applied mechanics

MDPI

Article

Scenario Identification and Classification to Support the Assessment of Advanced Driver Assistance Systems

Zafer Kayatas [1,*] and Dieter Bestle [2]

[1] Mercedes-Benz AG, Kolumbusstr. 19+21, 71063 Sindelfingen, Germany
[2] Chair of Engineering Mechanics and Vehicle Dynamics, Brandenburg University of Technology Cottbus-Senftenberg, Siemens-Halske-Ring 14, 03046 Cottbus, Germany; bestle@b-tu.de
* Correspondence: zafer.kayatas@mercedes-benz.com

Abstract: In recent years, driver assistance systems in cars, buses, and trucks have become more common and powerful. In particular, the introduction of AI methods to sensors, signal fusion, and traffic recognition allows us to step forward from actual level-2 assistance to level-3 Advanced Driver Assistance Systems (ADAS), where driving becomes autonomous and responsibility shifts from the driver to the automobile manufacturers. This, however, requires a high-precision risk assessment of failure, which can only be achieved by extensive data acquisition and statistical analysis of real traffic scenarios (which is impossible to perform by humans). Therefore, critical driving situations have to be identified and classified automatically. This paper develops and compares two different strategies—a traditional rule-based approach derived from deterministic causal considerations, and an AI-based approach trained with idealized cut-in, cut-out, and cut-through maneuvers. Application to a 10-h measurement sequence on a German highway demonstrates that the latter has the higher performance, whereas the former misses some of the safety-relevant events to be identified.

Keywords: advanced driver assistance system; real traffic situation; decision tree; machine learning; scenario identification; scenario classification

check for updates

Citation: Kayatas, Z.; Bestle, D. Scenario Identification and Classification to Support the Assessment of Advanced Driver Assistance Systems. *Appl. Mech.* **2024**, *5*, 563–578. https://doi.org/10.3390/applmech5030032

Received: 7 May 2024
Revised: 19 August 2024
Accepted: 23 August 2024
Published: 27 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

About 95% of traffic accidents are due to erroneous decision-making of the drivers [1]. Driver assistance systems may cut down the number of accidents if their reliability is higher than that of human drivers. However, as such assistance systems become more powerful, they additionally have to compensate for the effect that a confident driver will become less watchful. Reliability becomes even more of a critical issue when gradually switching from level-2 to level-3, or higher assistance [2], where automobile OEMs (original equipment manufacturers) must assume full responsibility for any failures of their control systems. Mercedes-Benz was the first to release an officially certified level-3 assistant for autonomous driving on German highways, called Drive Pilot [3,4]. Other major players, such as BMW and Volvo, are poised to introduce similar products with enhanced performance to the market.

High reliability and low failure rates will be essential for the acceptance of self-driving vehicles by both consumers and politicians. This, however, somehow contradicts the software architecture behind such products, as the realization of self-driving functionalities is closely correlated with the development of AI methods. In particular, the appearance of deep learning was a major game changer, also enforcing a major paradigm change compared to traditional product development: models are extracted from data correlations instead of causal deduction starting from proven principles. The more powerful such models become, the higher the interpretability of predictions lost [5]. To counteract this property, car manufacturers have to work on two fronts: (i) making AI algorithms safe enough for autonomous driving, and (ii) carefully assessing the failure rate of such systems.

The safety challenge cannot be tackled with traditional field testing since the mileage required for assessing the desired low failure rate would be much too high [6]. Therefore, software-in-the-loop simulation (SiL) is utilized, embedding the real ADAS and sensor models in virtual vehicle models [7–9]. To save costs, only critical traffic scenarios are simulated. The simulation will then provide conditional probabilities of failure of the ADAS w.r.t. to various scenarios. For a final assessment of the overall failure probability, the rate of occurrence of these scenarios has to be known.

In close cooperation of automobile manufacturers with research institutions, the German research project PEGASUS [10] defined various logical scenarios with associated parameters. Here, the three scenarios—cut-in, cut-out, and cut-through—will be considered in more detail; see Figure 1. Mercedes-Benz AG has been collecting a rich dataset over the years via a large measurement fleet of vehicles, called ego, equipped with radar, lidar, and several cameras monitoring the traffic in the surrounding area. Only a minor subset of this data was analyzed by human engineers to extract characteristic time snippets and label them by the corresponding scenario class. To automatically find such events in a large dataset of measurements, an algorithmic search has to be performed. One possibility is to use pattern recognition of activities of the ego vehicle and the surrounding traffic [6,11–14]. The question, however, arises if the above-mentioned scenarios may simply be identified from measurements of the longitudinal distance $s(t)$ between the ego and the first ego vehicle running ahead of the ego or from lateral distances $d(t)$ of vehicles running on neighboring lanes; se Figure 2. The measurements are typically made with a measurement frequency of $f_M = 50$ Hz resulting in time steps of size $\Delta t = 0.02$ s. These raw signals are then preprocessed by filling up dropouts and smoothing signals by filtering. The lateral data are additionally corrected by eliminating road curvature effects [15].

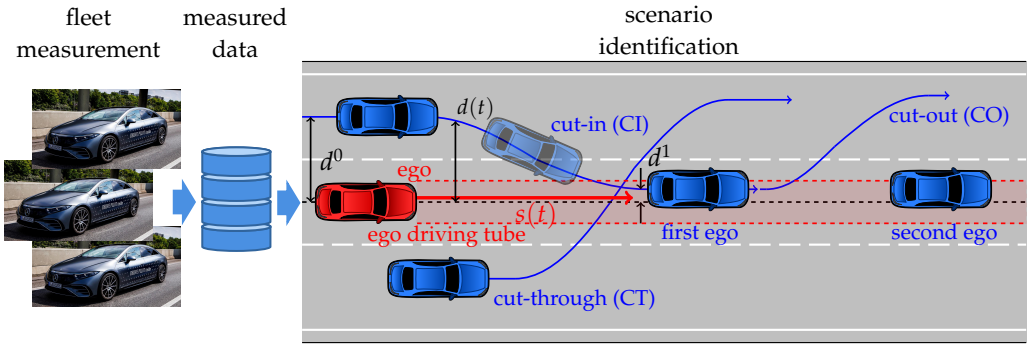

Figure 1. Scenario identification from fleet measurements.

Section 2 will use the $s(t)$ measurement to detect cut-in, cut-out, and cut-through scenarios from sudden jumps when a car is entering or leaving the ego driving tube, see the upper plot in Figure 2. Rules will be derived from kinematic considerations and their impact on the longitudinal distance signal. Section 3 will build a detection and classification algorithm upon measurements of lateral distance $d(t)$ and its characteristic time-behavior associated with the above maneuvers; see the lower plot in Figure 2. Preliminary studies [16] on several machine learning algorithms have shown that a strategy based on the Time Series Forest (TSF) approach [17] is suited best. As a machine learning approach, it requires training data, which is why Section 4 develops a simple procedure to generate idealized training maneuvers from pieces of $d(t)$ time series. Both classification algorithms are tested and compared in Section 5 using the measurements of a 10-h ride on a German highway.

