Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles

Moritz Klischat*, Edmond Irani Liu*, Fabian Höltke, and Matthias Althoff

***Abstract*— The safety validation of motion planning algorithms for automated vehicles requires a large amount of data for virtual testing. Currently, this data is often collected through real test drives, which is expensive and inefficient, given that only a minority of traffic scenarios pose challenges to motion planners. We present a workflow for generating a database of challenging and safety-critical test scenarios that is not dependent on recorded data. First, we extract a large variety of road networks across the globe from OpenStreetMap. Subsequently, we generate traffic scenarios for these road networks using the traffic simulator SUMO. In the last step, we increase the criticality of these scenarios using nonlinear optimization. Our generated scenarios are publicly available on the CommonRoad website.**

I. Introduction

Virtual testing is an important tool for validating the safety of automated vehicles, as it exposes potential defects of the algorithms under test. Having a large variety of challenging traffic scenarios is vital for effective and efficient testing of motion planning algorithms. While carrying out simulations using data recorded from test drives provides us with realistic scenarios, the required data collection is often overly expensive and time-consuming [1]. Even though the number of publicly-available datasets has increased over the last few years, e.g., [2]–[5], they usually feature only a small number of maps and require much effort to record.

Our framework generates a database of safety-critical scenarios for scenario-based testing [6] of motion planing algorithms for automated vehicles. It consists of

1) Extracting interesting road intersections worldwide from OpenStreetMap (OSM) [7] by using our Globetrotter tool (see Sec. III).
2) Generating safety-critical test scenarios by first populating the extracted intersections with traffic participants through the traffic simulator SUMO [8].
3) Optimizing the criticality of the obtained scenarios by using a generalizable criticality criterion (see Sec. IV).

A. Related Work

Below, we concisely review related works on approaches towards automatically creating virtual representations of road networks and generating critical test scenarios for automated vehicles.

*The first two authors have contributed equally to this work.
All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
{moritz.klischat, edmond.irani, fabian.hoeltke, althoff}@tum.de

1) Creating road networks: Generative approaches construct road networks, e.g., based on abstract specifications [9]. Moreover, a suite of road networks with a defined coverage of road curvatures is generated in [10] using satisfiability modulo theories. Road networks that lead to the failure of lane-keeping assistants are generated procedurally in [11] through mutating road networks using genetic algorithms.

Alternatively, road networks can also be created from external sources. In [12]–[16], the authors extract road networks from aerial and satellite images with the help of computer vision techniques. These works are capable of extracting high-level geometric information of road networks; however, lane-level information concerning motion planners of automated vehicles is not reconstructed. Promising works towards the reconstruction of road networks with lane-level detail from aerial images can be seen in [17], [18]. Similarly, while creating road networks for single lanes from OSM data is straightforward [19], creating those with lane-level detail is a more challenging problem [20].

2) Generating critical test scenarios: Creating test scenarios for automated vehicles using traffic simulators is proposed, e.g., in [21], [22]. Realistic scenarios can be obtained by calibrating their simulations through real-world measurements. By using criticality metrics, safety-critical scenarios are filtered [22]; however, these situations occur only rarely, in the main.

To efficiently obtain critical scenarios, importance sampling from large databases of recorded traffic data is proposed [23]. Criticality metrics are combined with the occurrence rates to efficiently sample critical scenarios representative of real-world driving conditions [24]. Other approaches use optimization to create critical scenarios based on these metrics [25], [26]. Similarly, falsification methods can detect scenarios that falsify a motion planner with respect to a given safety specification [27], [28]. Parameter regions for critical scenarios based on constraint satisfaction are computed in [29]. In our previous work, we presented an optimization-based method to increase the criticality of initially uncritical traffic scenarios [30], [31] by decreasing the space of possible solutions for the vehicle under test, called the *drivable area*.

B. Contributions

In contrast to previous work on generating test scenarios through simulation, our approach is particularly efficient since we combine the automatic extraction of road networks from OSM with a traffic simulation followed by an increase of its criticality. By exploiting the large variety of road networks around the world, we create complex, yet realistic

road networks which currently no procedural map generator is capable of producing. In contrast to existing work, our approach does not rely on test drives nor other real-world traffic data, thus it is able to efficiently create many traffic scenarios at low costs. The resulting scenarios are independent of the vehicle under test, due to our criticality metric based on the drivable area.

II. Overview

An overview of our approach is presented in Fig. 1. In the following subsections, we introduce each component.

