Entwicklung einer Konvertierungsmethode zum Import von OpenDRIVE-Daten in QGIS

Development of a Conversion Method for Importing OpenDRIVE Data into QGIS

Philipp Egert | Theresa Knoblach | Daniela Wenzel

Zusammenfassung
Im Bereich des autonomen Fahrens basieren zahlreiche Funktionen auf HD-Karten. Hierbei handelt es sich um Straßenraumkarten in HD-Qualität, die in standardisierten Formaten wie *OpenDRIVE* bereitgestellt werden. Der Fachbeitrag beschäftigt sich mit der Entwicklung einer Konvertierungsmethode zur Überführung von *OpenDRIVE*-Daten in eine GIS-konforme Datenstruktur und dem Import dieser Daten in *QGIS*. Dadurch eröffnen sich neue Möglichkeiten, HD-Karten durch kombinierte Analysen mit verschiedenen Geodatenquellen abzugleichen, um räumliche Realwelt-Informationen zu vervollständigen und zu erweitern sowie deren Geometrie, Topologie und Semantik anhand von z. B. Geobasisdaten zu validieren.

Schlüsselwörter: OpenDRIVE, Konvertierung, HD-Karte, autonomes Fahren, QGIS

Summary
In the field of autonomous driving, numerous functions are based on HD maps. These are high-definition maps of road spaces, provided in standardized formats such as OpenDRIVE. *This technical paper focuses on the development of a conversion method to transfer* OpenDRIVE *data into a GIS-compliant data structure and import this data into* QGIS. *This opens up new possibilities for matching HD maps through combined analysis with various geodata sources, in order to complete and extend spatial real-world information and validate the geometry, topology and semantics, for example by using geospatial base data.*

Keywords: *OpenDRIVE, conversion, HD map, autonomous driving, QGIS*

1 Projektbeschreibung

Im Forschungsfeld des autonomen Fahrens basieren KI-Training, Simulation, Lokalisierung und Navigation zunehmend auf HD-Karten. Für weitere Entwicklungen werden daher Straßenraumkarten in HD-Qualität und in standardisierten Formaten wie *OpenDRIVE* benötigt. Es gibt momentan noch keine etablierten Methoden, um HD-Karten flächendeckend zu erzeugen und zu validieren. Derzeit werden sie überwiegend aus individuell beauftragten Mobile-Mapping-Messkampagnen und lediglich teilautomatisierter Modellierung gewonnen.

HD-Karten sind räumliche Daten, die Objekte der realen Welt in Klassen beschreiben (z. B. Straße, Fahrbahn, Kreuzung, Verkehrsschild). So liegt es nahe, nationale und internationale Geodateninfrastrukturen (Geodaten, Metadaten und Dienste) sowie die Techniken der Geoinformatik zu nutzen, um die Erstellung und Optimierung von HD-Karten voranzubringen. Die Kombination von HD-Karten mit etablierten Geodaten-Produkten (z. B. ALKIS, 3D-Stadt- und digitale Geländemodelle) ist jedoch nicht direkt möglich, weil sich die räumlich-geometrischen Datenmodelle (Datenstruktur, Koordinatensysteme) von *OpenDRIVE*, einem weit verbreiteten HD-Kartenformat, und standardisierten Geodatenformaten grundlegend unterscheiden. In diesem Beitrag wird ein Datenmodell zur Integration von *OpenDRIVE* in die GIS-Domäne vorgestellt. Es wird gezeigt, wie die durch mathematische Funktionen beschriebenen Straßenverläufe von *OpenDRIVE* über das Parsing ihrer XML-basierten Struktur ausgelesen, angemessen diskretisiert und transformiert werden, um georeferenzierte punkt-, linien- und flächenförmige Objekte zu erzeugen und GIS-konform zu symbolisieren und zu visualisieren. Die vorgestellte Herangehensweise ist in einem eigens entwickelten Plug-in für *QGIS* umgesetzt, das interessierten Nutzern zur freien Verfügung gestellt werden kann.

Die entwickelte Konvertierungsmethode ermöglicht die Integration von *OpenDRIVE*-Daten im GIS-Umfeld, was zahlreiche Vorteile insbesondere hinsichtlich der Nutzung von Erfassungs- und Analysefunktionen mit sich bringt.

2 Einleitung

Für Fortbewegung und Verkehr sind kartographische Darstellungen der Erdoberfläche seit dem Altertum ein elementares Hilfsmittel. Dem aus der Weiterentwicklung der Fortbewegungsmittel erwachsenen Bedürfnis nach höherer Genauigkeit und Detailliertheit von Karten wurde über die Zeit durch konstanten technischen Fortschritt beim Erfassen der benötigten raumbezogenen Daten Rechnung getragen. Heute ist das hochtechnisierte Automobil zum selbstverständlichen Alltagsvehikel geworden, in dem zur Navigation fast ausschließlich digitale Karten verwendet werden, die über den Straßenverlauf hinaus weitere Informationen enthalten können, beispielsweise zu an der Straße angrenzender Infrastruktur. Gleichzeitig wird gegenwärtig zunehmend an Möglichkeiten für die Automatisierung der

  DOI 10.12902/zfv-0505-2025

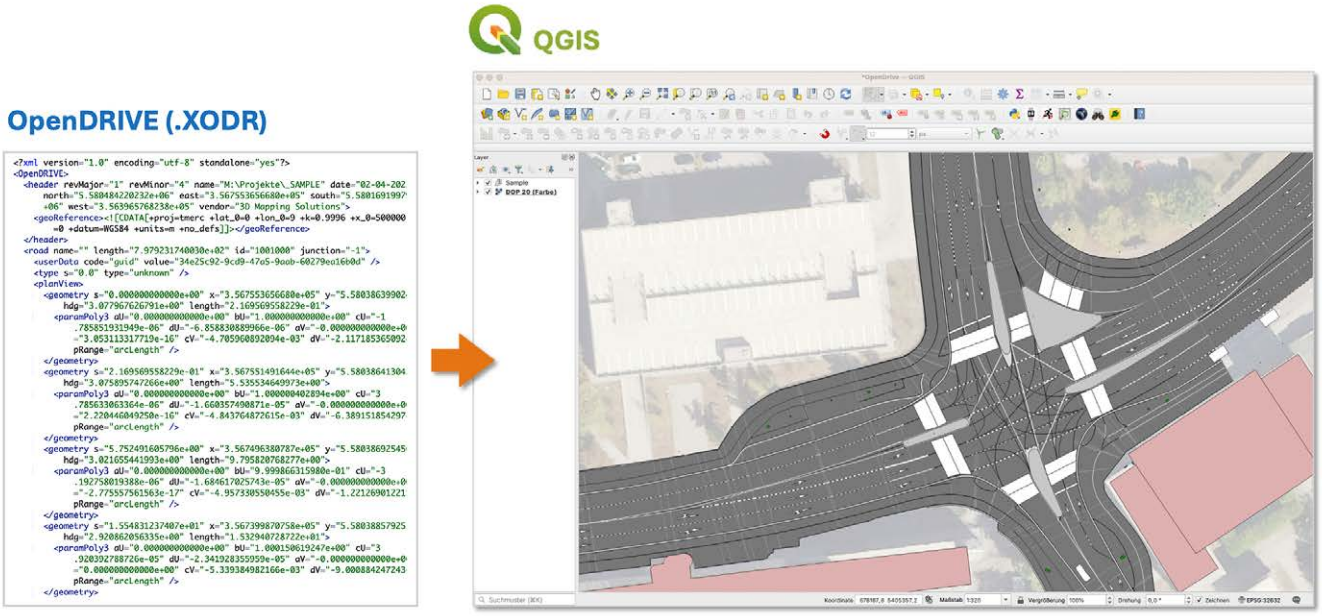

Abb. 1: Überführung einer *OpenDRIVE*-Datei in *QGIS*

Fortbewegung (autonomes Fahren) gearbeitet. Dabei stellt die selbstständige Verortung eines autonomen Fahrzeugs in seiner Umgebung einen elementaren Teilschritt dar, da erst dann kontextsensitive Entscheidungen und eine zielorientierte Navigation möglich sind. Gleichermaßen sind Modelle und Simulationen, beispielsweise von Verkehrsfluss und Fahrverhalten notwendig, um Tests und Kalibrierungen umzusetzen. Sowohl für das autonome Fahren als auch für die Entwicklungsarbeit dieser Technologie braucht es hochdetaillierte Karten, sog. HD-Karten. Ein standardisiertes Format für HD-Karten ist *OpenDRIVE*, das von zahlreichen Unternehmen aus der Automobilbranche unterstützt wird. Wenngleich es aus dem Bereich der Simulation von Straßenverkehr und Fahrdynamik kommt, haben nahezu alle enthaltenen Daten einen Raumbezug, der Lage und Verlauf von Straßen sowie von Objekten, die für den Verkehr relevant sind, in der realen Welt wiedergibt.

