Generation of Complex Road Networks Using a Simplified Logical Description for the Validation of Automated Vehicles*

Daniel Becker[1], Fabian Ruß[1], Christian Geller[2] and Lutz Eckstein[3]

Abstract— Simulation is a valuable building block for the verification and validation of automated driving functions (ADF). When simulating urban driving scenarios, simulation maps are one important component. Often, the generation of those road networks is a time consuming and manual effort. Furthermore, typically many variations of a distinct junction or road section are demanded to ensure that an ADF can be validated in the process of releasing those functions to the public.

Therefore, in this paper, we present a prototypical solution for a logical road network description which is easy to maintain and modify. The concept aims to be non-redundant so that changes of distinct quantities do not affect other places in the code and thus the variation of maps is straightforward. In addition, the simple definition of junctions is a focus of the work. Intersecting roads are defined separately, are then set in relation and the junction is finally generated automatically.

The idea is to derive the description from a commonly used, standardized format for simulation maps in order to generate this format from the introduced logical description. Consequently, we developed a command-line tool that generates the standardized simulation map format OpenDRIVE.

I. INTRODUCTION

In recent years, the interest in autonomous driving has considerably increased, both in research and society. To ensure the safety of automated driving functions (ADF), verification and validation methods need to be developed and established. The approval of an ADF for public road traffic demands compliance with safety requirements regarding its functionality. The requirements include an error-free behavior over several billion test kilometers. For example Wachenfeld and Winner [1] state that the distance to prove an interstate pilot amounts to 6.62 billion test kilometers, with a theoretically 50% chance this prove would say that the ADF is twice as good as a human driver. Since this distance seems unreal and economically not acceptable, Schuldt [2] advise a scenario based-testing. In order to shorten the test time, save costs and detect functional errors at an early development stage, simulation is well suited. It is also appropriate to create any number of scenarios and to explore critical driving situations without endangering people.

[1]Daniel Becker and Fabian Ruß are with the automated driving department of the Institute for Automotive Engineering (ika), RWTH Aachen University, 52074 Aachen, Germany {daniel.becker, fabian.russ}@ika.rwth-aachen.de

[2]Christian Geller is with RWTH Aachen University, 52062 Aachen, Germany christian.geller@rwth-aachen.de

[3]Lutz Eckstein is head of the Institute for Automotive Engineering (ika), RWTH Aachen University, 52074 Aachen, Germany lutz.eckstein@ika.rwth-aachen.de
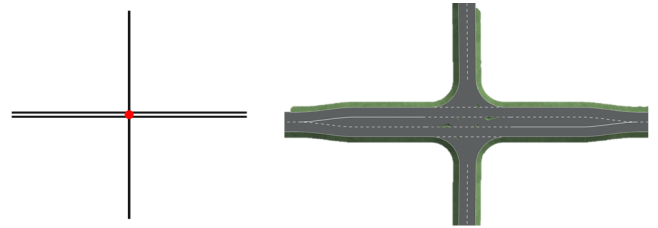
Fig. 1: Graphical example of input and output of the road generation tool. As a minimal input, two reference lines with assigned road types and a coupling point are sufficient to generate an intersection written in valid OpenDRIVE, visualized in CarMaker 8.0[1].

Scenario-based testing introduces three different stages to describe scenarios systematically: functional, logical and concrete, cf. [3]. The term functional states a linguistic description which allows experts to talk about scenarios at an early stage of the development process. In logical scenarios, parameters and ideally their probability distributions are provided. Finally, concrete scenarios specify a distinct value for each parameter which makes them feasible to be executed reproducible in a simulation or on proving grounds.

This kind of description uses the subdivisions of the six-layer model described by Bock et al. in [4], which is based on the layer model in Schuldt [2]. In this model, a scenario is divided in basic components and only an interaction between all the six layers represents a complete scenario. The model is illustrated in Fig. 2. The three lower layers describe the static part of the scenario. Layer four focuses on moving objects, whereas layer five and six describe environmental conditions and vehicle-to-everything (V2X) communication, respectively. Detailed discussion of layer four on German motorways can be found in Weber et al. [5].