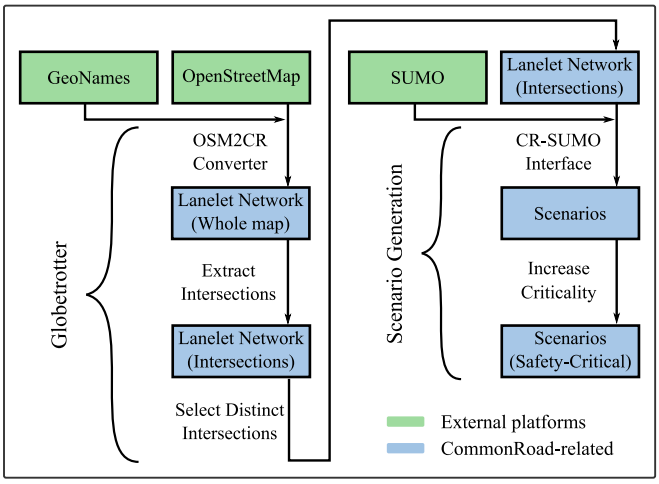


Fig. 1. Our pipeline for generating safety-critical scenarios.

A. Platforms

1) CommonRoad: The CommonRoad (CR) benchmark suite[1] is an open-source framework that provides a collection of traffic scenarios for motion planning algorithms. Scenarios in CommonRoad consist of *road networks*, *static obstacles*, and *dynamic obstacles* that represent all possible types of traffic participants. In this work, we focus on cars, trucks, and bicycles. *Road networks* in CommonRoad are described by lanelets [32] (see Fig. 2). Lanelets are defined by their left and right bounds, which are modeled by polylines. Furthermore, lanelets are connected through successor-predecessor and lateral-adjacency relations and contain additional information such as the speed limit. Additionally, we define *forking points* as the points on the centerlines of lanelets where lanelets split or multiple lanelets merge.

2) OpenStreetMap: OSM is an open-source project that provides geographic data worldwide. The main structure of OSM data is defined by three elements: *nodes*, *ways*, and *relations*. *Nodes* are geographic points defined by their latitude and longitude; *ways* are tuples of *nodes* and represent elements such as roads and boundaries of areas; *relations* are groups of *nodes*, *ways* and other *relations*. Fig. 3a shows a map taken from OpenStreetMap.

[1]https://commonroad.in.tum.de/

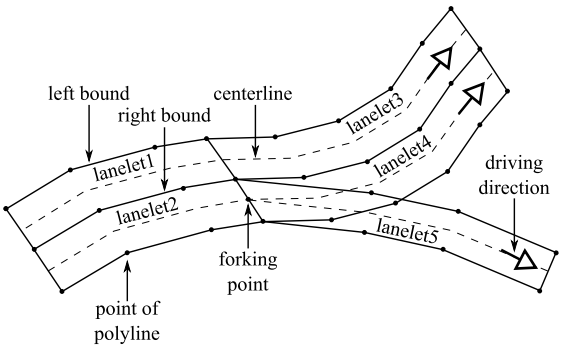


Fig. 2. Lanelet network representation.

3) GeoNames: GeoNames[2] is a free geographic database which covers all countries and contains over 11 million placenames of cities from all over the world. The provided geographical information includes global coordinates, postal codes, population, etc.

4) SUMO: This open-source microscopic traffic-simulation package is designed to handle large road networks. SUMO models individual vehicles and their interactions using models for car-following, lane-changing, and intersection behavior.

B. Converters

Since most of the software platforms mentioned above have individual formats and map representations, we use different converters and interfaces to bridge these platforms.

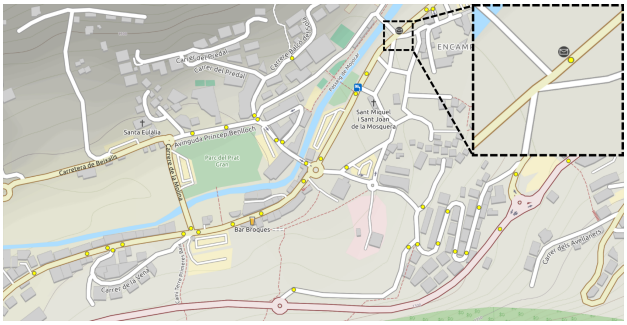
1) OSM2CR converter: It converts OSM maps to CommonRoad lanelet networks. While OSM provides map data for almost any place in the world, their level of detail is not yet suited for automated vehicles: The motion planners of automated vehicles and traffic simulators typically require lane-level information. To resolve this issue, in the first step, the topology of the lanelet network, i.e., the connections at intersections, needs to be estimated. Next, spatial information of individual lanes is deducted accordingly. Fig. 3b shows a converted lanelet network via this converter.

2) CR-SUMO interface: It enables the communication between CommonRoad and SUMO by a) converting the CR road network to SUMO format, b) generating configuration files for the simulation, and c) converting simulated vehicle trajectories to the CR format. We refer the interested reader to [33] for more details regarding this interface.

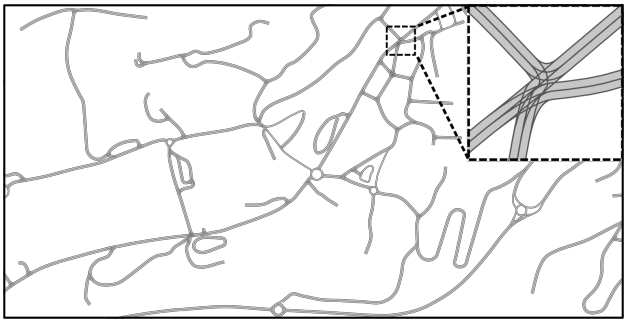
III. Globetrotter

To automatically extract interesting road networks from all over the world, we have developed the Globetrotter tool, which takes the road network data from OSM as its underlying input. As we want to create scenarios on distinct road networks, we mainly focus on extracting intersections. Below, we explain the major steps for extracting the intersections from OSM.

