Optimizing test-set diversity: Trajectory clustering for scenario-based testing of automated driving systems

Johannes Bernhard[1], Mark Schutera[2] and Eric Sax[3]

Abstract— **The developments in automated driving have raised questions concerning the safety of automated vehicles. The reliable behavior of automated driving functions has to be proven by a testing process. While real-world testing is the evident method for verification, there are significant concerns regarding test scalability. Simulative methods are economical and safe alternatives to assess driving functions' quality. A catalog with scenarios recorded in real-world traffic for re-simulation must be assembled, defining a set of requirements the driving function must pass before release. Therefore, to ensure safe behavior, the test catalog must include many diverse scenarios while reducing repetitive scenarios, as they do not pose additional challenges to the virtual vehicle and would unbalance the performance metrics. This work introduces a method to cluster and characterize vehicle behaviors based on trajectory data. The first step of the two-step bottom-up approach assigns individual coordinates of trajectories to clusters. Then, trajectory analysis is applied to histograms representing the distribution of retrieved cluster-IDs. Furthermore, we performed experiments, demonstrating the performance of our algorithm on various real-world data-sets. Using 1608 traffic scenarios, the method reduced the data-set by 61.0%, eliminating redundant scenarios while upholding scenario space coverage. This algorithm serves as a basis for traffic scenario understanding to optimize test-set diversity.**

I. INTRODUCTION

Through technical developments in deep learning methods achieved during the last decade, automated vehicles on public roads have become imminent. Machine learning algorithms are developed and applied to a series of automated driving tasks such as perception [1], [2], [3], trajectory estimation [4], [5], [6] and domain adaptation [7]. Despite remarkable achievements, machine learning algorithms in safety-critical applications remain a genuine concern, and system failures can cause fatal accidents [8]. Recent research revealed vulnerabilities and weaknesses of machine learning components, such as domain discrepancies during the development process [9], adversarial attacks [10], or a general lack of interpretability [11].

Safety is a crucial aspect during the release process of automated vehicles as public and political acceptance for those technologies directly depend on it [12]. Hence, the testing process has to prove the vehicle's reliable behavior. For example, the total distance that has to be tested to verify an interstate pilot's safety is estimated to be about 6.62 billion tested kilometers [12].

Real-world testing and field trials bear considerable downsides due to significant risks for participants, high costs for test execution, limited control over the test environment, and time constraints during product release processes [13]. Simulative testing can complement the verification process by addressing these issues. Software-in-the-loop and hardware-in-the-loop approaches test the driving function on system and component level versus a predefined model of its operational design domain to evaluate their performance under defined scenario properties. Simulations allow a safe way for repeatable and, further, scalable test execution [14]. Furthermore, simulative testing allows a precise formulation of the test scenario specifications, leading to total control over the traffic environment and traffic participants. Hence, the driving system can be tested on scenarios that pose a specific difficulty to the system and are challenging to test in real traffic.

Due to the multi-dimensionality of scenario components', such as the traffic participants' interaction, plain combinatorial sampling of test cases from scenario parameters or manual construction of test cases is infeasible. Therefore, data for vehicle behavior must be acquired by large-scale traffic recordings or traffic simulations [13]. However, traffic scenario acquisition and sampling can cause a shortage of computational resources, storage resources, and labeling resources, urging for a data selection process. Scenario clustering can avoid redundant data processing by focusing on cluster sub-sets. Hence, scenario clustering can be described as minimizing a data-set's size while upholding the data-set's variance. Traffic scenario selection focusing on vehicle behavior has seen increasing research, as scenario-based testing became more important over the past years [15]. For clustering of traffic participant interaction, feature representation learning using auto-encoder models has been introduced [16], [17]. Other approaches present meta-models for behavior description [18]. Further work on scenario clustering introduces the use of a random forest algorithm to structure scenario features such as vehicle speeds, relative angle, and road topology [19]. For analysis of scenarios with more than one vehicle, grid-based representations have been proposed with different layers representing a multitude of static and dynamic scenario information [20], such as vehicle movement and road geometry. Auto-encoder models trained on these representations can be used to detect out-

[1] J. Bernhard is with the Institute for Information Processing Technologies, Karlsruhe Institue of Technology, Germany and is also with Research and Development, ZF Friedrichshafen AG johannes.bernhard@zf.de

[2] M. Schutera is with the Institute for Automation and Applied Informatics, Karlsruhe Institue of Technology, Germany and is also with Research and Development, ZF Friedrichshafen AG mark.schutera@kit.edu

[3] E. Sax is with the Institute for Information Processing Technologies, Karlsruhe Institue of Technology, Germany eric.sax@kit.edu

of-distribution traffic scenarios and the evaluation of similarity between scenarios. Another branch of traffic behavior analysis focuses on the clustering of trajectories to obtain motion patterns. Therefore, shrinkage methods [21], longest-common-sub-sequence analysis [22] and spectral clustering [23] have been proposed.

This work introduces an unsupervised method for trajectory clustering to reduce repetitive scenarios while upholding scenario space coverage. First, using a Gaussian mixture model, we cluster individual vehicle states of a trajectory data-set. The results are transformed into a histogram representation, showing the distribution of vehicle state clusters for each trajectory. Afterward, a hierarchical clustering approach is applied to these representations for clustering on the trajectory level. Using the cluster information, traffic scenarios containing multiple vehicles are compared according to their trajectory data, enabling the detection of redundant scenarios. Additionally, we provide an evaluation and experiments on different data-sets [24], [25], [26], [27] to prove reliable state-of-the-art performance.

