Department of Informatics
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Deep Traffic Scenario Mining, Detection, Classification and Generation on the Autonomous Driving Test Stretch using the CARLA Simulator

Mining-, Erkennung-, Klassifizierung- und Generierung- von Verkehrsszenarien auf der Teststrecke für Autonomes Fahren unter Verwendung des CARLA-Simulators

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Walter Zimmer, M.Sc.

Author Aaron Kaefer

Date February 15, 2022 in Garching

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, February 15, 2022

(Aaron Kaefer)

Abstract

Over the last years the development of autonomous driving systems has experienced substantial progress [201]. The main reason for this development is the potential to bring numerous improvements to individuals and society. Some of these improvements are assumed to be increased road safety, reduced traffic congestion, and an improved ecological footprint [141].
In order to achieve these benefits, it is crucial for a large number of people to use autonomous vehicles. For this to be realized the trust and acceptance for automated vehicles (AVs) must be strengthened [144]. To gain consumer trust and safety, an autonomous vehicle must be capable of understanding the environment around it in order to act appropriately. This includes various tasks including object detection, semantic scene understanding [44], and path planning [121].
In recent years large successes have been achieved in the field of object detection, specifically due to the introduction of powerful vision-based object detection algorithms such as You Only Look Once (YOLO) [152] or Single-Shot Detector (SSD) [125] and the availability of many annotated datasets to train new perception models [28, 33, 37, 44, 70, 86, 172].
This progress has lead to various successful test-wise deployments of automated vehicles on urban and highway roads [9, 133]. While showing promising performance, various traffic incidents caused by automated vehicles [30, 52] have negatively influenced the consumer trust in the technology [162]. These incidents show that in the future large improvements must be achieved in the field of safety assessment for automated vehicles.
This work is part of the Providentia++ project, that strives to contribute to the above mentioned areas. The work has a special focus on the task of driving scenario generation and classification for the safety assessment of automated vehicles. It is guided by the scenario-based approach (SBA) [148].
One goal of this work is to create a collection of diverse driving scenarios, which are automatically classified and labeled by an algorithm that is capable of detecting various driving maneuvers and traffic scenes. The labeled driving scenarios are then added to a dataset. The resulting dataset can be used to train deep learning (DL) algorithms on tasks such as maneuver prediction, scenario detection, and classification. Additionally, the classified driving scenarios are transferred to the OpenSCENARIO (OSC) [93] format for the usage in various simulation environments. This enables a simulation-based safety assessment of automated vehicles in different traffic situations.

Zusammenfassung

In den letzten Jahren hat die Entwicklung von autonomen Fahrsystemen erhebliche Fortschritte gemacht [34]. Der wesentliche Grund für diese Entwicklung ist das Potenzial, zahlreiche Verbesserungen für Individuen und die Gesamtgesellschaft herbeizuführen. Einige dieser Verbesserungen sind voraussichtlich die Erhöhung der Verkehrssicherheit, die Verringerung von Verkehrsstaus und eine Reduzierung des ökologischen Fußabdrucks [141].
Um diese Nutzeffekte zu erreichen, ist es entscheidend, dass eine Vielzahl von Menschen autonome Fahrzeuge nutzt. Damit dies gelingt, muss das Vertrauen und die Akzeptanz für automatisierte Fahrzeuge gestärkt werden [144]. Um Vertrauen zu gewinnen und die Sicherheit der Verbraucher zu gewährleisten, muss ein autonomes Fahrzeug in der Lage sein, seine Umgebung zu verstehen und angemessen zu handeln. Dazu gehören verschiedene Fähigkeiten wie Objekterkennung, Umgebungsverständnis [44] und Bewegungsplanung [121].
In den letzten Jahren konnten große Erfolge auf dem Gebiet der Objekterkennung erzielt werden, insbesondere durch die Einführung leistungsfähiger bildgestützter Objekterkennungsalgorithmen wie You Only Look Once (YOLO) [152] oder Single-Shot Detector (SSD) [125] und die Verfügbarkeit zahlreicher annotierter Datensätze zum Training neuer Perzeptionsmodelle [28, 33, 37, 44, 70, 86, 172].
Diese Entwicklungen haben zu mehreren erfolgreichen Tests von automatisierten Fahrzeugen auf urbanen Straßen und Autobahnen geführt [9, 133]. Obwohl die Leistungen vielversprechend sind, haben verschiedene von automatisierten Fahrzeugen verursachte Verkehrsunfälle [30, 52] das Vertrauen der Verbraucher in diese Technologie negativ beeinflusst [162]. Diese Vorfälle zeigen, dass in Zukunft große Verbesserungen im Bereich der Sicherheitsbewertung für automatisierte Fahrzeuge erzielt werden müssen.
Diese Arbeit ist Teil des Providentia++ Projekts, das auf eine praktische Umsetzung der genannten Forschungsbereiche abzielt. Diese Arbeit hat einen besonderen Fokus auf die Aufgabe der Fahrszenariengenerierung und -klassifizierung für die Sicherheitsbewertung von automatisierten Fahrzeugen. Sie orientiert sich am scenario-based approach (SBA) [148].
Ziel dieser Arbeit ist es, eine Sammlung verschiedener Fahrszenarien zu erstellen, die von einem Algorithmus automatisch klassifiziert und beschriftet werden, der in der Lage ist, verschiedene Fahrmanöver und Verkehrsszenen zu erkennen. Die markierten Fahrszenarien werden dann zu einem Datensatz hinzugefügt. Der resultierende Datensatz kann zum Trainieren von Deep-Learning-Algorithmen für Aufgaben wie Manövervorhersage, Szenarienerkennung und -klassifizierung verwendet werden. Zusätzlich werden die klassifizierten Fahrszenarien in das OpenSCENARIO (OSC) [93] Format für die Verwendung in verschiedenen Simulationsumgebungen übertragen. Dies ermöglicht eine simulationsbasierte Sicherheitsbewertung von automatisierten Fahrzeugen in unterschiedlichen Verkehrssituationen.

Contents

List of Abbreviations

AD	Automated Driving
ADAS	Advanced Driver Assistance System
ASAM	Association for Standardization of Automation and Measurement Systems
AV	Automated Vehicle
BMVI	Federal Ministry of Transport and Digital Infrastructure
CNN	Convolutional Neural Network
CRISP-DM	Cross Industry Standard Process for Data Mining
DL	Deep Learning
DM	Data Mining
FOT	Field Operational Testing
HD	High-Definition
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KDD	Knowledge Discovery in Databases
LiDAR	Light Detection and Ranging
LSTM	Long-Short-Term Memory
ML	Machine Learning
Euro-NCAP	European New Car Assessment Program
NHTSA	National Highway Traffic Safety Agency
OD	OpenDRIVE
ODD	Operational Design Domain
OSC	OpenSCENARIO
PEGASUS	Projekt zur Etablierung von generell akzeptierten Gütekriterien, Werkzeugen und Methoden sowie Szenarien und Situationen zur Freigabe hochautomatisierter Fahrfunktionen

Radar	Radio Detection and Ranging
SAE	Society of Automotive Engineers
SBA	Scenario-Based Approach
SSD	Single-Shot Detector
TP	Traffic Participant
TTC	Time to Collision
UNECE	United Nations Economic Commission for Europe
XML	Extensible Markup Language
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Providentia++

This thesis is written in cooperation with Providentia++, which is a project in the field of autonomous driving. Its aim is to improve traffic flow and road safety by overcoming the limitations of sensor systems that are solely located on-board autonomous vehicles. The perception of autonomous vehicles is enlarged by adding a sensor-system composed of cameras, Light Detection and Ranging (LiDAR) sensors, and Radio Detection and Ranging (Radar) sensors to the road infrastructure. The system is used to detect and track traffic participants (TPs), located on a specific road segment. It can then pass information concerning the current traffic situation on to other vehicles close by via a 5G connection. With this information, the control unit of the receiving vehicles can choose more efficient routes and can be warned about dangerous situations early on [131].

Providentia++ was initiated in 2017 and since the beginning of 2020 is lead by the Chair of Robotics, Artificial Intelligence and Real-time Systems at the Technical University of Munich's Department of Informatics. It is funded by the Federal Ministry of Transport and Digital Infrastructure (BMVI). Additional cooperative partners supporting the project are Fortiss, Valeo, Intel, Cognition Factory, Elektrobit, Huawei, 3D Mapping Solutions, brighter AI, Siemens, and Volkswagen.

1.2 Problem Statement

In order to make intelligent decisions an autonomous vehicle must be able to perceive and understand its surrounding environment. This means in a first step it must be able to detect and localize surrounding objects and obstacles. Additionally, in order to plan appropriate maneuvers or react to dangerous situations it is highly important to understand the interactions between the detected objects and create a comprehensive understanding of the traffic situation the autonomous vehicle is approaching [121].

To train Deep Learning algorithms for the above-mentioned tasks, a dataset including labels to train and validate the algorithms is necessary. Depending on the specific task different types of data and labels are necessary. The main tasks in the domain of automated driving (AD) and the corresponding relevant data and labels are listed in Table 1.1 [12, 44, 91, 187, 203].

Task	Data Type	Label Type
Object Detection	Images, LiDAR point cloud, radar	Bounding Box (2D/3D location)
Trajectory Prediction	Trajectory data	Coordinate points with corresponding time stamps
Scene Understanding	Trajectory data, images	Trajectories labeled with maneuver types

Table 1.1: Overview of data type and labels necessary for different deep learning tasks related to autonomous driving (sources: [12, 44, 91, 187, 203]).

Two main challenges that arise when creating such a dataset are data coverage and correct data labeling. In the field of autonomous driving, these two challenges translate into very concrete problem definitions. A sufficient data coverage means to have a large enough pool of driving scenarios. These must firstly cover all driving situations that can occur and secondly be labeled correctly in order to be capable of training perception and navigation algorithms appropriately. Both of these problems are addressed and solved in this work.

Additionally, algorithms used in AVs must undergo a thorough safety assessment before a public deployment is possible. A safe operation however, can only be guaranteed if the AVs proper functionality is proven in all driving situations that can possibly occur in the real-world. Since this is practically infeasible [195], simulation approaches using driving scenarios are a common practice for the safety assessment of AVs [15]. In the literature this approach is often referred to as the SBA [23, 25, 56, 139, 148, 154, 155]. The main challenge for this approach is determining the relevant test cases and obtaining driving scenarios that cover all variations of these test cases [80]. In summary three main research questions arise when creating an implementation of the scenario-based approach.

1. “Which driving scenarios are relevant for the assessment of automated vehicles?”

2. “How can an extensive collection of driving scenarios be created?”

3. “How can driving scenarios be carried out to assess the safety of automated vehicles?”

These questions are addressed and solved by the contributions made in this work. A detailed overview of the contributions is given in the following section.

1.3 Contributions

This work focuses on the creation of a scenario dataset for the Providentia++ test stretch on which Deep Learning-based perception algorithms in the field of autonomous driving can be trained and validated for various tasks such as object detection, trajectory prediction, traffic scene understanding, and more.

To create labels for training such algorithms, an automatic recognition of specific vehicle maneuvers in traffic scenarios is necessary. The goal is to create a properly labeled scenario

dataset in an automated fashion. To achieve this, an algorithm is needed that can automatically detect and annotate vehicle maneuvers (turn left, slow down, accident, etc.) that occur based on recorded scenarios. Also, the scenario as a whole must be classified.

The work consists of six main contributions for creating, labeling, classifying, and visualizing driving scenarios. The contributions are visualized in Figure 1.1.

Figure 1.1: Overview of the main contributions made in this work (source: own illustration).