[2]https://www.geonames.org/



(a)



(b)

Fig. 3. (a) Map of Encamp, Andorra taken from OSM. (b) Conversion result into CommonRoad lanelet network via the OSM2CR converter.

A. Retrieving Candidate Regions

Clearly, there are intersections all over the globe, and they mostly vary according to region. Given that only 29% of the Earth's surface is covered by land[3], and that 10% of these regions accommodate 95% of the human population[4], sampling the Earth's surface with random coordinates is not very efficient. Assuming that intersections mostly occur near populated areas, we retrieve these populated candidate regions from GeoNames. To speed up the processing in the next steps, we can also divide the region into smaller subregions if the area of the region exceeds a certain value. The retrieved candidate (sub)regions are converted into lanelet networks via the OSM2CR converter.

B. Extracting Intersections

We denote the n-th forking point and the tuple of all forking points in a lanelet network as P_n and $\mathcal{P}$, respectively. For a given lanelet network, it is usually difficult to determine beforehand the number of intersections to be extracted. For this reason, instead of k-means-like algorithms [34], we apply the hierarchical agglomerative clustering (HAC) algorithm [35] to $\mathcal{P}$. HAC only requires a distance threshold d_{th} to limit the distances between clusters: a higher d_{th} entails larger intersections. Alg. 1 describes how the intersections are extracted from $\mathcal{P}$.

1) Clustering forking points (Alg. 1, lines 2-4): Initially, each forking point forms a cluster C_n with it being the only member: $C_n = \{P_n\}$. We denote the tuple of clusters by $\mathcal{C} := \langle C_1, C_2, \dots \rangle$, and the distance between two clusters C_i, C_j with single-linkage setting [35] by $d_{i,j}$:

$$d_{i,j} = \min\{\text{dist}(a,b) | a \in C_i, b \in C_j\},$$

where the operator $\text{dist}(\cdot)$ returns the Euclidean distance between two given forking points. In each iteration, the two clusters C_i, C_j with the minimum distance $d_{i,j} < d_{\text{th}}$ are merged into a new cluster $C' = C_i \cup C_j$. This process is repeated until no more clusters can be merged. Fig. 4a-4b show the dendrogram for clustering an exemplary lanelet network and the clustered forking points.

2) Creating intersections (Alg. 1, lines 5-10): For each remaining cluster $C' \in \mathcal{C}$, we determine the minimum radius r_{min} of a circle enclosing all forking points within the cluster. We enlarge this radius by a user-defined margin r_{mgn} to span a region of interest. We cut out all lanelets from this region, resulting in an intersection I, and additionally apply the following steps:

1) Lanelets that are not within the region of interest are removed.
2) Due to removed lanelets, we update the successor-predecessor and lateral-adjacency relations.

Fig. 4c shows the extracted intersections $\mathcal{I}$.

Algorithm 1 Extracting Intersections

Inputs: forking points $\mathcal{P}$, distance threshold d_{th}
Output: extracted intersections $\mathcal{I}$

1: $\mathcal{I} \leftarrow \emptyset$
2: ▷ Clustering forking points
3: $\mathcal{C} \leftarrow$ INITIALIZE($\mathcal{P}$)
4: $\mathcal{C} \leftarrow$ HAC($\mathcal{C}$, d_{th})
5: ▷ Creating intersections
6: **for** $C' \in \mathcal{C}$ **do**
7: $\quad I \leftarrow$ CUTLANELETS(C')
8: $\quad I \leftarrow$ POSTPROCESS(I)
9: $\quad \mathcal{I} \leftarrow \mathcal{I} \cup \{I\}$
10: **end for**
11: **return** $\mathcal{I}$

C. Selecting Interesting Intersections

Given the intersections $\mathcal{I}$ extracted from a lanelet network, we only keep those that are particularly interesting or distinct according to the following features:

- number of forking points;
- number of lanelets;
- number of crossing lanelets;
- number of predecessors and successors;
- area of lanelets;
- density of lanelets;
- angle between lanelets; and
- mean distance between forking points and their centroid.

