
    end if
end for
for $i = 0, \ldots, rfAltered.signals.length$ do
    $alteredSignal = null$
    if $listAlteredSignals.contains(rfAltered.signals[i])$ then
        $alteredSignal = listAlteredSignals.$get$(rfAltered.signals[i])$
        if $alteredSignal.$isGeometricalAlteration() then
            $rfAltered.signals[i].geometry = alteredSignal.geometry$
            $alteredSignal.$setGeometricalAlteration(false)
        else
            recalculateGeometry($rfAltered.signals[i]$)
        end if
    else
        recalculateGeometry($rfAltered.signals[i]$)
    end if
end for
```

The algorithm iterates through all object features and signal features of the altered road feature. It searches for any identified geometrical alterations for them and based on that either applies the geometrical update or recalculates their geometry based on the road feature's reference line using the OpenDRIVE standard's [15] track coordinate system. Then it checks whether the object feature or signal feature has been identified as geometrically altered. In that case the state of the geometrical alteration to *false* so that the application will skip this alteration in the next steps as it has been handled here already.

As geometrical alterations of object features and signal features do not affect any other elements in a road network, there is no need to go any further here.

Semantic Alterations

While topological and geometrical alterations of road features usually affect other features too, semantic alterations do not affect any other features. Therefore, these updates only have to be applied to the road feature's attribute solely.

Additions

New road features can be added to the road network as long as their geometries' start and end points match with other existing road features in the resulting road network. If the geometries differ a bit, an adjustment of the added road feature can be performed. This is the difference towards geometrical alterations of existing road features which is described before. However, greater differences require a more intense adjustment which is not performed in this master thesis but will be discussed in section 5.2.3.

Minor geometrical adjustments must only be performed on the added road feature. Altering the existing predecessor or successor road feature's geometry is not desired as the so far existing road network shall always be seen as correctly modeled as long as it is not altered specifically by the new road network. This will be also discussed in the second case study in section 3.6.2 where an existing road network is extended by another new road network.

Naturally it can happen that there is not only one single new road feature added but a sequence of road features instead. One option (which is visualized in figure 2.6) to adjust the geometrical continuity of the road features would be to move all of these road features so that the one road feature gets attached to the existing road feature correctly. However, the new road features may be connected to a whole road network which is connected to the existing road network at multiple locations. Moving all the road features may create a new geometrical gap at another location if the new road network is not modeled as precisely as the existing one.

Therefore, only the start or end part of the road feature's geometry that connects to the existing road network will be adjusted while the rest of the new road network stays untouched. Even though this is a functional solution, there are still cases in which the result will not be acceptable for the resulting road network. Road feature lanes should always describe a line with a curvature that is manageable for most vehicles. Altering this curvature to such a great extent that it would no longer be drivable must be prevented.

The next steps of the merging process for additions are also visualized in figure 2.7 as a flow chart.

So to adjust the new road feature's geometry adequately the difference between the start or end point (whichever is closer to the existing road network it connects to) should be less than ten percent of the complete geometry's length. Greater differences increase the risk of altering the curvature too much and will result in skipping the new road feature to be merged with the existing road network.

If the difference was small enough for a riskless adjustment, the start or end point of the new road feature's geometry will be set to the start or end point of the existing road

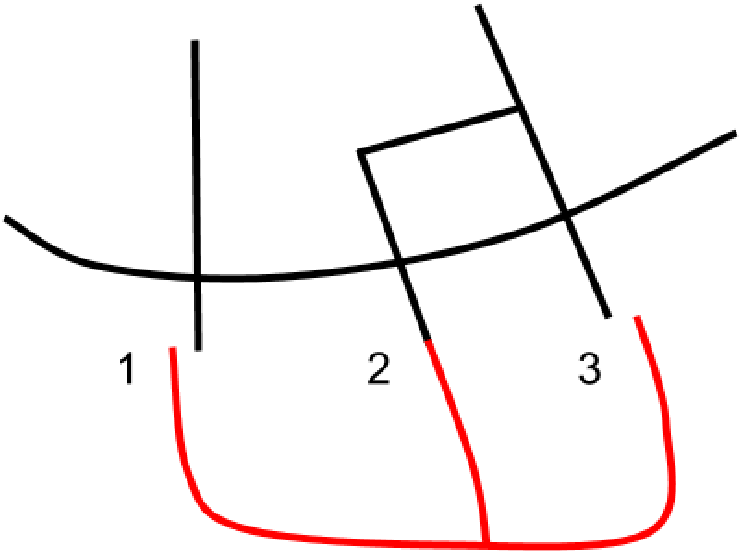


Figure 2.6: Visualization of how a new road network's geometries (red) do not match with the geometries of an existing road network (black) and moving the new road network's geometries to the left or right would cause only greater differences at the other connections (1-3)

feature's geometry it connects to. The geometrical functions which describe the road feature's geometry then have to be recalculated.

The recalculation is done by taking the first or last geometry entry (whichever shall connect to the existing road network) and redraw its line from its original beginning / end to the contact point of the existing road network.

After the road feature's geometry has been adjusted successfully, all possible child elements' geometries (object features and signal features) of the new road feature have to be recalculated, too. This is the same process as described before about the merging of geometrical alterations of road features.

If the road feature's geometry keeps unchanged, its child elements' geometries can naturally be adapted without further changes.

Furthermore, it has to be controlled whether there are any other object features or signal features that the new road feature has to refer to. This is especially the case for new road features in junctions. For example, if there is a traffic light which the road feature passes, it has to be either referred to via a signal feature reference (in case another road feature is the parent element of this signal feature) or the signal feature has to be added to the new road feature. The latter case can happen for signal features whose parent road feature got removed by an update while the signal features remain.

For new road features in a junction the controller and junction elements in the road network have to be updated, too. The same applies for the predecessor or successor information of the road feature the new road feature connects to.

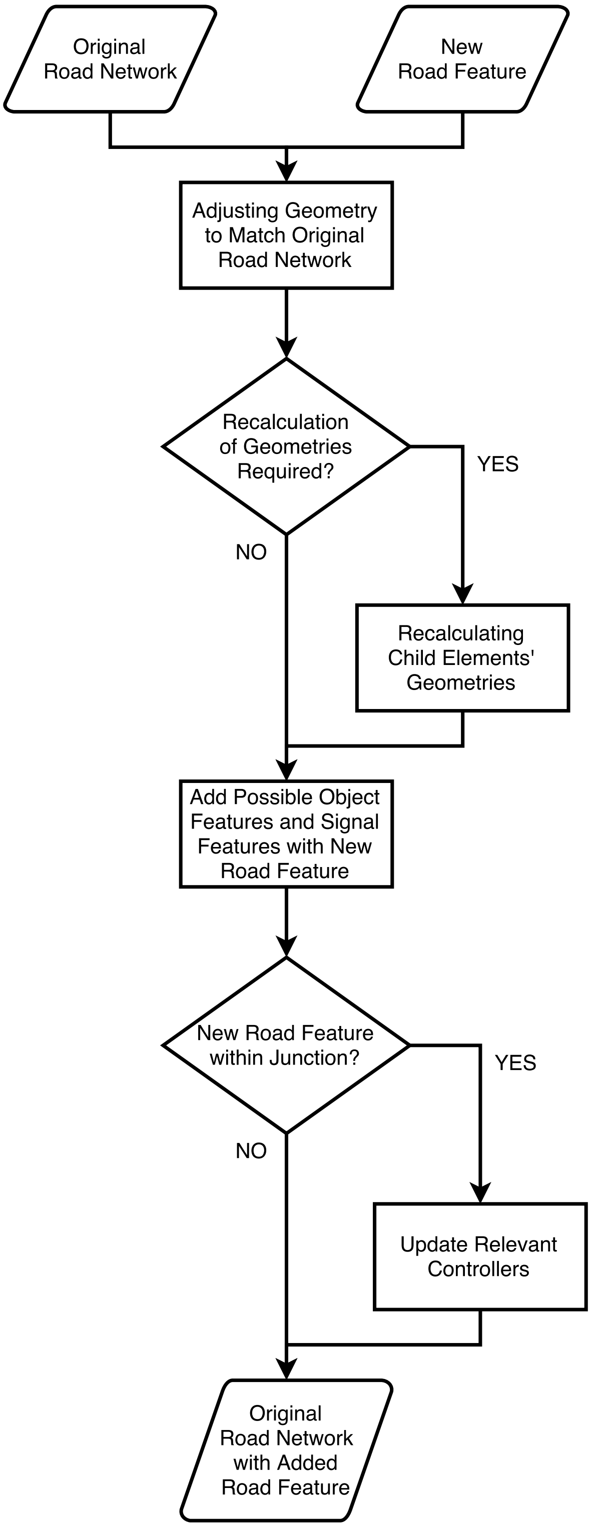

Original
Road Network
New
Road Feature
Adjusting Geometry to Match Original Road Network
Recalculation of Geometries Required?
YES
NO
Recalculating Child Elements' Geometries
Add Possible Object Features and Signal Features with New Road Feature
New Road Feature within Junction?
YES
NO
Update Relevant Controllers
Original Road Network with Added Road Feature

Figure 2.7: Overview of the merging process for additions of road features as a flow chart

Removals

The process of removing road features from the existing road network by an update is similar to adding a new road feature to it. All connections to other road features must be removed as well as all child elements as long as there is no other road feature referring to them (only valid for signal features). In that case the other road feature will become the parent road feature of this signal feature.

2.3.6 Merging Signal Features

Alterations of signal features can also affect all other elements in a road network. Therefore, the merging of these alterations is highly important, too. However, in the best case most or at least many of the alterations have been handled by merging the road features (section 2.3.5) before.

Geometrical Alterations

The method for the identification of geometrical alterations of signal features (section 2.2.1) restricts the possible difference in a signal feature's geometry so far that it cannot cause any effect on other elements in the road network. Therefore, the merging of geometrical alterations of a signal feature works by simply applying the new geometry to the existing signal feature's geometry.

However, as the OpenDRIVE standard [15] uses a track coordinate system for the signal features' locations, there are some transformations required to apply the new geometry to the signal feature. Algorithm 11 shows how the new geometry can be applied to the signal feature.