1. Scenario Catalog

   In the first stage various sources are analyzed in order to find traffic scenario types that are safety critical and relevant for the development of automated driving applications. These sources include documents from government agencies such as the National Highway Traffic Safety Agency (NHTSA) [143], European New Car Assessment Program (Euro-NCAP) [57], and United Nations Economic Commission for Europe (UNECE) [142]. The goal of this analysis is to find a good mix of common driving patterns and those situations that especially frequently lead to accidents. These

scenario types are then added to a shortlist from which a catalog specifically for the Providentia++ test stretch is created.

After the shortlist is finalized, it is determined which of the scenario types occur on the Providentia++ test stretch. This is an important step as many scenario types, such as entering a toll station or exiting a parking lot, cannot occur on the test stretch, based on the fact that they do not exist on the test stretch. Therefore, they can be ignored. Once the relevant scenario types for the Providentia++ test stretch are determined, they are added to the catalog, which provides an extensive scenario overview.

2. Scenario Generation Framework

   The "Scenario Generation Framework" additionally supports the artificial creation of driving scenarios that cannot be extracted from the Providentia++ data recordings because they have not occurred on the test stretch yet. This is usually the case for rare events such as accidents or vehicle failures. Thanks to the generation framework, new driving scenarios can be created very efficiently and exactly to the users' needs.

3. Scenario Extraction Pipeline

   The data recordings from the Providentia++ test stretch contain information about all detected vehicles and objects at each point in time. In order to form a complete driving scenario from the raw data recordings, a data extraction pipeline is created. The pipeline is referred to as the "Scenario Extraction Pipeline" in this work. It utilizes the sensor recordings to characterize the actors and their trajectories, as well as the environment conditions. This information is then transferred into a format that formally describes concrete driving scenarios.

4. Scenario Augmentation Framework

   In scenarios extracted from real-world data, some driving situations occur more often than others. This is normal since situations like accidents are much less likely to happen compared to lane changes, for example. To increase the number of data samples from underrepresented scenario types, the "Scenario Augmentation Framework" is introduced. This framework allows for a simple yet very effective variation of driving scenarios to increase the number of data samples. The variations are created using a stochastic sampling based approach. The user can control the stochastic sampling behavior by defining the sampling intervals. These can either follow a Gaussian distribution or a randomly sampled number inside a sampling range. In the first case the mean and standard deviation of the Gaussian distribution are provided as inputs by the user. In the latter case the user defines the beginning and end of the sampling interval. For example if the user wants to sample values between 0 and 10 the input parameters for the sampling range are 0 marking the beginning and 10 marking the end of the interval.

5. Maneuver Detection / Scenario Mining

   In the past, the driving scenario types that are included in the data recordings from the test stretch have been determined manually. This is done by re-watching each of the recordings and manually classifying the scenario type. However, to reduce the amount of manual labor the process of classifying driving scenarios is automated. Therefore, in this work a maneuver detection and scenario mining framework is created. It is composed of the following two components:

(a) Automatic Labeling Framework
Labeling framework for automated labeling of driving maneuvers.

(b) Maneuver Detector
Detection of driving maneuvers carried out by the actors in a driving scenario.

6. Scenario Statistics

For each driving scenario statistics are calculated containing information about the number of occurrences of each driving maneuver type as well as extreme values. These include the top speed and the maximum number of lane changes committed by a single vehicle. Based on the calculated statistics a classification of driving scenarios into various categories is possible.

7. Scenario Simulation / Visualization

The OpenSCENARIO standard is a file format for describing driving scenarios. Such files can be read by many simulation and visualization tools for testing purposes. To make use of these tools, a "Scenario Writer" is created, that parses extracted or synthetically generated driving scenarios into the OpenSCENARIO format.

Chapter 2

Theoretical Foundations

This work contains two main accomplishments. One is the creation of a labeled dataset for training Deep Learning algorithms for the Providentia++ test stretch. The second accomplishment is the creation of a driving scenario database for the simulation-based assessment of automated vehicles. The basic steps necessary to achieve these two goals and ensure a high quality of the data are outlined in the introduction of this work. This section describes the theoretical foundations necessary to fully understand each of the phases.

2.1 Open Source Datasets

In the area of autonomous driving, many complex tasks such as 3D map construction, object detection, and others are solved by trained Deep Learning models. However, in order to train these models large amounts of training data are necessary [86]. Since it is often tedious to extract data from sensor recordings to form a dataset, it is a common practice within the research community to share data. Such publicly available datasets fall into the category of open source datasets. The goal of making datasets publicly available is to help others researching in the same domain to overcome the difficulties of acquiring enough training data and to focus more on solving their research challenges [146].

2.2 Labeling

For an algorithm to learn from data and to be able to make predictions into the future, it needs to verify that the way it is interpreting the data is correct. This is comparable to a student, that is trying to solve mathematical problems and comparing his result to the solution given by the professor. If the result and solution match, the student has verified that he or she has correctly learned how the specific problem is solved [11].

In the case of machine learning (ML) models, in the training phase the model parameters are adjusted accordingly so that the input data is transformed to generate the correct output. To ensure the model parameters are adjusted appropriately by the optimization algorithm, it must be known what the correct output is for every input sample in the training data [129].

These outputs corresponding to input data must be generated in a process called “labeling”. In many famous datasets, including MNIST [42] for digit recognition and CIFAR-10 [110] for image recognition, the labels are generated manually by humans. This is the

case for most datasets for Machine Learning. However, there are also automated or semi-automated approaches that reduce the human labor necessary for labeling data [129].

2.3 Data Mining

The term data mining (DM) describes the analysis of usually large data collections with the help of computer-aided methods. This is a partly iterative process, which aims to identify correlations within the data. These patterns must meet certain criteria in order to be evaluated as meaningful. The criteria include that the patterns are valid over a large part of the data and that they describe previously unknown and potentially useful relationships within the dataset. The relationships are represented using logical or mathematical models. The discovered patterns are subsequently to be recognized in new unknown data as well. Furthermore, there is also the possibility to automatically detect new patterns, for example to derive predictions for the future [79].

2.4 Scenario Mining

Scenario mining is generally the same as Data Mining, only with a focus on extracting certain scenario types from recorded sensor data. The extracted information is used to automatically categorize data recordings of driving scenarios into different categories. Typical categories are lane change, accident, or cut-in. Scenario mining can also include the augmentation of existing scenarios or the artificial creation of new scenarios [69].

The development of autonomous driving systems relies on algorithms that perform complex tasks in highly diverse driving scenarios. To ensure the functionality of these systems, it is crucial to train and test them appropriately. For both the training and testing phase, it is important to have realistic data so that the system operates appropriately when deployed in the real world [106].

To ensure this, the algorithms that solve the various tasks associated with autonomous driving must be trained with an exhaustive list of traffic scenarios that commonly occur and traffic scenarios that could potentially occur in the future [80].

In practice, many scenario types such as lane changes or right turns at a crossing, are more common than for example an accident. Hence, a natural class imbalance can occur. Since this often is undesirable when training Deep Learning algorithms [24], it is common to mitigate this problem by augmenting or artificially creating underrepresented scenario types [38, 82].

2.5 Standardized Process Models for Data Mining

Unlike commonly assumed, data mining projects usually cannot be solved by a standardized approach. Rather, they are a creative process that requires interdisciplinary skills and knowledge. The success of the projects depends very much on the individual circumstances and peculiarities. Since these can be very different from project to project, it is very difficult to

define a standardized process [196].

However, some basic procedures can be found in all data mining projects. In order to increase the chances of success and to make the overall process more transparent, different procedure models have been developed, which represent all basic work steps and bring them into the overall context. On the one hand, this should significantly increase the chances of success, on the other hand, it also provides a better overview of the planned work steps for outsiders, in order to be able to estimate and understand the costs or duration of a project more accurately, for example [196].

Widely used process models are the Cross Industry Standard Process for Data Mining (CRISP-DM) [196] and the Knowledge Discovery in Databases (KDD)[62] [21]. In the section related work, the details including benefits and drawbacks of both process models for Data Mining are examined and explained.

2.6 Data Pre-Processing

In general, data is of high quality if the associated data mining goals can be met. Quality is determined by factors such as accuracy, completeness, credibility and the ability to interpret the data. In most real-world applications, the factors that determine quality are not fully developed, which is why pre-processing is necessary. [78] Data pre-processing represents the first data processing step of a data mining project after the collection of all data to be analyzed is completed. This process step is sometimes one of the most important, as it has a direct impact on the success of a project. For example, errors in the data during an analysis can lead to incorrect conclusions being drawn. Kotsiantis et al. [107] divide data pre-processing into the following steps, which can be processed one after the other and are explained in detail in this chapter:

1. Outlier Detection
2. Handling of Missing or Faulty Data
3. Discretization
4. Normalization
5. Feature Selection
6. Feature Construction

1. Outlier Detection

   Outliers are data values that lie outside the range of possibility or can be classified as extremely improbable, for example, because they lie outside a certain probability distribution of the total data points. There are different categories into which outliers can be classified according to their type, as can be seen in Table 2.1.

2. Handling of Missing or Faulty Data

   Incomplete datasets are very common in practice and can hardly be avoided. This is mainly due to the methods of data generation. First of all, it is important to understand why the missing data occurred in order to be able to initiate appropriate corrective

Problem	Outlier Type	Example
Illegal values	Cardinality	Gender > 2 induces problem
	max, min	All values should lie inside minimum and maximum value range
	Variance, Deviation	Variance and deviation of statistical values should not be higher than a certain threshold
Spelling errors	Feature values	Sorting on values often brings misspelled values next to correct values

Table 2.1: Overview of data type and labels necessary for different deep learning tasks related to autonomous driving (source: [107]).

measures. For example, data may be (1.) forgotten or lost during generation, (2.) not applied to a specific instance or non-existent for that instance, or (3.) contain a so-called "don't care" value due to irrelevance. [107]

Depending on the type of imperfections in the dataset, it is possible to decide between several methods for correction. Typical methods are:

(a) Removing all Instances with One or More Missing Values
In this process all instances of the dataset are removed, which have at least one missing value. This leads to a size reduction of the whole dataset. Advantages of this method are that only complete data are used and therefore no distortion of the results due to wrongly assumed substitute values for the missing parts can occur during the later model building. A disadvantage of this method is that with datasets with relatively few missing parts, which are however very evenly distributed over all instances, it can come to a substantial reduction of the data. In the worst case, too little data remains to perform further data mining steps [67].

(b) Most Common Value
Another method is to replace all missing values with the same-, most frequently occurring value within the same feature. This method has the advantage of preserving the size of the dataset, but it can also lead to a falsification of the results in later data analysis due to the uncertainty regarding the correctness of the replaced values. Nevertheless, in many applications, this method leads to an improvement in model performance [126].

(c) Concept, Most Common Value
In this method the handling of the missing values runs similarly to the "most common value" method. The difference is that reference is also made to the class in which the instance is located. Thus the dataset can be divided for example into several classes, which all contain a certain number of instances. A missing instance is now replaced with the most frequently occurring value of the same feature within the same class. Table 2.2 shows an example of the procedure. The goal of this approach, compared to the "most common value" method, is to select

the replacement values in such a way that they are better adapted to the respective class in which the instance is located, and a possible distortion of the data is further reduced [107].

Instance	Class	Feature value
1	1	1
2	1	1
3	1	3
4	1	1
5	1	1
6	2	2
7	2	2
8	2	Missing value
9	2	2
10	2	1

Class 1 Class 2

Table 2.2: Example for "concept, most common value" (source: own illustration).