Besides layer four which describes the dynamic behavior of all traffic participants, the static scenery has to be taken into account as well. The latter consists of the road network and all non moving objects which are relevant for the scenario. As mentioned, it is described within layer one to three of the six layer model and has to be defined in a logical and concrete manner, respectively. The open source standard format OpenDRIVE [6] is suitable for concrete descriptions of road networks and is widely used in industry and research. However, by now there is no standardized logical road description format which is easy to use and capable of being transferred into OpenDRIVE.
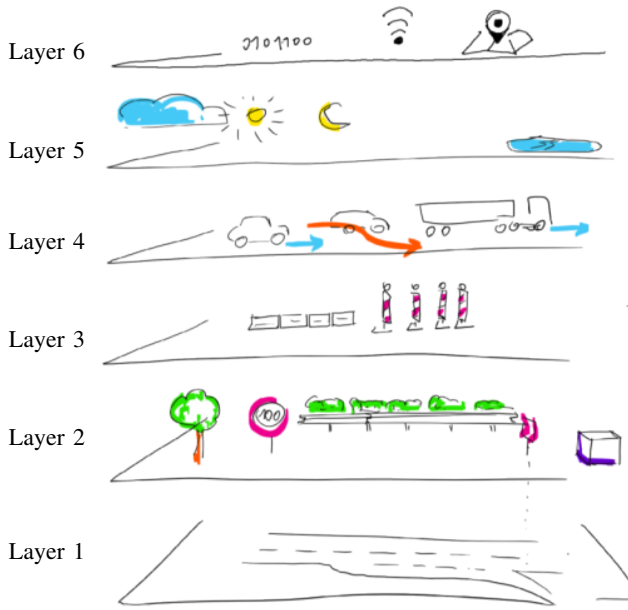
Fig. 2: The data layers for the description of a scenario following [4] and [7].

Current research initiatives such as SET Level 4to5 which is funded by the Federal Ministry for Economic Affairs and Energy (BMWi) of Germany, address simulative scenario-based testing with a focus on inner city traffic. Consequently, the concept presented in this paper aims to offer a description for road networks which allows easy modifications of the map due to a non-redundant design. Further, it is possible to provide limited information for intersections and lane definitions since some default configurations and dependencies for main and access roads are defined. The input shall be translated into a concrete standardized road description format. Therefore, with the logical description comes a road generation tool that calculates all required OpenDRIVE quantities from the simplified input. We use XML to store the input and the corresponding OpenDRIVE output is saved in XML as well. This way we attempt to close a gap in the pipeline of defining logical scenarios on intersections and make them executable in a concrete manner.

An example which outlines the idea of the concept is shown in Fig. 1 where an X-junction is defined by a main road intersected by an access road as the input and the result is a verified OpenDRIVE map which can be used in common automotive simulation software.

The remainder of this paper is organized as follows. First, Sec. II introduces related work to the topic, followed by Sec. III where input format is described in more detail. Then, details on the implementation are documented in Sec. IV. Next, in Sec. V several results are presented and finally a conclusion and outlook is given in Sec. VI.

II. RELATED WORK

During the research initiative PEGASUS [8] the concepts of scenario based testing have been developed and utilized within the domain German motorway. Furthermore, the main focus has been on layer four. Motorways usually consist of parallel lanes which follow curves with big radii. Such a road is not complicated to describe and generate. It can be done automatically with low effort. Menzel et al. [9] propose a method to create a standardized road network in the OpenDRIVE format from a logical (and functional) road description. A similar approach has been done by Noyer et al. [10] who created a format called SimplifiedRoad which is an XML schema to describe long streets without junctions. This logical description is primary meant to easily create concrete highway sections and cannot specify all details in OpenDRIVE. However, when moving the testing domain to inner city scenarios such as the driving behavior on intersections, layers one to three (cf. Fig. 2) are much more complex both in the logical description and in terms of the concrete format.

There are basically two different ways of creating virtual roads for simulation in the industry. In the first variant, the streets are build based on recorded real data. Disadvantages are the effort of collecting sufficient information of road networks and the restricted possibility in track variation. Examples for such systems are Road2Simulation presented by Richter and Scholz in [11] and services of the company atlatec[2]. The second variant is based on the manual creation of a the track with a visual track editor. This way, a user is required to assemble the road manually. Examples of such systems are Road Designer [12], CarMaker (IPG Automotive) and others. Those tools are useful for creating distinct road networks in an easy to use way but an automated generation of a huge number of similar road networks is hard to realize.