Algorithm 11 Updating a signal feature's geometry (pseudo code)

Require: Parent road feature (rf) of the altered signal feature ($sfAltered$) and the new geometry of it ($geomNew$)

Ensure: $geomNew$ transformed into track coordinate system and applied to $sfAltered$

```
for $i = 0, \ldots, rf.laneSections.length$ do
    if $rf.laneSections[i].s <= sfAltered.s$ then
        $nearestLaneSection = rf.laneSections[i]$
    end if
end for
$trackCoordinates$ = calculateTrackCoordinates($nearestLaneSection$, $geomNew$)
$sfAltered.s = trackCoordinates.s$
$sfAltered.t = trackCoordinates.t$
```

In the process of identifying any geometrical alterations the geometries are compared using a projected spatial reference system. The hereby calculated coordinates can then be used to find the nearest lane section of the parent road feature. To do that the s-coordinates of the lane sections and altered signal feature are compared. The nearest lane section is the one with the closest value to the s-coordinate of the signal feature but still smaller than or equal to it. This step is necessary to get the shape of the road feature at the nearest location to the signal feature, which is stored in the lane section elements. After the nearest lane section has been found, the (s,t)-coordinates of the signal feature can be calculated and applied as the new geometrical attributes to the signal feature.

Topological Alterations

Before any topological alteration can be merged, it is crucial to first update its geometry. Even if the signal feature's geometry may remain unchanged in the projected spatial reference system, it has very probably changed in the track coordinate system of its new parent road feature. Therefore, an update of the signal feature's geometry is mandatory before the topological update.

Topological alterations of signal features can cause effects on controllers and by that also on junction controllers as already shown in table 2.2. However, they can naturally only affect controllers, if they were contained by a controller already before. Controllers bundle signal features that are valid for the same (and opposite) driving direction in junctions. So if a signal feature is shifted to another parent road feature, this road feature may follow another direction and the controllers have to be updated, too.

Updating the controllers includes to test whether the altered signal feature is still valid for its previous controller or has to be removed from it and whether it has to be added to another controller. To not slow down the whole application by checking the controllers every time again when a signal feature has been altered, the controllers of the whole resulting road network are removed and recreated completely which does not influence the application's performance if only run once.

As junction controllers refer to the controllers which are effective for the respective junction, they are indirectly affected by these alterations, too. They are also getting removed and recreated by the application as this is a quick and easy solution.

Semantic Alterations

A signal feature has numerous attributes which can be altered. In most cases these semantic alterations do not affect any other features in the road network. For example, merging the value of a speed limit signal feature can be done simply by just changing the *value* attribute of the signal feature.

However, a semantic alteration of a signal feature can affect many other features in a road network, if the signal type changes. Signal features very often describe the topology of a road network, for example, by permitting or prohibiting specific turning directions for road lanes in junctions.

Figure 2.8 shows three examples of signal features from the German StVO [33], of which two describe the topology of the road network while the other is only valid for the traffic rules and not the topology. This example shows that specific rules are required for the update which may vary from country to country.

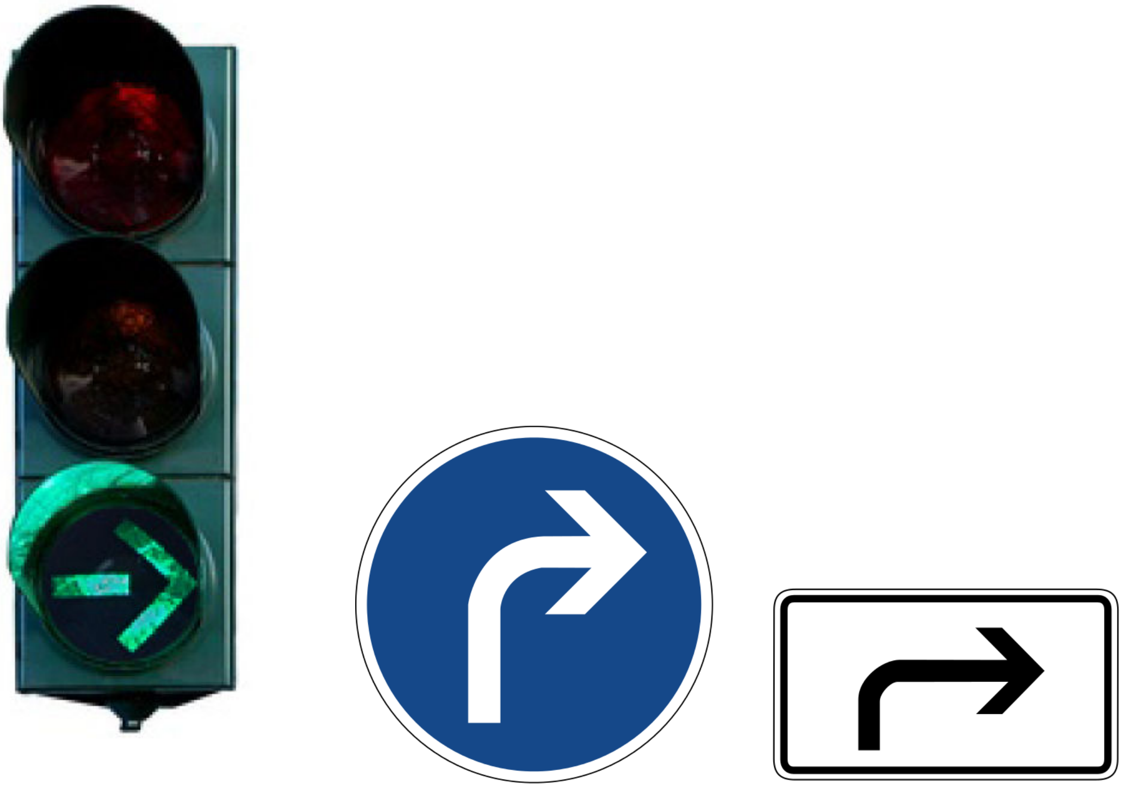


Figure 2.8: While the left and middle signal types describe the topology of the road network in the junction, the right signal type is only used for traffic rules though it describes a clear topological direction, too [37] [38] [39]

In case the semantic alteration of the signal feature causes a topological alteration in the road network, the topology of the road features must be updated.

Unfortunately, this is not possible, if no topological alteration for or addition / removal of the affected road feature has been identified. Therefore, the algorithm would return a warning that the update is not practicable. This restriction is further discussed in the following paragraph and in more detail in section 5.2.1.

For a signal feature like the one in the middle of figure 2.8 at a road feature which leads into several directions the application will be able to continue successfully by removing the respective connecting road features. However, in case the new signal feature does not remove a previously allowed turning direction but adds a new one, this becomes a bit tricky. The new connecting road features must exist in the new road network so that they can be added accordingly to the resulting road network. If they do not exist, it is impossible to create them based on the new signal feature only. The application would skip this alteration of the signal feature and inform the user about the noticed update error.

Additions

The same process described before is applied to new signal features. Further actions are not necessary.

Removals

For signal features that are getting removed by the update it must first be checked whether they are referred to by signal references of other road features and by (junction) controllers. These elements must be removed, too. Controllers only have to be removed, if the removed signal feature was the only element of the controller or the controller would then be a duplicate of another controller.

2.3.7 Merging Object Features

The update processes of object features is very similar to the ones of signal features, which are described before. The following sections give a brief overview of the update processes of object features and where any differences to the merging of signal features are.

Geometrical Alterations

Object features which have been altered geometrically only cause effects on the object feature itself. And as the object features' locations relative to their parent road features are stored exactly as the locations of signal features are stored, algorithm 11 can be used in the same way for the object features, too.

In contrast to signal features there are no references to or controllers for object features. Therefore, the merging process is complete by updating the object feature's geometry solely.

Topological Alterations

Updating topological alterations of object features does not differ a lot from signal features, too. Here it should be mentioned again, that the object feature's geometry must be also updated in this process as the track coordinate system of the OpenDRIVE standard [15] has been probably changed.

Semantic Alterations

Object features in a road network for driving simulations are often trees, benches, etc. These object features only serve the purpose for setting up the simulation's virtual environment and do not have any effect on the road network's connections.

However, road marks are also stored as object features and they can describe the topology of its parent road feature or its following one. Therefore, semantic alterations of object features must be merged in such cases similarly to semantic alterations of signal features (which is described before), if they alter the topology of the road network.

Alterations that do not have an effect on the topology can be simply applied to the object feature's attributes, respectively.

Additions

New object features which are added to the original road network are handled similarly to semantic alterations of object features and can cause the same effects on other elements in a road network for driving simulations.

Removals

Object features can be removed from its parent road feature without causing any effects on other elements in a road network for driving simulations.

2.4 Summary

In this chapter the methods used for the automated update of road networks for driving simulations are described. It is divided into the three major sections: finding corresponding features, identification of different alterations between them, and the final merging of these alterations. These parts are further divided into the specific methods for the different feature types. While the methods of the first two sections can work completely automated, some of the merging methods require manual interaction or data correction by the user. However, these methods still describe a high-level automation of the update process.

3 Implementation

The developed algorithms are implemented in a prototype as a proof of concept. It uses the OpenDRIVE standard [15] for describing the road networks and further software libraries which are summarized in the following section. Thereafter, the implementation of the algorithms is presented.

3.1 Used Software Libraries

The prototype is implemented solely in Java. The following software libraries have been used for the implementation (in alphabetical order).

GeoTools

The GeoTools library[3] already contains a great number of geo-spatial calculations, for example, map matching or spatial filtering.

JAXB

JAXB[4] enables reading and writing of XML-based files. The OpenDRIVE data format is based on XML (the file type is *.xodr*) [15]. Therefore, this library can be used to access the road networks in their original format.

JPA Hibernate

To persist the road networks in a database for making working with the data easier, the JPA Hibernate library[5] has a good selection of functions. It can also store geo-spatial data (e.g. in a PostGIS[6] database) with its spatial extension.