The dataset shown in Table 2.2 can be divided into two classes. According to the "concept, most common value" approach, the missing value of instance 8 would be overwritten with a "2", since the "2" represents the most common value within the same class. Compared to this, under the "most common value" system, the missing value would be replaced with a "1" since it is the most frequent across the entire dataset.

(d) Substitution by Average Value
To minimize the influence of a missing value on later analysis results, it is possible to replace the missing value with the average of all values of the same feature. This method turns out to improve the quality of the dataset in many cases [67]. As long as the values of the affected characteristic are subject to a normal distribution, the substituted average values have little influence on the results of a later data analysis [67].

(e) Regression or Classification
When dealing with error locations by regression or classification, a machine learning model for regression or classification can be used, depending on the application. The model is trained with the complete part of the dataset, whereby the feature whose errors are to be corrected in the incomplete part of the dataset is treated as an output. Subsequently, the trained model is applied to the incomplete part of the dataset. Based on all complete features, the values of the feature with missing values are determined. [78]

(f) Hot-Deck Method
In the hot-deck method, the most similar complete instance is determined for an instance with a missing value. The missing value in the incomplete instance is replaced with the value of the complete instance. This procedure is performed for all instances with missing values. [107]

(g) Treat Missing Values as Special Values

Another approach is the replacement of missing values with an uniform special value. The special values stand out from the "normal" values and can also be used as information. [107]

3. Discretization

The features in a dataset can be either categorical or continuous. The term "continuous" refers to features that contain numerical values or generally have a linearly ordered range of values [61].

In information theory, the term discretization refers to the conversion of values from a continuous set of values into values from a discrete set of values. The goal of such a process is to reduce the total number of possible values. This significantly increases the efficiency of processing algorithms, since fewer computational operations need to be performed [61]. This can be done using supervised and unsupervised algorithms. The supervised algorithms test the results of the discretization using a machine learning model. If the results are worse, the discretization is adjusted until this is no longer the case. In unsupervised discretization, it is common to determine the upper and lower bounds of the set of values and divide this range into k intervals of equal size. Each continuous value is assigned the value of the interval in which it is located [61].

Discretization algorithms can generally be divided into two categories, the "supervised" and "unsupervised" algorithms. The "unsupervised" algorithms do not make any references to the labels, while the "supervised" algorithms do [61].

The simplest type of discretization is the unsupervised method "equal size discretization". Here, the maximum and the minimum of a feature are determined and the span between them is divided into k intervals. Each interval has its own value and all continuous values within this interval are assigned this value. The operation of this form of discretization is illustrated in the example Table 2.3 [61].

Set of continuous values	Number of intervals	Partitioning of intervals	Value assignment
0	10	[0;1] →Interval 1	0 →1
0.263		[1;2] →Interval 2	0.263 →1
0.36		[2;3] →Interval 3	0.36 →1
1.1		[3;4] →Interval 4	1.1 →2
1.9		[4;5] →Interval 5	1.9 →2
2.464		[5;6] →Interval 6	2.464 →3
4.2		[6;7] →Interval 7	4.2 →5
5.234		[7;8] →Interval 8	5.234 →6
7.452		[8;9] →Interval 9	7.452 →8
8.1		[9;10] →Interval 10	8.1 →9
9.32			9.32 →10
9.7			9.7 →10
10			10 →10

Table 2.3: Example of the operation of the "equal size discretization" method (source: own illustration).

Many discretization methods are divided into top-down or bottom-up methods. The top-down methods start with a number of intervals chosen at the beginning, which are further divided with each iteration until no further improvement of the results occurs. In contrast, in the bottom-up method, adjacent intervals are gradually combined until the results degrade. Some of these methods require user parameters to influence the behavior of the discretization criterion or to set a threshold for the stopping criterion [8].

4. Normalization

   Numerous algorithms - especially in the field of machine learning - can only process input variables and draw meaningful results from them if they are in a certain form. This can imply, for example, that the input variables must lie in an interval between 0 and 1. Since in practice data is recorded from a wide variety of sources, the data values rarely lie in the interval required for the algorithm. Nevertheless, in order for information to be extracted from the data, the data must be transformed into the required form. This transformation step is called "normalization". Normalization is used to make data of any form compatible for a machine learning model [8].

   There are a few different forms of normalizing data. Two commonly used methods are "min-max" and "z-score" normalization:

   min-max normalization:

   $$v' = \frac{v - min}{max - min}$$

   z-score normalization:

   $$v' = \frac{v - mean}{standard\ deviation}$$

   Where v represents the input value and v' represents the normalized value [107].

5. Feature Selection

   Feature selection is the process of selecting the most relevant features from all the features available in the data. The goal of feature selection is to identify the features that contain information relevant for the task to be solved and to reduce the originally collected data to the features that really contribute to the solution of the problem [50].

   Feature selection represents a process step in which the most relevant features are to be recognized from all features occurring in the data. The resulting dimensionality reduction of the dataset allows processing algorithms to work faster and more effectively. Removing irrelevant features also improves the results of interpretation models. [50].

   Features are divided into three different classes by Kotsiantis et al. [107]:

   (a) Relevant: These features influence the output variable or contain information that allows to draw a conclusion about the output. In addition, the information contained in relevant features is not contained in the rest of the features.

(b) Irrelevant: Irrelevant characteristics do not contain any information regarding the outcome variable and thus have no influence on it. The values of irrelevant characteristics are random with respect to the output variable.

(c) Redundant: A feature is redundant if there is another feature that contains all the information of the first feature, or if all the remaining features contain this information. Redundant features are not needed for some applications.

Feature selection algorithms generally consist of two components. One component tries to find the features which together in a new subset lead to the best results in the later model building. The second component consists of an evaluation algorithm that provides a measure of how well the previously assembled subset is suited for model building. The process of feature selection is characterized that many different combinations of features are gradually assembled into separate subsets. Immediately after its generation, each subset is evaluated and given a score. The subset with the highest score is used for further model building and the features in it are thus "selected". To prevent the selection algorithm from running for an extremely long time or even infinitely, a termination criterion is required. This can be, for example, a fixed maximum number of iterations [50].

6. Feature Construction

The features recorded in the data are not always suited for a model optimally. For example, it may be that the recorded features contain the necessary information, but in a form that does not allow a model to make a statement. In such a case it is helpful to derive new features from the original features, which make important information accessible for a model. The newly generated features can thus lead to an increase in the prediction accuracy of a classification model. The generation of new features is called feature construction [107].

2.7 Traffic Scene Recognition

Traffic scene recognition is an important task for autonomous vehicles in order to make sense of the perceived environment and predict the behavior of surrounding traffic participants. Understanding a scene and the intentions of the surrounding actors helps the ego vehicle make smarter decisions when planning its future motion [39]. The aim of traffic scene recognition is to close the gap between the current performance of path planning algorithms and the visual reasoning capability of human beings [198].

Traffic scene recognition, or more general scene recognition, often includes various tasks like object recognition and object localization, semantic segmentation, and mapping [120]. In a first step objects and their location are detected. This usually contains the most information regarding a scene, since the main actors and their position are determined. The segmentation step separates the scene into meaningful regions [39]. This can contain additional information as, for example, an actor may behave differently when in the kitchen as opposed to in the pool. Similarly, mapping the scene can add additional information [120].

2.8 High Definition Maps

In recent years one of the foundations for many successful deployments of autonomous vehicle in real-world situations is the use of high-definition (HD) maps [90]. Compared to traditional digital maps, HD maps not only have knowledge of the road network, but also incorporate highly detailed information about the environment, which can be split into multiple levels of detail:

1. Road model
2. Lane model
3. Localization model

The road model, like traditional digital maps, includes information about the road network. This is used for planning general routes analogously to a navigation system. The road model of an HD map additionally contains information on which traffic rules apply to each lane. These can be static rules, like no lane change allowed or no left turn, but also dynamic rules coming from a traffic light [124]. Figure 2.1 (left) illustrates a typical road model.

To plan routes in high detail - for example which lane is used or when a lane change should be made - the road network is not sufficient. In some cases, when driving in a lane, it may be beneficial for the vehicle to take a route that does not follow the center line of the lane. Reasons for this can be that an obstacle must be avoided or that a trajectory that does not follow the center line may be more comfortable for the passengers. For planning the exact route inside a lane, the next two levels of information are required [147].

The lane model includes information such as the geometry of lanes including their boundaries, pedestrian ways, parking spaces, traffic signs, and many more. This information stored in HD maps can help the vehicle perform certain maneuvers like avoiding obstacles without endangering other traffic participants or parking the vehicle in a parking lot. Additionally, some shortcomings of perceptual sensors in areas on the road that are occluded or outside of the sensor range can be overcome with such detailed road information [124]. The lane model is visualized in Figure 2.1 (middle).

In the localization model static objects such as landmarks that can be detected by the sensors located of an autonomous vehicle are stored. This way, the vehicle can compare landmarks that are detected by its sensors with those stored in the HD map. Therefore, it is possible to determine the vehicle position with increased accuracy [41]. It can be differentiated between sparse and dense localization models. While a sparse model contains few landmarks at irregular distances, a dense model has a detailed representation of the environment at each point. In order to achieve such a dense representation often dense localization, models are created from high-resolution LiDAR point clouds [92]. An example for a dense localization model is shown in Figure 2.1 (right).

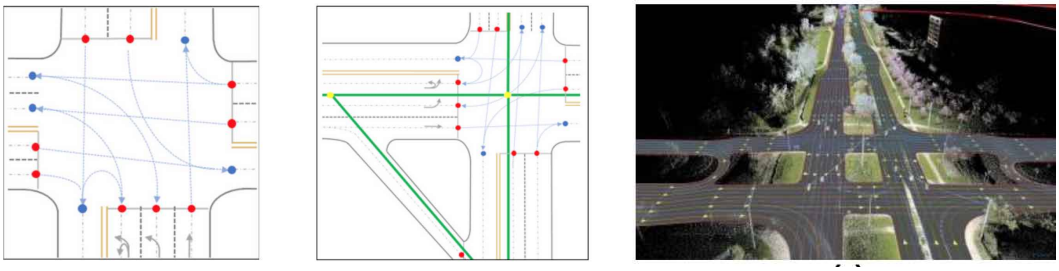

Figure 2.1: Left: Example of a lane model (source: [124]). Middle: Example of a road model (source: [124]). Right: Example of a dense localization model (source: [92]).

2.9 Trajectory Data

A trajectory describes the movement of an object over time. This can be defined by different sets of parameters, but is often described by a vector containing a set of coordinate points and a timestamp. For 2D coordinates the vector then can be represented as $p = (x, y, t)$. A trajectory is defined by a series of such points $p_1, p_2, ...p_n$ [10].

In this case, the coordinate points provide information about the location of the object, whereas the timestamps give insights to the temporal order in which the object positions are reached. The combination of the geographical and temporal components shows the direction in which the object is moving or the type of maneuver it is carrying out [167]. In Figure 2.2 an example trajectory is shown for a vehicle performing a left turn.

Figure 2.2: Example of a trajectory (source: own illustration based on [2]).

2.10 Simulation Environments

Currently it is common practice to test systems or processes in a simulation environment before deploying to the real world. Simulation environments are software based environments that mimic the real world as accurately as possible or needed. They can be used to test a virtual copy of a system or process and give a realistic feedback depending on the system's actions [87].

Motivations for using a simulation environment are saving costs because only the software for an autonomous system must be developed. Testing can then be carried out using a digital twin in a simulation environment without the need to acquire costly hardware like a vehicle. Another benefit is safety. If an autonomous system fails unexpectedly, this can lead to an accident, affecting not only the ego vehicle but also other traffic participants, causing personal and material damages. This means, by using a simulation environment, costs can be reduced significantly, while guaranteeing safety [122].