II. Methods

A. Scenario analysis

This work focuses on analyzing trajectory data acquired for a static scenario environment to analyze traffic participants' behavior. We can use this algorithm for test case selection. Data generation can either be performed by real-world data recordings by a vehicle's sensor system and stationary cameras or by using traffic simulations. Every scenario is observed from an ego vehicle's perspective, with other traffic participants being referred to as object vehicles. An interaction relates to the pair of the ego vehicle's trajectory $\mathcal{T}^e$ and an object vehicle's trajectory $\mathcal{T}^o$ at the same time. This trajectory pair is called *encounter* $\mathcal{E}$. Consequently, a scenario with three object vehicles generates three encounters. The proposed method for scenario reduction includes a clustering algorithm for encounters, assigning each encounter to a cluster, representing the interaction, such as following, crossing or leading the ego vehicle. We consider two scenarios to be similar if both scenarios' encounters are assigned to the same cluster, as shown in Fig. 1.

B. Trajectory clustering

For our proposed method, a trajectory is defined as a sequence of vehicle states

$$\mathcal{T} = \left\{ \begin{pmatrix} \mathbf{c_1} \\ \lambda_v * \mathbf{v_1} \end{pmatrix}, ..., \begin{pmatrix} \mathbf{c_i} \\ \lambda_v * \mathbf{v_i} \end{pmatrix}, ..., \begin{pmatrix} \mathbf{c_n} \\ \lambda_v * \mathbf{v_n} \end{pmatrix} \right\}, \quad (1)$$

with $\mathbf{c_i} \in \mathbb{R}^\mathbf{2}$ denoting the coordinate of the vehicle at time $i \in \{1, ..., n\}$ in a local coordinate system (measured in meter), $\mathbf{v_i} \in \mathbb{R}^\mathbf{2}, ||\mathbf{v_i}|| = \mathbf{1}$ denoting the driving direction of the vehicle at $i \in \{1, ..., n\}$ and $\lambda_v \in \mathbb{R}$ denoting a weight for the driving direction. Using vectors to describe the driving direction, rather than using angle values, is necessary since we assume the vehicle states to be in a metric space during clustering. Since the direction vector is normalized to length

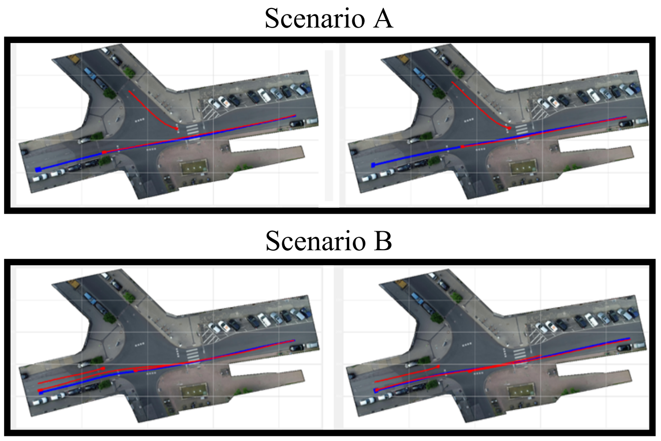

Fig. 1. Approach for scenario grouping with the ego vehicles in blue and the object vehicles in red. Each encounter between the object vehicle and an ego vehicle is assigned to a cluster. Two scenarios are considered similar if and only if they contain the same encounters.

$||v_i|| = 1$, the weight $\lambda_v = 2 * \sqrt{d_{max}}$ is used to scale the direction term, with d_{max} being the maximal euclidean distance between two recorded coordinates in the total data-set. An encounter $\mathcal{E}$ between an ego trajectory $\mathcal{T}^\mathrm{e}$ and an object vehicle $\mathcal{T}^\mathrm{o}$ is then defined by

$$\mathcal{E} = \{\mathcal{T}^\mathrm{e}, \mathcal{T}^\mathrm{o}\} = \left\{ \left\{ \begin{pmatrix} \mathbf{c_1^e} \\ \lambda_v \mathbf{v_1^e} \end{pmatrix}, ..., \begin{pmatrix} \mathbf{c_n^e} \\ \lambda_v \mathbf{v_n^e} \end{pmatrix} \right\}, \left\{ \begin{pmatrix} \mathbf{c_1^o} \\ \lambda_v \mathbf{v_1^o} \end{pmatrix}, ..., \begin{pmatrix} \mathbf{c_n^o} \\ \lambda_v \mathbf{v_n^o} \end{pmatrix} \right\} \right\}, \quad (2)$$

with $\mathcal{T}^e$ and $\mathcal{T}^o$ being the ego and object trajectories as defined in Eq. 1. The sampling rate for the trajectories is assumed to be equal.

Our proposed algorithm for unsupervised clustering encounters consists of a two-step bottom-up approach that is similar to the bag-of-words [28] method using two different clustering algorithms in sequence:

1) Fitting a Gaussian mixture model (GMM) to cluster vehicle states and assigning a cluster-ID to each vehicle state. Using the assignment IDs to create a histogram for each trajectory.
2) Applying hierarchical clustering (HC) on the histograms.

Encounters and trajectories vary in the number of recorded vehicle states due to variations in vehicle speed or the vehicle's entry and exit to the road geometry. Hence, we have to transform the trajectories into a standardized representation.