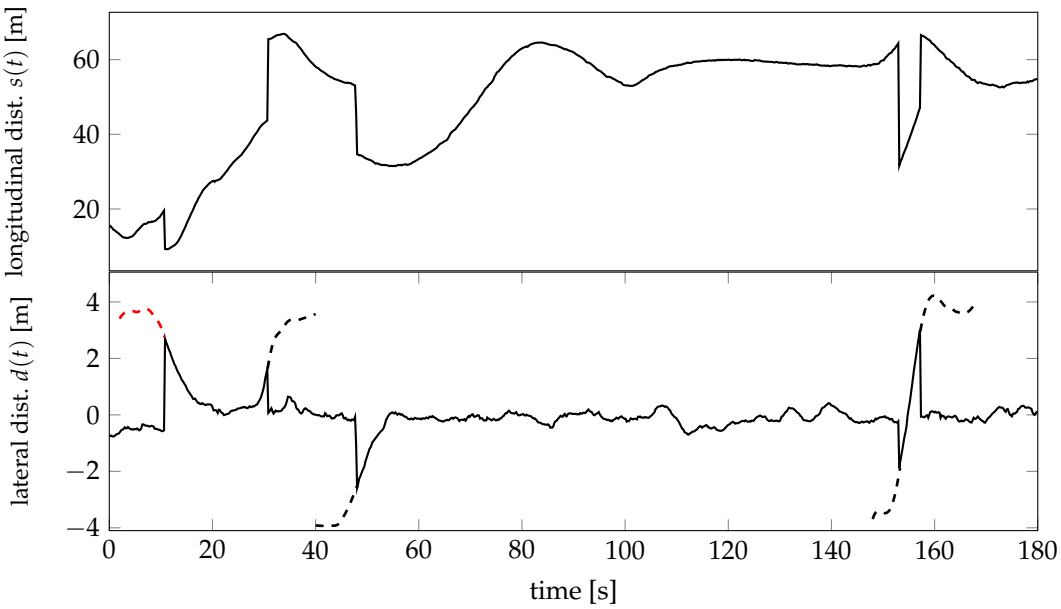

Figure 2. Measured data for longitudinal distance $s(t)$ and lateral distance $d(t)$.

2. Rule-Based Classification Tree

The basis for deterministic scenario detection involves distance measurements $s(t)$ in the longitudinal direction; see the upper plot in Figure 2. Whenever there is a sharp transition from high to low values, this indicates a cut-in (CI) maneuver. This occurs because the distance suddenly shifts from the first ego vehicle to a shorter distance to the vehicle that is cutting in. This vehicle enters the driving lane of the ego vehicle, becoming the new vehicle in front and reducing the gap; see Figure 1. In Figure 2, this occurs for $t \approx 10$ s. Subsequently, the ADAS reacts and enlarges the distance again to a proper, velocity-dependent distance. However, for $t \approx 30$ s, the distance value suddenly jumps from a low to a high value, indicating a cut-out (CO) maneuver, where the distance increases to the vehicle in front of the former first ego, called the second ego, after the latter has left the ego driving tube. Cut-through (CT) maneuvers combine both characteristics, the abrupt drop, and increase in $s(t)$ within a short period of time, where the distance first reduces to the crossing vehicle and then back to the former first ego. This can be observed for $t \approx 151$–158 s.

These relations may be embedded in a rule-based decision tree, as shown in Figure 3. The first three nodes exclude situations irrelevant to safety assessment. They check if the ego vehicle is driving on a highway with sufficient velocity since only then ADAS may take over control. Further ego lane changes are actually prohibited while driving in the level-3 mode. The more interesting fourth node detects jumps in the longitudinal distance $s(t)$ to the vehicle driving ahead, where the difference of subsequent measurements has to exceed a user-defined threshold ε_s, i.e., $\Delta s_i := |s(t_{i+1}) - s(t_i)| > \varepsilon_s$. This could indicate one of the relevant scenarios (cut-in, cut-out, or cut-through), which may be directly classified depending on the sign of the jump if there exists only a single neighbor object in the subsequent measurement snippet. A cut-in maneuver suddenly shortens the distance yielding a negative jump; cut-out enlarges $s(t)$ with a positive jump, and cut-through combines both jumps if they occur within a predefined time interval $\Delta T < \varepsilon_t$. In the following, $\varepsilon_s = 5$ m and $\varepsilon_t = 10$ s are chosen.

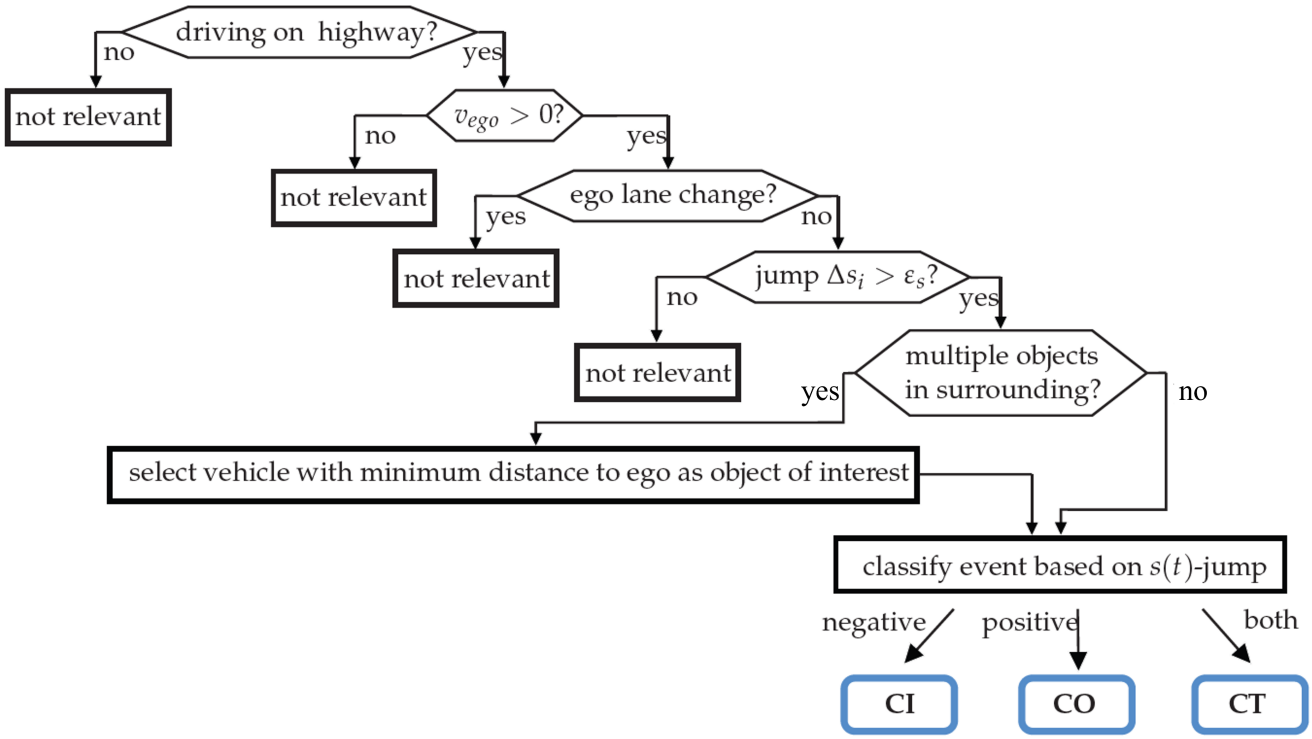

Figure 3. Rule-based decision tree for scenario classification.

If multiple objects are located in the surrounding neighborhood of the ego vehicle and perform maneuvers, the distances between the ego and all objects of interest are measured at time points, t^*, where a vehicle enters the ego-driving tube. The vehicle with the minimum distance is then the most critical and relevant one for scenario classification, whereas the other vehicles are ignored.