OpenDRIVE Modules

The DLR has already implemented a rich library of different modules for the work with road networks in the OpenDRIVE standard [15]. These include persisting modules to store road networks in local files or databases, enhancement modules to enrich road networks with new signal features [40], fillets for the 3D visualization, and validation

---

[3] `http://www.geotools.org/`

[4] `https://jaxb.java.net/`

[5] `http://hibernate.org/orm/`

[6] `http://postgis.net/`

modules to ensure the compatibility to the OpenDRIVE standard [15] and the topological connectivity of the road network.

Furthermore there are also modules to generate feature geometries or complete road networks automated. A result of the latter one is used in the second case study (section 3.6.2) to update another road network.

This work uses only three of the available modules: the persisting module to store the resulting road network in a local file or database, the module to enrich road networks with new signal features [40] to add and merge signal features, and the auto-generation module to generate geometries which follow the OpenDRIVE standard [15]. Validation modules are not used as already the underlying data for the case studies do not match all validation criteria. However, this does not affect the results of this work as this is only a proof of concept.

3.2 Usage of New Classes

As the existing software libraries are not built to handle updates of road networks, this master thesis uses some new classes to store features of the road network and identified alterations.

The following two sections 3.2.1 and 3.2.2 describe in detail what these new classes do.

3.2.1 Storing Features

The OpenDRIVE standard [15] uses a mix of a tree and graph structure for the different features. Unfortunately the relation from a child element to its parent is not set as an attribute for the child. For example, a road feature knows which signal features it contains. However, the other way round the signal feature does not know to which road feature it belongs to. Only by running through the whole tree of the road network data structure it is possible to find the corresponding parent element.

Consequently, this master thesis uses some new classes specifically to eliminate this difficulty but also for storing results from finding alterations together with the related features. In figure 3.1 the UML class diagram is shown for these new classes.

All features in a road network contain some attributes like names etc. Therefore, the *Feature* class is independent of any specific feature type and holds a dynamic list of any attributes given to it in form of a *Map* object of the *java.util.Map* class. With the *get*-functions of the *Map* class it is possible to quickly and easily retrieve the attribute values of the feature.

The two classes *PointFeature* and *RoadFeature* are a modified version of the object feature and signal feature (as point features) and road feature of the OpenDRIVE standard [15]. They both extend the *Feature* class to store these specific features only.

While the *PointFeature* class stores an indicator whether it represents a signal feature or object feature and its parent road feature's ID, the *RoadFeature* class only has to store the ID of its representing road feature.

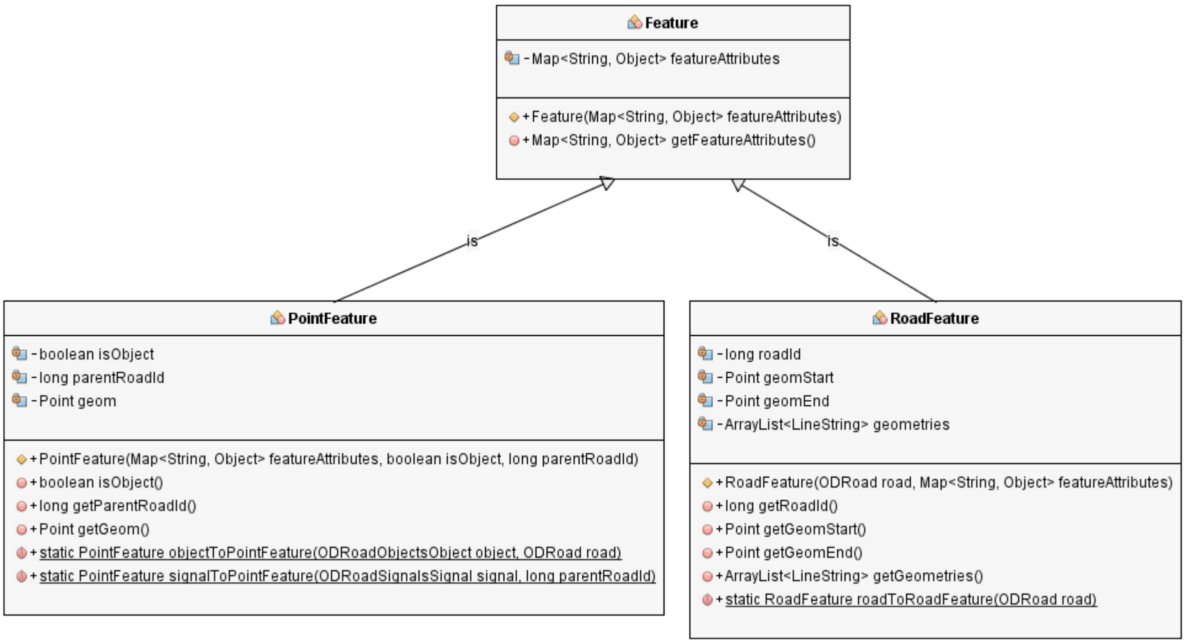

Feature
-Map<String, Object> featureAttributes
+Feature(Map<String, Object> featureAttributes)
+Map<String, Object> getFeatureAttributes()
is
is
PointFeature
-boolean isObject
-long parentRoadId
-Point geom
+PointFeature(Map<String, Object> featureAttributes, boolean isObject, long parentRoadId)
+boolean isObject()
+long getParentRoadId()
+Point getGeom()
+static PointFeature objectToPointFeature(ODRoadObjectsObject object, ODRoad road)
+static PointFeature signalToPointFeature(ODRoadSignalsSignal signal, long parentRoadId)
RoadFeature
-long roadId
-Point geomStart
-Point geomEnd
-ArrayList<LineString> geometries
+RoadFeature(ODRoad road, Map<String, Object> featureAttributes)
+long getRoadId()
+Point getGeomStart()
+Point getGeomEnd()
+ArrayList<LineString> getGeometries()
+static RoadFeature roadToRoadFeature(ODRoad road)

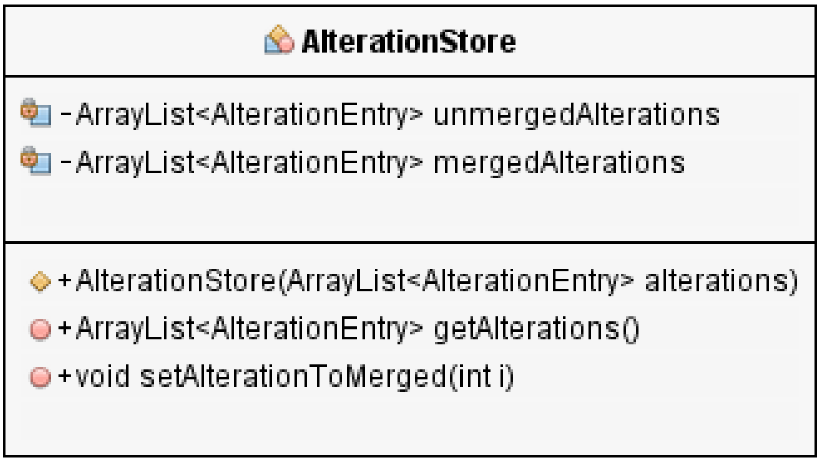
Figure 3.1: UML class diagram of the new feature classes

Both classes, *PointFeature* and *RoadFeature*, also store the feature's geometry. This functionality is not covered by the super class *Feature* as the way of storing the geometry differs for point features and road features. The *PointFeature* class stores the actual geometry while the *RoadFeature* class stores a list of the road feature's geometries. The reason for this kind of storing is that in the OpenDRIVE standard [15] the road's geometry is described via successive, so called "planview" geometries which are usually not one single geometry (e.g. a *MultiLineString*). Furthermore, the *RoadFeature* also stores the start and end point of the representing road feature. These points are later used for finding corresponding road features in the original road network.

3.2.2 Storing Alterations

Furthermore, this master thesis also uses new developed classes to store the identified alterations with their corresponding features of the original and new road network. These classes are shown in figure 3.2 and 3.3 as UML class diagrams.

To keep track of the identified alterations the *AlterationStore* class contains two lists of *AlterationEntry* objects (which are described in the next paragraph) for merged and unmerged alterations. It has two functions to return the list of unmerged alterations or to set an unmerged alteration at a given index to merged. The unmerged alteration is thereby moved from the list of unmerged alterations to the list of merged alterations. This way it is ensured that no information is getting lost and no alteration is forgotten during the merging process.



```
AlterationStore
-ArrayList<AlterationEntry> unmergedAlterations
-ArrayList<AlterationEntry> mergedAlterations
+AlterationStore(ArrayList<AlterationEntry> alterations)
+ArrayList<AlterationEntry> getAlterations()
+void setAlterationToMerged(int i)
```

Figure 3.2: UML class diagram of the alterations store classes

The *AlterationEntry* class stores the alterations for each feature of the original and new road network as a *Feature* (see section 3.2.1 before). The *AlterationEntry* class can be constructed with *PointFeature* objects or *RoadFeature* objects. Both constructors will compare the two given features directly to identify any alterations. The different, identified alteration types are stored as booleans. When a single alteration has been merged and in consequence set to unmerged, the other alteration types are still in their former state and will not be skipped this way.

In case the feature of either the original or the new road network is not given as an attribute, this indicates a removal or addition instead of another kind of alteration. While removals are stored via an attribute in the *AlterationEntry* class, additions are handled separately at another place in the application.