Zum Verarbeiten und Visualisieren dieser Informationen ist ein GIS, das der Erfassung, Verwaltung, und Analyse raumbezogener Daten dient, hervorragend geeignet. Zugleich ist es ein leistungsstarkes Werkzeug, um vorhandene *OpenDRIVE*-Datenbestände mit anderen Geodaten zu vergleichen und informativ anzureichern. Ein weit verbreitetes Geoinformationssystem ist *QGIS*. Da *QGIS* Open-Source-basiert ist, ist es prädestiniert für Forschungs- und Entwicklungsarbeiten, jedoch bestehen bislang nur begrenzte Möglichkeiten zur Nutzung von *OpenDRIVE*-Daten in *QGIS*. Details hierzu können dem Abschnitt *Stand der Forschung* entnommen werden. Um diese Lücke zu schließen und die Forschung zum autonomen Fahren voranzubringen, beschäftigt sich das hier vorgestellte Projekt mit der Entwicklung eines benutzerfreundlichen Plug-ins, durch das *OpenDRIVE*-Daten in *QGIS* überführt werden können (Abb. 1). Dabei steht die Diskretisierung, Transformation und Überführung der *OpenDRIVE*-Daten in ein GIS-konformes Datenmodell im Fokus.

Aufgrund der momentanen Ungewissheit über die weitere Entwicklung geeigneter Zielsysteme – im Rahmen von *real-world modelling* und *twin city* wird parallel zu GIS an anderen Plattformen für Datenanalyse und -speicherung geforscht – ist es daher umso wichtiger, Wege für die Nutzung verschiedener Arbeitsumgebungen wie Simulationssoftware, Datenbanken und GIS zu finden. An dieser Stelle trägt das hier vorgestellte Projekt dazu bei, Kompatibilität zwischen diversen Formaten, Datenmodellen und Anwendungsumgebungen zu schaffen.

3 Das *OpenDRIVE*-Format

Für die Beschreibung des Straßennetzes durch *OpenDRIVE*, bestehend aus zahlreichen miteinander verbundenen Straßen mit Fahrspuren und Objekten auf und entlang der Straße, spielen sowohl die Geometrie als auch die topologischen Beziehungen der Elemente und deren Attribute eine zentrale Rolle. Die Struktur von *OpenDRIVE* basiert auf XML und wird auf der Webseite von *ASAM*, eine Standardisierungsorganisation aus der Automobilbranche, in einer Dokumentation für die unterschiedlichen Versionen des Formats dargestellt (ASAM 2023). Die XML-basierte Struktur begründet den hierarchischen Aufbau und die Verwendung unterschiedlicher Tags.

Alles umrahmend in einer *OpenDRIVE*-Datei steht das *OpenDRIVE*-Tag, in dem sämtliche weiteren Elemente definiert sind. Die wichtigsten davon sind *header*-, *road*- und *junction*-Tag (Abb. 2).

header-Tag

Innerhalb des *header*-Tags sind Informationen, die die gesamte Datei betreffen, enthalten, darunter der Dateiname, die *OpenDRIVE*-Version der Datei, teilweise auch

```
<OpenDRIVE>
        <header> </header>
        <road>
                ...
                <planView> </planView>
                <elevationProfile> </elevationProfile>
                <lateralProfile> ... </lateralProfile>
                <lanes>
                        <laneOffset> ... </laneOffset>
                        <laneSection> ... </laneSection>
                </lanes>
                <objects>
                        <object> ... </object>
                <objects>
                ...
        </road>
        <junction> </junction>
        ...
</OpenDRIVE>
```

Abb. 2: Schematischer Aufbau einer *OpenDRIVE*-Datei

landesweit geltende Verkehrsregeln wie Rechtsfahrgebot oder Geschwindigkeitsbegrenzungen und der Ursprung des übergeordneten Koordinatensystems. Zudem werden Parameter dieses Koordinatensystems wie Hauptmeridian, Ellipsoid und Art der Projektion in einem CDATA-Abschnitt definiert. CDATA *(Character Data)* bedeutet, dass dieser Abschnitt vom XML-Parser als Rohtext behandelt und nicht analysiert wird. Innerhalb dieses CDATA-Abschnitts erfolgt dabei die Koordinatensystemdefinition in *Proj4*-Syntax, die durch die *pyproj-Library* festgelegt ist, einer Open-Source-Komponente zur Koordinatentransformation (PROJ contributors 2024).

road-Tag

Das n-fach vorhandene *road*-Tag enthält die für die Eigenschaften und Geometrie der Straße zentralen Informationen in diversen, untergeordneten Elementen (Abb. 3). Die wichtigsten hiervon sind *planView*-Tag, *elevationProfile*-Tag, *lateralProfile*-Tag, *lanes*-Tag und *objects*-Tag.

Im *planView*-Tag wird die zweidimensionale Geometrie der *road reference line* definiert, dem wichtigsten Element des *OpenDRIVE*-Formats, weil davon fast alle anderen Elemente der Straße geometrisch abhängig sind. Die *road reference line* fällt meist mit der Mitte der Straße zusammen und wird je nach geometrischem Verlauf in mehrere Abschnitte unterteilt. Sie kann dabei die folgenden vier bzw. fünf Geometrietypen annehmen:

- Geraden (Straight lines),
- Klothoiden (Spirals or clothoids with a linearly changing curvature),
- Kreisbögen (Arcs with a constant curvature),
- parametrische Funktionen mit Polynomen dritten Grades (Parametric cubic polynomials),
- kubische Polynome (Cubic polynomials).

Da der letztgenannte Geometrietyp veraltet ist und nicht mehr zum Einsatz kommt, findet er in diesem Projekt auch keine weitere Beachtung.

Den vertikalen Verlauf der Straße definieren die Elemente des *elevationProfile*-Tags, die in der XML-Struktur als Nächstes folgen. Sie legen die Höhe der *road reference line* fest, indem innerhalb mehrerer Abschnitte die Höhe aus einer Funktion dritten Grades in Abhängigkeit der Länge der *road reference line* berechnet wird. Höhenänderungen im Querprofil werden hier nicht berücksichtigt. Dafür steht *lateralProfile* als eigenes Tag zur Verfügung, das ein weiteres Element in der XML-Struktur darstellt. Es bietet durch *superelevation*-, *shape*- und *crossSectionSurface*-Tag unterschiedliche Möglichkeiten zur Höhendefinition des Querprofils der Straße.

Das in der XML-Struktur hierauf folgende, sehr umfangreiche *lanes*-Tag dient der Definition von Fahrspuren. Diese sind geometrisch von der Straßenmitte abhängig. Für die Fälle, in denen die Straßenmitte nicht mit der *road reference line* zusammenfällt, müssen die *laneOffset*-Elemente berücksichtigt werden, die eine Verschiebung quer zur Straßenachse angeben.

Die eigentliche Festlegung der Fahrspuren erfolgt in *lane sections*, also Teilabschnitten der Straße. Dabei sollte eine neue *lane section* zumindest immer dann gebildet werden, wenn sich die Anzahl der Fahrstreifen ändert. Innerhalb einer *lane section* werden die Fahrstreifen ausgehend von der Mitte nach rechts und links nummeriert. Die Breite der Fahrstreifen kann entweder durch sogenannte *width*-Elemente oder durch *border*-Elemente definiert werden. Des Weiteren sind in den *lane*-Elementen auch *road markings* vorhanden, die einen Teil der Straßenmarkierung in *OpenDRIVE* beschreiben. Einerseits sind sie geometrisch relevant, andererseits legen sie für die Darstellung der Fahrbahnbegrenzungen beispielsweise Breite und Farbe fest.