2.11 Reality Gap

There is a risk that programs that work well on simulated robots will fail completely on real robots. Due to differences in sensing and actuation, it is very difficult to model the actual dynamics of the real world [87]. The differences between a simulation and the real world are referred to as the "reality-gap". The reality gap prevents solutions developed in a simulation environment from performing well when deployed in the real world [43].

Robotics experts have repeatedly warned of the dangers associated with oversimplified, invalidated robot simulations. Many simulations are highly abstracted computer models rather than carefully constructed models of real robots. While these abstract models can be very useful for exploring some aspects of the problem of controlling autonomous agents, one must be very careful about drawing conclusions about real-world behavior from them. If the limitations of simulation models are not recognized, they can lead to both the study of problems that do not exist in the real world and the ignoring of problems that do exist in reality [87].

The effects of the "reality-gap" can be minimized by selecting a simulator that models the required physics appropriately [43].

2.12 Deep Learning

Nowadays, many important tasks such as speech recognition, visual object recognition, object detection, and many others have been solved by machines using methods of artificial intelligence. Specifically, the area of Deep Learning is mainly responsible for solving many of these problems with state-of-the-art performance [116].

Deep learning is a form of machine learning that enables computers to learn from experience and make decisions without the need to formally define rules of decision-making or specify the knowledge needed by the computer [73]. In Deep Learning, the information extraction process is ordered in a hierarchical scheme of multiple layers which enable the

machine to learn complex concepts by building these out of many simpler concepts. Because this architecture is composed out of many "stacked" layers, it is referred to as "Deep" Learning [194].

In order to "learn" the intricate structure in the data, Deep Learning uses the backpropagation algorithm to indicate how the machine should change its internal parameters to achieve the desired output [116].

Compared to classical algorithms deep learning has the disadvantages that large amounts of training data are necessary to achieve good results. Also, depending on the network size it can take weeks to completely train a model. Reasons for this are the non-linear optimization technique as well as the large amount of trainable model parameters.

2.13 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a form of deep learning that use bio-inspired convolutional operations to extract information. This type of network was originally introduced to extract information from visual data such as pictures and is still mainly used in this field [117].

The use of convolutional operations has many benefits over traditional algorithms and other deep learning network architectures, as less trainable parameters are needed and the overall model robustness to invariance in the data is increased. These benefits lead to faster training and the need of less training-data. Additionally, fewer data preprocessing steps are required [115]. Overall CNNs perform better compared to both traditional algorithms and other deep learning approaches and are currently the state of the art for image processing [173].

2.14 Long-Short-Term-Memory Networks

Long-Short-Term Memory (LSTM) networks are a special form of recurrent networks and were first introduced by Hochreiter and Schmidhuber (1997) [83]. Recurrent networks are designed to capture temporal dependencies in time-series data.

Classical recurrent network architectures have been prone to the vanishing gradient problem, which limits them to only capturing relatively recent (short-term) time dependencies [83]. This means that potentially important information which occurs at the beginning of a data sequence hardly gets respected when calculating the output of the network [74]. In contrast to this the LSTM network is able to capture both long-term and short-term dependencies due to its specific architecture which improves the gradient flow when training the network. The classic LSTM propagates both the cell state C_t and hidden state h_t to the next memory cell which is composed of a forget- and an update gate [83]. The components of the LSTM cell are pictured in Figure 2.3.

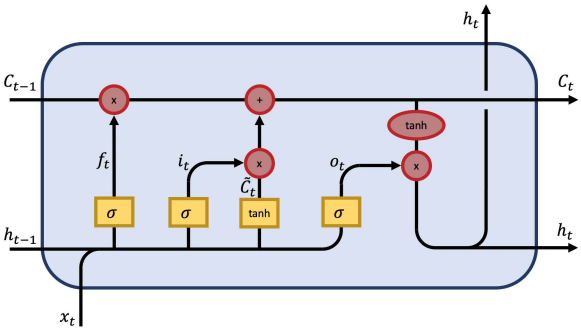

Figure 2.3: LSTM cell as described by Hochreiter and Schmidhuber (1997) (source: [83]).

2.15 Data Augmentation

In the field of Machine Learning, data augmentation refers to a method of reducing the variance of a model in its decision making [60]. In other words, the augmented data acts as a form of regularization and therefore often leads to an increase in model performance [174].

Deep Learning models rely on a large amount of data samples to avoid over-fitting. It is common problem that in many domains the acquisition of data is complicated or very costly, resulting in limited training data. As a result, the trained model could generalize poorly and could have a low performance. [163]

When applying data augmentation, an increased model performance is achieved by enhancing the size and quality of a dataset. By performing various transformations such as geometric rotations or applying filter masks, the dataset becomes larger and more versatile. Because of the larger amount of training data and the coverage of more variations, the chance of over-fitting is reduced [193]. Figure 2.4 shows an example of an image that is augmented by rotating the image by various angles.

Over the past years it has proven that data augmentation improves the performance of Deep Learning models in the field of image classification [47], text-to-speech [114], and time series classification [60].

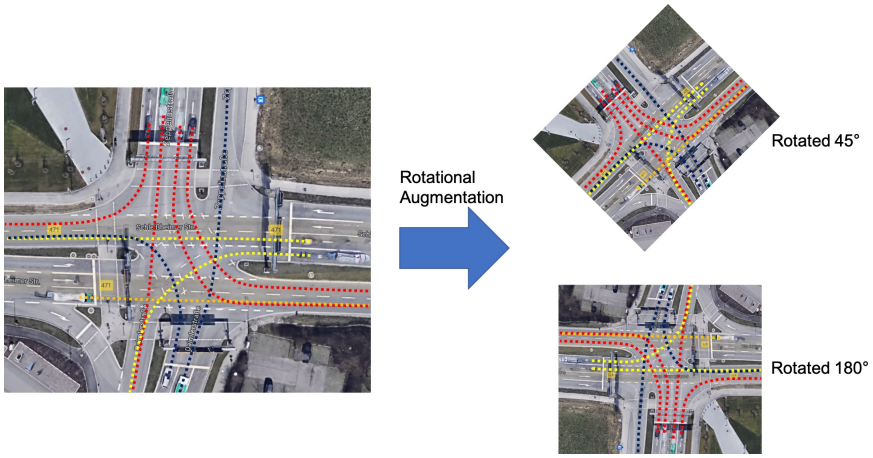

Figure 2.4: Example for a rotational augmentation of trajectory data (source: own illustration, [2]).

2.16 Robot Operating System (ROS)

The Robot Operating System (ROS) is an open-source operating system for various types of robots. It is not an operating system in classical sense of planning and scheduling tasks, but rather provides a communication layer on top of the host operating system of a heterogeneous computer cluster. The ROS layer allows for a simpler and more standardized communication between software and hardware components [151].

The main benefits of ROS are the modular approach, which enables the user to integrate multiple software and hardware systems easily. This makes it easier to reuse software originally written for individual tasks and integrate it into a complete robotic system. Also, ROS enables a more efficient debugging of individual task specific software components. For example, a robot that combines the tasks of computer vision and path planning to navigate its environment can assign each of the two tasks to an individual node. With ROS each of these nodes can then be debugged separately and eventually integrated into the robotic system [151].

Chapter 3

Related Work

3.1 Safety Assessment Approaches for Automated Vehicles

The verification of a systems safety is a critical aspect that must be addressed before deployment. This process is meant to detect any hazardous behavior from functional insufficiencies. Especially in the context of cyber-physical systems many approaches for the safety assessment of these systems exist. Riedmaier et al. [154] give an overview of seven relevant safety assessment approaches evaluating the driving capabilities of autonomous vehicles. These approaches are scenario-based, function-based, shadow mode, formal verification, real-world testing, staged introduction of AVs, and traffic-simulation-based. They are visualized in Figure 3.1.

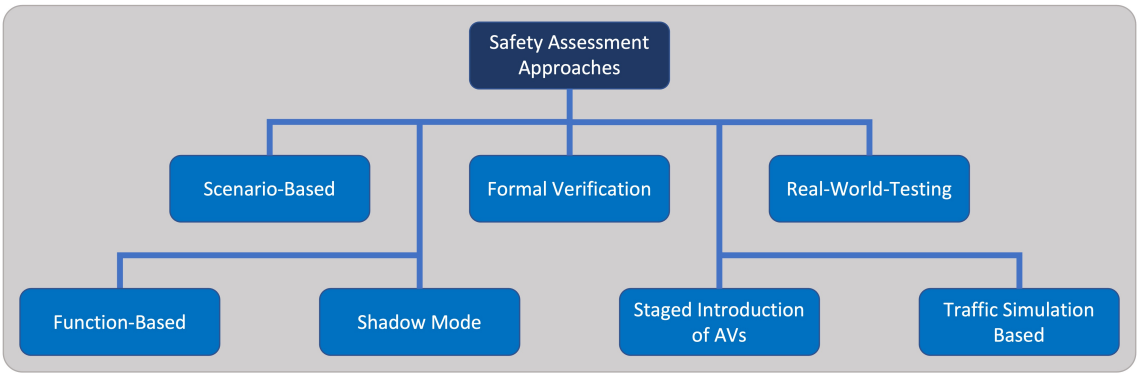

Figure 3.1: Overview of safety assessment approaches for automated vehicles (source: [154]).

In the following a critical review of each of the before mentioned safety assessment methods for automated vehicles is given.

3.1.1 Scenario-Based Approach

Because it has the potential to deliver an efficient and reliable safety assessment, the SBA is the most promising of the strategies currently reported in the literature [154]. In section 3.2, the approach is presented in depth. Only a brief overview is presented here in order to provide a clear distinction and differentiation from the other approaches.

A scenario is, by definition, a series of actions and events [181]. When examining a typical highway ride, for example, there is a significant amount of time when no actions or occurrences occur. The scenario-based approach, which is also used in research projects

(e.g., in Germany [84], Japan [16], and Singapore [36]), skips areas of the test that have no notable actions, thus reducing the test's scope. Furthermore, common situations that do not provide crucial information for the safety evaluation can be disregarded [139]. Such situations could for example include cut-in maneuvers with large relative distances where the leading vehicle's which performs the cut-in has a higher speed. In such a driving situation no action is required from the evaluated vehicle in order to maintain traffic safety. Therefore, the example could be excluded from the assessment process. However, it is still a challenge to determine which situations must be included in scenario-based assessments and where they should be located in order to make a meaningful judgment about the vehicle's safety [155].

The scenario-based approach is used to explore and microscopically evaluate individual scenarios. After a large number of these microscopic evaluations have been completed, the results can be used to derive a macroscopic safety statement.

3.1.2 Formal Verification

Formal verification is a mathematical method for formally demonstrating the safety of systems throughout the entire operational design domain (ODD). The verification is based on a mathematical model of the system under test. Additionally conditions are defined that determine the correct operation of the system and its components. If all defined conditions hold the system is validated. With this approach it is ideally possible to achieve a full coverage of the systems operational domain [96].

Formal verification is not based on driving scenarios, and therefore is not included in the SBA.

The disadvantages of this technique include both, that simplifying assumptions must frequently be made. Intel and MobilEye [161] for instance, make three strong assumptions:

1. All sensor inputs are correct
2. An exhaustive legal foundation for the domain of traffic will exist
3. Code can be generated in such a way that it can be formally assessed

This infers that only subsystems of the automated vehicle are addressed, rather than the entire vehicle. Compared to other approaches, this is a clear downside of formal verification based on the mentioned assumptions. Therefore, formal verification may be a promising technique in the future for a separate validation of various sub-systems.