The proposed approach is inspired by the bag-of-words method: In text analysis, each word of a text is assigned to a code word. Afterward, the document is transformed into a histogram that displays each category's number of occurrences. Bag-of-words is a standard technique in natural language processing [28] and computer vision [29]. However, applying bag-of-words methods were also used for trajectory analysis [30], using a bag-of-segments approach based on sub-trajectories.

We assign the individual vehicle states of a trajectory

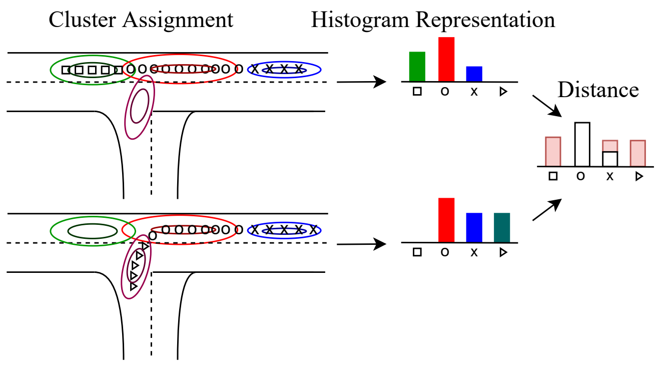

Fig. 2. Division of the environment in a 2-dimensional example. The fitted Gaussian mixture model assigns every vehicle state coordinate to a cluster. Hence, a trajectory can be transformed into a standardized histogram representation. The histogram contains information on how much time the vehicle spent in which region.

to clusters according to their location in the coordinate system and driving direction for trajectory clustering. We can interpret each cluster as a region, such as the colored ellipses in Fig. 2. The more vehicle states of a trajectory are mapped to a region, the more time the vehicle spent in the corresponding region, indicating low velocity. Our method interprets other trajectories that show a similar number of vehicle states assigned to this section as similar. A simple approach for the selection of regions is the partitioning of the coordination system into a grid. While this does not require any prior knowledge about the traffic environment, it does not differentiate between multiple lanes or sidewalks if the grid cells are too large. Alternatively, the region selection can be made by including expert knowledge by dividing the road geometry into manually chosen segments. In contrast to these predefined methods, we propose a Gaussian mixture model to detect regions with a high density of recorded vehicle states.

Fig. 2 shows an example of applying the bag-of-words model to trajectories by using a GMM to assign trajectory points to cluster-IDs. Transforming cluster-ID occurrences to histograms creates a standardized interface that allows us to assess similarities between trajectories or encounters. For clustering on the vehicle state level, we fit a GMM based on the four-dimensional space shown in Eq. 1, and Eq. 2. This data representation includes information on the position $\mathbf{c}$ and the direction $\mathbf{v}$ of the vehicle. Therefore we avoid the grouping of coordinates from vehicles that are driving in different directions. With this method, we map each trajectory to a histogram representation. We concatenate two histograms for the clustering of encounters, one based on the ego vehicle trajectory and one for the object vehicle trajectory.

We cluster the histogram representations with a hierarchical clustering algorithm in the second step of our bag-of-words approach. We consider encounters as similar according to the distance between their histogram representations. In contrast to clustering methods that need a predefined number of clusters, hierarchical methods can perform threshold-based clustering, setting the desired level of diversity inside the clusters. The threshold can be set by using expert knowledge or evaluation metrics for clustering algorithms. The encounters that fall in the same clusters are assumed to represent the same interaction between ego and object vehicle.

1) Gaussian mixture clustering:
To fit a clustering model on vehicle state level, an initial set of N trajectories $\mathcal{D} = \{\mathcal{T}^1, ..., \mathcal{T}^N\}$ shall be given. All vehicle states of all available trajectories in $\mathcal{T}$ are aggregated into a separate set

$$\mathcal{X} = \bigcup_{i=1}^{N} \mathcal{T}^i = \left\{ \begin{pmatrix} \mathbf{c_1} \\ \lambda_v \mathbf{v_1} \end{pmatrix}, ..., \begin{pmatrix} \mathbf{c}_\lambda \\ \lambda_v \mathbf{v}_\lambda \end{pmatrix} \right\},\ \lambda = \sum_{i}^{N} \# \mathcal{T}^i. \tag{3}$$

In general, mixture models are based on the assumption that a data-set was not sampled from a single distribution but from multiple sub-distributions. Hence, instead of fitting a single distribution function, the data-set should be modeled by fitting several distribution functions. A GMM assumes that the total data-set $\mathcal{X}$ was drawn from $k \in \mathbb{N}$ different multivariate Gaussian standard distributions $\mathcal{N}(\boldsymbol{\mu}_{i=1,...,k}, \boldsymbol{\sigma}^2_{i=1,...,k})$ with $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$ being the mean and variance of the sub-distribution $i \in \{1, ..., k\}$, and k being the pre-specified number of sub-distributions. The mixture density function is then given by

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i * f(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2) \tag{4}$$

with α_i being the proportion of data samples that are assigned to sub-distribution i. The probability $P(\boldsymbol{x}_j \in \mathcal{C}_i|\boldsymbol{x}_j)$ of sample $\boldsymbol{x}_j$ to belong to sub-distribution i is used to assign data points to the most probable sub-distributions. Optimization of $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$ is done by the expectation maximization algorithm with the number of sub-distributions k being pre defined.