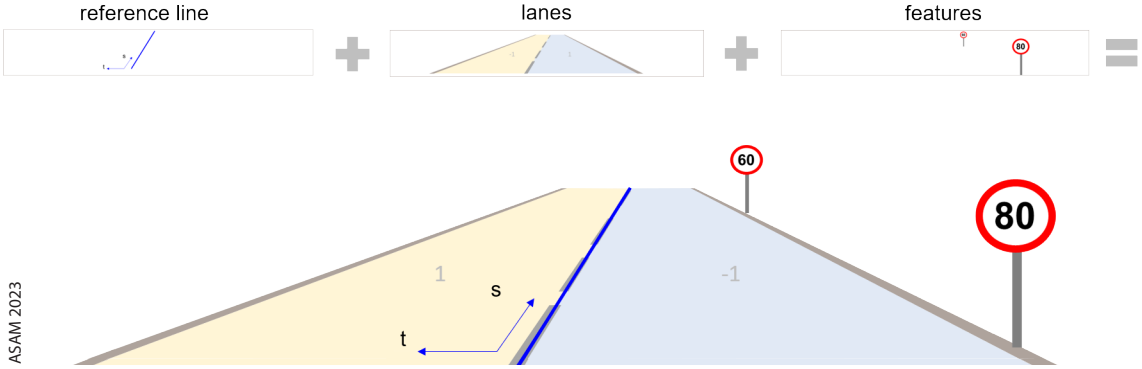

Abb. 3: Schematische Darstellung der Bestandteile einer Straße

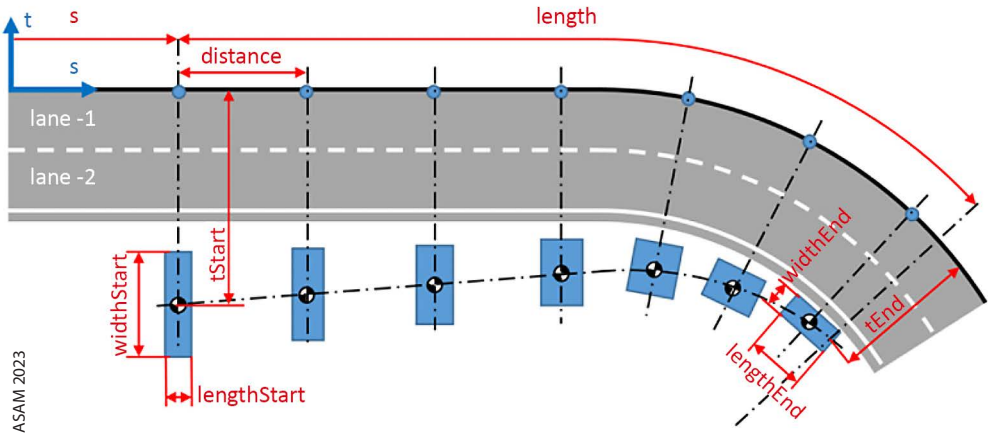

Abb. 4:
Schematische Darstellung von *repeating objects* mit Parameter *distance* > 0

Darüber hinaus ist das *height*-Element bei *lanes* geometrisch wichtig. Hier kann für einen Fahrstreifen definiert werden, dass dieser um einen bestimmten Wert erhöht sein soll, zusätzlich zu den zuvor vorgestellten Möglichkeiten aus dem *lateralProfile*-Tag. Ein Einsatzbereich hierfür sind beispielsweise Gehsteige, die um einige Zentimeter höher sind als die Fahrspuren der Fahrzeuge.

Durch das Attribut *level*, das auf der obersten Ebene des *lane*-Elements steht, können Angaben für das Querprofil durch das zuvor beschriebene *lateralProfile*-Tag unberücksichtigt bleiben, sodass eine *lane* ein horizontales Querprofil aufweist.

Auf das sehr umfangreiche *lanes*-Tag folgt bei *OpenDRIVE* das *objects*-Tag. Dem zugeordnet sind alle straßennahen Objekte, die eine Straße beeinflussen, indem sie ihren Verlauf erweitern, begrenzen oder ergänzen (ASAM 2023), und die somit auch für ein autonomes Fahrzeug relevant sein können, so beispielsweise nahe Mauern, Gebäude oder Fußgängerüberwege. Für deren Beschreibung samt ihren Eigenschaften stehen eine Reihe von Unterarten zur Verfügung, die durch XML-Tags markiert werden: *repeating objects, object outline, object outlines, object skeleton, object material, lane validity for objects, access rules to parking spaces, object marking, object borders, object CRG surface, object reference, tunnels, bridges*. Davon sollen nur *repeating objects*, *object outline* und *object outlines* als die wichtigsten näher betrachtet werden.

repeating objects, die durch das Tag *repeat* markiert werden, definieren entweder mehrere Objekte oder ein größeres zusammenhängendes. Beispiele hierfür sind eine lange Mauer oder Pfosten, die in regelmäßigen Abständen am Straßenrand stehen. Ein *repeating object* wird dabei für eine gewisse Distanz in Abhängigkeit der *road reference line* platziert, zu der ein Offset angegeben ist (Abb. 4).

Mit *object outline* kann die Form eines Objekts durch mehrere Punktkoordinaten vorgegeben werden. Hierbei bestehen zwei Möglichkeiten: entweder *cornerRoad*- oder *cornerLocal*-Elemente. Bei *cornerRoad*-Elementen wird für die Festlegung eines Punktes zu einer Distanz entlang der *road reference line* ein Offset spezifiziert (Abb. 5). Bei *cornerLocal*-Elementen hingegen wird ein lokales Koordinatensystem im Mittelpunkt des Objekts definiert, wo die Koordinaten der Eckpunkte des Objekts gelagert sind. Zum Ursprung werden, analog zu einem *cornerRoad*-Element, Distanz und Ablage sowie ein Winkel, der die Orientierung des Koordinatensystems gegenüber der *road reference line* beschreibt, angegeben (Abb. 6).

Object outlines ist hinsichtlich der Definition der Geometrie identisch mit *object outline*. Der Unterschied liegt

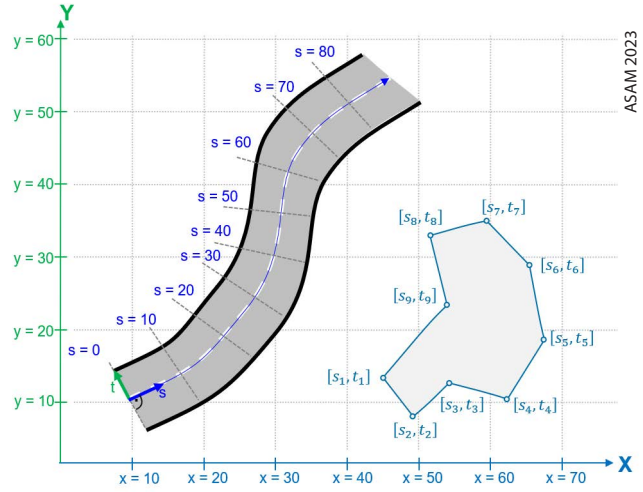

Abb. 5: Schematische Darstellung von einem *outline object* mit *cornerRoad*-Elementen

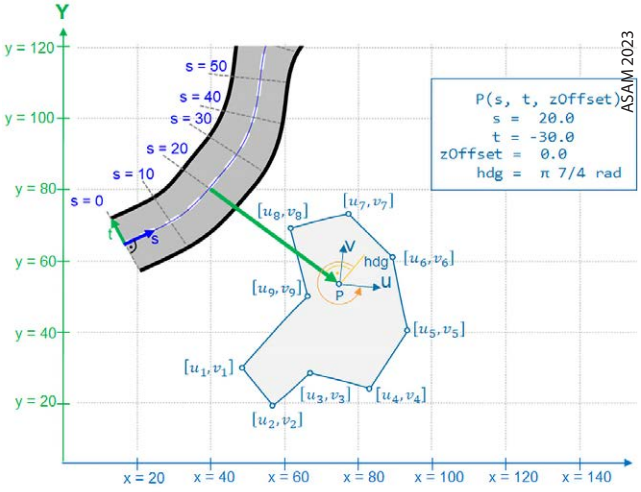

Abb. 6: Schematische Darstellung von einem *outline object* mit *cornerLocal*-Elementen

darin, dass mehrere der *outline objects* in einem *outlines object* vorkommen können.

junction-Tag

Durch das *junction*-Tag werden Straßen beschrieben, die in komplexeren Kreuzungen miteinander verbunden sind. *OpenDRIVE* stellt hier zahlreiche Möglichkeiten bereit, um Spezialfälle umzusetzen. Grundlegend wird jedoch durch das *connection*-Element anhand der IDs der *roads* festgelegt, welche Wege ein Fahrzeug nehmen kann.