III. LOGICAL ROAD DESCRIPTION

The proposed logical road description is derived from the reference line concept used in OpenDRIVE [6]. This means that each lane is described by an offset from a reference line defined in the x, y-plane (topview). However, in contrast to OpenDRIVE we introduce some simplifications and eliminate redundancies along the reference line. Each road segment consists of the geometric primitives line, arc and spiral, which is motivated by the federal guidelines for road constructions in Germany cf. [13] and [14]. A simplified illustration of the road network's XML structure is shown in Fig. 3. On the first level under the `roadNetwork` tag all `segments` are defined and links which connect them can be stated (node "seg. assembly"). In addition, "loose ends" of segments can be automatically connected within the `closeRoadNetwork` tag. As illustrated, segments might be T-/X-junctions, roundabouts and connection roads.

In the following sections, the reference line is examined in more detail, followed by the description of the input for the logical road network concept. The documentation is based on the nodes shown in Fig. 3.
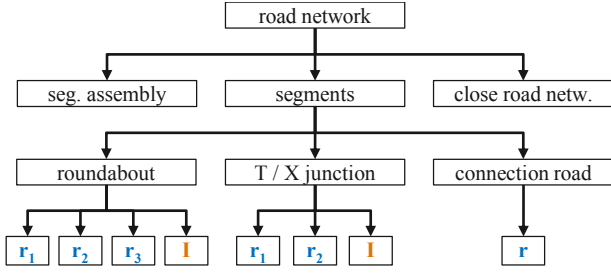
Fig. 3: Hierarchical structure of the road network. On the first level all segments and their relation to each other are defined. In addition, it is possible to define the automatic connection of two road ends. Segments are defined as roundabouts, junctions and connection roads. r_i and I describe roads and intersection definitions, respectively.
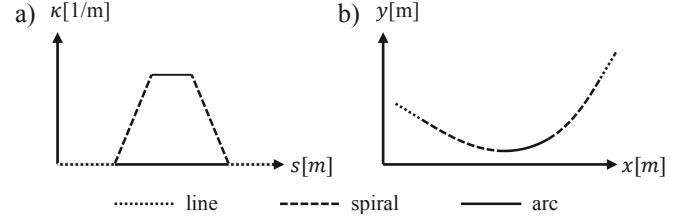

Fig. 4: Qualitative example for a curve defined by the curvature graph. The plan view results from two integrations over s. Geometric primitives: line-spiral-arc-spiral-line. Positive curvature values correspond to left turns in positive s direction.

A. Reference Lines

In order to ensure a smooth driveability along the road network, we define that the curvature has to be continuous along roads where curvature is defined as the reciprocal of the radius. Therefore, the definition of curvature over distance, a start point and initial heading is sufficient to uniquely describe a road's course. At the same time, this implies that in contrast to OpenDRIVE it is not necessary to state the start point and heading of each geometric element (i.e. line, spiral, arc). Fig. 4 shows the definition of a road segment that is composed of the geometric primitive sequence line-spiral-arc-spiral-line.

As shown in Fig. 4 a), the curvature $\kappa(s)$ is a piecewise linear defined function over distance. Since we use lines, spirals and arcs, three definitions for those pieces are required. A line is described by $\kappa(s) = 0$ which means, only a length L needs to be specified. An arc's course is defined by a constant curvature $\kappa(s) = \kappa_{\mathrm{arc}} = \mathrm{const}$. Therefore, an arc is defined by a length L and a radius R. Note that we use radii in the input description because they are more intuitive than the curvature. Finally, a spiral is characterized by a linear change of curvature: $\kappa(s) = as$, where a is constant (for details on spirals cf. [13]). Consequently, in addition to the length L, the start and end radius R_s and R_e need to be defined for a spiral to be unique. Those definitions are then being concatenated to a curvature graph and result in an unambiguous road course in the x, y-plane as in 4 b), if either the initial values for position and orientation at the start are provided or the road is connected to another segment without a curvature jump.

B. Segments

In the following, the segments which might be defined in the road network input are described starting with connection roads, followed by junctions and roundabouts.