3.1.3 Real-World Testing

At increased levels of automation, a distance-based evaluation of safety through field operational testing (FOT) is no longer economically practicable. According to [97], 11 billion miles would have to be driven in the United States to conclude with sufficient certainty that automated vehicles exceed the safety level of human drivers by 20 percent. In this case, surpassing indicates that there are fewer deadly accidents. Similar statistical studies have been conducted in Germany, with Wachenfeld and Winner [185] concluding that a motorway chauffeur requires around 6.6 billion test kilometers. These two studies are backed by [202] which find that it takes 38 years of driving to be involved in a situation leading to

a noteworthy traffic accident and 6877 years of driving to be involved in a fatal accident. Testing in the actual world is the norm when there is a low degree of automation. However, from automated driving level 3 onwards, the operational driving domain grows drastically. Whereas driver assistance system cover few very limited use-cases, a fully automated vehicle must perform well in all possible driving situations. At this point real-world testing is no longer economically possible. This is due to the transfer of all driving tasks and responsibilities to the automated vehicle [171].

3.1.4 Function-Based Approach

In function-based testing, the system's functions are individually tested on the basis of requirements. This is a common Advanced Driver Assistance System (ADAS) technique. Current ISO standards (e.g., ISO 15622 for Adaptive Cruise Control (ACC) [168]) and UNECE regulations (e.g., UNECE R131 for Advanced Emergency Braking Systems [58]) for ADAS use this method, defining a series of precisely specified tests for individual systems that check basic functionality and thus ensure a minimum level of safety. On the one hand, the decreased testing effort is achievable with ADAS since the systems' functional scope is limited, while on the other hand, the driver is required to constantly monitor the system [85].

3.1.5 Shadow Mode

Wang and Winner [186] describe a system known as shadow mode, in which the autonomous driving function in production cars is conducted passively. The driving function has access to the sensors' real-time inputs but not to the vehicle's actuators. Simulation may be used to assess the autonomous driving function's judgments and, as a result, the level of safety. Car manufacturers, such as Tesla, utilize a similar method to test new systems or new versions of old ones [149].

However, because other road users also plan and execute their actions depending on the activities of the AV, the conduct of the possible conflict partner (other traffic participants) in the simulation does not correlate to reality. If the passive driving function in a circumstance makes a different decision from the vehicle's actual (active) driving function or the human driver, another traffic participant may have made a different decision, and the simulation's findings are only partially valid.

3.1.6 Staged Introduction of Automated Vehicles

The goal of the incremental introduction of automated driving functions is to lower the vehicle's operational driving domain. Consequently, the number of traffic scenarios that the automated vehicle must be able to navigate safely, is strongly reduced. This allows for a cost-effective safety evaluation based on real-world testing. Driving on a certain portion of a road for a few hundred meters or kilometers only when visibility conditions are excellent, is an example of a highly limited driving domain. The safety approach also includes a certified safety driver who may act instantly if the system makes a mistake. If the vehicle is determined to be safe in this driving domain, the domain can gradually be enlarged and possibly the safety driver can be removed [148].

This strategy might be useful for introducing Level 4 cars in a certain urban environment, for example. However, according to the Society of Automotive Engineers (SAE) it is not acceptable for the validation and certification of Level 5 systems in practice, because these systems have a limitless driving domain by definition [157].

3.1.7 Traffic-Simulation-Based Approach

The idea of traffic simulation is to model not just one situation, but a whole road network with hundreds of traffic participants (so-called agents). As a result, this technique is particularly well suited to determining the safety of automated vehicles at a macro level. It is also possible to simulate and examine the effect of automated vehicles on human traffic participants here. It may also be calculated how the chance of scenarios occurring varies as a result of the introduction of automated vehicles, as well as the impact an increasing number of automated vehicles has on total traffic.

Because the full ODD can be replicated in traffic simulation, the technique based on traffic simulation may be utilized to improve the efficiency of the stepwise introduction of automated vehicles. This technique is no longer practicable for Level 5 systems, just as it is for the phased introduction. [100, 156, 159] provide further information on this subject.

3.2 Scenario-Based Approach

In section 3.1 a brief overview of the seven different methods for the assessment of automated vehicles was given. This work strongly focuses on the scenario-based approach, since it has been found to be the most promising approach. Therefore, the scenario-based approach is discussed in detail in this section.

3.2.1 Process Overview

The main challenge of the scenario-based approach is to identify a set of driving scenarios that allow for an adequate safety assessment of automated vehicles. To achieve this it is key to cover all possible driving scenario types. Ideally also variations of each scenario are included in the test set [80]. Ponn [148] derives a workflow for the scenario-based approach from existing literature and major research projects [16, 84]. The entire workflow is shown in Figure 3.2.

The workflow consists of several stages that start with the acquisition of driving scenarios. From there on the scenarios are classified into categories and stored in a scenario data-base. The scenario data-base plays the central role in the process of evaluating automated vehicles. Here all driving scenarios that can be utilized for testing purposes are collected. New scenarios can be added to the data-base anytime and test cases can be formed from the scenarios stored here. For the validation of automated vehicles, proper driving scenarios are selected to form test cases. The test cases can then be executed in real-world traffic or in a simulation environment. Finally after completing the test cases the performance of the automated vehicle is evaluated. In the following sections each category of the workflow is discussed in detail.

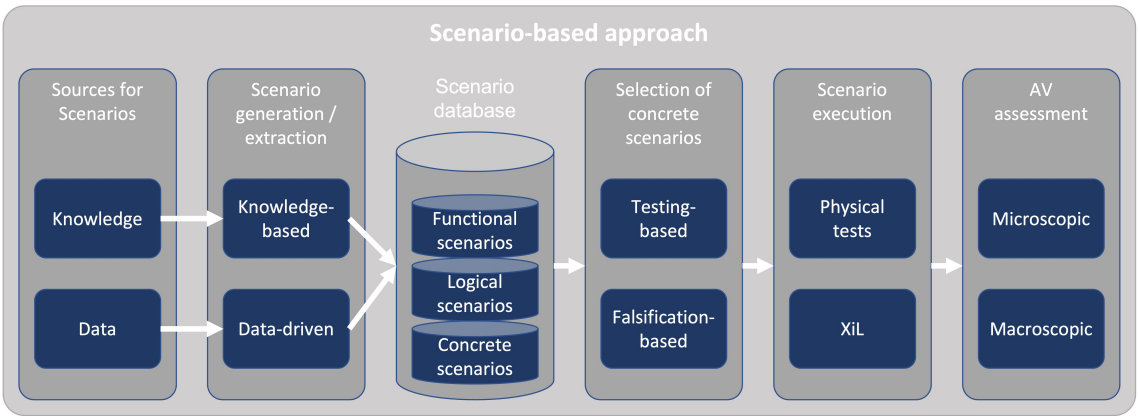

Figure 3.2: Description of the workflow for the scenario-based approach (source: [148]).

3.2.2 Sources for Scenarios

There are two main sources for scenarios. These can be knowledge in various forms or data driven. In the first case knowledge is usually referred to as expert knowledge of how a driving scenario is structured and which components must be included. However, also road and traffic guidelines or experience from existing traffic statistics can be included [96].

The data driven sources for scenarios are usually detailed datasets created from real-world traffic situations. To be able to use data driven sources for scenarios the availability of such datasets is crucial. These can either be created independently or acquired from publicly available sources [68]. In recent years, a number of institutions have made publicly accessible datasets available. [76, 98] provide a summary of the existing datasets. Zhu et al. [204] also present an overview of datasets and seek to integrate them.

Krajewski et al. [108] show a new way for gathering real-world driving data. In their approach a drone is used to record traffic, and computer vision techniques are then used to extract the trajectories of individual road users. This technique has the main benefit of requiring no costly and time-consuming test cars with complex sensor equipment to be set up, as well as ensuring that the recording of data has no impact on traffic. A downside of the approach is that only a limited road sector of 420 meter length is observed.

Similarly in the Providentia++ project traffic data is recorded from highway, rural, and urban road areas. This is done by adding multiple sensor-systems composed of camera-, LiDAR-, and radar-sensors to the road infrastructure. The sensor systems are mounted on gantries or custom masts. A major benefit of this system is that once installed it records the traffic 24 hours a day. This way it is more likely to record rare traffic events which again are interesting scenarios for testing and validating automated vehicles [131].

3.2.3 Scenario Extraction and Generation

In the previous section 3.2.2 the two main sources for the creation of driving scenarios were discussed. These are either knowledge based or data driven. Depending on the source different approaches for the generation and extraction of scenarios exist. These approaches are visualized in Figure 3.3.

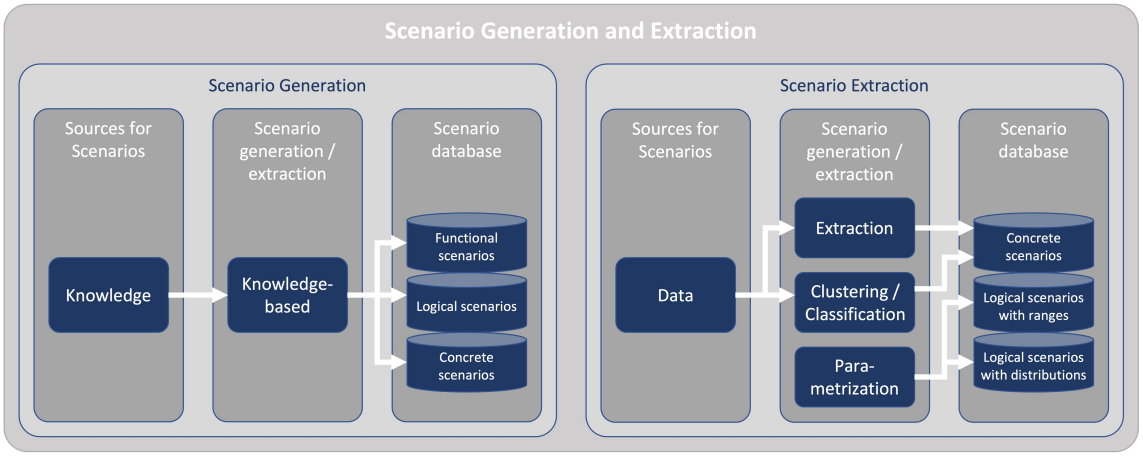

Figure 3.3: Knowledge-based scenario generation and data-driven scenario extraction methods (source: [148]).

3.2.3.1 Knowledge-Based

The systematic transfer of information into scenarios is the core principle of knowledge-based scenario design. To put it another way, abstract data is utilized to create functional, logical (with parameter ranges), or directly concrete situations.

As a knowledge base, things like road traffic legislation, regulations, accident data, consumer testing, ethics guidelines, safety analysis methodologies, or expert information may be utilized. Expert knowledge is frequently used to generate scenarios from existing knowledge-based sources. Ontologies are often utilized to store and arrange expert knowledge in this manner [23, 40, 71, 102, 123, 197].

3.2.3.2 Data-Driven

The data-driven creation of driving scenarios can be split in three main approaches. These include the direct extraction, classification, and parametrization of driving scenarios. Often but not exclusively these approaches rely on machine learning methods.

Recorded traffic data is used as a basis for the scenario extraction method. Hereby information about vehicles and actors including their trajectories is extracted. This information is then processed to form concrete driving scenarios based on real-world traffic situations. Corner-case scenarios can then for example be found by detecting very unique features in a scenario [113]. Alternatively, corner-case scenarios can also be determined based on the difficulty of predicting the traffic participants' behavior [31]. This method can also be used to generate new concrete driving scenarios based on the recorded traffic data [88, 89, 109].