Die gesamte *OpenDRIVE*-Struktur betrachtend, liegt der Schwerpunkt für das hier entwickelte Projekt auf der Auseinandersetzung mit dem *header*- und dem *road*-Tag. Vor allem bei letzterem wird die Komplexität des *OpenDRIVE*-Formats deutlich. Dazu tragen auch die verschiedenen Koordinatensysteme bei.

Koordinatensysteme von *OpenDRIVE*

Drei verschiedene Koordinatensysteme sind im *OpenDRIVE*-Format definiert (Abb. 7): das übergeordnete Koordinatensystem *(inertial coordinate system)*, das *road reference line*-Koordinatensystem *(road reference line coordinate system)* und lokale Koordinatensysteme *(local coordinate system)*.

Das übergeordnete Koordinatensystem ist ein dreidimensionales rechtshändiges Referenzsystem, das die Lagerung aller Objekte aus einer *OpenDRIVE*-Datei festlegt. Dabei zeigt die x-Achse nach rechts und die y-Achse nach oben, sodass die y-Achse dem North-Wert und die x-Achse dem East-Wert entspricht.

Das *road reference line*-Koordinatensystem hingegen ist ein lokales Koordinatensystem mit der Achsenbeschriftung *s*, *t* und *h*. Es kann dabei in jedem Punkt der *road reference line* platziert werden, wobei die *s*-Achse immer entlang der Tangente der *road reference line* ausgerichtet ist. Weil es sich hier ebenso um ein rechtshändiges Koordinatensystem handelt, zeigt die *t*-Achse bezogen auf die *road reference line* nach links. Da auch die Winkel, anders als in der Geodäsie üblich, keinen positiven Drehsinn aufweisen, sondern gegen den Uhrzeigersinn gemessen werden, wird dadurch die Berechnung sämtlicher Koordinaten erschwert. Die *h*-Achse steht senkrecht auf der durch die *s*- und *t*-Achse aufgespannten Ebene.

Bei der dritten Art von Koordinatensystem, den lokalen Koordinatensystemen, deren Achsen mit *u*, *v* und *z* bezeichnet werden, liegt der Ursprung abseits der *road reference line*. Auch hier handelt es sich um rechtshändige Koordinatensysteme mit Winkeln mit negativem Drehsinn. Die Orientierung entspricht dabei, wenn keine Rotationswinkel angegeben sind, der Ausrichtung des *road reference line*-Koordinatensystems in dem Punkt der *road reference line*, der als Bezugspunkt für das lokale Koordinatensystem verwendet wird. Ein solches lokales Koordinatensystem kommt teilweise bei Objekten zum Einsatz.

4 Stand der Forschung

Zur Konvertierung von *OpenDRIVE*-Daten in andere Formate ist die Arbeit von Althoff et al. aus dem Jahr 2018 einschlägig. *Lanelet* ist dabei das gewählte Zielformat. Es ist flexibel und leichtgewichtig und definiert Straßen als atomare, miteinander verbundene und befahrbare Segmente. Das Werkzeug wurde in *Python* programmiert und muss die Architektur des *OpenDRIVE*-Formats um die *road reference line* in einzelne Elemente aufgliedern, da *lanelets* Fahrstreifenabschnitte sind, die durch eine linke und eine rechte Polylinie beschrieben werden (Althoff et al. 2018). Dazu werden die *OpenDRIVE*-Daten zuerst geparst und in einzelne Teile zerlegt. Hierbei sowie im Vorgehen zur anschließenden Ermittlung von Punkten der *road reference line* und der einzelnen *lanes* bestehen Ähnlichkeiten zu diesem Projekt. Maierhofer et al. (2021) haben den Beitrag von Althoff et al. (2018) um zusätzliche Kartenexportformate erweitert, die jedoch keinen Bezug zu GIS haben

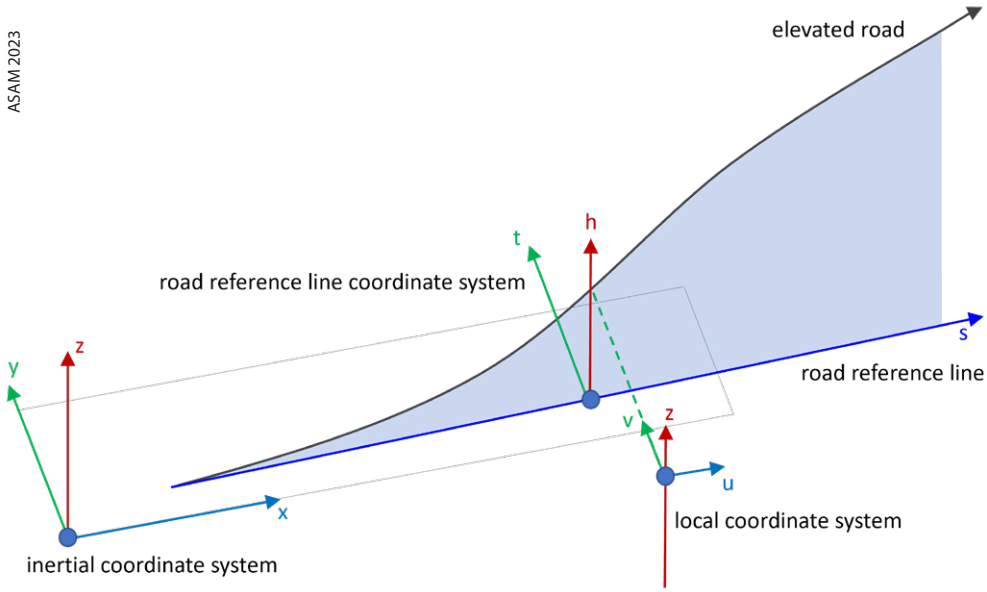

Abb. 7: Koordinatensysteme von *OpenDRIVE* (ASAM 2023)

und daher nicht standardisierten Geodatenformaten entsprechen.

Eine Möglichkeit, um *OpenDRIVE*-Daten in ein GIS-kompatibles Format zu überführen, stellt demgegenüber die *C++ Library ad_map_access* dar. Diese ist ein ausgegliedertes Element des *Carla-Simulators*, mit dem Fahrstrategien in der Simulation trainiert und validiert werden können (Dosovitskiy et al. 2017). Die *C++ Library ad_map_access* enthält Werkzeuge, um HD-Karten im *OpenDRIVE*-Format zu bearbeiten, darunter auch ein Werkzeug, um diese Daten in *QGIS* zu visualisieren (CARLA Simulator Team, o. J.). Allerdings kann diese *Library* nicht in einer *Windows*-Umgebung genutzt werden, da das Programm für *Ubuntu* entwickelt wurde. Zudem scheint die Überführung von Klothoiden in *QGIS* nicht integriert zu sein (Gassmann 2021), was jedoch nicht nachgeprüft werden konnte.

Ein anderes Verfahren zur Konvertierung von *OpenDRIVE*-Daten liefert der Straßenraumtransformator *r:trån* von Schwab et al. (2020), wobei die *OpenDRIVE*-Daten in das *CityGML*-Format überführt werden. Derzeit ist eine getestete Unterstützung für die *OpenDRIVE*-Version 1.4 vorhanden, jedoch können auch andere Versionen experimentell gelesen und verarbeitet werden (r:trån 2024). *CityGML* ist zwar ein standardisiertes Geodatenformat, jedoch ist ein verlustfreier Import in viele GI-Systeme nur mit zusätzlichen Plug-ins oder Spezialsoftware möglich. Im Zuge des Projekts wurde das durch *r:trån* konvertierte *CityGML*-Format in *QGIS* importiert. Jedoch ließ sich dabei keine korrekte Wiedergabe erzielen, da beispielsweise Abschnitte der *road reference line* mit falschen Radien abgebildet wurden oder gänzlich fehlten, außerdem waren die Fahrspuren nicht vorhanden.