Connection roads consist of single `road` tags defined by a reference line and definition for lanes and signals in the s, t-coordinate system. The latter is a moving coordinate system along the distance s. The t-coordinate is defined as the offset orthogonal to the reference line at position s. Lane

extension and merging follow 3rd-degree polynomials which are stated by their start and end point in s, t-coordinates, the corresponding polynomial coefficients are calculated by the road generation tool.

At least two and at most three `road` entities are required to create a T-junction. Often one road is the dominant or main road in an intersection, which can be exploited by defining it as a single road that is intersected by an access road. Alternatively, three single roads can be combined to form a T-junction by defining one reference road and the angles at which the other two roads are connected. The main idea of the intersection concept is that each road is defined in its own local coordinate system (cf. Fig. 5 a)) and is combined afterwards, as shown in Fig. 5 b). E.g., the T-junction in Fig. 5 consists of a main road described by an arc and an access road whose course follows a line. The main road is intersected by the access road approximately at its center (s_1) whereas the additional road is connected at its origin ($s_2 = 0$) under the angle α. The reference coordinate system of the segment now is at the junction's center with the x-axis parallel to the tangent of the main road at this point. The described road combinations are defined within `intersectionPoint`-tags which are part of the I-knots of Fig. 3.

In addition to topological information of the junction, it is possible to define the intersection area more detailed in a `coupler`-tag. First, for each road arm the distance from the intersection point to the beginning of the junction can be specified. The latter means the point where turning lanes start to leave the course of the reference line. They are specified by their minimal radius, the concrete course is calculated automatically (cf. Sec. IV-B). Fig. 7 shows the concept of the intersection area. Furthermore, additional lanes for left and right turns can be added ad libitum. The descriptions of T- and X-junctions are a lot alike and thus X-junctions can be created analogously to the preceding paragraph about T-junctions.

Roundabouts are defined similar to junctions with the difference that the reference road is of circular form (for now a circle) and every access road is assigned at a position s with an angle α to this reference element.
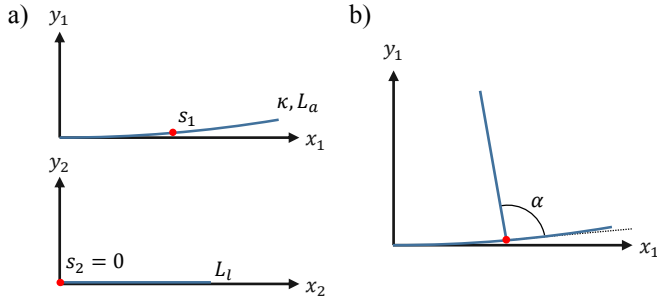
Fig. 5: Two roads are coupled to form a T-junction. Road 1 is defined as an arc by length and curvature whereas road 2 is a straight line with length L_l. On both roads the intersection point along s is defined. In addition, the angle α at this point is stated.
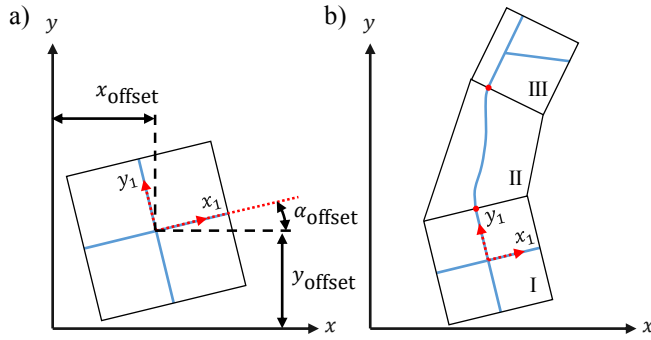

Fig. 6: Three segments are concatenated to form a road network. First, one reference segment has to be set (I). For further connections only the two endpoints need to be specified because concatenation follows the continuous curvature condition.

C. Combine Segments

Once all desired segments are specified, those may be concatenated to form a road network. In general, all segment types can be combined with each other. The only condition is that the curvature at the connection point is mutual to fulfill the continuous curvature condition we introduced for reference lines. However, in the current state of the road generation tool the curvature has to be zero at the connection to combine two segments. Fig. 6 illustrates the combination of an X-junction, connection road and T-junction. In this example, segment I provides the reference coordinate system which can be placed at any position and orientation $x_{\text{offset}}, y_{\text{offset}}$ and α_{offset} in world coordinates. Next, the connections between I and II as well as II and III are defined in a linking tag by stating which segment endings are linked. With this information, the road generation tool is able to generate an OpenDRIVE map uniquely.