In order to categorize various driving scenarios clustering and classification methods are often used. While for classification methods the scenario-categories are known in advance, this is not the case for clustering methods. Classification methods usually build on supervised learning, while clustering is mainly done with unsupervised learning. Both methods have their individual benefits and are generally used for different purposes. Since in clustering approaches the scenario-categories are not defined in advance the approach can be used to find previously unknown scenario-categories. Classification approaches however have the benefit that the correct scenario-class is directly assigned. This can reduce the manual effort of labeling scenarios by their type.

In the literature different clustering methods are found. These include the use of similarity measures [111, 112], hierarchical agglomerative clustering [189], or procedures based on Bayesian models [188]. However, the preferred approach in the literature is scenario classification. The approaches using this technique are based on various metrics. These include the collision potential [191] or relative movement between two or more vehicles [56]. The classification algorithms are based on various machine learning techniques such as support-vector-machines, nearest-neighbors, or neural networks [22, 27, 51, 75].

As described in Section 3.2.3.1, logical scenarios are defined by various parameters. These parameters are bounded either by a specific range or distribution. Logical scenarios can be derived from real-world traffic recordings. For each scenario class the parameters to describe the scenarios are determined from the traffic recordings. The parameter bounds for each scenario class are then defined so that all scenarios of this class lie inside these bounds. In other words, the most extreme scenarios of each class mark the parameter bounds.

3.2.3.3 Scenario Database

In the scenario-based approach many different test scenarios are used evaluate the capabilities of an automated vehicle. Depending on the test cases different driving scenarios are of interest. For efficient management and selection of the scenarios of interest a scenario database is required. The scenario database represents the core element of the scenario-based approach. Here all driving scenarios are stored and can be selected depending on the user's needs.

The PEGASUS[1] project [150] provides an interface for inserting various data sources and processing them into a standardized format. Within this project a database for highway scenarios is created [84, 150]. Another framework including a scenario database is the Commonroad project [13]. The framework uses driving scenarios to assess the performance of trajectory planning algorithms.

3.2.3.4 Selection of Concrete Scenarios

Before the evaluation of an automated vehicle is possible, an adequate selection of concrete driving scenarios must be made to form the test cases. These scenarios can be selected from the scenario database and should validate the safety of the automated vehicle. One difficulty is dealing with the large number of different driving scenarios each of which unlimited variations exist. For a practical assessment however, only a manageable number of driving scenarios can be included in the test cases. Ideally the operational safety is proved with the minimal number of test cases. To achieve this, a technique for the proper selection of concrete driving scenarios is necessary. In the literature two main approaches exist for the selection of concrete driving scenarios. These are the testing-based and the falsification-based approach. The two approaches follow individual strategies for guaranteeing the fulfillment of all requirements. In the testing-based method the proper functionality of an automated vehicle is validated if a subset of the driving scenarios is completed while fulfilling all requirements. In contrast the falsification approach seeks to find specific driving scenarios where the automated vehicle does not fulfill all requirements.

[1] PEGASUS stands for: Projekt zur Etablierung von generell akzeptierten Gütekriterien, Werkzeugen und Methoden sowie Szenarien und Situationen zur Freigabe hochautomatisierter Fahrfunktionen (PEGASUS)

Testing-Based Selection

In testing-based approaches the driving scenarios are categorized and selected based on different key performance indicators (KPIs). KPIs can be the occurrence of certain driving maneuvers, accidents, criticality measures, or guidelines and standards. There are various approaches to select a representative subset of driving scenarios from the scenario database. The selection techniques follow two main goals:

1. High test case coverage of operational driving domain

   The first goal of the sampling technique is to ensure that driving scenarios of all types are adequately represented in the test set. This ensures that AVs are assessed in all kinds of driving situations including rare ones like accidents. To ensure the safe operation of the AV under assessment, it is important to include especially critical driving scenarios in the test cases. Critical driving scenarios contain demanding traffic situations where the AV is likely to fail. The identification and selection of such driving scenarios is one of the key challenges in the selection process [22, 25, 65, 80, 138].

2. Representative statistical traffic safety evaluation

   Another important goal of scenario-based testing is the creation of a representative statistical traffic safety evaluation of the assessed AV. A major difficulty is the correct selection of certain driving scenario types in order to achieve a realistic safety evaluation comparable to real-world testing. To achieve this the amount of driving scenarios of each scenario type must be selected based on its probability of occurrence in real-world traffic [17, 94, 97, 195].

Depending on the goal of the scenario selection strategy different approaches have already been addressed.

Falsification-Based Selection

In contrast to testing-based selection the falsification-based approach tries to identify driving scenarios in the ODD where the assessed AV fails to meet all performance requirements. To find such driving scenarios different approaches are found in the literature. They are summarized in the following.

1. Accident Data

   One approach to determine critical driving scenarios that an AV may fail is using scenarios based on real-world traffic accidents [63, 169, 170]. Some approaches additionally create variations of existing accident scenarios to achieve a higher test-case coverage [59, 165]. The underlying assumption is that driving situations that are demanding for human drivers are also demanding for AVs. This may be true for some cases. However, AVs operate very differently to human drivers and therefore the assumption made is likely not to hold for all accident scenarios. Additionally this approach does not cover the ODD of the AV sufficiently as other non-accident driving scenarios are not included in the testing process.

2. Critical driving scenarios

   Another approach is based on the use of critical driving scenarios. These are scenarios that especially incorporate relevant and demanding driving situations. In order to obtain such scenarios, two main techniques are found in the literature. The first technique evaluates driving scenarios based on various parameters. These are usually the

occurrence of certain driving maneuvers and other criticality measures like the distance or the time to collision (TTC) to the lead vehicle. [145] follows this approach.

The second technique generates critical driving scenarios. Here scenarios are modified to increase their criticality. This is achieved by applying an optimization algorithm to the driving scenarios. The technique is applied in [14, 101].

3. Complex driving scenarios

   Mining or creating complex driving scenarios is another popular approach to generate test cases. Complexity is a measure that defines how difficult it is to navigate a driving scenario safely. In general with a higher level complexity both human drivers and AVs have more difficulties to complete the driving scenario. To determine such scenarios, often general performance criteria are used [65, 199, 200]. These criteria determine the complexity level of a scenario and include various factors like weather conditions, illumination, road geometry, behavior of other traffic participants and more [76, 103]. Based on these factors and their extent the overall complexity of a driving scenario can be determined.

4. Simulation-based falsification

   The goal of simulation-based falsification is to efficiently find driving scenarios where the AV fails by using an optimization strategy. For this the performance of the AV is evaluated in driving scenarios based on various KPIs. The results of the KPIs are then used to determine how close the AV is to failing the specific driving scenario. Based on this information the next driving scenario is selected specifically to increase the chance of the AV failing [148].

   In the literature many different optimization strategies for simulation-based falsification exist. In the following an overview is given.

   (a) Reinforcement learning [45, 104, 118, 119]
   (b) Differential evolution genetic optimization [26, 29]
   (c) Particle swarm optimization [26, 29]
   (d) Adaptive search-algorithms [134, 135]
   (e) Bayesian optimization [64]
   (f) Random forest models [138]
   (g) Simulated annealing [6, 176–179]
   (h) Gradient descent optimization [180]
   (i) Forward / backward search [105]

3.2.3.5 Scenario Execution

When the test cases are determined the selected concrete driving scenarios are executed. This can be done in various surroundings including real-world or simulation-based environments. Real-world testing is typically conducted as field tests i.e., on public roads or as test site tests. Simulation-based testing is done in a virtual environment that typically mimics the physics of the real world including the automated vehicle and other traffic participants. Using simulation environments for testing purposes has two main benefits. Firstly, the costs for testing can be reduced drastically as no hardware such as a vehicle and sensor system are

necessary. Secondly, no safety hazards can occur if the system does not behave as intended. Various commercial simulation tools are available. Also, in the literature different simulation frameworks are mentioned [5, 77, 99].

3.2.3.6 Automated Vehicle Assessment

The assessment of automated vehicles is done microscopically or macroscopically. The safety of a vehicle however can be determined best in a microscopic test environment. For this it is beneficial to use various gradual performance indicators rather than binary ones. A gradual performance indicator for instance is the time-to-collision [81], which exists in various forms [95, 160, 184]. It is a measure for the criticality of a maneuver based on the distance and differences in velocity and acceleration of the automated vehicle to other vehicles. Other criticality measures are compared in [127]. Binary performance indicators only indicate if a driving scenario is completed successfully or not. A downside of binary performance indicators is the lack of a detailed driving style analysis. For example, an automated driving function could be regarded as safe because all driving scenarios are completed successfully. However, the automated vehicle could complete the scenarios in an unsafe driving style. This would not be reflected in the evaluation.

The microscopic assessment of an automated vehicle in critical driving scenarios can be used to draw conclusions over the macroscopic behavior of the vehicle. In [94] the transition from many microscopic to a macroscopic vehicle assessment is discussed.

The previously mentioned scenario selection methods testing- and falsification-based both microscopically evaluate the safety in each of the driving scenarios. Hereby the testing-based method seeks to achieve a high stochastic coverage of the automated vehicle's driving domain. The falsification-based method concentrates on identifying corner case driving scenarios, often with increased difficulty [148].

The macroscopic analysis aims to evaluate the compatibility of an automated vehicle with the surrounding traffic situation. For this it is beneficial to cover a large number of scenarios in the operational driving domain to draw further macroscopic conclusions. In return this means that the testing-based approach is better suitable as it has a higher stochastic coverage of the driving domain. In contrast the falsification-based approach mainly includes corner cases therefore limiting the stochastic coverage of the driving domain [148].

3.2.3.7 Conclusion

The scenario-based approach shown in Figure 3.2 includes all major process steps for the evaluation of automated vehicles. The assessment techniques build upon a suite of driving scenarios from which test cases are formed. The driving scenarios are stored inside the database which represents the central element of the scenario-based approach. The main tasks involved in the approach are the generation of concrete driving scenarios and the appropriate selection of driving scenarios to form test cases and evaluate the system of the automated vehicle. The generation process deals with filling the scenario database with a large number of driving scenarios. It aims to form an exhaustive collection of driving scenarios from which test cases can be derived. The scenario selection process is then applied to reduce the number of driving scenarios in test cases while guaranteeing a complete assessment of the automated vehicle. The goal of the selection process is to find a minimal set of driving scenarios that covers all driving situations necessary to prove the correct functionality

of the automated vehicle.

Testing of automated vehicles is mainly done in two environments. Either in the real-word on test-tracks or as field tests on public roads or alternatively in a simulation environment. Real-world testing is time-consuming, subject to high expenses, and can potentially lead to safety hazards. For these reasons testing in a simulation environment is beneficial.

In total there are many promising solutions for each stage of the scenario-based approach. Many of these however are limited to specific areas of the approach. A big challenge remains the creation of a scalable implementation of the scenario-based approach.

3.3 Standards for Describing Driving Scenarios

In the following section the two standards OpenDRIVE (OD) [54] and OpenSCENARIO are presented. Both standards are created and maintained by the Association for Standardization of Automation and Measurement Systems (ASAM) [7] and are used for describing road networks and dynamic content such as vehicles maneuvers. These standards are widely used across the industry for developing and testing ADAS and AD [18]. The scenarios created as part of this work are based on the two standards OpenDRIVE and OpenSCENARIO.

3.3.1 OpenDRIVE

OpenDRIVE created by the ASAM is a standardized format to describe road networks. An OpenDRIVE file includes information about the geometry of roads, lanes, and objects. This includes road markings and traffic signals [14].