Diesbezüglich geeigneter zur Überführung von *OpenDRIVE*-Daten ist die *libOpenDRIVE-Library*, eine Ende 2020 entstandene und frei über *GitHub* verfügbare *C++*-Bibliothek zum Parsen von *OpenDRIVE*-Dateien der Version 1.4 und zum Generieren von 3D-Modellen (Pagel 2020a). Aus dem dort veröffentlichten Code ist ersichtlich, dass zuerst die Daten geparst werden, um anschließend die einzelnen Teile unterschiedlich weiter zu verarbeiten. So werden dann sowohl die Koordinaten der *road reference line* entsprechend dem vorliegenden Geometrietyp als auch die Fahrstreifenbegrenzungen der Straße berechnet. Die *Library* ist dabei ein zuverlässiges Werkzeug, auch wenn manche Features der Daten nicht umgesetzt werden, beispielsweise *road shapes*.

Zweckdienlich für das vorliegende Projekt ist der Webviewer für *OpenDRIVE*-Dateien, der auf der *libOpenDRIVE-Library* basiert (Pagel 2020b). Er verarbeitet fast alle Elemente und Tags des *OpenDRIVE*-Formats und kann daher zur visuellen Kontrolle sowie als Referenz für das Ergebnis dieses Projekts herangezogen werden.

Die beiden Arbeiten von Orozco Idrobo (2015) sowie Scholz und Orozco Idrobo (2017) stellen einen Versuch dar, die *Geospatial Data Abstraction Library (GDAL)* bzw. die *OpenGIS Simple Features Reference Implementation (OGR)* um eine Möglichkeit zu erweitern, *OpenDRIVE*-Daten in *QGIS* zu importieren (Orozco Idrobo 2015). Der aufgezeigte Weg, um Daten aus der XML-Struktur von *OpenDRIVE* zu extrahieren sowie die *road reference line* in x/y-Koordinaten zu überführen, bleibt jedoch auf zwei Geometrietypen – Linien und Kreisbögen – beschränkt, sodass ein lückenhaftes Netz der *road reference lines* entsteht. Außer der Mittellinie werden keine weiteren Elemente, durch die eine Straße zusätzlich beschrieben wird, beispielsweise Fahrspuren, Verkehrszeichen und Objekte am Rand der Straße, behandelt.

Weiterführend ist hier die 2024 für die *Geospatial Data Abstraction Library (GDAL)* bzw. die *OpenGIS Simple Features Reference Implementation (OGR)* entwickelte Erweiterung (Scholz und Bardak 2024a), die die *libOpenDRIVE-Library* verwendet. Sie kann folgende Elemente des *OpenDRIVE*-Formats in *QGIS* bilden: *road reference line*, *road lane*, *road lane border*, *road marks*, *road objects* und *road signals* (Scholz und Bardak 2024b). Dabei handelt es sich um eine konsolengesteuerte Anwendung, die jedoch die *OpenDRIVE*-Version 1.4 nicht vollständig konvertiert.

RoadRunner stellt demgegenüber eine kommerzielle Software dar, mit der 3D-Szenen für die Simulation und das Testen automatisierter Fahrsysteme entworfen werden können (MathWorks, o. J.). Es besteht unter anderem die Möglichkeit, *OpenDRIVE*-Daten bis Version 1.7 zu importieren und zu exportieren (MathWorks, o. J.). Im Projekt sind als Exportformate nur das *dxf*-Format von *AutoCAD* und *GEOJson* relevant. Beide Formate sind zwar *QGIS*-kompatibel, liefern jedoch keine GIS-typische Datenstruktur. Zudem besteht keine Möglichkeit, die Diskretisierung zu beeinflussen.

Zusammenfassend lässt sich konstatieren, dass trotz der bisher vorliegenden Ansätze und Methoden zur Datenkonvertierung von *OpenDRIVE* die Entwicklung eines auch unter *Windows* lauffähigen Programms, das den Import wesentlicher Elemente des *OpenDRIVE*-Formats in der aktuellen Version 1.8 in ein GIS zulässt, noch aussteht.

5 Konvertierung von *OpenDRIVE*

Bei der Konvertierung von *OpenDRIVE*-Daten in ein GIS-konformes Datenmodell werden zuerst die *OpenDRIVE*-Daten aus der XML-Struktur extrahiert. Im nächsten Schritt wird durch Diskretisierung und Transformation die Überführung der Objektgeometrie realisiert, wobei die unterschiedlichen Raumbezüge in einen übergeordneten Bezugsrahmen gebracht werden, der sich in einem Geoinformationssystem darstellen lässt.

Datenstruktur im Zielsystem

Um im GIS die Daten in der Attributtabelle des jeweiligen Layers bereitzustellen, müssen möglichst alle Attribute atomar sein. Dies ist durch Normalisierung erreichbar. Dabei wird eine Menge von Relationen mit minimaler

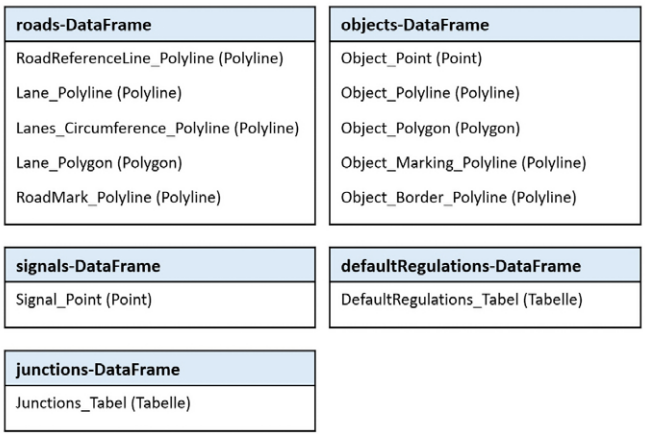

Abb. 8: Layer- und Tabellenstruktur in *QGIS*

Datenredundanz erstellt, um die Konsistenz zu wahren und korrekte Einfügungen, Löschungen und Änderungen zu erleichtern (Bahmani et al. 2008).

Beispielsweise wird der *roads*-DataFrame in mehrere Layer zerlegt, wodurch sich Listen eliminieren lassen. Außer der Normalisierung hat auch die begrenzte Möglichkeit, 3D-Objekte in *QGIS* wiederzugeben, Einfluss auf die Bildung einiger Layer. Damit trotz dieser Einschränkungen keine Informationen verloren gehen, sollen aus den DataFrames Layer und Tabellen entstehen, die in Abb. 8 aufgelistet werden.

Extraktion von Daten aus der XML-Struktur

Für den Vorgang der Datenextraktion wird die *Python Library ElementTree* verwendet. Diese ermöglicht es, durch die Funktionen *find()*, *findall()*, *get()* einzelne Elemente aus der hierarchischen XML-Struktur auszulesen. Der Fokus liegt dabei auf den Inhalten des *header*-, *junction*- und insbesondere des *road*-Elements, die in mehreren *pandas* DataFrames gespeichert werden sollen. Von zentraler Bedeutung sind aus dem *road*-Element die Angaben zur Geometrie der *road reference line*, auf der alle anderen Strukturen der Straße aufbauen.

Überführung der Elemente samt Geometrie

Die Geometrien folgender zentraler Elemente werden in der angegebenen Reihenfolge überführt:

- *road reference line* der Straßen mit vertikaler Geometrie,
- Fahrstreifen mit vertikaler Geometrie,
- Objekte,
- Verkehrszeichen,
- Kreuzungen.