After fitting the parameters, the mixture model assigns for each trajectory $\mathcal{T}$ every sample $\boldsymbol{x} \in \mathcal{T}$ to the sub-distribution with the highest probability. The assignments are transformed into a histogram $\boldsymbol{h} \in \mathbb{N}^k$, displaying the number of occurrences of points from each sub-distribution in the trajectory. The histogram construction is displayed in Fig. 2. The whole process converts an encounter $\mathcal{E} = \{\mathcal{T}^e, \mathcal{T}^o\}$ into two histograms $[\boldsymbol{h}^e, \boldsymbol{h}^o] \in \mathbb{N}^{2*k}$ that are concatenated together.

2) Hierarchical clustering:
To cluster the histogram representations created by the GMM, we used an agglomerative hierarchical clustering approach, as it does not require a predefined number of clusters. Instead, the number of final clusters depends on a set threshold that specifies how dissimilar encounters inside a cluster are allowed. Agglomerative hierarchical clustering incorporates a bottom-up approach that starts by assigning each data point to its own cluster. Afterward, clusters are aggregated step-by-step, depending on their similarity.

Fig. 3 shows how this method creates a cluster structure. The dendrogram shows the bottom-up process for merging

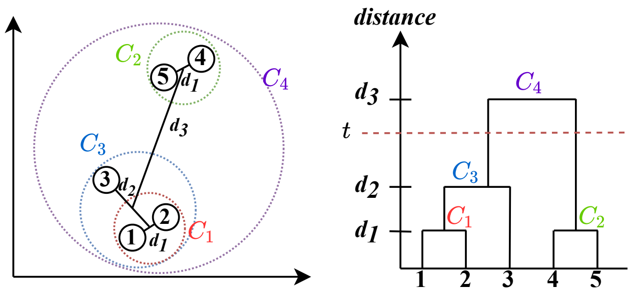

Fig. 3. Hierarchical clustering process. On the right side, a dendrogram is shown. The graph is starting with each data point in its own cluster. Then clusters are merged depending according to their inter-cluster distance, as defined in Eq. 6. The dendrogram can be cut off to get the cluster structure by setting a threshold t for the maximum distance between clusters. We used the Davies-Bouldin Index to optimize the threshold.

the clusters as the distance is increasing. By setting a threshold t, we can cut of the structure and hold the number of cluster variable dependent on the data.

Hierarchical clustering depends on two functions:

1) The distance function $d(\cdot,\cdot)$ to measure dissimilarity between histogram representations (intra-cluster difference), as defined in Eq. 5.
2) The linkage-function $D(\cdot,\cdot)$ to measure the dissimilarity between clusters (inter-cluster difference), as defined in Eq. 6.

The distance function between two normalized histogram representations $\boldsymbol{h}_1, \boldsymbol{h}_2 \in [0,1]^n$ is calculated using the Chi-squared distance [31]

$$d(\boldsymbol{h}_1,\boldsymbol{h}_2) = \frac{1}{2}\sum_{i=1}^{n} \begin{cases} \frac{(h_{1,i}-h_{2,i})^2}{h_{1,i}+h_{2,i}}, & \text{if } h_{1,i}+h_{2,i} > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

The linkage-function between two clusters $C^1 = \{\boldsymbol{h}^1_1, ..., \boldsymbol{h}^1_i\}$ and $C^2 = \{\boldsymbol{h}^2_1, ..., \boldsymbol{h}^2_j\}$ is defined as the distance between the two cluster centroids:

$$D(C^1, C^2) = ||(\frac{1}{\#C^1}\sum_{\boldsymbol{h}^1 \in C^1} \boldsymbol{h}^1) - (\frac{1}{\#C^2}\sum_{\boldsymbol{h}^2 \in C^2} \boldsymbol{h}^2)||_2, \quad (6)$$

with $||\cdot||_2$ being the Euclidean norm.

3) Hyper-parameter optimization:
The proposed trajectory clustering method requires two hyperparameters to be set:

- Number of Gaussian distributions k: The amount of Gaussian distributions controls the granularity of regions that cover the road geometry.
- Threshold for hierarchical clustering t: The maximum value for dissimilarity in a given cluster. The number of clusters can be held variable by setting a threshold. Furthermore, this allows for extending the algorithm to an iterative process.

Choosing k too low may cause too much generalization of the road geometry. On the other hand, a high k may cause too fine-grained clusters. The same principles apply to the threshold t for the clustering of the trajectories and encounters.

We used the Davies–Bouldin Index [32] (DBI) to implement this work. The DBI sets the intra-cluster variation relative to the inter-cluster variation. The calculation of the DBI for a data-set that is clustered in k different subsets requires calculating a pair-wise measure $r_{i,j}$ for the clustering fit between the clusters i and j. This value is defined by

$$r_{i,j} = \frac{\sigma_i + \sigma_j}{d(\mathbf{c_i}, \mathbf{c_j})}, \quad (7)$$

with σ_i being the standard deviation of cluster $\mathcal{C}_i$, $\mathbf{c_i}$ being the centroid of cluster $\mathcal{C}_i$ and $d(\cdot,\cdot)$ being a distance function. Hence, a high $r_{i,j}$ indicates a high similarity between cluster $\mathcal{C}_i$ and $\mathcal{C}_j$. Then the DBI can be calculated by averaging for each cluster the highest similarity value to a different cluster. Hence, the smaller the DBI, the better the clustering.