When applying the rule-based decision tree to the measurement section in Figure 2, it detects four relevant scenarios; see Figure 4. Additionally, it extracts corresponding 20 s snippets of the lateral distance $d(t)$ around the respective time point t^* when the $s(t)$-jump arises. They confirm the correct classification: in the first CI-event the cut-in vehicle enters from the left adjacent lane (distance about 3.8 m corresponding to the lane width), in the second CI-event from the right adjacent lane (distance about -4 m). During cut-out, the vehicle leaves the ego lane to the left. Finally, the cut-through vehicle crosses from right to left.

At first glance, what appears logical and has thus become state-of-the-art, has two drawbacks: First, noisy data, sensor dropouts, and ghost objects can disrupt the deterministic classification method described. Second, tolerances for jump-through (ε_s) and cut-through durations (ε_t) have to be chosen by experience, which may have a major impact on the robustness of classification. AI-based approaches, as discussed in the next section, are often more robust since they learn such parameters from training data.

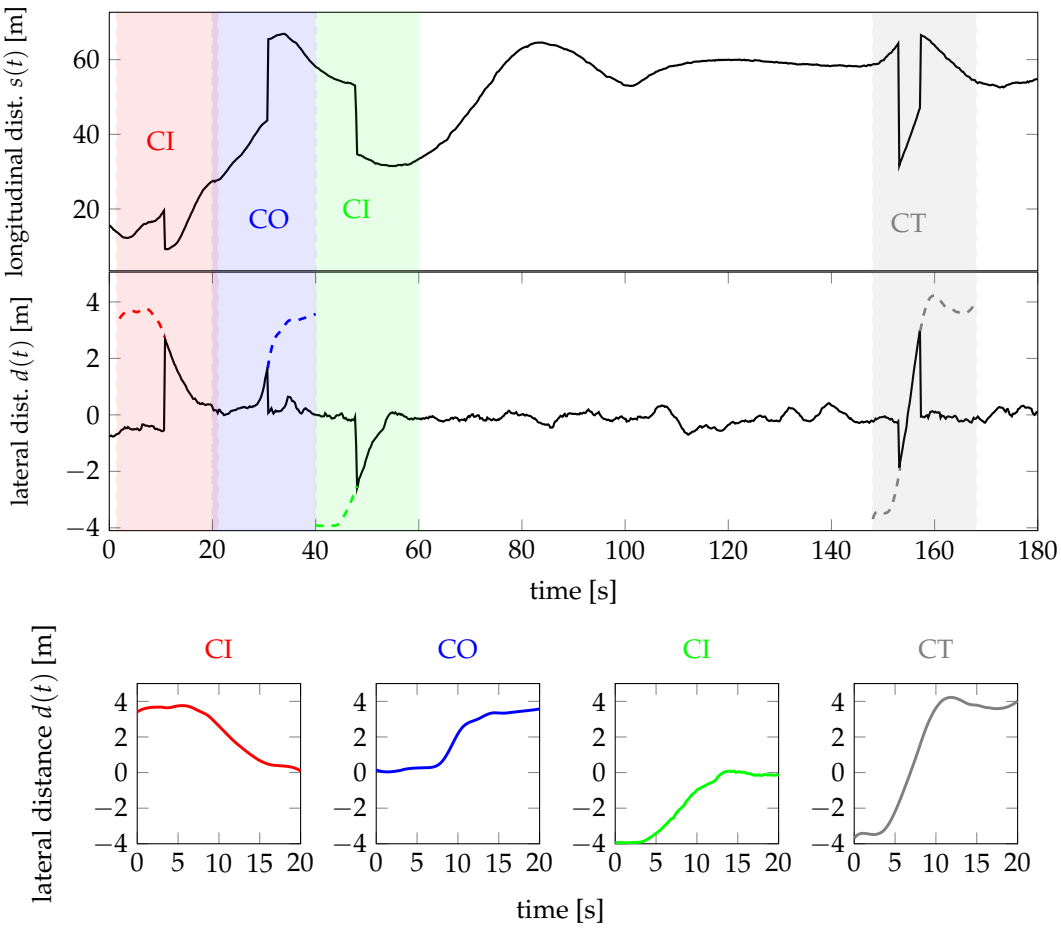

Figure 4. Scenarios identified from distance measurement $s(t)$ with corresponding snippets of lateral distance $d(t)$.

3. AI-Based Classification Scheme

Analogous to the rule-based classification, the general conditions—such as the ego vehicle driving on a highway at sufficient velocity without performing a lane change—remain valid. But the main difference involves using the lateral distance behavior $d(t)$ instead of $s(t)$ for classification, to align more closely with real drivers, who primarily rely on lateral information to assess driving maneuvers [18]. To identify scenario classes from a long sequence of measured data $d(t)$, a moving window approach is applied; see Figure 5. The sliding window of length $T_W = 20$ s extracts pieces of the $d(t)-$measurement to classify each window separately. Figure 5b shows some selected 20 s-sequences in the time range from 30 s to 50 s. From visual inspection, it can already be concluded that the $d(t)-$trajectory pieces have different characteristics. The first three windows show a jump from almost 4 m to -4 m in the lateral direction moving from right to left because the window moves in time from left to right. This may be considered a cut-through scenario. After some time, the jump diminishes, and the next three selected windows exhibit a gradual increase in $d(t)$ already observed for a typical cut-in maneuver. Again, after a while, this increase shifts to the left, and the last three windows display an almost straight line, which does not fit any of the three defined scenarios and is, therefore, classified as "other".