Bei der *road reference line* müssen zunächst die kontinuierlichen Geometrieelemente diskretisiert werden. Deshalb sind außer bei Geraden in regelmäßigen Abständen Koordinaten zu ermitteln, die die *road reference line* repräsentieren. Als Standardabstand wird 0,5 m eingestellt, er kann aber auch durch den Nutzer frei gewählt werden. Die Vorgehensweise ist abhängig von der geometrischen Form des jeweiligen Objekts. So lässt sich die Gerade durch polares Anhängen berechnen, während bei den Kreisbögen erst der Mittelpunkt bestimmt werden muss, um dann von diesem Punkt aus polares Anhängen durchführen zu können. Hierbei wird der Richtungswinkel durch Inkrementierung so lange erhöht, bis der Zentrierwinkel überschritten ist.

Bei Klothoiden ist eine Fallunterscheidung notwendig, je nach Größe der Anfangs- und Endkrümmung sowie deren Vorzeichen. Zuerst werden die Anfangs- und Endpunkte im Zielkoordinatensystem bestimmt, sodass anschließend die lokalen, durch die Näherungsformel ermittelten Koordinaten durch eine 3-Parameter-Transformation in das Zielsystem gebracht werden können.

Bei Parametrischer Funktion mit Polynomen dritten Grades ergeben sich lokale Koordinaten durch Einsetzen der *s*-Koordinaten (Abb. 7) in zwei Polynome dritten Grades. Zur Überführung in das Zielsystem muss im Anschluss ebenfalls eine 3-Parameter-Transformation angewendet werden.

Hinsichtlich der *z*-Koordinaten der *road reference line* sind Angaben zur vertikalen Geometrie von *elevation-Profile*- und *road shape*-Elementen zu berücksichtigen.

Nach der *road reference line* werden die Fahrstreifen der Straße ermittelt. Dies erfolgt unter Beachtung des *lane Offset*, um die Lage der *center lane*, also des mittleren Fahrstreifens, zu bestimmen. Danach lassen sich Punkte für die Fahrstreifen in regelmäßigem Abstand berechnen. Hierbei wurden zwei Varianten (Umsetzungsmöglichkeiten) untersucht:

1. Es werden zuerst Punkte der *center lane* für die gesamte Länge der ersten *lane section*, in diesem Beispiel der ganzen Straße, anschließend die Punkte des linken Fahrstreifens ermittelt, weil dieser eine numerisch größere ID hat. Im Anschluss wird die Berechnung für den rechten Fahrstreifen mit der ID = –1 vorgenommen. Bei einer Straße mit mehreren *lane sections* können analog dazu dieselben Schritte mit der nächsten *lane section* wiederholt werden (Abb. 9).
2. An wichtigen Distanzen der *road* und in regelmäßigen Abständen, z. B. 0,5 m, werden alle Punkte des Querprofils, d. h. die Punkte aller Fahrstreifenbegrenzungen und der Punkt der *center lane*, gleichzeitig berechnet. Dies wird sukzessiv für sämtliche Distanzen vollzogen (Abb. 10).

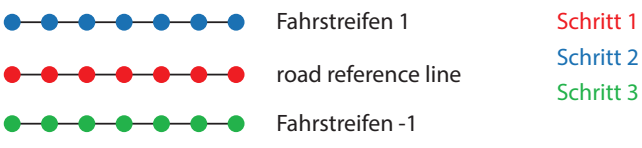

Abb. 9: Schematisches Vorgehen Variante 1

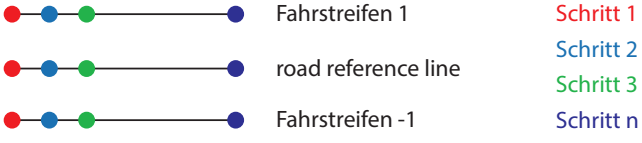

Abb. 10: Schematisches Vorgehen Variante 2

Im Projekt wurde die zweite Variante umgesetzt, da *width/border*-Elemente und *lane sections* mit einer Länge, die kleiner ist als das Intervall, in Variante 1 nicht repräsentiert werden, wenn nicht zufällig eine der Distanzen in diese kurzen *width/border*-Elemente und *lane sections* fällt. Zudem wird bei Variante 1 nur ein festes Schema an Intervallen abgearbeitet, weswegen sie recht starr und unflexibel ist.

Unabhängig von der Variante muss für die Berechnung der Lage der Fahrstreifenbegrenzungen eine Unterscheidung zwischen *border*- oder *width*-Elementen durchgeführt werden. Mit entsprechenden Anpassungen wird hierbei unter anderem eine Funktion eingesetzt, die aus dem Algorithmus zur Diskretisierung abgeleitet wird. Sie muss zu einer gegebenen Distanz eine Koordinate auf der *road reference line* bestimmen und die Tangentenrichtung in diesem Punkt angeben.

Bei der Überführung von Fahrstreifen ist ebenfalls die vertikale Geometrie zu berücksichtigen. Neben Angaben durch *elevationProfile*- und *road shape*-Elemente sind *superelevation*- und *lane height*-Elemente sowie der Wert des Attributs *level* zu beachten. Zusätzlich werden auch die Straßenmarkierungen aus *roadMark*-Elementen in eine GIS-nutzbare Form gebracht.

Bei der Berechnung der Geometrie von Objekten, die unabhängig von den Fahrstreifen in Bezug zur *road reference line* erfolgen kann, wurden die wichtigsten Objektarten – *Objekte ohne separate Geometriedefinition*, *repeating objects*, *outline objects* und *outlines objects* – überführt. Je nach Geometriedefinition muss eine Diskretisierung sowie eine 3-Parameter-Transformation in das übergeordnete Koordinatensystem durchgeführt werden.

Bei Verkehrszeichen, sog. *signal*-Elementen, handelt es sich um Punktobjekte, weshalb nur der Einfügepunkt des Verkehrszeichens bzw. der Ampel zu berechnen ist.

Informationen zu Kreuzungen aus *junction*-Elementen enthalten keinen Raumbezug, da sie rein relationale Daten sind. Daher ist keine Geometrie zu bestimmen.

Für die Überführung aller angeführten Elemente einer *OpenDRIVE*-Datei wurden Funktionen in Python erstellt. Häufig ist auch eine kombinierte Anwendung der Funktionen notwendig, beispielsweise bei der Berechnung der Fahrstreifen und von Punkten auf der *road reference line*.

6 *QGIS*-Plug-in

Als Benutzerschnittstelle wurde ein *QGIS*-Plug-in realisiert, das die oben genannten Funktionen integriert und die Visualisierung umsetzt. Das Plug-in wurde mit Hilfe von *QGIS Plugin Builder* generiert, die grafische Benutzeroberfläche wurde mit dem *Qt Designer* erstellt (Abb. 11). In den Quellcode werden dabei zunächst alle zuvor beschriebenen Teilschritte integriert und aus der *run*-Funktion heraus in der korrekten Reihenfolge aufgerufen.

Die Benutzerinteraktion beschränkt sich auf die Auswahl der *OpenDRIVE*-Datei, die Benennung des Gruppenlayers (für die Layerorganisation in *QGIS*) sowie die Definition des Punktabstandes der Diskretisierung.

Die Layer werden automatisiert erstellt, indem die Geometrie berechnet wird, alle Attribute generiert und mit Attributwerten versehen werden. Da bis zu diesem Schritt nur Punktgeometrien berechnet wurden, werden nun darauf aufbauend Polylinien und Polygone gebildet. Dabei müssen *s*-Koordinaten und IDs beachtet werden, um die Punkte in der korrekten Reihenfolge zu verbinden. Im Anschluss werden die Layer spezifisch, teilweise attributabhängig symbolisiert.

Um eine einheitliche Struktur und Ordnung in der Layerverwaltung zu gewährleisten und die unterschiedlichen Symbole möglichst gut sichtbar zu machen, werden die Layer in einer Gruppe gesammelt und in eine bestimmte Reihenfolge gebracht. Abb. 12 zeigt das Import-Ergebnis eines kurzen Straßenabschnitts in *QGIS* im Vergleich zur Visualisierung mit dem webbasierten *OpenDRIVE* Viewer.