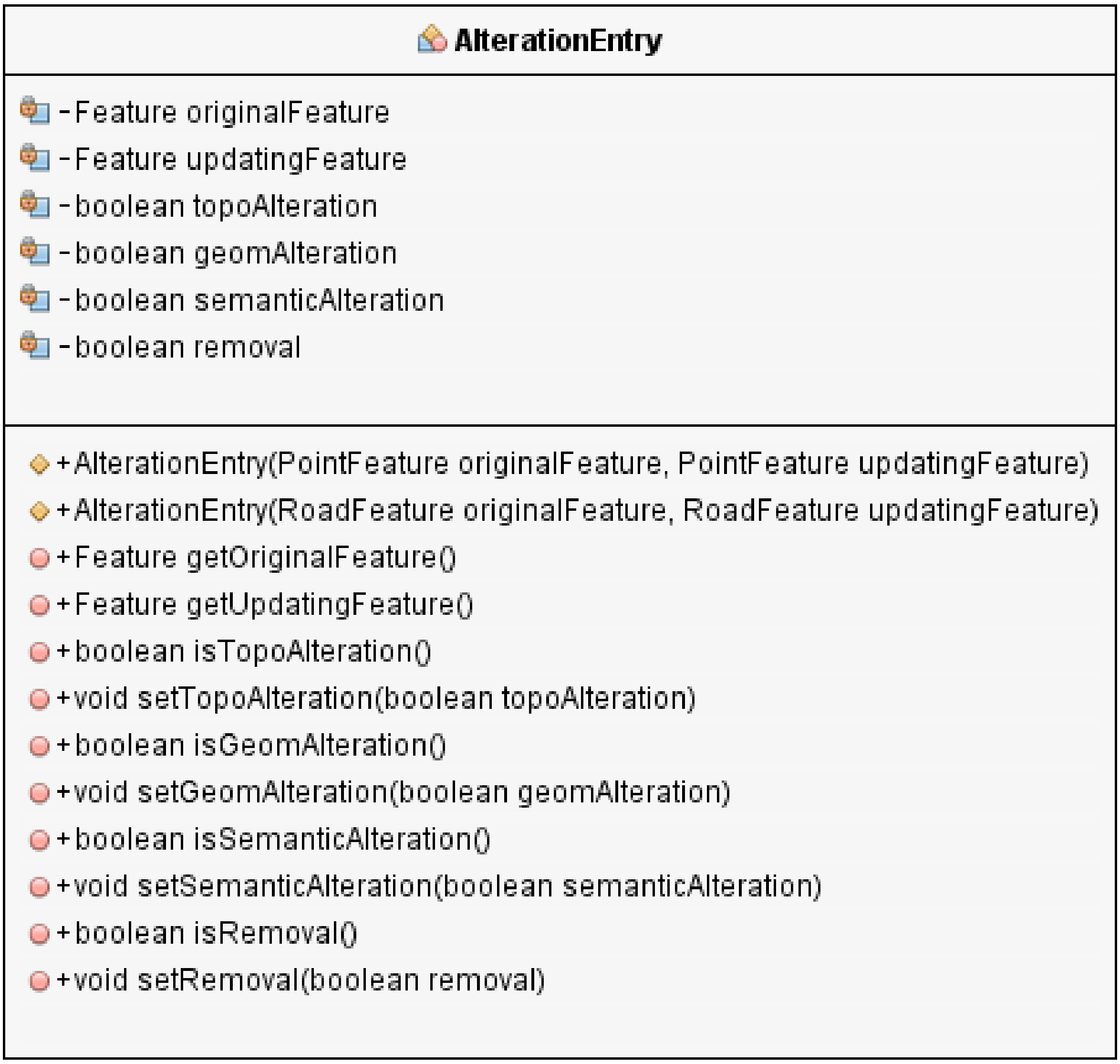
3.3 Finding Corresponding Features in Road Networks

The implementation of the algorithms from section 2.1 can be done precisely like they are described in pseudo code. Only finding corresponding road features is more complex.

The geometries of road features can be stored differently, for example, as arcs or lines, and have to be transformed to comparable *LineString* objects of the GeoTools library. After this transformation the implemented algorithm operates exactly as described above.

To skip this task for the next methods, the *LineString* objects are stored with the road feature in the new class, which was already described in section 3.2.1.

Furthermore, the connections between road features to each other is more complex as each road feature can have more than one preceding or succeeding road feature respectively. However, this is described in more detail in the section 3.4.2 in combination with the identification of alterations of these topological connections.



AlterationEntry

-Feature originalFeature
-Feature updatingFeature
-boolean topoAlteration
-boolean geomAlteration
-boolean semanticAlteration
-boolean removal

+AlterationEntry(PointFeature originalFeature, PointFeature updatingFeature)
+AlterationEntry(RoadFeature originalFeature, RoadFeature updatingFeature)
+Feature getOriginalFeature()
+Feature getUpdatingFeature()
+boolean isTopoAlteration()
+void setTopoAlteration(boolean topoAlteration)
+boolean isGeomAlteration()
+void setGeomAlteration(boolean geomAlteration)
+boolean isSemanticAlteration()
+void setSemanticAlteration(boolean semanticAlteration)
+boolean isRemoval()
+void setRemoval(boolean removal)

Figure 3.3: UML class diagram of the alterations classes

3.4 Identification of Alterations in Road Networks

This section describes how the methods from section 2.2 are implemented. They follow mostly the pseudo code in its structure but some of them also have some differences to it, for example, the identification of topological alterations of road features which is more complex in the OpenDRIVE standard [15] than it is stated in algorithm 6.

3.4.1 Geometrical Alterations

Algorithm 5 already describes very precisely how geometrical alterations for different feature types can be identified. It compares the geometries directly without iterating through their coordinates or something similar as a simplification.

Fortunately, the GeoTools software library offers a handy function to compare geometries for equality with the method *equalsExact*. This function is not only faster than normal *equals* functions but compares the coordinates and the underlying geographic reference system of two geometries [41].

The *equalsExact* function can identify the equality of geometries even if they use different reference points along their shape so the split geometries of a road feature could also be merged to a single geometry for the comparison. However, the road feature's geometry is described as mathematical functions in the OpenDRIVE standard [15]. Therefore, the geometry may still be the same but the mathematical functions may have changed, for example, from a combination of an arc and a line to a cubic polynomial function. To update these descriptions the implemented algorithm compares all geometries of a road feature individually.

3.4.2 Topological Alterations

In contrast to the relatively simple solution which is described in algorithm 6 the identification of topological alterations of road features is more complex in the OpenDRIVE standard [15]. While point features have an easy to compare topology of one parent element, road features have a predecessor and successor. However, this element does not have to be a single road feature but is in many cases in fact a junction. This junction contains numerous connecting road features which are the indirectly preceding and succeeding road features of the incoming or outgoing road features.

Therefore, it does not reach out to compare the link attributes of the road features. It is also necessary to compare the connections. In this master thesis these junction connections are supposed to be correct in the new road network. This means that they can be compared and updated directly. Generating connections based on road features' geometrical and topological attributes is not part of this implementation as it would require much more work on the basis of the DLR's OpenDRIVE modules. Road feature lanes, for example, would have to be stored with a geometry (in a projected spatial reference system) to generate valid connections. However, currently only the reference line of the road feature is stored as its only geometry in a projected spatial reference system.

This requirement does not restrict this master thesis a lot as new road networks should always be valid. Otherwise an update with them may not lead to a proper result, if there are errors in the new road network already. So as the junction connections can be assumed as correct in the new road network, topological alterations can be identified via comparisons of the connections, too.

What makes the identification of topological alterations more complex than the other types of alterations is the fact that a feature's topology does not consist solely of the feature itself. The features it connects to are also relevant. And here it can happen that the unique ID of another feature might be altered. The topology may be unchanged but the IDs have to be updated for the link attributes and junction connections.

3.4.3 Semantic Alterations

The identification of semantic alterations can be implemented exactly as it is described in section 2.2.3 in pseudo code. All features of a road network are stored in the new developed classes (see section 3.2.1) so that their semantic attributes are stored in a list. Consequently the implemented algorithm only has to iterate through this list to compare the attribute's value with the same key to find any semantic alteration.

3.5 Merging Road Networks

Most of the methods described in section 2.3 can be implemented without requiring any further work. Therefore, only those parts are described here where some more work is required or useful functions shall be emphasized. Methods which are not described here are implemented precisely as described in the methodology section.

3.5.1 Merging Road Features

While the implementation of merging semantic alterations and removals of road features works precisely like described in section 2.3.5, topological alterations, geometrical alterations, and additions are described here a bit further as there are some specific features for them.

Topological Alterations

As explained before in section 3.4.2 there is a small difference between the implementation and the methodology regarding topological alterations. As the connections are assumed to be valid in the new road network, either they or the link attributes of the road features are merged by applying the new values to the ones of the original road network.

In any case, if a road feature's topology is merged, its child elements' and predecessor's alterations will be merged directly afterwards before the application continues with the next step.

Geometrical Alterations

The geometries of road features in the OpenDRIVE standard [15] are described via the so called planview geometries and the road feature lanes. Merging geometrical alterations of road features can therefore be achieved by applying these attributes from the new road network's road feature to the original road network's road feature.

As the object features and signal features of the road feature define their geometry via the road feature's reference line, their geometries has to be updated, too.

Additions

When merging new road features into the original road network it can happen that due to minor differences in the geometries these have to be adjusted. However, it is not always desired to alter the geometry more extensive by adjusting a road feature's geometry to connect with another one. Therefore, an indicator is given to the application whether geometrical adjustments shall be performed additionally to merging the identified geometrical alterations.

Such geometrical adjustments are implemented for the geometries in the projected spatial reference system. To transform the geometries back into the OpenDRIVE standard [15] the geometries generation module of the DLR (see section 3.1 before) is used.

The rest of the merging process of new road features follows in general the descriptions of section 2.3.5.

3.5.2 Merging Signal Features

During my internship at the DLR I already developed some methods for the enhancement of road networks for driving simulations using the OpenDRIVE standard [15]. This includes the enhancement of such road networks with new signal features.

The methods described in section 2.3.6 are therefore to a large amount already implemented in a software module, which is mentioned in section 3.1, too.

However, this work uses this module more selectively. In the previous work [40] a road network with nearly no signal features got a list of signal features which were added to it. It was not the goal to update the topology etc. of the road network in any way but just to enhance it with further data, the signal features.

In this work the module shall not return an enhanced road network but specific information, for example, what kind of signal feature an element is and which effect it can have on the road network's topology or in another scenario to which road feature and even more specifically to which road feature lane a new signal feature belongs to. To achieve these results, new methods have to be implemented and included into this work's application.

3.5.3 Merging Object Features

As already mentioned in section 2.3.7, the process of merging object features is similar to the one of signal features. The module for enhancing road networks with signal features can also process object features and can therefore be used for the same work that is described in section 3.5.2 before.

3.6 Case Studies

The implemented methods are tested in two case studies. Both studies use (partial or complete) road networks which are based on the real world, to be more precise from the town Brunswick in Germany. The DLR has digitized great parts of the town's inner city ring road through mobile mapping and modeled it with the OpenDRIVE standard [15] to a highly-precise road network [13].