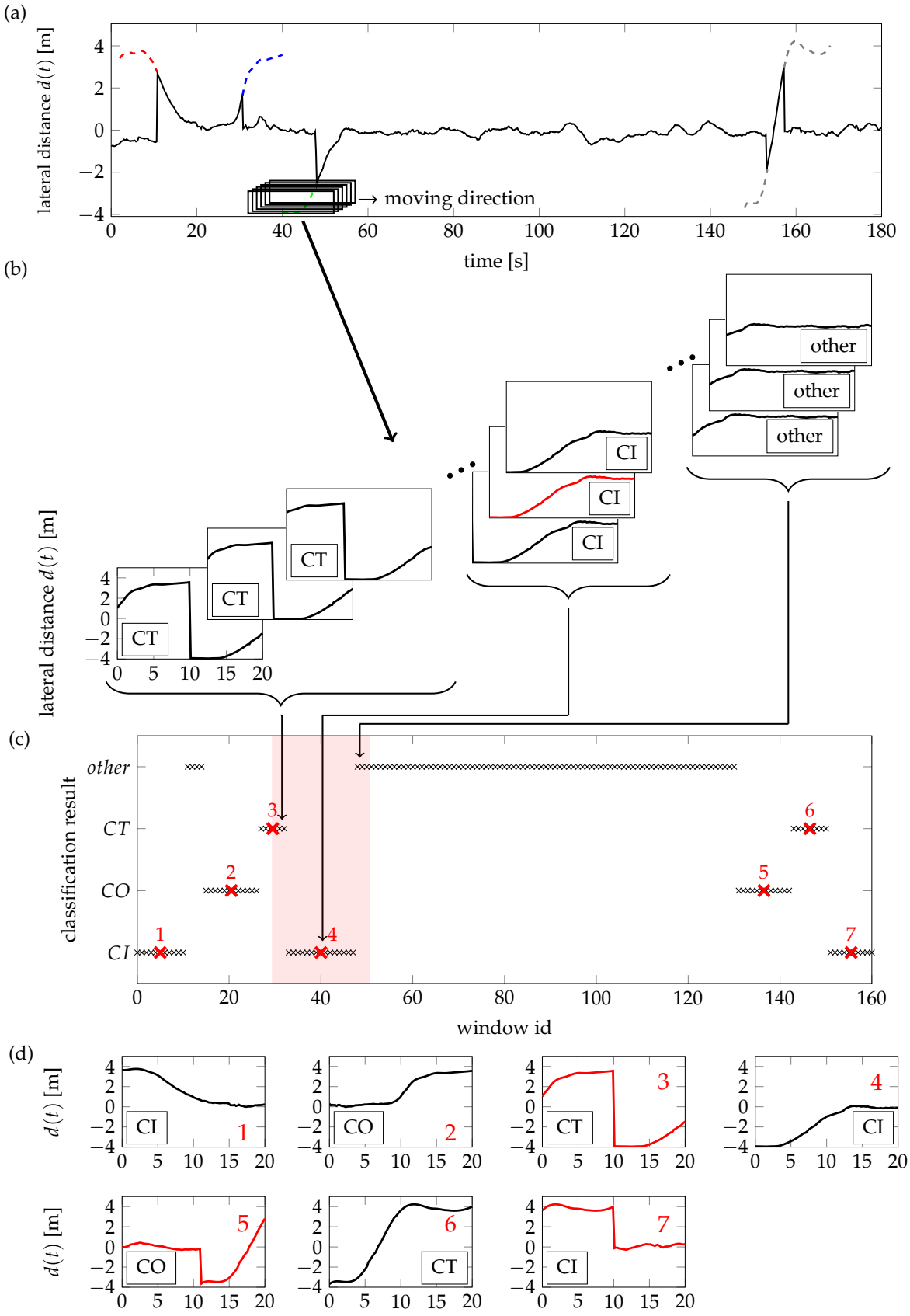

Figure 5. Selected moving windows (**b**) extracted from a lateral distance measurement (**a**) and corresponding classification (**c**), which finally results in aggregated scenarios (**d**).

This procedure results in the classification sequence in Figure 5c, where the time-step of moving the window is 1 s, which is short enough to ensure that every driving maneuver is included in at least one of the analysis windows. Obviously, adjacent windows mostly yield the same classification results associated with the same event. Therefore, these sliding window classification results have to be aggregated in final scenario events. Here, we determine that a sequence of at least five identical classification results can be summarized as a single event, each with associated time frames centered around the time points marked by red crosses for the scenario classes CI, CO, and CT. In particular, the middle crosses are taken as references, respectively, where the final windows with the corresponding lateral trajectories $d(t)$ of these events are illustrated in Figure 5d.

The kernel of the procedure is the assignment of a label $c \in C = \{\text{CI, CO, CT}\}$ to each window, W, referring to "cut-in", "cut-out", or "cut-through", where classification is solely based on the measured lateral distances $d(t_j \in W)$; see Figure 6. If none of these scenarios apply to W, the label $c \in \overline{C} = \{\text{other}\}$ is assigned. Presuming a measurement frequency of 50 Hz, each window of length $T_W = 20$ s contains $M = 1000$ data points. Thus, the data basis of 20 ssequences extracted from the measurement campaign of real driving tests is represented as a time series $\mathbf{d}$ with an associated class c to be determined:

$$\begin{aligned} &(\mathbf{d}, c) \text{ with } \mathbf{d} = (d_1, d_2, \ldots, d_M),\ d_j = d(t_j \in W), \\ &c \in C \cup \overline{C},\ C = \{\text{CI, CO, CT}\},\ \overline{C} = \{\text{other}\}. \end{aligned} \tag{1}$$

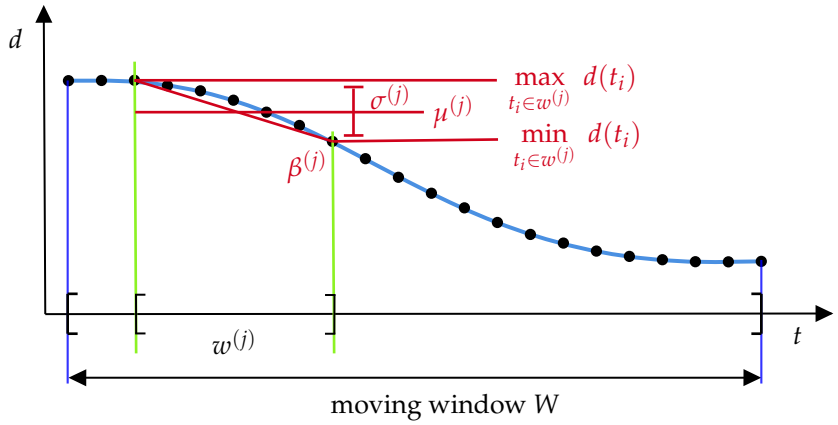

Figure 6. Feature extraction from subinterval $w^{(j)} \subset W$.

The applied classification method is based on the time series forest (TSF) approach [17] using a specific feature-engineering procedure. It splits the window, W, into randomly selected subintervals $w^{(j)}$ and assigns features like the mean, variance, and slope to each subinterval. In total, N subintervals are obtained by the so-called bagging, a randomized stripping of the whole window, W, into fixed, equal-width subintervals $w^{(j)} \in W$ with replacement (here the bagging interval size is set to 2s resulting in $|w^{(j)}| = 100$ data points, c.f. Figure 6). In the following, $K = 3$ interval features are considered and summarized in the feature vector, as follows:

$$\mathbf{x}_f(w^{(j)}) = \mathbf{x}_f^{(j)} := \left[\{\mu^{(j)}, \sigma^{(j)}, \beta^{(j)}\}\right]^T, \quad j = 1, \ldots, J. \tag{2}$$

The extracted features are the mean $\mu^{(j)}$, standard deviation $\sigma^{(j)}$, and the slope $\beta^{(j)}$ of a regression line, Figure 6, as follows:

$$
\begin{aligned}
x_{f,1}^{(j)} &:= \mu^{(j)} = \frac{1}{|w^{(j)}|} \sum_{t_i \in w^{(j)}} d(t_i), \\
x_{f,2}^{(j)} &:= \sigma^{(j)} = \frac{1}{|w^{(j)}| - 1} \sum_{t_i \in w^{(j)}} (d(t_i) - \mu^{(j)})^2, \\
x_{f,3}^{(j)} &:= \beta^{(j)} = \frac{d(\hat{t}_j) - d(\check{t}_j)}{\hat{t}_j - \check{t}_j}, \hat{t}_j = \arg\max_{t_i \in w^{(j)}} d(t_i), \check{t}_j = \arg\min_{t_i \in w^{(j)}} d(t_i)
\end{aligned} \tag{3}
$$

The calculations of the feature vectors $\mathbf{x}_f^{(j)}$ are repeatedly applied to all randomly stripped subintervals, also known as sample generation. These samples are then used to build up a decision tree by learning splitting criteria $x_{f,k}^{(j)} \leq \tau_k$ as the base component of a classification tree (see, e.g., Figure 7), classifying CI or non-CI. When satisfying this decision rule, the respective instances of the sample are sent to the left child node (blue), otherwise, to the right child node (yellow). These subsets are then further split until either the subsets become too small or a user-defined maximum depth N_f of the classification tree is reached (in the following, N_f is limited to the maximum depth 8). In the example in Figure 7, N_f is only three, and only binary classification is demonstrated for simplicity, although TSF is capable of dealing with multi-class problems, as in our application.