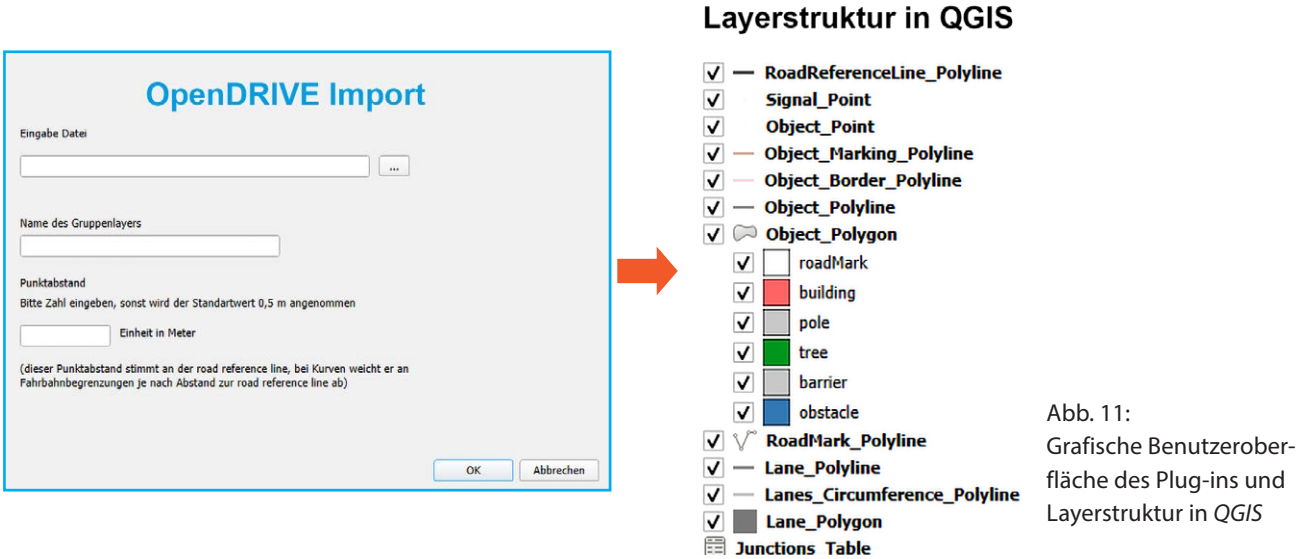

Abb. 11: Grafische Benutzeroberfläche des Plug-ins und Layerstruktur in *QGIS*

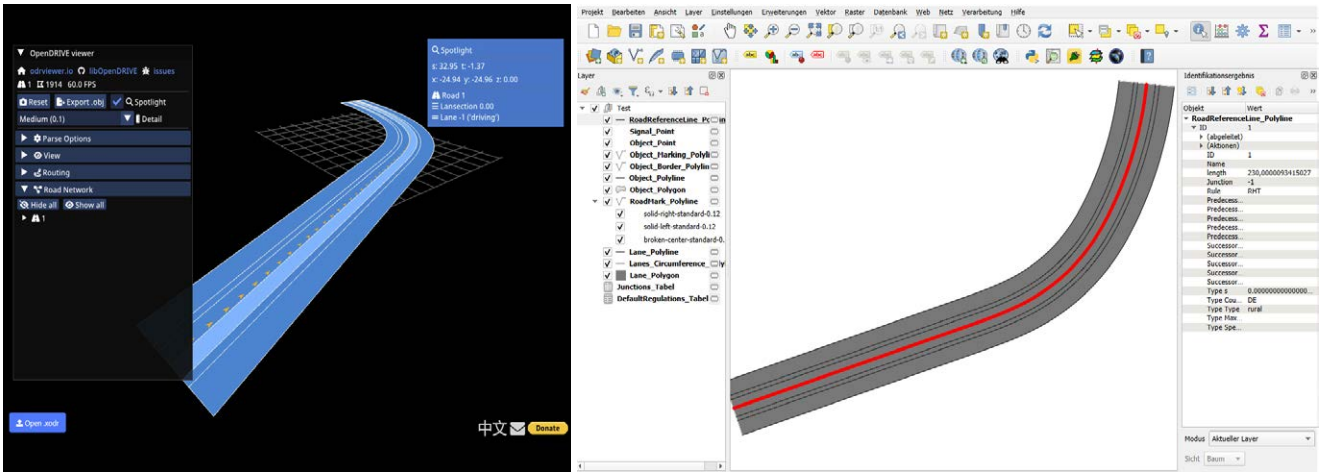

Abb. 12: Zwei Interpretationen einer *OpenDRIVE*-Datei im .xodr-Format: *OpenDRVE* Viewer (links) und *QGIS* mit *OpenDRIVE*-Import-Plugin (rechts)

7 Fazit und Ausblick

Im Projekt wurde eine neue Methode zur Konvertierung von *OpenDRIVE*-Daten in GIS-konforme Datenstrukturen entwickelt. Dabei wurden die durch mathematische Funktionen beschriebenen Straßenverläufe von *OpenDRIVE* diskretisiert und in ein übergeordnetes Koordinatensystem transformiert, um georeferenzierte punkt-, linien- und flächenförmige Objekte zu erzeugen, GIS-konform zu symbolisieren und zu visualisieren. Die Nutzung der Konvertierungsmethode erfolgt über ein benutzerfreundliches *QGIS*-Plug-in.

Fünf zentrale Schwierigkeiten während des Entwicklungsprozesses werden abschließend kurz reflektiert:

1. Sowohl die starke Verschachtelung der Attribute von *OpenDRIVE* und die daraus entstehenden Listen als auch das Fehlen von XML-Elementen und somit von Attributen führten zu sehr aufwändigem Parsing.
2. Vor allem das Koordinatensystem von *OpenDRIVE* bringt es mit sich, dass die kontinuierlichen Funktionen der *road reference line* in einzelne Punkte diskretisiert werden müssen. Besonders bei Klothoiden stellte dies eine Herausforderung dar, da hier Fallunterscheidungen je nach Anfangs- und Endkrümmung zu treffen sind und teilweise keine der beiden Krümmungen den Wert 0 hat.
3. Bedingt durch das an der *road reference line* ausgerichtete Koordinatensystem und die Verschachtelung der Elemente gestaltete sich die Berechnung der Fahrstreifenbegrenzungen ebenfalls als sehr anspruchsvoll. Auch hier stellt die komplexe Struktur das Hauptproblem dar, da in einer Straße eine Vielzahl von *lane sections* mit diversen *lanes* vorhanden sein können, die wiederum aus etlichen *width*- oder *border*-Elementen bestehen.
4. Bei der Entwicklung der Funktionen zur Überführung der verschiedenen Elemente von *OpenDRIVE* war die *ASAM*-Dokumentation (ASAM 2023) sehr hilfreich. Dennoch konnten manche strukturellen Zusammenhänge auf dieser Basis nur teilweise durchdrungen werden, beispielsweise bei Objekten. Die Bereitstellung einer Referenzimplementierung von *ASAM OpenDRIVE* 1.8 könnte an dieser Stelle Abhilfe schaffen, was jedoch bisher noch nicht erfolgt ist.
5. Bei *QGIS* als Zielsystem erweist sich die begrenzte 3D-Fähigkeit als ein Nachteil. Um den Verlust von Höheninformationen bei 3D-Objekten zu vermeiden, war es daher notwendig, zusätzlich zum Polygon-Layer einen 3D-Polylinien-Layer der Umrisse zu erstellen.

Trotz dieser Herausforderungen wurde durch das Projekt ein wichtiger Fortschritt für einen effizienten Austausch zwischen verschiedenen Programmen und Formaten erzielt und so eine Möglichkeit für die Nutzung diverser Arbeitsumgebungen – Simulationssoftware, Datenbanken und GIS als besonders leistungsstarkes Werkzeug für die Entwicklungsarbeit rund um HD-Karten – geschaffen.

Die neue Konvertierungsmethode des *OpenDRIVE*-Formats in die GIS-Welt und das entwickelte Plug-in für *QGIS* eröffnen Möglichkeiten der Erweiterung bzw. Validierung der HD-Kartendaten durch andere vorhandene Datenquellen, wie beispielsweise die flächendeckend verfügbaren Geobasisdaten. Die GIS-konforme Überführung des *OpenDRIVE*-Formats erlaubt zudem den Transfer in eine relationale Datenbank wie *PostgreSQL/PostGIS* mit all ihren Vorteilen.