In the next two sections the setup of the case studies is explained while their results will be summarized in chapter 4.

3.6.1 Case Study 1: Updating a Road Network by Altering its Features

In the first case study an update of the K47 junction (Rebenring / Hagenring, see figure 3.4) as the original road network is performed. This junction is used regularly for different kind of surveys by the DLR and is therefore the probably best modeled junction in the existing road network.

The new road network will consist of an exact copy of the K47 junction with some alterations. These alterations are also based on the real world as the town's administration has altered two lanes of the Hagenring. The current state of the junction is visualized on the satellite image in figure 3.5.

Figure 3.6 shows a comparison of the former and current state. In the former state there were two road lanes heading straight towards Brucknerstraße. The right one of these has been altered from a straight-right to a right-only road lane. Furthermore, the remaining straight-only road lane now connects to the right road lane of Brucknerstraße and no longer to the left one.

This update includes geometrical, topological, and semantic alterations of road features as well as signal features and object features.

3.6.2 Case Study 2: Updating a Road Network by Merging It with Another Road Network

In the second case study, the north-east part of the highly-precised modeled inner city ring road of Brunswick is extended by an auto-generated road network. Figure 3.7 visualizes these areas of interest. The auto-generated road network covers the roads to the east and west of the road Hagenring, which enters the previously described K47 junction from the south.

The goal is to extend one road network with another road network. As the two road networks were created in different ways (manually generated and highly-precise vs auto-generated and less precise), there are differences in the data which have to be handled. This case study shall show to what extent it is possible to identify these differences and to merge them into a new network.

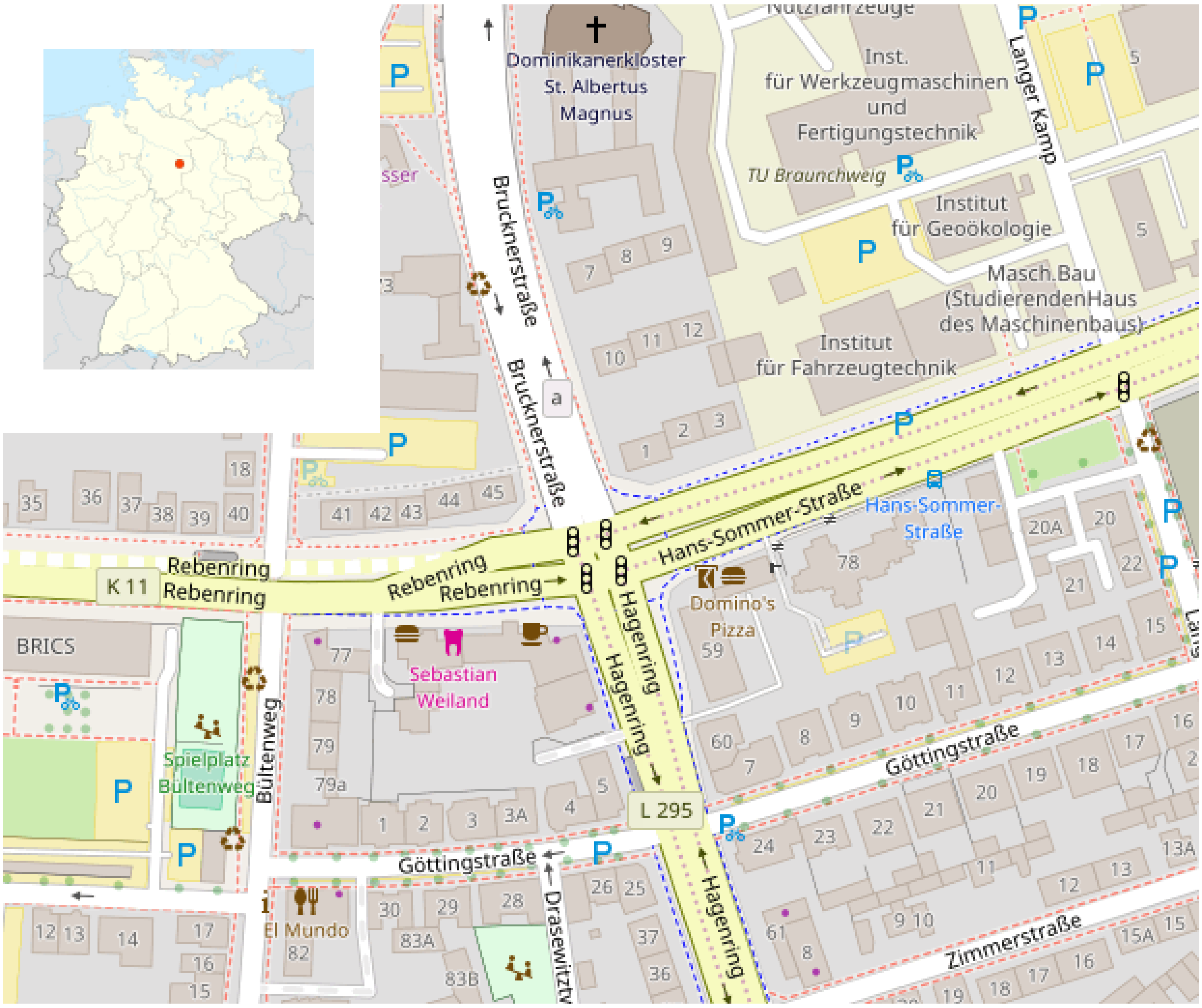


Figure 3.4: Map view of the K47 junction (Rebenring / Hagenring) [42] with an overview of the location of Brunswick in Germany [43]

3.7 Summary

In this chapter it is shown how the implementation of the methods described in chapter 2 is possible with the Java programming language. This is only a proof of concept which is demonstrated in two case studies with different focuses and requirements on the update process. The implemented application uses numerous other software libraries. Most of its code is written based on the methods developed in this master thesis, though.

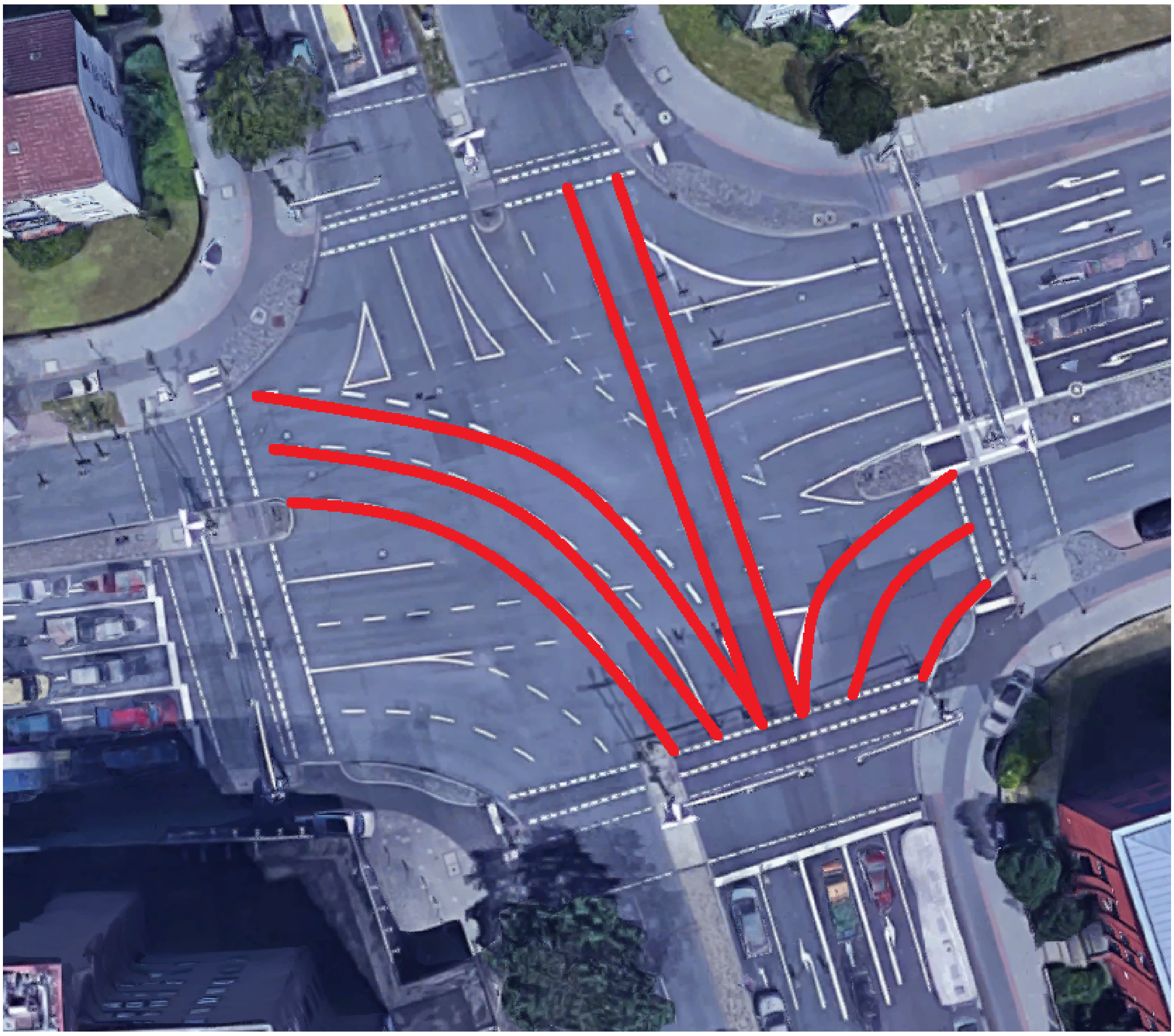


Figure 3.5: Current state of the K47 junction as seen in Google Maps [44] with the connecting roads coming from Hagenring highlighted in red

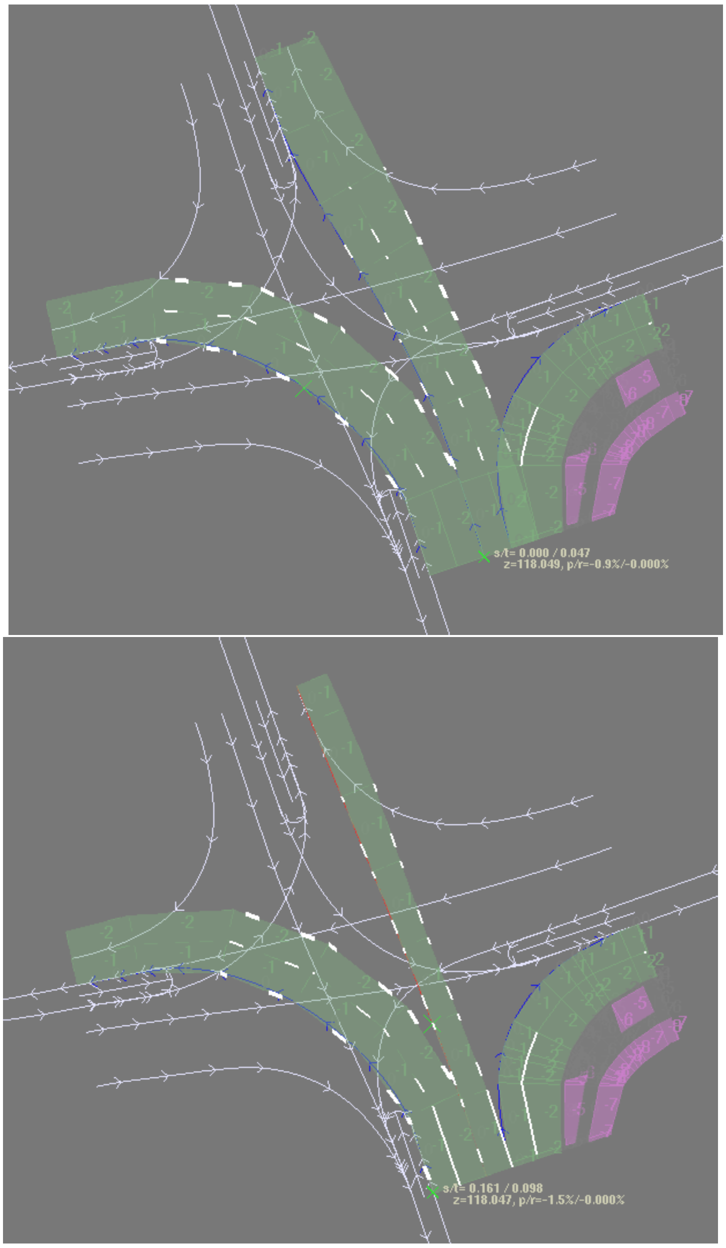


Figure 3.6: K47 junction modeled in the OpenDRIVE standard [15] with the connecting roads coming from Hagenring selected (top: former state, bottom: current state). The white lines represent the reference lines of the road features, the dotted ones road marks, and the green and purple polygons are road feature lanes of type driving and sidewalk.