We associate the interestingness of intersections with the dissimilarity between their features and those of other

[3]https://www.noaa.gov/
[4]https://ec.europa.eu/jrc/en

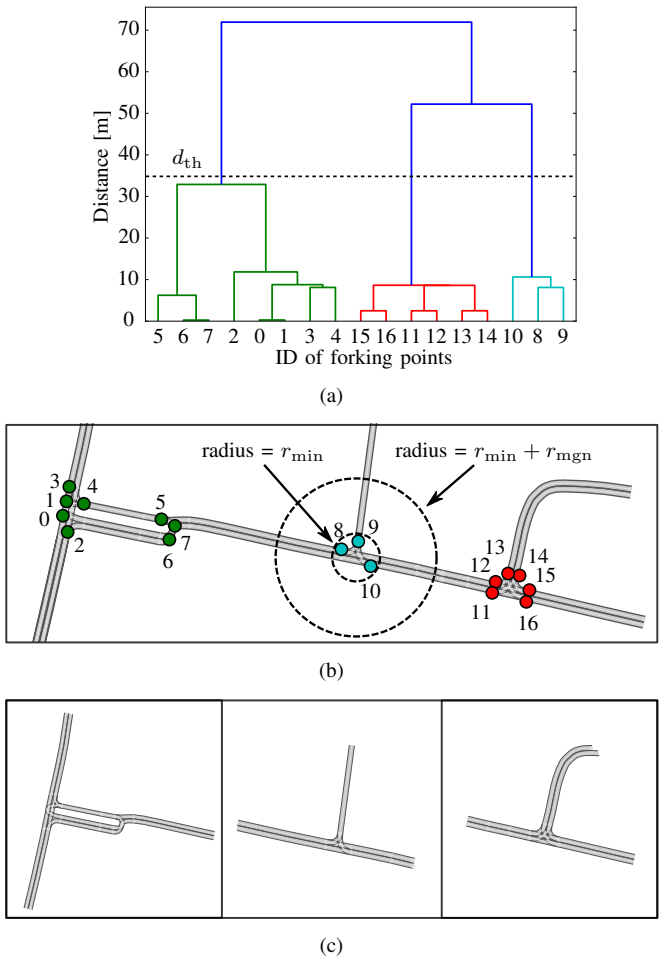


Fig. 4. (a) Dendrogram of the clustering result. Three clusters are generated with d_{th} set to 35 meters. (b) Forking points within one cluster have the same color. r_{mgn} is set to 15 meters. (c) Intersections extracted from the input lanelet network.

intersections. By doing so, we turn the selection of interesting intersections into a multivariate outlier (anomaly) detection problem. To solve this problem, we use the isolation forest (iForest) algorithm [36], since it is unsupervised, capable of efficiently handling multiple dimensions of features, and requires limited effort to hand-tune its parameters. In the training phase, a total of k isolation trees (iTrees) are trained with sets of randomly-selected intersections; in the detection phase, an anomaly score $s \in [0, 1]$ is assigned to each intersection by the iTrees [36], where an intersection with a score above a threshold s_{th} is considered an outlier. Fig. 5 presents a collection of distinct intersections. It should be recalled that we divide the candidate region into subregions if it is overly large, thus rendering the adopted iForest algorithm computationally tractable.

IV. Generation of Safety-Critical Scenarios

On the extracted maps, we simulate traffic participants using our previously-introduced CR-SUMO interface next. Since scenarios simulated with SUMO often yield uncritical, easy-to-solve motion planning problems, we subsequently increase their criticality using our approach [30], [31] for reducing the solution space. We first parametrize the initially obtained trajectories of other traffic participants in Sec. IV-B and, following that, formulate a nonlinear optimization problem with a criticality criterion specified in Sec. IV-D.

A. Motion Planning Problem

The system dynamics of the ego vehicle is defined by

$$\dot{x}_{\text{e}}(t) = f(x_{\text{e}}(t), u(t)),$$

where $x_{\text{e}}(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathcal{U}$ is the input vector with the set of admissible inputs $\mathcal{U} \in \mathbb{R}^m$.

The trajectories of n_{tp} other traffic participants are given by $x_i(t; p)$, $i \in \{1, ..., n_{\text{tp}}\}$ with parameters $p \in \mathbb{R}^{n_{\text{p}}}$, initial time t_0, and final time t_f. Initial candidates for these trajectories are obtained from SUMO; their parametrization is explained in more detail in Sec. IV-B. The occupied space $\mathcal{O}_i(t; p) \subset \mathbb{R}^2$ of a traffic participant is obtained through the $\text{occ}(\cdot)$ operator, i.e., $\mathcal{O}_i(t; p) = \text{occ}(x_i(t; p))$. We define the motion planning problem for the ego vehicle as a classical reach-avoid problem: given an initial state $x_{\text{e},0} = x_{\text{e}}(t_0)$, an input trajectory $u(t)$ has to be found to steer the ego vehicle into a goal region while not leaving the road surface $\mathcal{W}_{\text{lanes}} \in \mathbb{R}^2$ and avoiding the space $\mathcal{O}(t; p)$ occupied by all obstacles, i.e.,

$$\forall t \in [t_0, t_f] : \text{occ}(x_{\text{e}}(t)) \subseteq \mathcal{W}_{\text{lanes}} \backslash \mathcal{O}(t; p). \tag{1}$$

We obtain a motion planning problem by deleting a selected vehicle in a scenario simulated with SUMO and storing the initial state $x_{\text{e},0}$ of this vehicle. Vehicles with interesting maneuvers are automatically selected by using thresholds on the velocity and acceleration profiles or by identifying lane changes, turns or vehicles driving nearby.