One more option of the road network concept is the automated generation of a compound curve (composed of two spirals with an arc between them) to close the gap between two loose road ends. This might be required if the road network should be used for endurance simulations where traffic agents shall drive in circles infinitely often

and no sources and sinks are implemented in the simulation environment. It is not possible to calculate a curvature graph which matches a desired endpoint in the x, y-plane from an known start point analytically because of the algebraic properties of spirals (cf. Sec. IV-A). Therefore, an optimization problem needs to be solved which attempts to find combinations of segment lengths s_i and a radius R which form a curve similar to Fig. 4 that matches the desired end point with $\kappa = 0$.

IV. ROAD GENERATION

The specified input may be translated into the road network standard format OpenDRIVE by a command-line tool. Since the input describes a simplified representation of the road network, missing information needs to be calculated to provide a verified OpenDRIVE map. In the following, some steps of these calculations are described starting with details on geometric properties of roads and lanes followed by the generation of intersections.

A. Geometry

As introduced in Sec. III-A, we assume that a road's reference line is curvature continuous and therefore, a single initial position and heading is sufficient to calculate its course. However, OpenDRIVE requires a start point and heading for each line, arc or spiral. In order to provide information on x, y-coordinates and orientation φ for each geometric primitive, a transformation depending on the geometry type is necessary. For a straight line, the following trivial equations result:

$$\varphi(s) = \varphi_0, \quad x(s) = s\cos(\varphi_0), \quad y(s) = s\sin(\varphi_0). \quad (1)$$

With a segment of a circle, however, the orientation also changes along s. With constant curvature κ and the angle $\varphi(s) = s\kappa + \varphi_0$ this results in:

$$x(s) = \frac{1}{\kappa}\left(\sin\varphi(s) - \sin\varphi_0\right)$$
$$y(s) = \frac{1}{\kappa}\left(\cos\varphi_0 - \cos\varphi(s)\right). \quad (2)$$

For a spiral the coordinates along s can be described with a parameter $a = \frac{1}{\sqrt{2Rs}} = \sqrt{\frac{\kappa}{2s}}$ which is constant at any point of the spiral (cf. [13]) and the following equations hold:

$$x = \frac{1}{a}\int_0^{as}\cos\left(t^2\right)dt, \quad y = \frac{1}{a}\int_0^{as}\sin\left(t^2\right)dt. \quad (3)$$

The formulas in (3) are called Fresnel integrals and it can be shown that there is no analytical solution for them. However, they can be approximated by a power series expansion for sine and cosine. With help of (1) to (3) missing information for each reference line can be calculated such that Open-DRIVE requirements are fulfilled.

In a further step, lanes are added to the reference line. Each is described by a lane type, a marking, and especially by a width w, which for each lane i corresponds to a polynomial $w_i(s) = as^3 + bs^2 + cs + d$ with coefficients a, b, c, d. The width for each lane is defined separately and the final outer t-coordinate of the road results from the sum of all
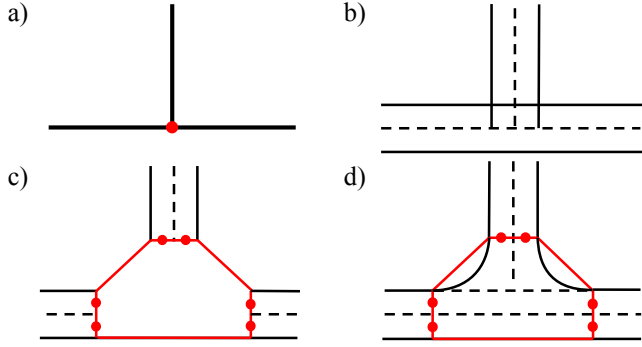
a) b)

c) d)

Fig. 7: From logical definition to concrete intersection. After the reference lines are in the correct position a), corresponding lanes are generated (b). Subsequently, the intersection area is cut free (c) and all required connection lanes are generated automatically (d).