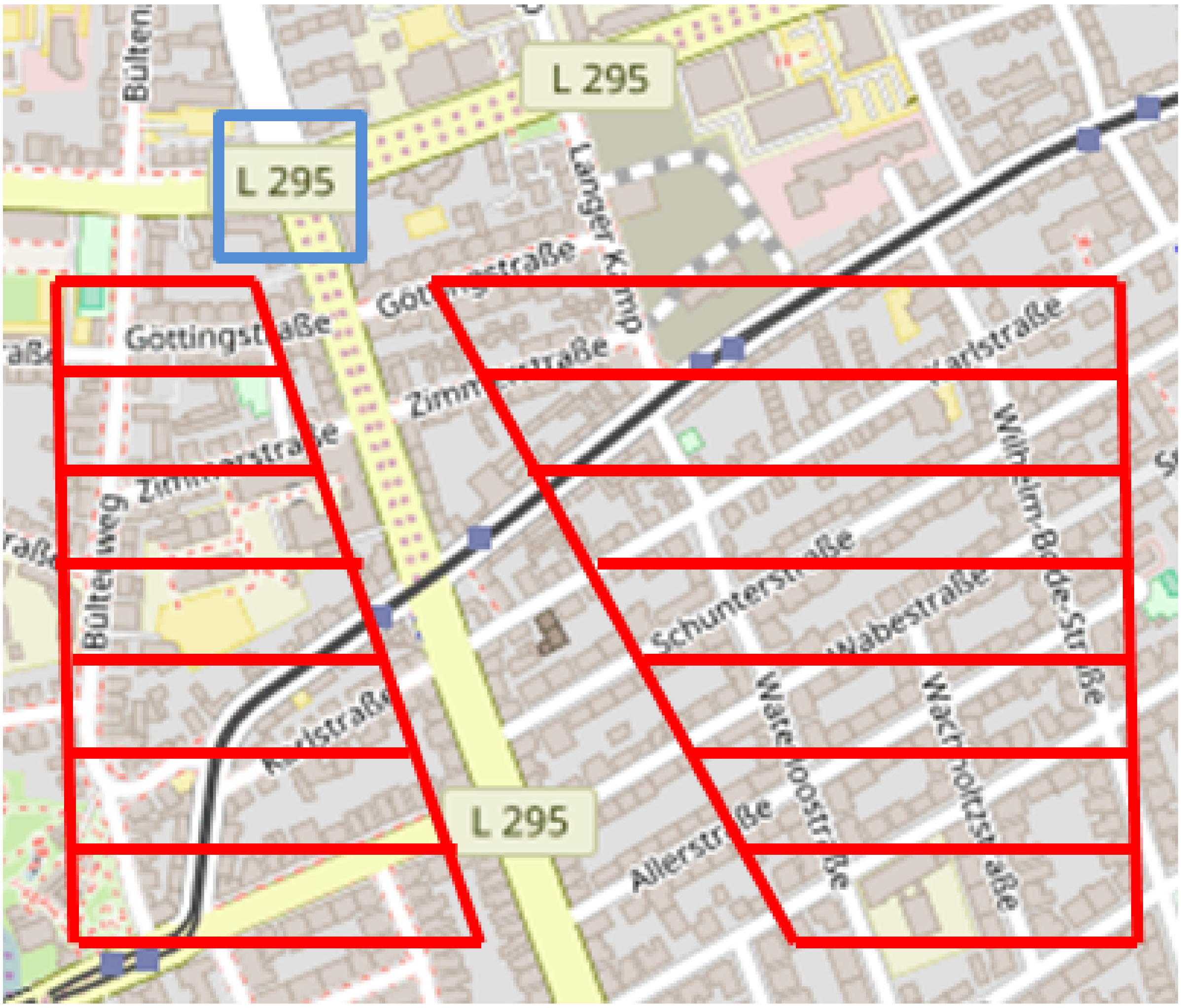


Figure 3.7: Map of the second case study showing the areas of the auto-generated road network (red areas) which are on the eastern and western side of the Hagenring, entering the K47 junction (blue rectangle) from the south, which is highly-precised modeled

4 Results

This chapter summarizes the results of the two case studies from the implementation section 3.6 and how they have been validated.

4.1 Case Studies

The following two sections summarize the results of the previously described case studies.

4.1.1 Case Study 1: Updating a Road Network by Altering its Features

The new road network, which shall update the original road network, has been created by manually altering some of its features. Table 4.1 shows an overview of the manually altered features as well as the results of the application: which alterations could be identified and merged successfully.

Altered Features	Manual Update	Identified	Merged
Object Features	2	2	2
Road Features	2	2	2
Signal Features	0	1	1

Table 4.1: Number of alterations from the manual update and the application's results with the actually identified and merged alterations of different feature types in the first case study

In table 4.2 a more detailed overview of the identified alterations is shown.

The resulting road network equals the new road network from figure 3.6. This is caused by the fact that the update also contained the unchanged features of the original road network. However, even by only having the altered features in the new road network, the result will be the same as the implemented application ignores unchanged features any way.

The greatest update is the altered road feature which crosses the K47 junction straight from south to north or from Hagenring to Brucknerstraße (see figure 3.4). Even though the original and new road features' geometries differ a lot, the developed application

Feature Types	Alteration Types		
	Geometrical	Topological	Semantic
Object Features	0	0	2
Road Features	1	1	1
Signal Features	1	0	0

Table 4.2: Overview of the identified alterations sorted by feature type and alteration type

was able to identify these two road features as corresponding road features and their alterations.

The other altered road feature is the road Hagenring. In the new road network it was not altered as such but its connection in the K47 junction was altered so that it is now succeeded by the new road feature. The result of the update was that this topological alteration was identified and merged successfully.

The altered object features are two pictograms on the Hagenring in front of the K47 junction which describe the road feature's driving directions in the K47 junction. As object features do not have a *type* and a *subtype* attribute like signal features which are defined by official guidelines (e.g. the German StVO [33]), they are defined by their *name* attribute. In this case study the name was changed from straight-right to straight for both object features. These alterations were successfully identified by the application.

To make the identification more difficult, one of the two altered object features also got another unique ID. Nevertheless, the alterations were identified successfully though.

The identified alteration of a signal feature is a result of the altered geometry of its parent road feature. As signal features define their geometry via the reference line of the corresponding road feature, the geometrical attributes of the signal feature had to be updated too, even though its geometry in the projected spatial reference system was kept unchanged in the manual update.

4.1.2 Case Study 2: Updating a Road Network by Merging It with Another Road Network

The results of the second case study show the limitations of updating a road network for driving simulations. In contrast to the first case study this one is rather a merging of two road networks of different areas instead of a typical update. However, to merge these two road networks they have to match at the points where they should connect. Figure 4.1 shows with an example of two corresponding road features from the two road networks that the geometrical difference between the road networks is too large. Scenarios as in this example can be found at nearly all connections between the two road networks.