B. Scenario Parametrization

In order to optimize the criticality of scenarios, we parametrize trajectories by the parameter vector p. We describe trajectories in lane-based coordinate systems, in which a state is defined as $x = [s_\xi, \dot{s}_\xi, s_\eta, \dot{s}_\eta]^T$. The subscripts ξ and η denote the longitudinal and lateral coordinates with respect to the centerline, respectively (see Fig. 2).

For the n_{tp} traffic participants, we only parametrize the longitudinal trajectory using translations $p^{\text{s}} \in \mathbb{R}^{n_{\text{tp}}}$, initial velocity variations $p^{\text{v}} \in \mathbb{R}^{n_{\text{tp}}}$, and acceleration variations $p^{\text{a}} \in \mathbb{R}^{n_{\text{tp}}}$, yielding $p = \left[p^{\text{s}}, p^{\text{v}}, p^{\text{a}}\right]^T$. The parametrized longitudinal position trajectory is given by

$$s_{\xi,i}(t; p_i) = \hat{s}_{\xi,i}(t) + p_i^{\text{s}} + p_i^{\text{v}} t + \frac{1}{2} p_i^{\text{a}} t^2. \tag{2}$$

From (2), the centerlines, and the dimensions of the vehicle, we obtain the occupied space $\mathcal{O}_i(t, p)$ of each traffic participant.

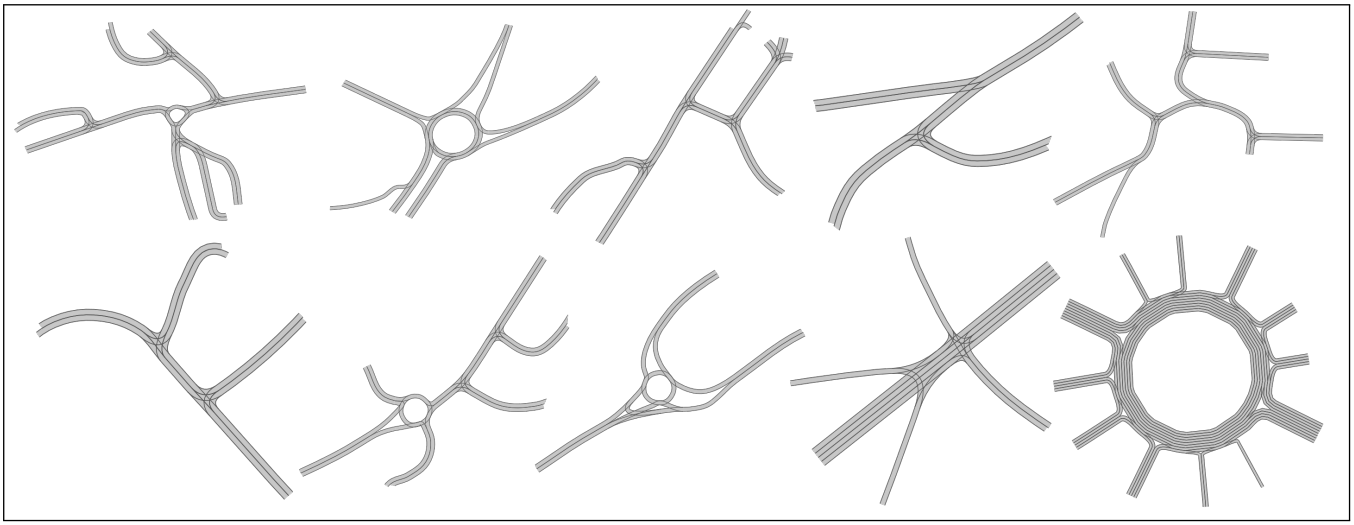


Fig. 5. Selected road intersections generated by Globetrotter ($s_{\mathrm{th}} = 0.9$).

C. Drivable Area

We denote a feasible solution to the motion planning problem defined in Sec. IV-A as $\chi(t; x_0, u(\cdot))$, where $u(\cdot)$ refers to the entire trajectory instead of a particular value $u(t)$ at time t. To quantify the criticality of a scenario, we use the solution space, which corresponds to the set of reachable states for $t \in [t_0, t_f]$ without collisions:

$$\mathcal{R}(t; x_0, \mathcal{O}(\,\cdot\,; p)) = \Big\{ \chi(t; x_0, u(\cdot)) \;\Big|\; \forall \tau \in [t_0, t_f] : u(\tau) \in \mathcal{U}, \; \mathrm{occ}\big(\chi(\tau; x_0, u(\cdot))\big) \subseteq \mathcal{W}_{\mathrm{lanes}} \backslash \mathcal{O}(\tau; p) \Big\}.$$

By applying the projection operator $\mathrm{proj}(x) : \mathbb{R}^n \to \mathbb{R}^2$, which projects the state space to the position domain, we obtain the *drivable area*

$$\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p)) = \bigcup_{x \in \mathcal{R}(t; x_0, \mathcal{O}(\,\cdot\,; p))} \mathrm{proj}(x). \tag{3}$$

An example for the drivable area in presence of an obstacle is depicted in Fig. 6.