OpenDRIVE provides a road network description for the use in simulation environments to develop ADAS and automated driving functions. The main goal behind OpenDRIVE is to create a standardized format that can be used by different simulators without costly conversion steps. The road data can be created manually or originate from map data or scans of real-world roads [130].

An OpenDRIVE network is mainly defined by a reference line which is the basis of every road. Roads and lanes are both defined according to the reference line. Additionally, the position of objects such as traffic signs can be defined either by an offset from the reference line or by a set of coordinates in the global coordinate system of the road network. This is illustrated in Figure 3.4 on the left side. The blue line in the middle of the road segment is the reference line, and the lanes (blue and green) as well as the traffic signs are defined as an offset from this line in the s/t-coordinate system [19].

Additionally, the dependencies between different road segments in a traffic network can be defined. For example, two road segments that interconnect can be defined as a junction. On the right side of Figure 3.4 an example of how a junction is defined in the OpenDRIVE format is shown. The definition of dependencies between road segments can be important for understanding the driving logic of other vehicles when planning a route. With this information a vehicle knows when it is approaching an intersection or other critical road areas and can act accordingly [19].

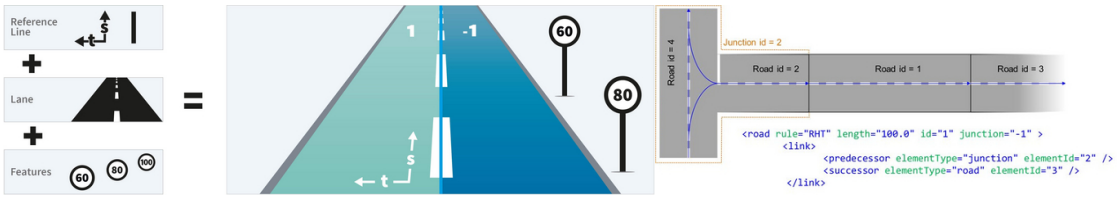

Figure 3.4: Description of a road network defined by a reference line in the OpenDRIVE format (left). Example of the junction definition in the OpenDRIVE format (right) (source: [136], [19]).

Finally, the lanes between road segments are also linked with each other corresponding to the driving logic. This is especially important in junctions where the reference lines of corresponding roads are not necessarily aligned with each other [19]. This is illustrated in Figure 3.5.

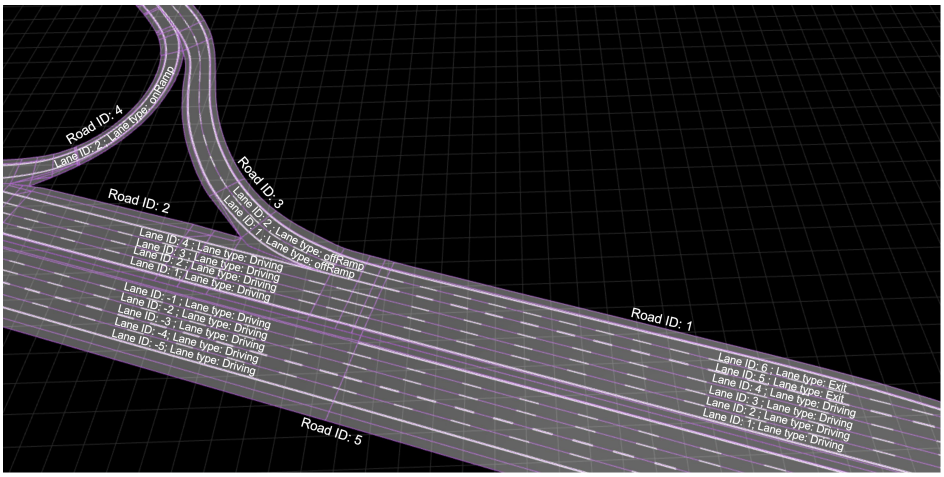

Figure 3.5: Illustration of the road linkage across a junction in the OpenDRIVE format (source: [19]).

3.3.2 OpenSCENARIO

The OpenSCENARIO standard created by the ASAM is a file format meant for the description of dynamic content in a driving scenario. This includes complex maneuvers that can include multiple traffic participants such as vehicles, pedestrians, and others. There are two ways of describing the movement of each participant. One way is to script the movement by adding driver actions like performing a lane change or making a turn. Another possibility is to use recorded trajectories that define the movement of each traffic participant over time. Other components such as environment conditions, traffic, pedestrians, and more can also be specified [20].

The vehicle maneuvers are defined as stories that describe the driving maneuvers of a single vehicle. A story can contain acts which tell a traffic participant to perform a certain maneuver whenever a defined condition is met. Conditions can be, for example, exceeding the allowed speed limit, approaching a leading vehicle at a certain distance, and others. The use of sequences then allows multiple vehicles to perform maneuvers as a reaction to a condition met [20].

The exact driving behavior is described by events and actions. Events refer to a maneuver

which occurs, while actions define which kind of maneuver is carried out. Actions not only include vehicle maneuvers such as accelerating, braking, or changing lanes, but also include environment changes like a traffic light switching from red to green. Actions can also be defined as routes or trajectories a traffic participant should follow [20].

ASAM has released two versions 1 and 2 of the OpenSCENARIO standard. Version 1 was first released in the beginning of 2020 while version 2 was released in December 2021. Both versions basically share the same features. The main difference of the two version lies in the addition of a more detailed set of actions and attributes for the relevant simulation models. This allows for a more comprehensive scenario description. Additionally, OpenSCENARIO version 2 uses a domain specific language for the description of driving scenarios. In the future ASAM plans to merge the two versions together. At the time of this work only OpenSCENARIO version was released [20]. Therefore OpenSCENARIO files are generated in this version and it is referred to version 1 when mentioning OpenSCENARIO.

3.4 CARLA Simulator

CARLA Simulator is an open-source simulator developed for performing research in the domain of autonomous driving. More precisely, it supports the development, training, and validation of autonomous systems in urban environments. CARLA Simulator already provides some building blocks like cars, buildings, urban layouts, and more for creating realistic driving scenarios. Also an autonomous vehicle can be simulated with different sensor suites affecting what the vehicle perceives [53]. An example for a CARLA Simulator scenario from the Providentia++ test stretch is pictured in Figure 3.6.

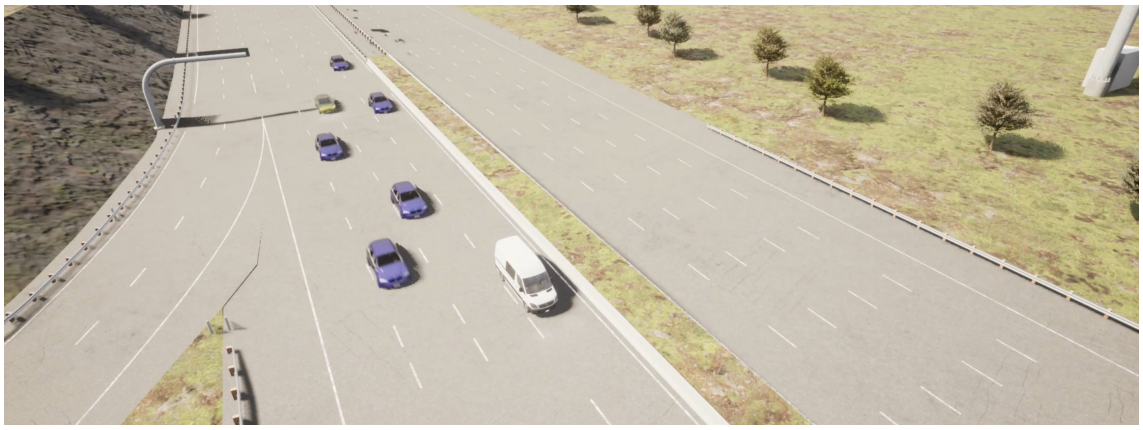

Figure 3.6: Example of a traffic scenario from the Providentia++ test stretch simulated with CARLA Simulator (source: [136]).

In this work scenarios are generated according to the OpenSCENARIO standard and can then be simulated with CARLA Simulator to achieve graphically realistic traffic scenarios. These can then be used for training Deep Learning models on tasks like object detection, image segmentation, and more.

3.5 ScenarioGeneration

"ScenarioGeneration" [192] is a Python package designed to enable a simple creation of OpenDRIVE and OpenSCENARIO files. The package contains the two modules "xodr" which handles the creation of OpenDRIVE road networks and the "xosc" module for defining driving scenarios in the OpenSCENARIO format. Since an OpenDRIVE map for the Providentia++ test stretch has already been created in previous works, only the driving scenarios are created in this work. Therefore in the following only the "xosc" module is explained in detail.

The "xosc" module addresses the OpenSCENARIO related parts and covers all parts of OpenSCENARIO version 1.0 and most of OpenSCENARIO version 1.1 The module is an Extensible Markup Language (XML) file generator that allows the user to efficiently create a complete OpenSCENARIO hierarchy without having to explicitly define all abstraction levels.

This module is intended to provide a better accessibility to the components of OpenSCENARIO without confronting the user with all the XML levels that do not contain essential information for the user. Therefore, some of the XML elements contained in the OpenSCENARIO file do not appear in the class structure, but are compressed to a level where the user can set the required parameters.

XML files for both versions 1.0 and 1.1 of OpenSCENARIO can be generated by the module.

3.6 ScenarioRunner

The ScenarioRunner module allows to define traffic scenarios and their execution within CARLA Simulator. To define scenarios, they can either be imported from the OpenSCENARIO standard or created using a Python interface. Complex traffic scenarios can be created with ScenarioRunner in order to test an autonomous driving agent before deployment in real-world traffic [35].

To create a scenario from scratch with the Python interface, the ego vehicles and their starting position, as well as the simulation world, must be initialized. All other vehicles and their trajectories can be defined within an XML file and are then loaded into the scenario [35].

3.7 Scenario Labeling

The proAnno[2] labeling tool is part of the Providentia++ project and is utilized to annotate data recordings with labels. The tool is specialized for the domain of autonomous driving and the Providentia++ test stretch. With proAnno it is possible to label bounding boxes, vehicle and road user types, vehicle IDs, and more [136]. An exhaustive list of label types that can be annotated with proAnno is shown in Table 3.1. Figure 3.7 shows proAnno and how bounding boxes are visualized by the tool.

[2] proAnno is based on the 3D Bounding Box Annotation Toolbox [205]

Label type	Explanation	Tasks for which label is used
2D/3D bounding box	A box that encloses an object inside itself. Marks the size and position of an object in 2D or 3D, depending on the bounding box type. Additionally, to the object position, the object orientation can also be defined.	Object detection, Tracking, Trajectory prediction
Road user	Type of road user. Examples are car, truck, pedestrian, bicycle, motorcycle, van, special vehicle.	Object classification
Vehicle ID	An ID unique for the same vehicle across multiple data frames	Tracking, Trajectory prediction

Table 3.1: Label types included in proAnno (source: [136]).