The road features' reference lines have different angles which already make a matching difficult but the largest difference is the shift of the start and end points of the two road

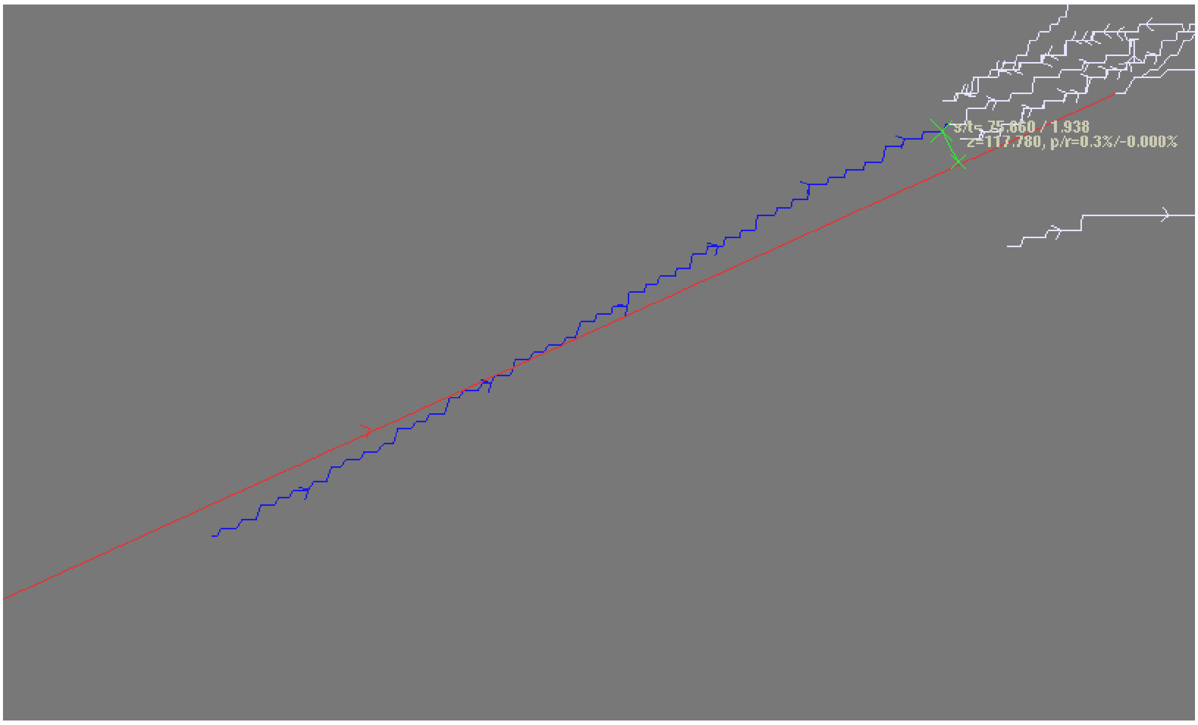


Figure 4.1: This figure shows the large geometrical differences between the two road networks. The blue line represents the reference line of a road feature from the original road network while the red one is its counterpart from the new, auto-generated road network (the steps in the blue reference line are only a visual error of the software, the reference line is actually a straight line). The first difference is that the angle of the two road features already differs noticeably (the green line showing the space between the two road features should not be visible). In addition the start and the end points of the two road features differ by several meters, which makes a matching impossible.

features. These points are not only so far away from each other but also from the start or end point of their predecessor or successor in the other road network, that it is impossible to merge these features.

Nevertheless, this is a result which helps to define the constraints of the developed algorithms. To merge two road networks that cover different areas it is necessary that the generation of their geometries follows the same rules. For example, in the original road network the road features often end at the holding lines at junctions. If such a rule is used for both road networks, the geometrical differences should be reduced to a minimum where a merging is possible.

Therefore, it is a good practice to use general rules for the generation of road networks for driving simulations to enable updates on them even at greater extent.

4.2 Validation

The developed application returns beside the resulting road network also a file with the update logs. All successful and failed merging steps are listed there as well as all identified alterations. This allows a faster validation of the result than to compare the road networks in a graphical animation.

As the extents of the road networks from the first case study are very limited, the resulting log file can be manually evaluated. Before the application ran through the road networks, a list of required alterations and consequent updates was developed by the author. This list works as a comparison to the resulting logs and ensures that no difference between the expected and actual results gets lost.

The second case study contains much more data and its results are therefore not as quickly evaluated as in the first case study. However, only the connections between the two road networks are important here. This still enables a manual validation of the results by comparing the results at each connection with the defined rules from chapter 2 and their expected outcome.

5 Discussion

The two case studies are only an exemplary proof of concept and do not cover all possible scenarios. Nevertheless, their results are a good basis for a discussion of the efficiency of the application that was developed in this master thesis. Its benefits and restrictions are discussed in the following two sections.

5.1 Benefits

The results from chapter 4 showed that the strategies of this master thesis can work properly to update a road network for driving simulations. The two major benefits are the reduced costs and time that is required for such updates.

With the implemented application it is possible to update most of the alterations automatically. And if an alteration cannot be merged, for example, because the difference between the road networks is too large or there is some data missing, a manual update of these few alterations is still a lot quicker than a manual update of the whole road network.

And the automated identification of geometrical, topological, and semantic alterations of different feature types is a very strong benefit of this master thesis. All alterations could be identified in the two case studies even if, for example, the unique ID, geometry, and semantic attributes of a feature were altered. This shows that a successful finding of corresponding and identification of their alterations is possible with the here described algorithms.

As the first case study used a new road network with manual updates, it is possible to make a rough comparison about the time reduction achieved by this master thesis. The manual update required several hours of work (nearly up to one work day) though this was "only" an update of a single road feature in a junction plus the corresponding object features. The implemented application needed in average less than three minutes for the update from which it was occupied for a good part with the loading of the road networks from the database.

This kind of small update for a road network is actually a very typical use case in the normal work routine when working with road networks for driving simulations. Unfortunately such updates are not delivered by the town or state departments and have to be first created by the developers themselves and then manually merged into the existing road network. As it costs several work days to perform these updates, they are often simply not done at all. Therefore, this master thesis may help to improve the currentness of road networks for driving simulations by reducing the required costs and time significantly.

5.2 Restrictions

Besides the great benefits, this work has also some restrictions. These restrictions are listed in the following sections.

5.2.1 Generating New Road Features Based on Alterations of Object Features and Signal Features

The major restriction deals with the generation of new road features. As already explained in section 2.3 it can happen that object features or signal features describe a topology which is altered from the original road network's topology. If an altered road feature causes such change, it will update the relevant object features and signal features accordingly. These features follow specific guidelines, for example, the German StVO [33], which makes it possible to update them even if some information is missing. An example for such an update can be that a right-turn sign is changed into a straight-only sign. However, it is not possible to perform such an update the other way round, if there is no further information given about the affected road feature. This means that when a object feature or signal feature indicates an altered topology of the road network but no such alteration exists for any road feature, then it is impossible to generate the road feature based on the information of the object feature or signal feature only.

The problem is that road features (in general) must not intersect each other and have to fit into the environment's geometry. The road feature's shape and geometry, which shall be generated, may be extracted from additional data sources, for example, satellite imagery, but the goal of this master thesis is to update an existing road network for driving simulations with the information of another road network for driving simulations. Therefore, no further data sources are used and the generation of new road features based on alterations of object features and signal features is not performed here.

Nevertheless, this is an interesting topic for future work and is further discussed in section 6.1.

5.2.2 Calculation of the Elevation Profile for New Road Features

When a new road feature has to be created as an update of the original road network, it needs an elevation profile for the driving simulation. Otherwise there would be a vertical gap between the roads in the driving simulation. If the new road feature is part of the new road network, it may have an elevation profile already. In case the new road feature does not have an elevation profile already or the road feature is created as a consequence of a topological alteration triggered by a road mark, the elevation profile has to be calculated.

To calculate the elevation profile, further data is required, for example, a digital elevation model of the environment. However, the calculation of this elevation profile is not part of this work. It would be necessary to compare the elevation models of the original and the new road network. Then the calculated elevation profile would have to match the start and end points' elevation of the road features it is connected to. This is beyond the scope of this master thesis.

5.2.3 Ontology for Road Networks for Driving Simulations

Sometimes the road networks differ too much from each other that the methods of this work alone do not reach out to merge them. In such cases an ontology for road networks for driving simulations could be helpful. With an ontology the comparison of features in a road network can be performed easier. Though there are no ontologies for road networks for driving simulations yet, such results can be expected from other fields of research where ontologies improved the comparison, for example, in the retrieval of images by using ontologies with defined attributes for its elements [45].

This leads to the restriction of this work that road networks which differ too much from each other cannot be merged, as the second case study of this master thesis also showed. The application would break with an error message informing the user that an update of the original road network is not possible with the given dataset. It is therefore necessary to update an existing road network with data that is comparable, for example, by using the same standard like the OpenDRIVE standard [15]. Road networks which are based on different standards may not be comparable as they use different kind of features, hierarchies, or reference systems. Furthermore, the road networks have to be modeled in the same spatial reference system. Otherwise a transformation is required beforehand as the geometries could not be compared then.

This restriction is also mentioned by Mustière et al. [29] who also excluded an ontological approach from their work for the same reason.

To enable an update with at least the road features that do not differ too much, the application can also be started with an "update-if-possible" attribute. Then road features (and their corresponding object features and signal features) with a too great difference will be skipped and their unique ID will be added to a list of update errors which is returned with the resulting road network in a special file.

6 Conclusion

This master thesis showed that a partially automated update of road networks for driving simulations is possible. The constraints of such an update are that the road networks' features have to follow common rules when being created. Otherwise the differences may be too large which then requires a manual intervention.

However, for road networks with common rules even a fully automated update is possible. This is a great improvement towards the so far used techniques which always required a manual interaction with the data as no other updating mechanism of road networks for driving simulations exists yet (see section 1.3). Manual updates require a lot of time and money which can now maybe be saved by applying the here developed algorithms to the updating process.

6.1 Future Work

For some future work one could think about extending the algorithms to reduce the amount of manual work that is still required. The developed application in this master thesis is already automated in so far that it can update a road network for driving simulation via another road network. However, this road network must be complete on its update. Missing road features, for example, cannot be created, as explained in section 5.2.1 and other parts of this master thesis.

This could be possible by using satellite imagery to identify the shapes of road features and calculate their geometries and topologies based on that. With this additional data road features would not have to be modeled in the new road network to update the existing one but could be created based on the information given by the satellite imagery, the signal features, and the object features.