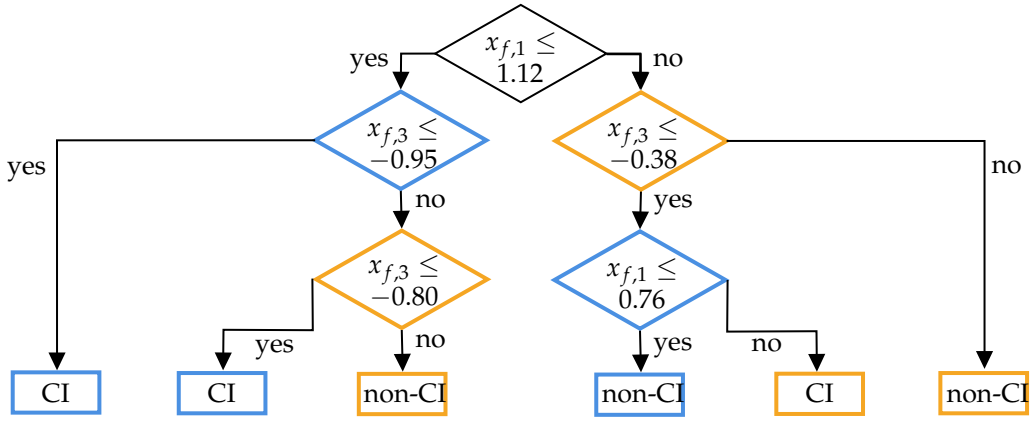

Figure 7. Exemplary binary time series decision tree of depth $N_f = 3$, where non-CI represents CO, CT, or other.

Similar to the rule-based decision tree, time series trees also employ a top-down, recursive strategy. However, in this approach, each node independently learns the best-split feature, $x_{f,k}$, and the best threshold τ_k for the split criterion from training, instead of pre-defining it. To achieve this, the best thresholds τ_k are found for all features $x_{f,k}$, $k = 1, 2, 3$, individually by maximizing a combination of entropy gain and margin [17]. Let $E_k(\tau_k)$ denote the maximum values of the split for each feature $x_{f,k}$; the one with the highest value is then selected for this specific node, as follows:

$$
k^* = \arg\max_k E_k(\tau_k). \tag{4}
$$

The resulting tree and, thus, its classification result, depends on the training set used. To make the classification procedure more robust, ensemble learning is applied: several different training subsets $W^{(l)} \subset W$ are generated by bagging, each resulting in its own decision tree ($\mathrm{DT}^{(l)}$). Applied to any test data, these multiple decision trees will predict different classification outcomes $c^{(l)}$. To end up with a unique result, this tree ensemble is combined, such that the majority of classes $c^{(l)}$ define the final class c; see Figure 8. In this paper, the classification model consists of $L = 200$ time series decision trees, each resulting from $J = 50$ subintervals $w^{(j)}$ consisting of $|w^{(j)}| = 100$ data points, respectively. In order to train the classification trees, a training set with hundreds of examples for each class

is required. Since manual labeling of real measurement data is rather time-consuming, trajectories are generated as idealized driving maneuvers, as described in Section 4. The advantage is that the labels are known a priori.

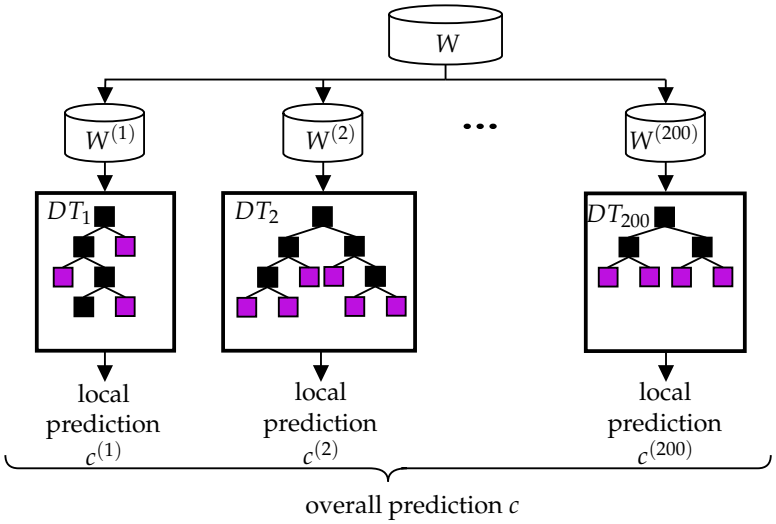

Figure 8. Exemplary ensemble of time series decision trees $(\mathrm{DT}^{(l)})$ with individual results $c^{(l)}$ combined to determine the final predicted class c.

The obtained time series-based machine learning classification model may then be applied to real-world data for automatic labeling of long-term measurements by performing classification of the moving windows. Figure 5 shows a short section of such a measurement with corresponding classification results. Similar to the rule-based approach in Figure 4, the AI approach finds the same four events labeled 1, 2, 4, and 6 in Figure 5c,d. In addition to the rule-based approach, the AI classification finds another three events, 3, 5, and 7, highlighted in red in Figure 5d. These, however, are misclassifications. The jump in event 3 from $d \approx 4$ to $d \approx -4$ was misinterpreted as a cut-through, although it actually resulted from switching the focus from the cut-in vehicle to another vehicle performing the next cut-in. Similarly, events 5 and 7 are misinterpretations of the different phases of a single cut-through maneuver 6. It should be noted that all these are speculations since the lack of causality in AI methods makes the interpretability of predictions difficult or even impossible. To be honest, the sequence in Figure 5 was selected to demonstrate the possibility of misclassification. Therefore, the above comparison should not be taken too seriously, but a rigorous assessment of the TSF performance must be made according to Section 5.

4. Generation of Idealized Maneuvers as Training Data

Apart from proper feature extraction and the model setup, the availability of enough training examples plays an important role in machine learning. In principle, labeled maneuvers obtained from measurements would be the best choice if available in a sufficient amount, otherwise, training data may be generated artificially [15,19]. To demonstrate the power of machine learning strategies, extremely idealized maneuvers will be used here, which are deduced from measurement data. According to Figure 1, ideal maneuvers have an initial phase with constant lateral distance $d(t)$, an S-shaped middle part describing the lane change, and a final phase that is also constant. Such an idealized cut-in maneuver is shown in Figure 9 as a red curve and can be described as a monotonic function $y_i = y(t_i)$ by a cubic Hermite spline with three sections: (I) a constant initial phase with value d^0 for $t_i \leq t^0$, (II) a 3rd-order polynomial, and (III) a constant end phase with value d^1 for $t_i \geq t^1$. Realistic values of the parameters determining this curve may be obtained by regression of a measured lateral trajectory $(t_i,\ d_i)$, $i = 1, .., 100$; see the blue curve in Figure 9. Thereby,

we assume 20 s time snippets consisting of 100 points similar to the specifications above. For simplicity, the original real-time t_i is substituted by its associated index, i, i.e.,

$$t_i \in \{1, 2, \ldots, 100\}, \quad t^0, t^1 \in [1, 100], \quad t^0 < t^1. \tag{5}$$

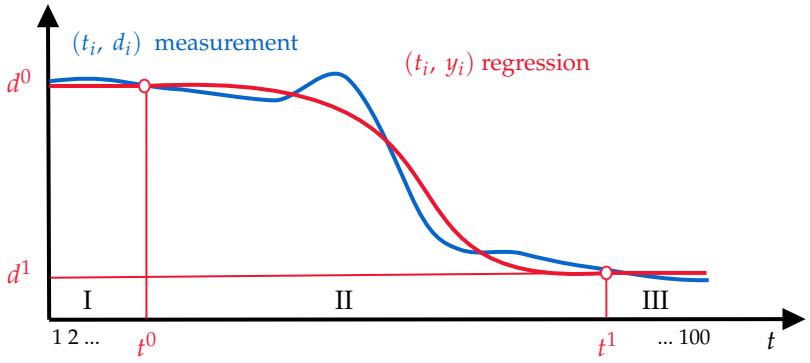

Figure 9. Idealized parametric model (red) describing a real cut-in maneuver (blue) by the three-section Hermite-spline regression.