4) Clustering evaluation metric:
We apply the Hungarian method [33] to determine the optimal assignment between ground truth groups and predicted clusters. The method optimizes the label assignment to minimize the number of faulty predicted trajectories. Using the optimal assignment, the correct clustering rate (CCR) [34] is calculated by

$$CCR = \frac{1}{N}\sum_{c=1}^{K} p_c, \quad (8)$$

with N denoting the number of trajectories, p_c denoting the number of trajectories correctly matched to cluster c, and K denoting the number of total clusters.

III. Experiments

We validate the presented clustering algorithm by holding it against state of the art traffic trajectory data-sets. Hence, we can compare the method with alternative clustering algorithms. The data-sets contain trajectories as defined in Eq. 2, they include labels that assess the quality of clustering algorithms on traffic trajectories quantitatively.

In the second part of the experiments, we apply our model to the InD (**In**tersection **D**rone data-set) traffic trajectories [27] recorded by a drone. Although the recordings do not provide labels, they allow extracting driving scenarios, including vehicle interactions and encounter trajectories.

A. Implementation

The implementation started by assembling the data-set $\mathcal{X}$ (see Eq. 3). The trajectories were interpolated to simulate additional ten coordinates between each vehicle state due to the original trajectories' low recording rate. We smoothed all trajectories using a Gaussian window to smooth trajectory edges. Random noise, sampled from $\mathcal{N}(0,1)$, is added to the data-points during the sampling process to prevent the Gaussian distributions from overfitting along single trajectories.

Model fitting was done utilizing the Scikit-learn toolbox [35] for Python programming on an NVIDIA Tesla P100 GPU (16GB GPU memory). An implementation of the model can be found on: `https://github.com/VanLock9988/Trajectory_Clustering.git`.

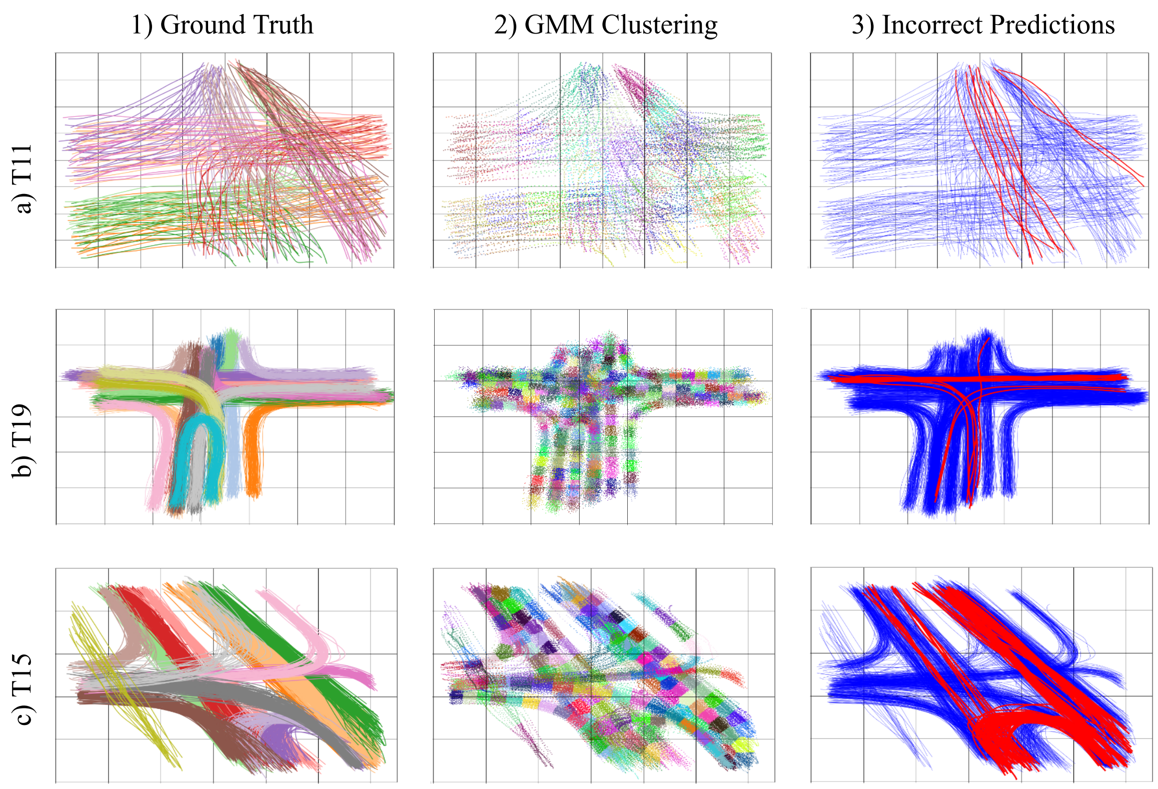

Fig. 4. Results for each of the three evaluation data-sets detailed in III.A. Column (1) shows the trajectories with the color representing the ground truth. Column (2) shows the output of the Gaussian mixture models for each data-set. Column (3) shows the faulty predicted trajectories in red and correctly predicted trajectories in blue.

B. *Trajectory clustering*

1) Data:
There are several data-sets available for the evaluation of the proposed trajectory clustering algorithm.