(a) find helping point   (b) arc construction   (c) connected road

Fig. 8: Process for connecting roads in junction area.

$w_i(s)$. Lanes are divided into lane sections in which those polynomials are valid. Note that the s coordinate is zero at the beginning of a road but not at the beginning of a lane section. In the simplest case, w_0 is constant and $a = b = c = 0, d = w_0$ holds. If a lane widening occurs, the lane width increases from $w_i(s_0) = 0$ to the final width $w_i(s_0 + ds) = w_0$ over a given length ds. Additional conditions for modeling the four polynomial coefficients are continuous transitions ($w_i'(s) = 0$) at start and end of the widening s_0 and $s_0 + ds$, respectively. This case, the four coefficients can be uniquely solved with help of the described conditions. A lane lapse behaves analogously but starting with $w_i(s_0) = w_0$ and ending with $w_i(s_0 + ds) = 0$.

B. Intersections

Besides the reference lines, the intersections and round-abouts specified in the input file have to be processed in order to form valid OpenDRIVE entities. Each intersection or roundabout consists, as shown in Fig. 3, of corresponding roads r_i as well as more detailed information I. Besides the size of the intersection area, it is also specified which lanes of the individual roads are connected or whether additional lanes are added before the intersection.

A segment is constructed such that the junction midpoint lies in the origin and intersects the reference road without any angle. Further roads r_i intersect the origin at the specified road position $s_{I,i}$ at a defined angle $\alpha_{I,i}$, so that the starting point and the starting angle of each geometry of this road must be shifted by a translatory offset $\tilde{x}, \tilde{y}$ and rotated around the angle $\tilde{\varphi} = \alpha_{I,i}$. The translatory offset can be determined from the coordinates at position $s_{I,i}$ so that all segments with their starting points x_0, y_0, φ_0 can be transformed as follows.

$$x = \cos(\tilde{\varphi})(x_0 - \tilde{x}) - \sin(\tilde{\varphi})(y_0 - \tilde{y})$$
$$y = \sin(\tilde{\varphi})(x_0 - \tilde{x}) + \cos(\tilde{\varphi})(y_0 - \tilde{y}) \qquad (4)$$
$$\varphi = \varphi_0 - \tilde{\varphi}.$$

Before inserting new connecting roads, the junction area has to be generated, which requires a cut in the reference line of each road $r_i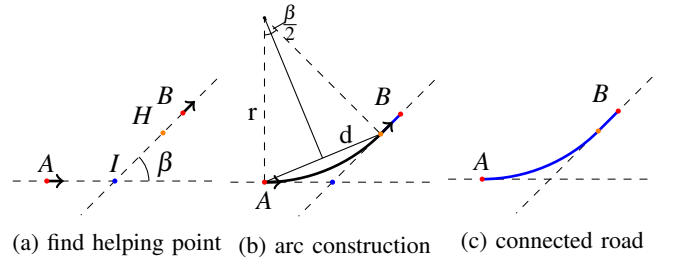$. From the junction midpoint at $s_{I,i}$ we define s_A as the length along s which spans the junction area in one direction. Note that for simplicity we assume s_A in both directions of the reference line in the documentation. In practice those are two variables. From the defined reference line of r_i, two parts are extracted: the part up to the beginning of the junction $(-\inf, s_I - s_A]$ and the part behind the junction $[s_I + s_A, \inf)$. The two relevant parts of the reference line are now iteratively copied over all geometries to generate the new exit roads of the intersection. Depending on the cut of a straight line, arc or spiral, the respective segment of the reference line has to be adapted. The result of this step can be illustrated as shown in Fig. 7 c).

After every street is cut and correctly positioned, the next step is calculating the reference lines of the lanes inside of the intersection area. Each lane is generated as an individual street with its reference line, whereby the single-lane lies to the right of the reference line. Based on starting point A and endpoint B as well as the angles φ_A and φ_B the new reference line of a lane is calculated. As a simplification, only straight lines and arcs are used since the discontinuity in the curvature can be neglected at low speeds. Note that the assumption still yields continuous angles along s. The setup for one connection road is illustrated in Fig. 8. A is the end of the road entering the intersection area and B is the corresponding start of the newly generated road leaving the junction. In the following, the geometric construction of the new reference line is described.