The goal of regression is to find the transition points $t^0,\ t^1$ as well as the initial and final position values $d^0,\ d^1$ in an optimal sense. This may be formulated as a two-stage minimization problem, as follows:

$$\min_{1 \le t^0 < t^1 \le 100} \ \min_{d^0, d^1} \sum_i (d_i - y_i)^2. \tag{6}$$

Generally, a cubic polynomial with boundaries $(t_0,\ f_0,\ \dot{f}_0)$ and $(t_1,\ f_1,\ \dot{f}_1)$ may be described as follows:

$$f(t) = f^0 G_0 + f^1 G_1 + \dot{f}_0 H_0 + \dot{f}_1 H_1 \tag{7}$$

where

$$G_0 = 2\xi^3 - 3\xi^2 + 1,\ G_1 = -2\xi^3 + 3\xi^2,\ H_0 = (\xi^3 - 2\xi^2 + \xi)\Delta,\ H_1 = (\xi^3 - \xi^2)\Delta \tag{8}$$

with

$$\xi = \frac{t - t_0}{\Delta},\ \Delta = t_1 - t_0. \tag{9}$$

For zero derivatives $\dot{f}_0 = \dot{f}_1 = 0$, Equation (7) simplifies to the following:

$$f(t) = f^0 G_0 + f^1 G_1. \tag{10}$$

Applied to the three sections, for the given t^0 and t^1, we obtain the following regression conditions:

$$\begin{aligned} 1 &: y_i = f(t_i) \equiv d^0 \ \stackrel{!}{=}\ d_i && \forall t_i \le t^0, \\ 2 &: y_i = f(t_i) \equiv d^0 G_0 + d^1 G_1 \ \stackrel{!}{=}\ d_i && \forall t^0 \le t_i \le t^1, \\ 3 &: y_i = f(t_i) \equiv d^1 \ \stackrel{!}{=}\ d_i && \forall t_i \ge t^1. \end{aligned} \tag{11}$$

With submatrices $\mathbf{1} = [1 \ldots 1]^T$, $\mathbf{G_\bullet} = [\ldots G_\bullet(\xi_i) \ldots]^T$ and $\mathbf{d} = [\ldots d_i \ldots]^T$, this may be summarized as an overestimated system of linear equations for d^0, d^1:

$$\underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{G_0} & \mathbf{G_1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} d^0 \\ d^1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{d_1} \\ \mathbf{d_2} \\ \mathbf{d_3} \end{bmatrix}}_{\mathbf{b}} \tag{12}$$

where $\mathbf{A} \in \mathbb{R}^{100\times2}$, $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{b} \in \mathbb{R}^{100}$. The values d^0 and d^1 then result from the least-squares solution $\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ and the final Hermite-spline is given as $\mathbf{y} = [\dots y_i \dots]^T = \mathbf{A}\mathbf{x}^*$.

The outer minimization problem in Equation (6) may be solved by any nonlinear optimization algorithm. In order to avoid constraint (5) for t^0 and t^1, we can re-parametrize the problem with normalized parameters $p_1, p_2 \in [0, 1]$:

$$\begin{aligned} B &:= 1 + (N_t - 2)p_1 && \in [1, N_t - 1], \\ t^0 &= 1 + (N_t - B - 1)p_2 && \in [1, N_t - 1], \\ t^1 &= t^0 + B && \in [2, N_t]. \end{aligned} \tag{13}$$

A two-phase solution strategy is chosen, where global optimization by differential evolution [20] as the first phase is combined with local optimization by sequential quadratic programming (SQP), with the BFGS update [21] as the second phase to find global optimal parameters p_1, p_2 and, thus, t^0, t^1 for the regression problem (6). Each evaluation of the cost function requires a least-square solution of Equation (12), which is pretty fast. Differential evolution explores the parameter space to find an initial guess for the global minimum, while the SQP algorithm refines this solution to find a more precise minimizer.

As a result, we find the idealized maneuvers in Figure 10. In the upper row, cut-in maneuvers start from lateral distances of about $d^0 \approx 3.8$ m, which is the typical lane width of German highways, and ends at about $d^1 \approx 0$. Cut-out maneuvers begin from the ego lane with $d^0 \approx 0$ and involve a lane change to the neighboring lane, ending with $d^1 \approx 3.8$ m. Cut-through maneuvers typically start from the right neighboring lane with $d^0 \approx -3.8$ m and proceed to $d^1 \approx 3.8$ m. In the lower row, analogous training maneuvers from the other side are shown. Obviously, measurements reveal a wide variety of slow and fast lane changes, which have to be detected by the algorithms described in Sections 2 and 3. It should be noted that the training samples in Figure 10 could also be generated artificially by randomly selecting the parameters t^0, t^1, d^0, and d^1 within proper ranges and computing $y_i = y(t_i)$ along Equation (11). Here, it should also be noted that the used classification model is trained with around 8000 generated trajectories from each scenario.

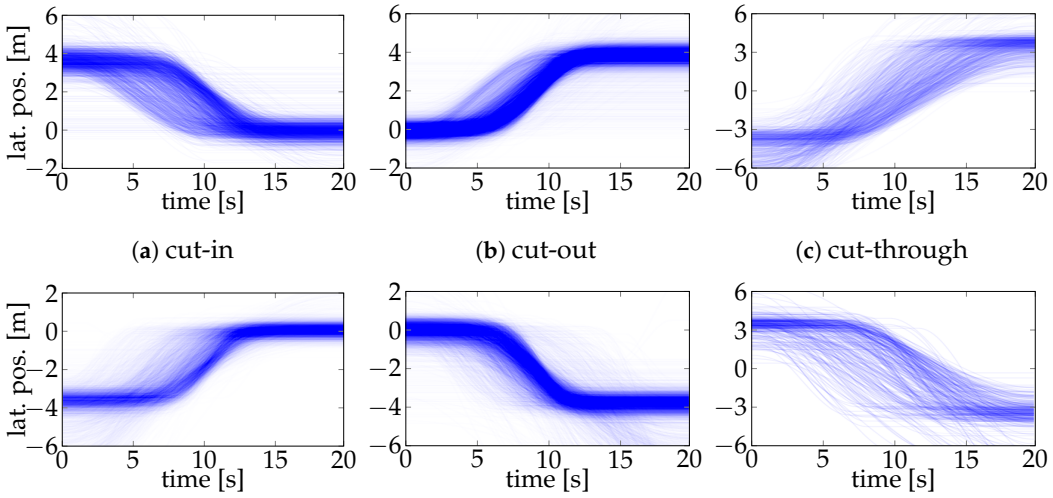

Figure 10. Samples of idealized maneuvers: (**a**) cut-in from left and right, (**b**) cut-out to left and right, and (**c**) cut-through from right and left.

5. Comparison of Classification Strategies

In order to test the performances of the two classification approaches presented in Sections 2 and 3 on real measurement data, 10 h of motorway driving are labeled, manually resulting in 77 cut-in (CI) maneuvers, 111 cut-out (CO) maneuvers, and 10 cut-through (CT) maneuvers. This is considered the ground truth. The classification quality of the rule-based

(RB) decision tree and TSF-based model may then be assessed from the confusion matrices in Table 1.

Table 1. Confusion matrix for the scenario classification.