- T11 [24]: 220 recorded real-world trajectories from eleven different groups. Each group contains 20 trajectories, which makes the data-set balanced, though also sparse (Fig. 4 a)).
- T19 [25]: 1900 simulated vehicle trajectories from 19 different groups. The 19 groups describe 19 possible driving maneuvers, including a turnaround. The set is balanced, as there are 100 trajectories for each label (Fig. 4 b)).
- T15 [26]: 1500 recorded real-world trajectories from different 15 groups. The data-set is significantly unbalanced, as each group contains between 18 and 278 trajectories. The groups describe various driving maneuvers, including two different turnarounds (Fig. 4 c)).

2) Baseline:
We use the work done by Xu et al. [21] to compare our model against alternative approaches for traffic trajectory clustering on the T11, T15, and T19 data-set. Alongside their own model (AKMS + K-means), they implemented multiple alternative clustering:

- AKMS + K-means [21]: By using a so-called *adaptive multi-kernel-based shrinkage* method, trajectories are shrunk towards their local motion patterns, which aims to eliminate sampling noise. Afterward these shrunk trajectories are clustered using K-means clustering.
- Heat-map [36]: Trajectories are transformed into a grid-based heat-map that displays the trajectories course. After standardizing, clustering is performed on the heat-map format.
- DTW + K-means [21]: Applying K-means clustering on raw trajectories with *dynamic time warping* as the distance measure.

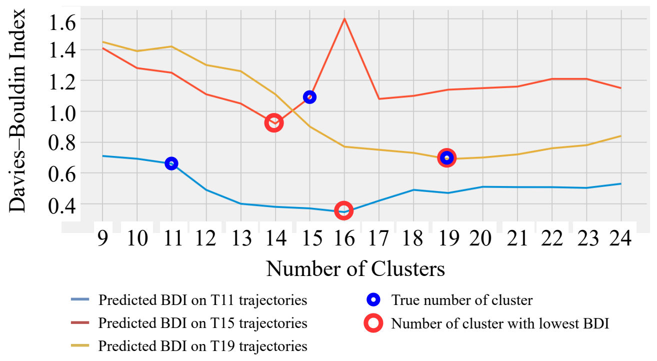

Fig. 5. Davies–Bouldin Index for parameter optimization of clustering algorithm for the trajectory data-sets T11, T15, and T19. The red circles denote the expected number of clusters, while the blue circles denote the real number of clusters.

TABLE I

CORRECT CLUSTERING RATE OF OUR PROPOSED MODEL AGAINST THREE ALTERNATIVE MODELS ON THE DATA-SETS T11 [24], T19 [25] AND T15 [26] IN %.

	T11	**T19**	**T15**
DTW + K-means	92.1	90.0	83.2
Heat-map [36]	95.0	91.8	82.0
AKMS + K-means [21]	**99.5**	99.1	87.4
GMM + HC (ours)	92.7	**99.6**	**90.7**

3) Results:

The clustering performance of our model is displayed in Table I. The CCR values are calculated as stated in Eq. 8. The number of trajectory clusters is determined using the Davies-Bouldin Index (see Fig. 5). Additionally, Fig. 4 shows the clustering predictions on the vehicle state level and the errors on trajectory level.

- T11: Our proposed model achieves a CCR score of 92.7%, trailing the AKMS + K-means model, which achieves a CCR score of 99.5%. In contrast to the other data-sets, the trajectories of T11 are uniformly located across each group. There is not a high density of trajectories in the middle of each group's track. Hence, the Gaussian distributions we fitted are unstable at the edges of the tracks.
- T19: Our proposed model outperforms the baseline methods with a CCR score of 99.5%, compared to the AKMS + K-means model with a CCR score of 99.1%. The faulty predicted trajectories primarily lie between clusters where the clusters overlap, and the GMM assigns vehicle states to the wrong Gaussian distributions. These results show the beneficial effect of our method if there is a high density of trajectories in the middle of the road.
- T15: Our proposed model achieves a CCR score of 90.7%, outperforming the AKMS + K-means model, which achieved a CCR score of 87.4%. Similar to the T19 data-set, the model performs strongly if there is a high density of trajectories in the center of each road. However, our trajectory clustering algorithm made the most faulty predictions on the turnaround groups as it tends to struggle with trajectory groups that share substantial parts of the course but separately afterward.

C. Scenario Analysis

1) Data:

For encounter clustering, no labeled scenario data-set is available. However, there are unlabelled traffic recordings that we used for encounter clustering experiments. For example, the InD data-set [27] contains trajectories that drones have recorded on different traffic intersections in Aachen, Germany. For an experimental application of the scenario reduction model, the *Frankenburg* intersection out of the InD data-set is used. The data-set provides data for seven different recording sessions of about 45 minutes each. A top-down view of the intersection is shown in Fig. 1.

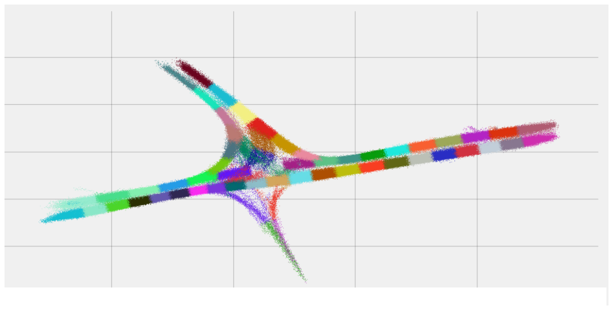

Fig. 6. Output of Gaussian mixture model on the InD data-set with 37 Gaussian distributions.