In the first step, the endpoints are extended with straight lines to calculate the intersecting point I.

$$I_x = \frac{B_y - A_y + (\tan(\varphi_A)A_x - \tan(\varphi_B)B_x)}{\tan(\varphi_A) - \tan(\varphi_B)}$$
$$I_y = \tan(\varphi_A) \cdot I_x + (A_y - \tan(\varphi_A)A_x). \qquad (5)$$

The euclidean distances $d_A = \|A - I\|$ and $d_B = \|B - I\|$ between endpoints and intersection point are then being calculated and the road with greater distance is extended by a line until the new distances are equal, which introduces the new helping point H at the end of the extended line. Finally, the two points are connected by an arc with radius r and length l. The following equation determines the parameters

Fig. 9: OpenDRIVE output of a road network generated by the tool which processes the developed logical road description. The road network consists of three junctions, one roundabout, one connection road and four automatically generated road network closing roads. Visualization with internal Unity[3] tool.

of the circle, where $\beta = \varphi_B - \varphi_A$:

$$r = \frac{\sqrt{(H_x - A_x)^2 + (H_y - A_y)^2}}{2 \cdot \sin\left(\frac{\beta}{2}\right)} \quad (6)$$

$$l = |r \cdot (\varphi_B - \varphi_A)|.$$

This procedure is repeated until all connection roads, i.e. connection lanes, inside the junction are calculated.

V. RESULTS

Recall Fig. 1, which shows a generated X-junction. The input description of this map consists of 32 lines whereas the resulting OpenDRIVE file is 870 lines long. This shows one benefit of the proposed road description which is straightforward to maintain.

In Fig. 10 the OpenDRIVE output of two concatenated T-junctions is shown. Each segment consists of a main road which follows an arc and one access road that is modeled as a straight line. The angle between both roads is $\alpha = \frac{\pi}{2}$. Further, a link between both junctions is applied at the end of the respective access road. Both junctions are defined mostly the same, the only differences for the upper junction are that the radius of the main road is smaller and that an option is activated which automatically generates lane widenings for left turns on main roads. In fact, only two chars are different in the respective junction definitions which makes the description easy to vary. In addition, the input file consists of four defined roads whereas the resulting OpenDRIVE file contains 18 road definitions. These effects have impact on the overall amount of code which can be seen in Table I line 2. The input file requires only 5.3% as many chars as the concrete OpenDRIVE file.

As a general overview, in Fig. 9 the result for a considerably large road network is illustrated. It is described entirely by the presented logical description and then is instantiated as OpenDRIVE with help of the developed tool. The network consists of three intersections, one roundabout, one connection road and four instances of the close road network function which generates compound curves between segments automatically. As shown in Table I line 3, the input file is significantly smaller in terms of code length than the resulting OpenDRIVE file. In addition, only little parts of
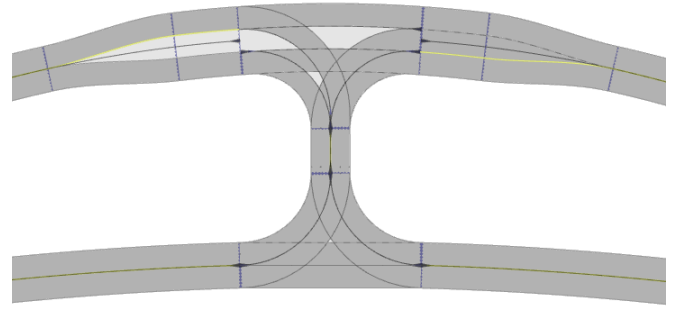
[3]https://unity.com/

Fig. 10: OpenDRIVE Output of two concatenated T-junctions. Both junctions are almost identically defined. The only differences for the upper junction are that the radius of the main road is changed and an option that automatically generates lane widening for left turns is activated. Note that black lines indicate lane courses, not their marking. Visualization in CarMaker 8.0.

TABLE I: Comparison of OpenDRIVE and logical description in terms of amount of code.