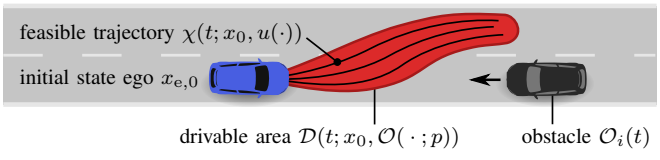


Fig. 6. Example of a drivable area for the time interval $[t_0, t_f]$.

To quantify the solution space, we introduce the function $\mathrm{area}(\mathcal{X})$ returning the area of a set. We write

$$A(t; p) \coloneqq \mathrm{area}\big(\mathcal{D}(t; x_0, \mathcal{O}(\,\cdot\,; p))\big)$$

to obtain the area profile of the drivable area over time. We compute the drivable area using our approach as in [37].

D. Optimization Problem

For increasing the criticality of the motion planning problem, we optimize the parameter vector p to obtain a desired, critical area profile $A_{\mathrm{crit}}(t)$:

$$\underset{p}{\mathrm{argmin}}\, \kappa(p), \quad \kappa(p) = \int_0^{t_f} \big(A(t; p) - A_{\mathrm{crit}}(t)\big)^2 \tag{4}$$

$$\text{subject to} \quad \forall t, \forall i, \forall j \neq i : \quad \mathcal{O}_i(t; p) \cap \mathcal{O}_j(t; p) = \emptyset. \tag{5}$$

The constraint in (5) ensures that no traffic participants collide with each other. In this work, we use the drivable area computed without any traffic participants and the scalar $\gamma \in]0, 1[$ which quantifies the reduction of the drivable area: $A_{\mathrm{crit}}(t) = \gamma \cdot \mathrm{area}(\mathcal{D}(t; x_0, \emptyset))$.

Since the drivable area is highly nonlinear with respect to the trajectories of other traffic participants and possibly subjected to local minima, we use particle swarm optimization [38] as in our previous work [30]. Furthermore, we implement a repair algorithm that enforces the collision constraint (5). To that end, we formulate the collision constraints as linear inequality constraints and correct infeasible solutions by computing the closest feasible solution using linear programming. A more efficient optimization is ensured by an a priori computation of relevant parameter intervals as presented in [30].

V. Evaluation

We demonstrate our approach by generating scenarios on a large variety of road networks from various places across the world. First, we obtain 576 road networks from 8 countries and 46 cities from Globetrotter, for each of which we simulate multiple scenarios using our CR-SUMO interface. After selecting interesting ego vehicles, we obtain 1402 scenarios for which we optimize the criticality. The resulting scenarios are added to our website[5].

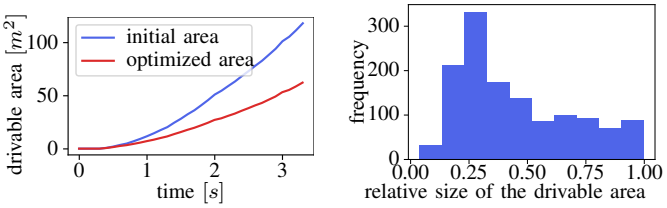
[5]https://commonroad.in.tum.de/

In Fig. 7a we compare the area profiles $A(t;p)$ of the drivable area in the optimized scenarios against the initial scenario obtained from SUMO: our approach is able to significantly decrease the drivable area, and it thus increases the criticality. Fig. 7b shows the distribution of the achieved reduction of the critical area. For most of the optimized scenarios, the drivable area ranges between 0.2 – 0.3 of its initial size.

TABLE I

PARAMETERS FOR CRITICALITY OPTIMIZATION

Drivable area computation	
max. acceleration ego vehicle $\lvert a_{max} \rvert$	$5.0\,\mathrm{m/s^2}$
time step size Δt	$0.1\,\mathrm{s}$
time horizon t_f	$3.4\,\mathrm{s}$
Constraints for optimization	
initial velocity variation	$[-3, 3]\,\mathrm{m/s}$
acceleration variation	$[-5, 2]\,\mathrm{m/s^2}$



(a) (b)

Fig. 7. (a) Size of the drivable area $A(t;p)$ over time, averaged over all scenarios. (b) Histogram of the size of the drivable area in the optimized scenarios relative to the initial scenarios.