In contrast to the data-sets used in Section III-B, the InD data-set provides the trajectories temporal context. Due to the drone's bird-eye view, the data-set also provides accurate localization of the objects and their trajectory. That allows the analysis of the traffic participants' interaction if they cross the intersection simultaneously. As detailed in Section II-A, a scenario consists of an ego trajectory and multiple object trajectories. However, the data-set does not contain ego trajectories, as traffic recordings from vehicles would. Hence, trajectories have to be designated as ego trajectories manually to construct scenarios around them. We assigned every vehicle that fully crosses the intersection as the ego vehicle for its own scenario, ignoring parking vehicles. Our work focuses on analyzing vehicle encounters, ignoring encounters with and between vulnerable road users, such as pedestrians and cyclists. Using this approach, we extracted 1608 scenarios. Based on this set, each ego trajectory is paired to each object trajectory detected on the intersection simultaneously to create a set of encounters.

2) Results:

We applied the proposed cluster model to the total encounter data-set to find redundancies in the data-set. As displayed in Fig. 1, two scenarios are considered equal if the close interactions can be matched and the ego takes the same path across the road geometry.

Fig. 6 shows how the GMM model manages to partition the trajectory vehicle states in meaningful sections based on vehicle position and driving direction. In addition, the model manages to cleanly separate the different lanes on the straight and the crossing. Like the T19 data-set, the GMM model profits from a high density of trajectories in the middle of each track that resembles a Gaussian normal distribution density function. Furthermore, there is an absence of multi-track lanes or lanes with a sparse density of data points.

The models' ability to extract unique interactions and scenarios iteratively is displayed in Fig. 7. The number of unique encounters and scenarios increases, although the slope decreases, indicating fewer *new* scenarios found per driven distance:

- Fig. 7 a) shows the number of encounter clusters extracted from the total data if fed iteratively to the model. 2533 total encounters are extracted from the

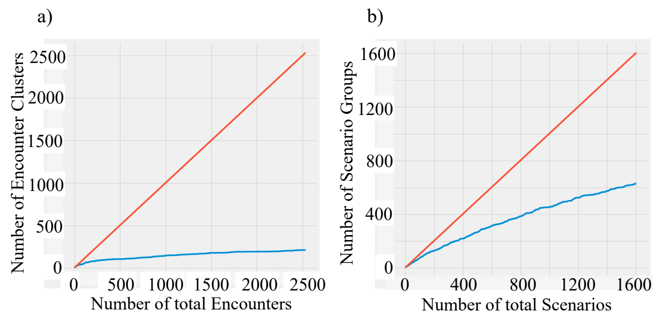

Fig. 7. Development of detected clusters for encounters and scenarios if the data is iteratively added to the model. The red line describes the identity function, while the blue line describes the unique clusters detected.

drone data and grouped in 208 clusters that represent unique interactions.

- Fig. 7 b) shows the number of extracted unique scenarios if the data is fed iteratively to the model. From the 1608 total scenarios, 627 unique scenarios are extracted, achieving a reduction of 61.0% of the total scenario set. With an increasing number of acquired scenarios, the ratio between total scenarios and scenario groups decreases, indicating an increasing amount of redundant scenarios. Hence, the probability of collecting a new unique scenario decreases with the number of already collected scenarios.

IV. Discussion

The experiments that we performed allowed us to assess the properties of our proposed model. We held our model against alternative approaches. We managed to outperform these approaches on multiple data-sets. We achieved satisfying results, especially on trajectory groups with a sufficient data-set and a dense distribution of trajectories in the middle of the road. This property is convenient for vehicle traffic, where traffic regulations restrict movements. The model partitions multiple lanes in the same driving direction, or different driving directions on the same lane. If the data-set is insufficient with a sparse trajectory distribution, the assignment of vehicle states to their respective distributions becomes inaccurate. Hence, while the model can achieve precise results, they rely on comprehensive data acquisition. Furthermore, the algorithm may group trajectories or encounters if they share a substantial part of their course, even if they separate in different directions.

Our experiments demonstrated how redundancies in traffic scenario data-sets can be identified. For the simulation process during an automated driving systems validation, the test effort is reduced while upholding the diversity inside the data-set. The development of newly found scenarios provides insight into the coverage of scenarios found this far and the estimated data that has to be gathered to detect the next distinct scenario.

V. Conclusion and Outlook

With this work, we introduce an effective method for unsupervised trajectory clustering. We combined Gaussian mixture clustering and hierarchical clustering to achieve accurate results on various benchmark trajectory data-sets. These data-sets include single-vehicle motion behavior as well as pair-wise traffic participant interaction. Furthermore, we demonstrated how the method can be used for traffic scenario reduction in the context of scenario-based testing of automated driving functions. Our approach structures a scenario data-set to maintain scenario space coverage while preventing redundancies.

There is a range of extensions and future applications for our model. For example, apart from unsupervised clustering, we can use the model for abnormal behavior detection by searching for isolated feature representations or trajectories that move through uncharted areas that the GMM does not cover. Furthermore, we want to include a retrieval-based approach to find critical scenarios that pose a particular hazard towards a driving function in future work.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

[2] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE International Conference on Intelligent Transportation Systems*, pp. 1–8, 2017.

[3] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *arXiv preprint arXiv:1605.06409*, 2016.