Darüber hinaus können die nun konvertiert vorliegenden Daten eine wichtige Datengrundlage für Geoinformationssysteme von Kommunen, der öffentlichen Verwaltung, Wirtschaft und Industrie bilden. Gerade im Bereich Verkehr bietet sich eine Vielzahl an Anwendungsmöglichkeiten, angefangen von Analysen zur lokalen Verkehrsbelastung und Lärmschutzanalysen samt deren Visualisierung bis hin zu Verkehrs- und Baustellenplanung.

Perspektivisch ist ein Ausbau des Umfangs der aus *OpenDRIVE* überführten Elemente, wie beispielsweise die Objektarten *skeleton*, *tunnel* und *bridge*, denkbar.

Ebenfalls eine sinnvolle Ergänzung über das *OpenDRIVE*-Format hinaus ist der GIS-Import des eng ver-

wandten *OpenCGR*-Formats, das die Oberfläche von Straßen beschreibt, um auf diese Weise weitere objektspezifische Informationen für die Optimierung des autonomen Fahrens im GIS zur Verfügung zu stellen.

Im Hinblick auf eine Verbesserung der Performance des Plug-ins kann das Vorgehen beim Auslesen und Verarbeiten der Daten modifiziert werden. Dabei wäre ein Ansatzpunkt, nicht alle Straßen, die in einer *OpenDRIVE*-Datei gespeichert werden, in einem DataFrame zwischenzuspeichern, um sie im Anschluss zu verarbeiten und als Layer darzustellen, sondern dies für jede Straße einzeln vorzunehmen, damit sich so die Größe der temporär benötigten Variablen verringert.

Basierend auf den Höheninformationen der Punktdaten und den Fahrstreifenbegrenzungen als Bruchkanten lässt sich ein Digitales Geländemodell (DGM) erstellen. Diese Methodik bietet insbesondere für den Export und die Nutzung der Daten in vollwertigen 3D-Anwendungen außerhalb von *QGIS* entscheidende Vorteile, setzt jedoch voraus, dass die aktuelle GIS-Software Möglichkeiten zu einer vollständigen 3D-Topologie-Analyse und 3D-Datenverarbeitung enthält. Gegebenenfalls sind Erweiterungen oder alternative Softwarelösungen notwendig, um den Anforderungen solch komplexer räumlicher Datensätze gerecht zu werden.

Eine größere zukünftige Aufgabe ist darüber hinaus die Rückkonvertierung von Daten mit Raumbezug, die mit einem GIS verarbeitet werden können, in *OpenDRIVE*. Dabei liegt die Herausforderung darin, mathematische Funktionen zu finden, die die *road reference line* und Fahrstreifenbegrenzungen korrekt wiedergeben. Ein Ansatz hierfür wäre die Nutzung eines sich bewegenden Gleitfensters, in dem dann jeweils eine Funktion gesucht wird, die die Geometrie am besten annähert, wie es bereits in einem Projekt zur Konvertierung von OSM-Daten in *OpenDRIVE* versucht wurde (Oberlerchner 2019). Ein weiteres Problem für die Rückkonvertierung stellen die Ausgangsdaten dar, da hier eine möglichst weit verbreitete und gleichzeitig hochgenaue Datengrundlage gefunden werden müsste, die als Standard festgelegt werden kann.

Literatur

Althoff, M., Urban, S., Koschi, M. (2018): Automatic Conversion of Road Networks from OpenDRIVE to Lanelets. 2018 IEEE International Conference on Service Operations and Logistics and Informatics (SOLI). Singapore, 157–162. https://ieeexplore.ieee.org/document/8476801/, letzter Zugriff 08.11.2024.

ASAM (2023): OpenDRIVE 1.8.0 Specification. https://publications.pages.asam.net/standards/ASAM_OpenDRIVE/ASAM_OpenDRIVE_Specification/latest/specification/index.html, letzter Zugriff 29.05.2024.

Bahmani, A. H., Naghibzadeh, M., Bahmani, B. (2008): Automatic database normalization and primary key generation. 2008 Canadian Conference on Electrical and Computer Engineering. Niagara Falls, 11–16. https://ieeexplore.ieee.org/document/4564486, letzter Zugriff 08.11.2024.

CARLA Simulator Team (o. J.): Overview – ad_map_access. https://admap-access.readthedocs.io/en/latest/ad_map_access/Main/#mapdatamodel, letzter Zugriff 29.05.2024.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V. (2017): CARLA: An Open Urban Driving Simulator. Proceedings of the 1st Annual Conference on Robot Learning, 1–16. https://carla.org//, letzter Zugriff 08.11.2024.

Gassmann, B. (2021): Unable to load xodr files Issue #72. GitHub. https://github.com/carla-simulator/map/issues/72, letzter Zugriff 29.05.2024.

Maierhofer, S., Klischat, M., Althoff, M. (2021): CommonRoad Scenario Designer: An Open-Source Toolbox for Map Conversion and Scenario Creation for Autonomous Vehicles. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). Indianapolis, 3176–3182. https://ieeexplore.ieee.org/document/9564885, letzter Zugriff 08.11.2024.

MathWorks (2024): RoadRunner Documentation. https://de.mathworks.com/help/roadrunner/index.html?s_tid=hc_product_card, letzter Zugriff 09.08.2024.

MathWorks (o. J.): RoadRunner. https://de.mathworks.com/products/roadrunner.html, letzter Zugriff 09.08.2024.

Oberlerchner, D. (2019): Road Data-Modelling for Autonomous Driving. Master's Thesis. Graz. https://repository.tugraz.at/publications/my8f9-gk135, letzter Zugriff 08.11.2024.

Orozco Idrobo, A. M. (2015): Extension of the Geospatial Data Abstraction Library (GDAL/OGR) for OpenDRIVE Support in GIS Applications for Visualisation and Data Accumulation for Driving Simulators. Master's Thesis. München. https://elib.dlr.de/103827/, letzter Zugriff 08.11.2024.

Pagel, S. (2020a): libOpenDRIVE. https://github.com/pageldev/libOpenDRIVE, letzter Zugriff 05.06.2024.

Pagel, S. (2020b): OpenDRIVE viewer. https://odrviewer.io/, letzter Zugriff 25.10.2024.

PROJ contributors (2024): PROJ coordinate transformation software library. https://proj.org/, letzter Zugriff 30.06.2024.

r:trån (2024): Processing Library for Road Space Models. https://rtron.io/, letzter Zugriff 08.11.2024.

Scholz, M., Bardak, G. (2024a): OpenDRIVE-HD-Karten mittels GDAL ins GIS bringen. FOSSGIS 2024 Tagungsband. Hamburg, 114–116. https://fossgis-konferenz.de/2024/programm/, letzter Zugriff 08.11.2024.

Scholz, M., Bardak, G. (2024b): libopendrive GitHub Repository. https://github.com/DLR-TS/gdal/tree/libopendrive, Zugriff am 5.6.2024, inzwischen nicht mehr verfügbar 22.08.2024.

Scholz, M., Orozco Idrobo, A. M. (2017): Supporting the Implementation of Driving Simulator Environments Through Established GIS Approaches by Extending the Geospatial Data Abstraction Library (GDAL) with OpenDRIVE. In: Kemeny, A., Colombet, F., Merienne, F., Espié, S. (Eds.): Proceedings of the Driving Simulator Conference 2017 Europe VR. Stuttgart, 51–54. https://elib.dlr.de/110123/, letzter Zugriff 08.11.2024.

Schwab, B., Beil, C., Kolbe, T. H. (2020): Spatio-Semantic Road Space Modeling for Vehicle – Pedestrian Simulation to Test Automated Driving Systems. Sustainability, 12 (9), 3799. https://www.mdpi.com/2071-1050/12/9/3799, letzter Zugriff 08.11.2024.

Kontakt

M. Eng. Philipp Egert
egertphilipp@gmail.com

Prof. Dr. Theresa Knoblach
Ostbayerische Technische Hochschule Regensburg
theresa.knoblach@oth-regensburg.de

Prof. Dr. Daniela Wenzel
Technische Hochschule Würzburg-Schweinfurt
daniela.wenzel@thws.de

Dieser Beitrag ist auch digital verfügbar unter www.geodaesie.info.