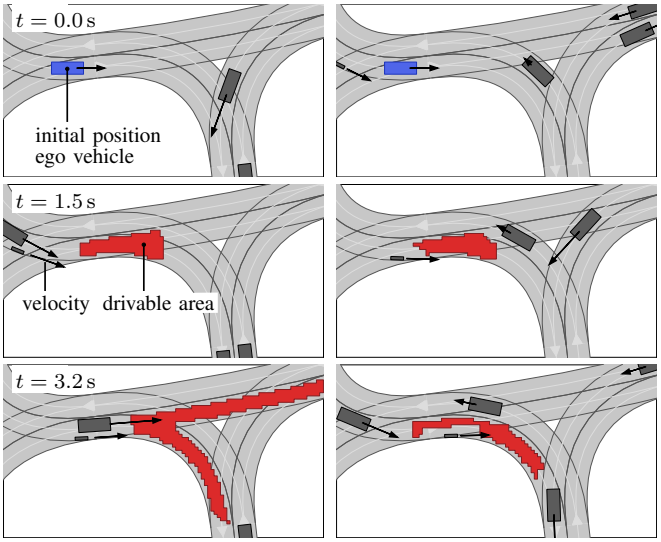
Let us present some concrete examples for demonstrating our algorithm. The first example is an intersection from the town Pula, Croatia. In Fig. 8, we compare the drivable area of the initial scenario obtained from SUMO with the optimized scenario. Note that we restrict the allowed road surface $\mathcal{W}_{\text{lanes}}$ to lanelets that the ego vehicle is allowed to drive in. In the initial scenario, the ego vehicle could either turn freely to the right or drive straight. However, after the optimization, two turning vehicles and a bicycle restrict possible maneuvers of the ego vehicle.

The second example is a four-way intersection from the town Putte, Belgium. In the optimized scenario, the ego vehicle must either respect an oncoming vehicle when turning left or a bicycle when driving straight. As a result, the drivable area is split into two parts, as shown in Fig. 9.

VI. CONCLUSIONS

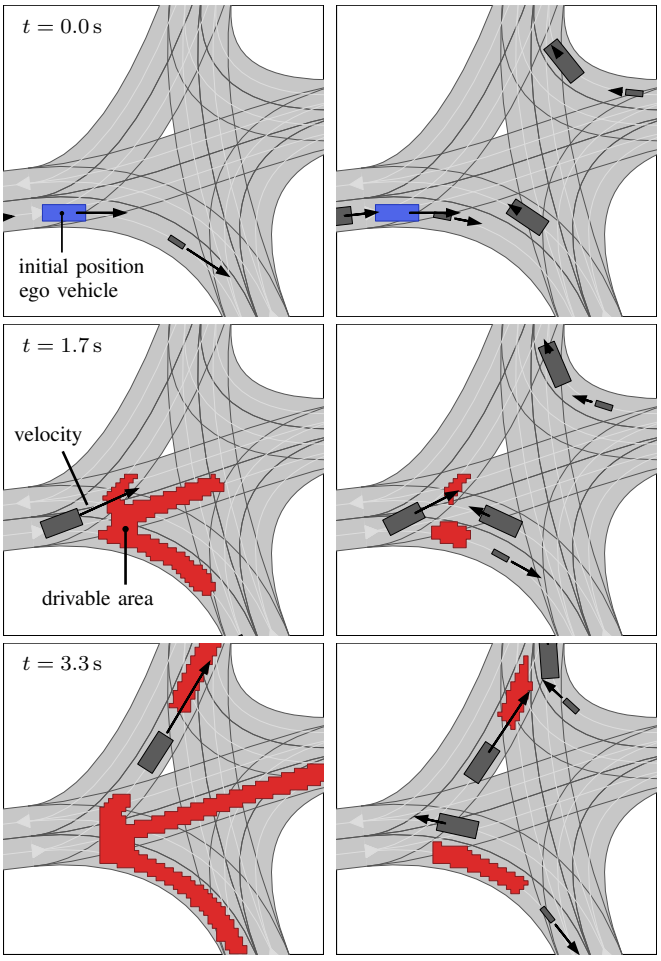
We present an approach to automatically generate a large number of test scenarios for automated vehicles. Our results show that we are able to extract a large number of distinct road networks from OpenStreetMaps, for which we simulate traffic scenarios using the traffic simulator SUMO. Our approach subsequently yields challenging scenarios by decreasing the solution space for motion planning algorithms.

The generated, publicly-available scenarios render the virtual testing of motion-planning algorithms in challenging situations easier. In the future, the explicit consideration of traffic rules during the generation of our critical scenarios will further improve our test cases.



(a) Initial scenario from simulation. (b) Optimized, more critical scenario.

Fig. 8. Example 1: Comparison of the drivable areas at different times.



(a) Initial scenario from simulation. (b) Optimized, more critical scenario.

Fig. 9. Example 2: Comparison of the drivable areas at different times.

ACKNOWLEDGMENTS

The authors would like to thank Chuxuan Li for her work on the SUMO interface and Maximilian Rieger for his work on the OSM2CR converter. We gratefully acknowledge the financial support from the Central Innovation Programme of the German Federal Government under grant no. ZF4086007BZ8 and the German Research Foundation (DFG) within the Priority Programme SPP 1835 *Cooperative Interacting Automobiles* under grant no. AL 1185/4-2.