[4] H. Xue, D. Q. Huynh, and M. Reynolds, "Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision*, pp. 1186–1194, 2018.

[5] M. Schutera, S. Elser, J. Abhau, R. Mikut, and M. Reischl, "Strategies for supplementing recurrent neural network training for spatio-temporal prediction," *at-Automatisierungstechnik*, vol. 67, no. 7, pp. 545–556, 2019.

[6] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE International Conference on Intelligent Transportation Systems*, pp. 399–404, IEEE, 2017.

[7] M. Schutera, M. Hussein, J. Abhau, R. Mikut, and M. Reischl, "Night-to-day: Online image-to-image translation for object detection within autonomous driving by night," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[8] V. A. Banks, K. L. Plant, and N. A. Stanton, "Driver error or designer error: Using the perceptual cycle model to explore the circumstances surrounding the fatal tesla crash on 7th may 2016," *Safety science*, vol. 108, pp. 278–285, 2018.

[9] M. Schutera, F. M. Hafner, H. Vogt, J. Abhau, and M. Reischl, "Domain is of the essence: Data deployment for city-scale multi-camera vehicle re-identification," in *2019 IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 1–6, 2019.

[10] A. Ranjan, J. Janai, A. Geiger, and M. J. Black, "Attacking optical flow," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, October 2019.

[11] O. Willers, S. Sudholt, S. Raafatnia, and S. Abrecht, "Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks," in *International Conference on Computer Safety, Reliability, and Security*, pp. 336–350, Springer, 2020.

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

[13] J. Bach, J. Langner, S. Otten, E. Sax, and M. Holzäpfel, "Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems," in *2017 International Conference on Engineering, Technology and Innovation*, pp. 203–210, IEEE, 2017.

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[15] D. Nalic, T. Mihalj, M. Bäumler, M. Lehmann, A. Eichberger, and S. Bernsteiner, "Scenario based testing of automated driving systems: A literature survey," in *FISITA Web Congress*, 2020.

[16] S. Li, W. Wang, Z. Mo, and D. Zhao, "Cluster naturalistic driving encounters using deep unsupervised learning," in *2018 IEEE Intelligent Vehicles Symposium*, pp. 1354–1359, 2018.

[17] W. Wang, A. Ramesh, and D. Zhao, "Clustering of driving scenarios using connected vehicle datasets," *arXiv preprint arXiv:1807.08415*, 2018.

[18] R. Pfeffer, J. He, and E. Sax, "Development and implementation of a concept for the meta description of highway driving scenarios with focus on interactions of road users.," in *VEHITS*, pp. 440–447, 2020.

[19] F. Kruber, J. Wurst, and M. Botsch, "An unsupervised random forest clustering technique for automatic traffic scenario categorization," in *2018 International Conference on Intelligent Transportation Systems*, pp. 2811–2818, IEEE, 2018.

[20] L. Ries, J. Langner, S. Otten, J. Bach, and E. Sax, "A driving scenario representation for scalable real-data analytics with neural networks," in *2019 IEEE Intelligent Vehicles Symposium*, pp. 2215–2222, 2019.

[21] H. Xu, Y. Zhou, W. Lin, and H. Zha, "Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage," in *2015 IEEE International Conference on Computer Vision*, pp. 4328–4336, 2015.

[22] M. Y. Choong, L. Angeline, R. K. Y. Chin, K. B. Yeo, and K. T. K. Teo, "Modeling of vehicle trajectory clustering based on lcss for traffic pattern extraction," in *2017 IEEE International Conference on Automatic Control and Intelligent Systems*, pp. 74–79, IEEE, 2017.

[23] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *2005 IEEE International Conference on Image Processing*, vol. 2, pp. II–602, Ieee, 2005.

[24] W. Wang, W. Lin, Y. Chen, J. Wu, J. Wang, and B. Sheng, "Finding coherent motions and semantic regions in crowd scenes: A diffusion and clustering approach," in *European Conference on Computer Vision*, pp. 756–771, Springer, 2014.

[25] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 312–319, 2009.

[26] W. Hu, X. Li, G. Tian, S. Maybank, and Z. Zhang, "An incremental dpmm-based method for trajectory clustering, modeling, and retrieval," *2013 IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 5, pp. 1051–1065, 2013.

[27] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," *arXiv preprint arXiv:1911.07602*, 2019.

[28] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[29] G. Qiu, "Indexing chromatic and achromatic patterns for content-based colour image retrieval," *Pattern Recognition*, vol. 35, no. 8, pp. 1675–1686, 2002.

[30] Y. Zhou and T. S. Huang, "Bag of segments for motion trajectory analysis," in *2008 IEEE International Conference on Image Processing*, pp. 757–760, 2008.

[31] V. Asha, N. U. Bhajantri, and P. Nagabhushan, "Glcm–based chi–square histogram distance for automatic detection of defects on patterned textures," *International Journal of Computational Vision and Robotics*, vol. 2, no. 4, pp. 302–313, 2011.

[32] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[33] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[34] Z. Zhang, K. Huang, and T. Tan, "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *2006 International Conference on Pattern Recognition*, vol. 3, pp. 1135–1138, IEEE, 2006.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] W. Lin, H. Chu, J. Wu, B. Sheng, and Z. Chen, "A heat-map-based algorithm for recognizing group activities in videos," *2013 IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 11, pp. 1980–1992, 2013.