The current restrictions from section 5.2 can also be used as a base for future work in this field of research. With an appropriate elevation model of the environment, alterations of road features do not lead to a removal of their elevation profile. This could be recalculated and applied to the altered road feature. Such information is very important for a good driving simulation so that the vehicle is not driving through a flat but uneven environment, which is closer to reality.

There exists already an implemented application for this task at the DLR. However, merging the elevation profiles of two digital terrain models (DGM) requires new strategies. It is possible, for example, to simply recalculate all elevation profiles of the original road network. However, then it can happen that an elevation profile of higher accuracy gets lost due to applying a new DGM on the road network.

References

[1] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three Decades of Driver Assistance Systems: Review and Future Perspectives", *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[2] SAE, *Automated Driving*, 2014. [Online]. Available: `http://www.sae.org/misc/pdfs/automated_driving.pdf` (visited on 03/09/2017).

[3] State of California - Department of Motor Vehicles, "Autonomous Vehicle Disengagement Reports 2016", 2016. [Online]. Available: `https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/disengagement_report_2016` (visited on 02/08/2017).

[4] National Highway Traffic Safety Administration (NHTSA), "Automatic Vehicle Control Systems", 2017. [Online]. Available: `https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF` (visited on 03/28/2017).

[5] Google, "Google Self-Driving Car Project Monthly Report March 2016", 2016. [Online]. Available: `https://static.googleusercontent.com/media/www.g.u.00rz.com/en//selfdrivingcar/files/reports/report-0316.pdf` (visited on 03/28/2017).

[6] Mitsubishi Electric, "Mitsubishi Electric Develops Technologies for Automated Mapping and Extraction of Transitions in Mapping Landscape for High-precision 3D Maps Essential in Autonomous Driving", 2017. [Online]. Available: `http://www.mitsubishielectric.com/news/2017/pdf/0316.pdf` (visited on 03/28/2017).

[7] Honda, "Honda Targeting Introduction of Level 4 Automated Driving Capability by 2025", 2017. [Online]. Available: `http://hondanews.com/releases/honda-targeting-introduction-of-level-4-automated-driving-capability-by-2025` (visited on 06/09/2017).

[8] Federal Ministry for Economic Affairs and Energy (BMWi), "Pegasus Research Project", 2017. [Online]. Available: `http://www.pegasusprojekt.de/en/home` (visited on 02/09/2017).

[9] Tesla, *Model S Owner's Manual*, 2016. [Online]. Available: `https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf` (visited on 03/29/2017).

[10] H. Friedl and A. Richter, "Fusion heterogener Geodaten zur Erstellung realer 3D-Welten am Beispiel einer Fahrsimulation", *GEOINFORMATIK 2012*, 2012.

[11] A. Richter and M. Scholz, “Systematische und effiziente Erhebung hochgenauer Straßengeodaten für Geoinformationssysteme und Fahrsimulationen”, *INFORMATIK 2016*, pp. 1579–1589, 2016.

[12] German Aerospace Center (DLR), “DLR Braunschweig - Dynamic Driving Simulator”, 2015. [Online]. Available: `http://www.project-syntropy.de/en/portfolio-item/dlr-braunschweig-dynamic-driving-simulator/` (visited on 03/14/2017).

[13] A. Richter, M. Scholz, H. Friedl, T. Ruppert, and F. Köster, “Challenges and experiences in using heterogeneous, geo-referenced data for automatic creation of driving simulator environments”, *Simulation*, vol. 92, no. 5, pp. 437–446, 2016.

[14] T-online.de and dpa, “Stuttgart warnt vor Navi-Nutzung auf der A8”, 2016. [Online]. Available: `http://www.t-online.de/auto/news/id_78515468/stuttgart-warnt-autofahrer-sollen-auf-a8-nach-sueden-navi-ignorieren.html` (visited on 02/09/2017).

[15] VIRES, *OpenDRIVE Specification*, 2016. [Online]. Available: `http://opendrive.org/` (visited on 02/10/2017).

[16] National Highway Traffic Safety Administration (NHTSA), *Preliminary Statement of Policy Concerning Automated Vehicles*, 2013. [Online]. Available: `www.nhtsa.gov/staticfiles/rulemaking/pdf/Automated_Vehicles_Policy.pdf` (visited on 03/09/2017).

[17] T. M. Gasser and D. Westhoff, “BASt-study: Definitions of Automation and Legal Issues in Germany”, in *Proceedings of the 2012 Road Vehicle Automation Workshop*, 2012.

[18] B. W. Smith, “SAE Levels of Driving Automation”, 2013. [Online]. Available: `http://cyberlaw.stanford.edu/blog/2013/12/sae-levels-driving-automation` (visited on 03/09/2017).

[19] ALK Technologies, “CoPilot GPS - Navigation”, 2017. [Online]. Available: `https://play.google.com/store/apps/details?id=com.alk.copilot.mapviewer&hl=en` (visited on 06/12/2017).

[20] NDS Association, “Navigaiton Data Standard”, 2017. [Online]. Available: `http://www.nds-association.org/` (visited on 07/26/2017).

[21] 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.

[22] A. Nåbo, C. J. Andhill, B. Blissing, M. Hjort, and L. Källgren, “Known roads : Real roads in simulated environments for the virtual testing of new vehicle systems”, Körsimulering och visualisering, SIM, Tech. Rep. 2015-2, 2016, p. 40.

[23] D. Kurteanu and E. Kurteanu, “Open-source road generation and editing software”, MSc Thesis, Chalmers University of Technology, University of Gothenburg, 2010.

[24] H. Shi, "Automatic generation of OpenDrive roads from road measurements", MSc Thesis, Chalmers University of Technology, University of Gothenburg, 2011.

[25] A. Richter, H. Friedl, and M. Scholz, "Beyond OSM – Alternative Data Sources and Approaches Enhancing Generation of Road Networks for Traffic and Driving Simulations", *Proceedings of the SUMO2016*, vol. 30, pp. 23–31, 2016.

[26] T. Wu, L. Xiang, and J. Gong, "Updating road networks by local renewal from GPS trajectories", *ISPRS International Journal of Geo-Information*, vol. 5, no. 9, p. 163, 2016.

[27] R. Toledo-Moreo, D. Bétaille, and F. Peyret, "Lane-Level Integrity Provision for Navigation and Map Matching with GNSS, Dead Reckoning, and Enhanced Maps", *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 100–112, 2010.

[28] 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.

[29] S. Mustière and T. Devogele, "Matching Networks with Different Levels of Detail", *GeoInformatica*, vol. 12, no. 4, pp. 435–453, 2008.

[30] 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.

[31] L. Bernard, "Vorlesung Geoinformatik I Einführung in die Geoinformatik", 2008. [Online]. Available: `https://tu-dresden.de/bu/umwelt/geo/geoinformatik/ressourcen/dateien/studium/opal/material/V-GeoinfoI-WS0809-1-Einfuehrung.pdf` (visited on 03/13/2017).

[32] Wikipedia, "Geoobjekt", 2017. [Online]. Available: `https://de.wikipedia.org/wiki/Geoobjekt` (visited on 03/13/2017).

[33] Deutscher Verkehrssicherheitsrat (DVR), "Verkehrszeichen in der StVO", 2014. [Online]. Available: `http://www.dvr.de/multimedia/downloads/verkehrszeichen.htm` (visited on 03/10/2017).

[34] Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), *Richtlinien für die Anlage von Autobahnen (RAA)*, FGSV Verlag GmbH, 2008. [Online]. Available: `http://www.forschungsinformationssystem.de/servlet/is/275112/`.

[35] Federal Ministry of Transport and Digital Infrastructure (BMVI), *Allgemeine Verwaltungsvorschrift zur Straßenverkehrs-Ordnung (VwV-StVO)*, 2001. [Online]. Available: `http://www.verwaltungsvorschriften-im-internet.de/bsvwvbund_26012001_S3236420014.htm`.

[36] Wikipedia, “Zeichen 297 - Richtungspfeile”, 2006. [Online]. Available: `https://commons.wikimedia.org/wiki/File:Zeichen_297_-_Richtungspfeile,_StVO_1970.svg` (visited on 07/24/2017).

[37] MOTOR-TALK, “GRÜN HEISST STOP! VERWIRRUNG ÜBER POLNISCHE PFEILE”, 2012. [Online]. Available: `https://www.motor-talk.de/news/gruen-heisst-stop-verwirrung-ueber-polnische-pfeile-t3938103.html` (visited on 07/26/2017).

[38] Wikipedia, “Zeichen 209-20”, 2013. [Online]. Available: `https://de.wikipedia.org/wiki/Datei:Zeichen_209-20.svg` (visited on 07/14/2017).

[39] ——, “Zusatzzeichen 1000-21”, 2006. [Online]. Available: `https://de.wikipedia.org/wiki/Datei:Zusatzzeichen_1000-21.svg` (visited on 07/14/2017).

[40] M. Runde, “External Semester - Final Report”, 2017, Ask author for access via marius.runde@uni-muenster.de.

[41] GeoTools, *Geometry Relationships*, 2016. [Online]. Available: `http://docs.geotools.org/stable/userguide/library/jts/relate.html` (visited on 06/12/2017).

[42] OpenStreetMap-Contributors, “Openstreetmap”, 2017. [Online]. Available: `www.openstreetmap.org` (visited on 06/27/2017).

[43] Wikipedia, “Braunschweig”, 2017. [Online]. Available: `https://de.wikipedia.org/wiki/Braunschweig` (visited on 06/27/2017).

[44] Google, “Google Maps”, 2017. [Online]. Available: `https://www.google.de/maps/@52.2755122,10.5355802,89m/data=!3m1!1e3` (visited on 07/13/2017).

[45] V. Mezaris, I. Kompatsiaris, and M. G. Strintzis, “An Ontology Approach to Object-Based Image Retrieval”, in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, IEEE, vol. 2, 2003, pp. II–511.

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit über *„Strategies for Automated Updating of Road Networks for Driving Simulations“* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

Vorname Nachname, Braunschweig, 18th August, 2017

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

Vorname Nachname, Braunschweig, 18th August, 2017