REFERENCES

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. Part A: Policy Pract.*, vol. 94, pp. 182–193, 2016.

[2] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION dataset: An INTERnational, Adversarial and Cooperative moTION dataset in interactive driving scenarios with semantic maps," *arXiv:1910.03088*, 2019.

[3] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft Level 5 AV dataset 2019," https://level5.lyft.com/dataset/, 2019.

[4] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo Open dataset," *arXiv:1912.04838*, 2019.

[5] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2118–2125.

[6] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[7] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, 2008.

[8] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO–simulation of urban mobility: an overview," in *Proc. of Int. Conf. Adv. Syst. Simul.*, 2011, pp. 63–68.

[9] C. Campos, J. M. Leitão, J. P. Pereira, A. Ribas, and A. F. Coelho, "Procedural generation of topologic road networks for driving simulation," in *Iberian Conf. Inf. Syst. Technol.*, 2015, pp. 1–6.

[10] B. Kim, A. Jarandikar, J. Shum, S. Shiraishi, and M. Yamaura, "The SMT-based automatic road network generation in vehicle simulation environment," in *Proc. of the ACM Int. Conf. Embed. Softw.*, 2016, pp. 1–10.

[11] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. of the 28th ACM SIGSOFT Int. Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[12] G. Máttyus, W. Luo, and R. Urtasun, "DeepRoadMapper: Extracting road topology from aerial images," in *Proc. of the IEEE Int. Conf. Comput. Vision*, 2017, pp. 3438–3446.

[13] M. Maboudi, J. Amini, M. Hahn, and M. Saati, "Road network extraction from VHR satellite images using context aware object feature integration and tensor voting," *Remote Sens.*, vol. 8, no. 8, 2016.

[14] P. Li, Y. Zang, C. Wang, J. Li, M. Cheng, L. Luo, and Y. Yu, "Road network extraction via deep learning and line integral convolution," in *Proc. of the Int. Geosci. Remote Sens. Symp.*, 2016, pp. 1599–1602.

[15] Y. Zang, C. Wang, Y. Yu, L. Luo, K. Yang, and J. Li, "Joint enhancing filtering for road network extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 3, pp. 1511–1525, 2016.

[16] Y. Y. Chiang and C. A. Knoblock, "Automatic extraction of road intersection position, connectivity, and orientations from raster maps," in *Proc. of the ACM Int. Symp. Adv. Geogr. Inf. Syst.*, 2008, pp. 183–192.

[17] P. Fischer, S. M. Azimi, R. Roschlaub, and T. Krauß, "Towards HD maps from aerial imagery: Robust lane marking segmentation using country-scale imagery," *Int. J. Geo-Inf.*, vol. 7, no. 12, 2018.

[18] A. Zang, Z. Li, R. Xu, and D. Doria, "Lane boundary extraction from satellite imagery," in *Proc. of the ACM SIGSPATIAL Workshop High-Precis. Maps Intell. Appl. Auton. Veh.*, 2017, pp. 1–8.

[19] A. Artunedo, J. Godoy, and J. Villagra, "Smooth path planning for urban autonomous driving using OpenStreetMaps," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 837–842.

[20] D. Krajzewicz, G. Hertkorn, and J. Ringel, "Preparation of digital maps for traffic simulation; part 1: approach and algorithms," in *Proc. Ind. Simul. Conf.*, 2005, pp. 285–290.

[21] D. Nalic, A. Eichberger, G. Hanzl, M. Fellendorf, and B. Rogic, "Development of a co-simulation framework for systematic generation of scenarios for testing and validation of automated driving systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 1895–1901.

[22] P. Riegl, A. Gaull, and M. Beitelschmidt, "A tool chain for generating critical traffic situations for testing vehicle safety functions," in *IEEE Int. Conf. on Vehicular Electronics and Safety*, 2019, pp. 1–6.

[23] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 595–607, 2017.

[24] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part I: Methodology," *arXiv:1905.03419*, 2020.

[25] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *Proc. of the Int. Conf. Comput. Safety, Rel., Security*, 2019, pp. 69–84.

[26] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1129–1134.

[27] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2016, pp. 1470–1475.

[28] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 2879–2886.

[29] A. Nonnengart, M. Klusch, and M. Christian, "CriSGen : Constraint-based generation of critical scenarios for autonomous vehicles," in *Proc. of the Int. Workshop on Formal Methods for Autonomous Systems*, 2019.

[30] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2352–2358.

[31] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *Proc. of the IEEE Intell. Veh. Symp.*, 2018, pp. 1326–1333.

[32] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[33] M. Klischat, O. Dragoi, M. Eissa, and M. Althoff, "Coupling SUMO with a motion planning framework for automated vehicles," in *SUMO: Simulating Connected Urban Mobility*, 2019.

[34] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[35] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?" *J. Classif.*, vol. 31, no. 3, pp. 274–295, 2014.

[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. of the IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[37] M. Klischat and M. Althoff, "A multi-step approach to accelerate the computation of reachable sets for road vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020.

[38] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. of the Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.