(**a**) rule-based decision tree

		Real Scenario			
		CI	CO	CT	other
prediction	CI	66	-	-	8
	CO	-	97	-	7
	CT	-	-	9	2
	other	11	14	1	-

(**b**) TSF-based model

		Real Scenario			
		CI	CO	CT	other
prediction	CI	69	-	-	2
	CO	-	103	-	4
	CT	-	-	10	1
	other	8	8	0	-

The absolute frequencies or counts of predictions from the corresponding classification models, compared to the actual scenario events that occurred, are displayed. For example, Table 1a can be interpreted as follows: the rule-based decision tree correctly classifies 66 of the 77 CI events but misses 11 CI events and incorrectly classifies 8 other events as CI. Obviously, both approaches avoid misclassification with respect to the considered scenario types—CI, CO, or CT—but they do sometimes miss events or misclassify non-interesting measurement parts (= other) as one of the interesting scenarios. Therefore, cross-comparing the classification errors of the two classification models might be interesting. Table 2 highlights the falsely predicted events, which are labeled as 'other', in comparison to the predicted events from the other method for the same ones.

Table 2. Cross-comparison between prediction errors of the different classification approaches.

(**a**) wrong RB and corresponding TSF prediction

		TSF Prediction			
		CI	CO	CT	other
wrong RB	CI = 11	4	-	-	7
	CO = 14	-	6	-	8
	CT = 1	-	-	1	-
	other	-	-	-	-

(**b**) wrong TSF and corresponding RB prediction

		RB Prediction			
		CI	CO	CT	other
wrong TSF	CI = 8	1	-	-	7
	CO = 8	-	-	-	8
	CT = 0	-	-	-	-
	other	-	-	-	-

Based on these results, it can be concluded that the TSF complements the rule-based classification approach, e.g., in overall scenario classes, the TSF correctly predicts a relatively significant number of events that were falsely labeled by other methods. In contrast, the rule-based classifier correctly identifies only one event mislabeled in the TSF predictions. At first glance, the TSF appears to perform better, but an objective assessment requires additional metrics.

To apply the usual classification quality metrics to this multi-class classification problem, a transformation is needed that decomposes it into several binary tasks. The simplest approach is the One-vs-all (OVA) transformation shown in Figure 11. It generates three binary problems for the three classes, $c \in C = \{\text{CI, CO, CT}\}$, where each problem discriminates a selected class from the other three classes, including $c =$ "other" [22]. The resulting binary classification then only distinguishes two discrete classes, e.g.,

$$c_b = \begin{cases} 1 & \text{if } c = \text{CI (cut-in)}, \\ 0 & \text{if } c \neq \text{CI (non-CI)}. \end{cases} \tag{14}$$

In our case, Table 1 can be transformed into the binary confusion matrices shown in Table 3.

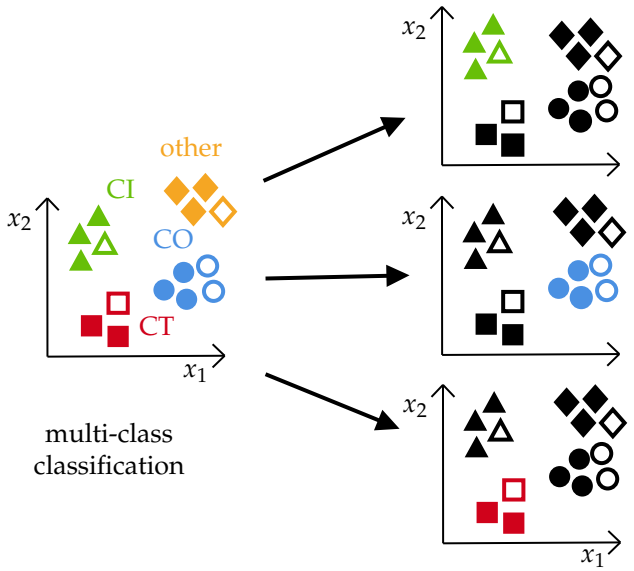

Figure 11. One-vs-all transformation of multi-class classification problems into binary classification problems (colors and marker types refer to predicted classes, filled markers refer to correct classification, unfilled markers refer to misclassification).

Table 3. Confusion matrix for cut-in (CI) after OVA transformation.

	(**a**) generally			(**b**) rule-based decision tree			(**c**) TSF-based model		
		Real			**Real**			**Real**	
		CI	non-CI		CI	non-CI		CI	non-CI
pred.	CI	t_p	f_p	**pred.**	CI	66	8	**pred.** CI	69	2
	non-CI	f_n	t_n		non-CI	11	130	non-CI	8	126

An ideal classifier would classify all cut-in events as CI and all non-CI maneuvers as non-cut-in. Misclassification occurs when a cut-in maneuver is classified as non-CI, or a non-CI event is classified as CI. These two variants of misclassifications have rather different implications. They are, therefore, distinguished when determining the classification quality.

We can count the following four cases of classification results after the OVA transformation; see Table 3a:

- t_p: number of true-positive outcomes (CI is classified as CI).
- t_n: number of true-negative outcomes (non-CI is classified as non-CI).
- f_p: number of false-positive outcomes (non-CI is classified as CI).
- f_n: number of false-negative outcomes (CI is classified as non-CI).

Based on the absolute frequencies of these four cases, commonly used criteria may be applied. For instance, the accuracy, i.e.,

$$s_{ACC} = \frac{\text{correct predictions}}{\text{all events}} = (t_p + t_n)/(t_p + t_n + f_p + f_n) \tag{15}$$

indicates how well a binary classifier predicts the correct result. According to Table 3, the rule-based decision tree has a lower accuracy of $s_{ACC,DT} = 196/215 \approx 91.2\%$ compared to $s_{ACC,TSF} = 195/205 \approx 95.1\%$ of TSF and, thus, has a lower classification performance.

Regarding problems with several classes, it is important to check the recall and precision of each class. The goal is to achieve maximum recall $s_R = t_p/(t_p + f_n)$ and precision $s_P = t_p/(t_p + f_p)$ for each label, which directly results in high accuracy. Recall is the probability that a real event is classified correctly, whereas precision describes the probability that the prediction of an event is correct. In other words, high precision means that the identified scenarios are likely to belong to the corresponding class, whereas high recall means that a large proportion of the real scenarios is identified.

The results of the rule-based decision tree (DT) and time series forest (TSF) for all three scenario types are listed in Table 4. Additionally, the supervised TSF (sTSF) was investigated, which is an extension of TSF, where feature intervals are not randomly selected from the entire time series but are instead determined based on their discriminative ability, enhancing the efficiency of TSF [23]. Moreover, additional features such as median, minimum, maximum, and interquartile range were extracted. All three classification approaches demonstrate high precision and recall performances for all three scenarios. In contrast to the impression gained from the discussion in Section 3, the slight differences confirm that TSF outperforms the other two classifiers. Notably, its highly sensitive approach results in outstanding precision, and its robustness—bolstered by a large number of decision trees with significant depth N_f—yields almost the best recall values for all scenario types.

Table 4. Precision and recall of various classification methods.

Classification Models	**Cut-in**		**Cut-out**		**Cut-through**	
	s_P	s_R	s_P	s_R	s_P	s_R
rule-based decision trees (DT)	89.2%	85.7%	93.3%	87.4%	81.8%	90.0%
time series forest (TSF)	**97.2**%	**89.6**%	**96.3**%	92.8%	**90.9**%	**100.0**%
supervised time series forest (sTSF)	95.8%	88.3%	95.4%	**93.7**%	83.3%	**100.0**%

Finally, an overall measure summarizing all three scenarios will be computed. Thereby, it typically should be taken into account that, due to the different rates of occurrence of the different scenario classes, the evaluated dataset is imbalanced. However, because of their equal relevance for risk assessment, here, each scenario class is taken into account equally to define total precision $s_{P,t}$ and total recall $s_{R,t}$ as arithmetic means

$$s_{P,t} = \frac{1}{3}\sum_{c\in C} s_{P,c}, \quad s_{R,t} = \frac{1}{3}\sum_{c\in C} s_{R,c}. \tag{16}$$

This ultimately results in Table 5, highlighting that over all scenarios, the TSF outperforms the two other classification methods.