File	OpenDRIVE		Logical Description	
	Lines	Chars	Lines	Chars
XJunc (Fig. 1)	870	32,468	3.7%	4.1%
TwoTJunc (Fig. 10)	937	36,414	5.0%	5.3%
HugeRoadNet (Fig. 9)	5,477	210,215	4.0%	4.8%

the input have to be changed (e.g. the curvature of one connection road) in order to obtain a road network of similar topology but with thousands of changes in the OpenDRIVE file. This way, parameter spaces for distinct quantities may be explored and many concrete road networks can be generated using one *logical* definition.

All generated OpenDRIVE files are verified against current OpenDRIVE XSD schemas of version 1.4[4] and 1.5[5]. In addition, we evaluated selected tracks with ODR Viewer[6],

[4]http://www.opendrive.org/tools/OpenDRIVE_1.4H.xsd
[5]http://www.opendrive.org/tools/OpenDRIVE_1.5M.xsd
[6]http://www.opendrive.org/tools/odrViewer.zip

CarMaker 8.0 and RoadRunner[7] by making sure the import works, the visualization is plausible and a vehicle is able to pass the generated track.

VI. Conclusion and outlook

In this paper, we introduced a prototypical concept to describe road networks in a logical manner with the possibility to generate a standardized concrete format which can be used in most simulation environments. The results are ready to use simulation maps which can be easily varied.

A next step could be the analysis of real road networks to calibrate the assumed parameters, e.g. for main and access roads or radii and width of lanes inside junctions. Further, the logical input and road generation tool might be extended with more features such as nested junctions or objects next to the road. In addition, the concatenation of segments could be improved. Curvatures unequal to zero might be allowed at the connection between two segments, or the close road network function could be enhanced such that linking elements among two junctions can be generated automatically.

References

[1] W. Wachenfeld and H. Winner, "The release of autonomous vehicles," in *Autonomous driving*. Springer, 2016, pp. 425–449.

[2] F. Schuldt, "Ein Beitrag für den methodischen Test von automatisierten Fahrfunktionen mit Hilfe von virtuellen Umgebungen," Ph.D. dissertation, 2017.

[3] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1821–1827.

[4] J. Bock, R. Krajewski, L. Eckstein, J. Klimke, J. Sauerbier, and A. Zlocki, "Data basis for scenario-based validation of had on highways," in *27th Aachen Colloquium Automobile and Engine Technology*, 2018.

[5] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki, and L. Eckstein, "A framework for definition of logical scenarios for safety assurance of automated driving," *Traffic Injury Prevention*, vol. 20, no. sup1, pp. S65–S70, 2019, pMID: 31381437. [Online]. Available: https://doi.org/10.1080/15389588.2019.1630827

[6] M. Dupuis, E. Hekele, and A. Biehn, *OpenDRIVE Format Specification, Rev. 1.5*, 1st ed., VIRES Simulationstechnologie GmbH, Feb 2019.

[7] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1813–1820.

[8] J. Mazzega, "Pegasus method: An overview," Lilienthalplatz 7, 38108 Braunschweig, Germany, May 2019, contributions: PEGASUS Project Partners.

[9] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and M. Maurer, "From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2383–2390.

[10] U. Noyer, A. Richter, and M. Scholz, "Generation of highway sections for automated test case creation," in *Driving Simulation Conference Europe 2018 VR*, A. Kemeny, F. Colombet, F. Mérienne, and S. Espié, Eds., 2018, pp. 171–174. [Online]. Available: https://elib.dlr.de/117113/

[11] A. Richter and M. Scholz, "Deploying guidelines and a simplified data model to provide real world geodata in driving simulators and driving automation," *Transportation Research Part F: Traffic Psychology and Behaviour*, Mai 2017. [Online]. Available: https://elib.dlr.de/110094/

[12] *ROD - Road Designer - Product Data Sheet*, VIRES Simulationstechnologie GmbH, Grassinger Strae 8, 83043 Bad Aibling, Germany, June 2013.

[13] R. Baier, *Richtlinien für die Anlage von Stadtstraßen: RASt 06*, korr. nachdr ed. Köln: FGSV-Verl., 2008.

[14] Straßen.NRW, "Straßenbauplanung in NRW: Der Weg vom Planungsauftrag bis zum Bau," 2019. [Online]. Available: https://www.strassen.nrw.de/de/planung-bau/mit-planung-zum-bau.html

[7] https://www.vectorzero.io/roadrunner