Table 5. Total precision and recall for various classification methods.

Classification Models	**All Scenarios**	
	$s_{P,t}$	$s_{R,t}$
Rule-based decision trees (DT)	88.1%	87.7%
Time series forest (TSF)	**94.8**%	**94.1**%
Supervised time series forest (sTSF)	91.5%	94.0%

6. Conclusions

Validating highly automated driver assistance systems requires accurate risk assessment of safety-relevant driving situations. Detecting and classifying such scenarios in large datasets of measurements is a crucial initial step in the validation process. The currently applied rule-based scenario classification fails to meet all quality needs and is therefore replaced here by a time series forest (TSF) approach, which achieves better classification results. Due to its use of lateral distances to all neighboring vehicles, the TSF approach misses fewer critical scenarios than the rule-based decision tree, which only looks ahead along the ego-driving tube. It is also more robust against misclassifications of non-critical scenarios, thus leading to more reliable predictions of failure probabilities. It should be emphasized that highly idealized samples have been used for training the machine learning model. To further improve its robustness, measured snippets of more realistic artificial maneuvers generated by [15] may be used instead. Additionally, features may be extracted from $s(t)$ and combined with the $d(t)$-features already used in the paper.

7. Patents

A patent application is in the preliminary examination.

Author Contributions: Methodology and paper writing, Z.K. and D.B.; software and investigation, Z.K.; supervision, D.B. All authors have read and agreed to the published version of the manuscript.

Funding: The second author is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project no. 501840485.

Data Availability Statement: The datasets presented in this article are not available due to the privacy restrictions of the OEM. Requests to access the datasets should be directed to correspondence.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Europäisches Parlament. Safer Roads: New EU Measures to Reduce Car Accidents. Available online: https://www.europarl.europa.eu/topics/en/article/20190307STO30715/safer-roads-new-eu-measures-to-reduce-car-accidents (accessed on 7 August 2024).
2. Society of Automotive Engineers. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*; ISO, J3016_202104; SAE International: Warrendale, PA, USA, 2021.
3. van der Aalst, W. Six Levels of Autonomous Process Execution Management. *arXiv* **2022**, arXiv:2204.11328.
4. Richter, A.; Walz, T.P.; Dhanani, M.; Häring, I.; Vogelbacher, G.; Höflinger, F.; Finger, J.; and Stolz, A. Components and their Failure Rates in Autonomous Driving. In Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023), Southampton, UK, 3–7 September 2023.
5. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
6. Reichenbächer, C.; Rasch, M.; Kayatas, Z.; Wirthmueller, F.; Hipp, J.; Dang, T.; Bringmann, O. Identifying Scenarios in Field Data to Enable Validation of Highly Automated Driving Systems. In Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems-VEHITS, Online, 27–29 April 2022; SciTePress: Setúbal, Portugal, 2022; pp. 134–142. ISBN 978-989-758-573-9; ISSN 2184-495X. [CrossRef]
7. Wagner, J.; Häring, J. Validierung von hochautomatisierten Fahrfunktionen mit cloud-basierter Simulation. *ATZ* **2020**, *122*, 50–55. [CrossRef]
8. Pfeffer, R.; He, J.; Sax, E. Development and implementation of a concept for the meta description of highway driving scenarios with focus on interactions of road users. In Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems–VEHITS, Prag, Czech Republic, 2–4 May 2020.
9. Bock, J.; Krajewski, R.; Eckstein, L.; Klimke, J.; Sauerbier, J.; Zlocki, A. Data Basis for Scenario-Based Validation of HAD on Highways. In Proceedings of the 27th Aachen Colloquium Automobile and Engine Technology, Aachen, Germany, 8–10 October 2018.
10. PEGASUS. 2019. Available online: https://www.pegasusprojekt.de/en/pegasus-method (accessed on 14 April 2024).
11. Muslim, H.; Endo, S.; Imanaga, H.; Kitajima, S.; Uchida, N.; Kitahara, E.; Ozawa, K.; Sato, H.; Nakamura, H. Cut-Out Scenario Generation with Reasonability Foreseeable Parameter Range from Real Highway Dataset for Autonomous Vehicle Assessment. *IEEE Access* **2023**, *11*, 45349–45363. [CrossRef]
12. Zlocki, A.; König, A.; Bock, J.; Weber, H.; Muslim, H.; Nakamura, H.; Watanabe, S.; Antona-Makoshi, J.; Taniguchi, S. Logical Scenarios Parameterization for Automated Vehicle Safety Assessment: Comparison of Deceleration and Cut-In Scenarios from Japanese and German Highways. *IEEE Access* **2022**, *10*, 26817–26829. [CrossRef]
13. Ma, X.; Ma, Z.; Zhu, X.; Cao, J.; Yu, F. *Driver Behavior Classification under Cut-In Scenarios Using Support Vector Machine Based on Naturalistic Driving Data*; SAE Technical Paper 2019-01-0136; SAE: Warrendale, PA, USA, 2019. [CrossRef]
14. De Gelder, E.; Paardekooper, J.P.; Saberi, A.K.; Elrofai, H.; den Camp, O.O.; Kraines, S.; Ploeg, J.; De Schutter, B. Towards an Ontology for Scenario Definition for the Assessment of Automated Vehicles: An Object-Oriented Framework. *IEEE Trans. Intell. Veh.* **2022**, *7*, 300–314. [CrossRef]
15. Kayatas, Z.; Bestle, D.; Bestle, P.; Reick, R. Generation of Realistic Cut-in Maneuvers to Support Safety Assessment of Advanced Driver Assistance Systems. *Appl. Mech.* **2023**, *4*, 1066–1077. [CrossRef]
16. Pirouz, A. Identifizierung Kritischer Fahrszenarien aus Messdaten Mittels Machine Learning Methoden. Master's Thesis, University of Stuttgart, Stuttgart, Germany, 2021.
17. Deng, H.; Runger, G.; Tuv, E.;Vladimir, M. A Time Series Forest for Classification and Feature Extraction. *Inf. Sci.* **2013**, *239*, 142–153. [CrossRef]
18. Roesener, C.; Harth, M.; Weber, H.; Josten, J.; Eckstein, L. Modelling Human Driver Performance for Safety Assessment of Road Vehicle Automation. In Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018. [CrossRef]
19. De Gelder, E.; Hof, J.; Cator, E.; Paardekooper, J.P.; den Camp, O.O.; Ploeg, J.; De Schutter, B. Scenario Parameter Generation Method and Scenario Representativeness Metric for Scenario-Based Assessment of Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 18794–18807. [CrossRef]

20. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
21. Fletcher, R. *Methods of Optimization*; Wiley: Chichester, UK, 1987.
22. Rifkin, R.; Klautau, A. In Defense of One-vs-All Classification. *J. Mach. Learn. Res.* **2004**, *5*, 101–141.
23. Cabello, N.; Naghizade, E.; Qi, J.; Kulik, L. Fast and Accurate Time Series Classification through Supervised Interval Search. In Proceedings of the IEEE International Conference on Data Mining, Sorrento, Italy, 17–20 November 2020. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.