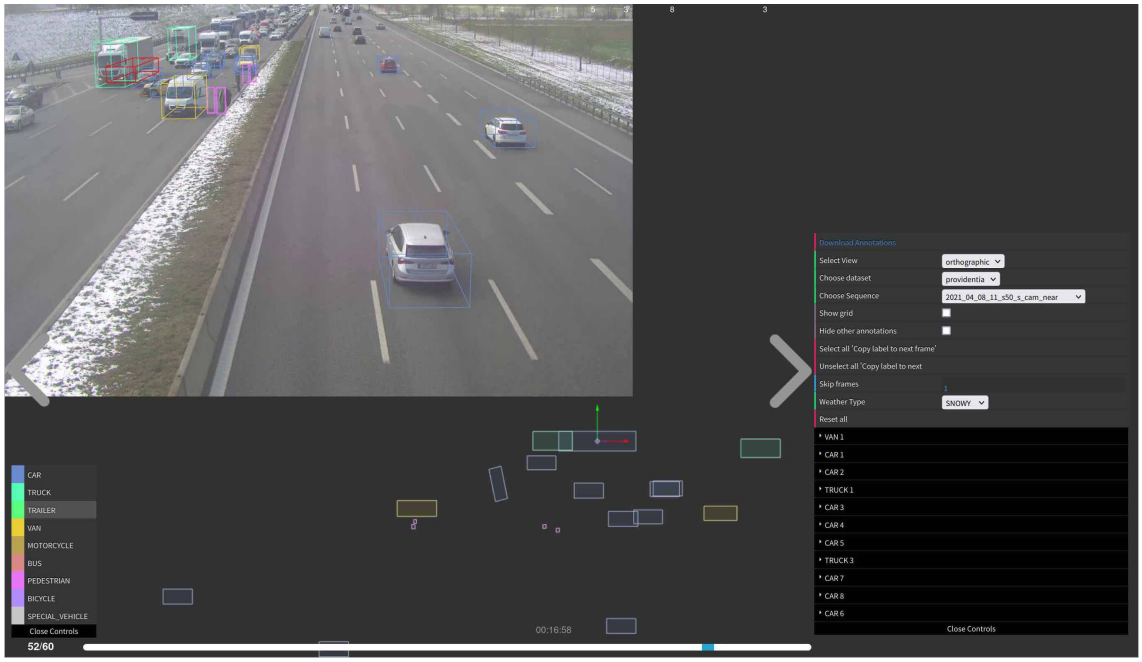

Figure 3.7: Image of the labeling tool proAnno (source: [136]).

In the above mentioned state, the tool can create labels for the training of Deep Learning algorithms on the tasks of object detection and classification and trajectory prediction [136]. In the course of this work, proAnno was extended to support the creation of labels for semantic scene understanding which refers to traffic scene understanding and maneuver prediction in case of the Providentia++ project.

3.8 Knowledge Discovery in Databases (KDD)

Knowledge Discovery in Databases describes a non-trivial process in which potentially useful information and insights can be discovered from collected data [62]. It consists of several phases which are run through one after the other in order to ultimately achieve new insights from the originally collected data. An overview of this is shown in Figure 3.8. The KDD process can include numerous iterations and have loops between any two process steps [62].

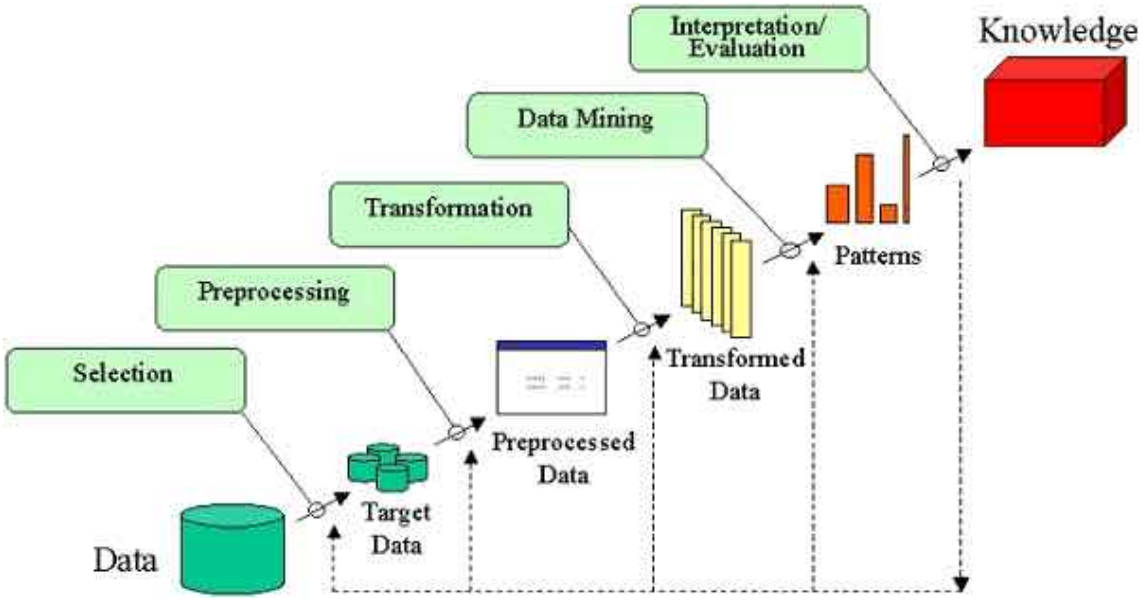

Figure 3.8: Overview of the KDD-process (source: [62]).

In the following, important steps of the KDD process according to Brachmann and Anand [32] are described.

1. At the beginning of the process, it is important to generate an understanding of the application area and the required prior knowledge. It is also important to identify the goal of the KDD process from the end user's point of view.

2. In the next step, the data utilized for the data analysis should be collected and arranged into a dataset, which is the target data. The following steps of KDD are based on this data.

3. In the third step, the pre-processing of the target data takes place. This phase is used to remove data noise or outliers from the collected raw data. This process is also called data cleansing. Furthermore, if necessary, strategies can be created and applied for the treatment of missing elements in the dataset. These are, for example, the removal of features or samples, if they have a high number of missing parts.

4. Feature selection is used to select the most relevant and useful features and reduce the amount of data initially collected for a project. The associated reduction of the data leads to an improvement in the accuracy of the results obtained later, since data that is

irrelevant and potentially hindering the model building is sorted out at an early stage. A further advantage is the reduction of the used data quantity since the programmed algorithms must accomplish fewer computations and thus the program execution is accelerated.

5. In the following step, data mining methods adapted to the goals of the project are selected. Common applications are for example regression, classification, or clustering.

6. After the data mining method is defined, the algorithms and selection methods used for the execution are selected. This includes a decision process, which models and parameterizations could be useful. In addition, it is necessary to consider the exact success criteria of the KDD process during the selection process. For example, it may be more interesting for a user to understand the basis on which a model makes decisions, rather than just having high predictive accuracy.

7. The seventh step is the actual data mining step. The aim here is to look for interesting patterns in the data or to better understand the approach of the applied model. For example, the understanding of a classification model can be improved by uncovering decision rules.

8. In the eighth step, the uncovered patterns are interpreted and evaluated. If necessary, it is possible to return to phases 1 to 7 to extract further knowledge. This step also includes visualization of the extracted patterns and models, or visualization of the data read into the model.

9. The final step is to exploit the knowledge collected during the KDD process. This can be the direct use of the collected knowledge, its integration into another system, or simply its documentation and presentation to the interested parties.

3.9 Cross Industry Standard Process for Data Mining (CRISP-DM)

The Cross Industry Standard Process for Data Mining (CRISP-DM) was developed in 1996 with the support of the EU and several industry partners. It builds upon earlier attempts to define the data mining process [8, 32, 62, 153]. The standard serves to standardize the approach to data mining projects across the industry, thus providing more transparency for customers and developers. The goals of the CRISP-DM approach are to increase the cost effectiveness, reliability, reproducibility, manageability, and efficiency of data mining projects [196].

According to the CRISP-DM model, data mining projects are divided into six different phases. These phases are to be seen as iterative, not unique. Depending on the requirements, some phases or even blocks consisting of several steps can or must be repeated in order to achieve the desired result. This can be the case, for example, if a targeted solution path turns out to be unsuitable or only partially successful. In such a case, the data mining process must be restarted in the phase in which the problem occurred. The repetition of certain phases is almost unavoidable due to the complexity and unpredictability of data mining projects and

usually occurs frequently. This is illustrated by the arrow directions in Figure 3.9 for the CRISP-DM approach [196].

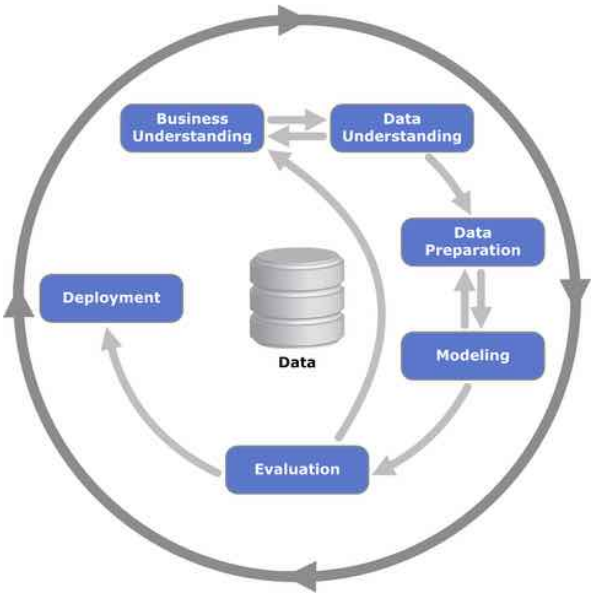

Figure 3.9: Illustration of the different phases and their relationships in the CRISP-DM procedure (source:[46]).

In the following, a detailed explanation of the individual steps is provided [196]:

1. The CRISP-DM process begins with the phase "Business Understanding". The main objective here is to define the basic goals and requirements of the project. This is then used to determine the rough approach and the exact task.

2. In the second phase, "Data Understanding", the data used in the project are first collected and an initial examination of its quality is carried out. It is important to identify possible problems with the structure or quality of the data.

3. In the following third phase, "Data Preparation", the aim is to create a final dataset from the collected data, which is adapted to the requirements of the models created. On the one hand, corrections such as outliers or the elimination of missing parts in the dataset are made. On the other hand, data that is not important for model building can be removed. Usually this phase is the most time-consuming.

4. After all data has been brought into the desired form, the fourth phase, "Modeling", begins. Data mining methods are used, which often, but not necessarily, involve the implementation of several machine learning algorithms. Different approaches are tested, so that eventually the model with the highest performance can be selected.

5. In the fifth phase "Evaluation", the results of the different implemented models are analyzed and evaluated. The model that best meets all the requirements is selected for the final implementation of the project.

6. The final phase of the data mining process is the "Deployment" phase. Now that the results of the data mining are available, the models created can be integrated into systems or applications to perform tasks there. Furthermore, it is necessary to prepare and document the obtained results, for example, to present them to the client that commissioned the project.

The phases "Business Understanding" and "Data Understanding" as well as "Data Preparation" and "Modeling", which are connected with opposite arrows in the diagram, each represent very closely interwoven process steps.

For example, if the goals or requirements of the project change in the "Business Understanding" phase, this has a direct impact on the "Data Understanding" phase, as new or adapted data may need to be collected in order to achieve the new goals. How the data is to be understood or interpreted may also change.

Similarly, between the "Data Preparation" and "Modeling" phases. For example, different Machine Learning models may have different requirements for the data they read in. In practice, this may mean that some algorithms within a model can only produce useful results if the input data has been previously normalized to values between 0 and 1. Other algorithms can get by without such normalization. In other cases, certain features in a dataset may also have a negative impact on the accuracy of a model. In such situations, depending on the model, it makes sense to remove the features that are detrimental to performance from the dataset [196].

From these examples it is obvious that in order to achieve the best possible performance of a model, a precise model-specific adjustment of the data is necessary. Thus, the applied dataset often has to be adjusted separately for each model [196].

Chapter 4

Solution Approach

In this section the solution approach is explained in detail for the contributions mentioned in Section 1.3 of this work. The workflow consists of five main stages with a total of six contributions until the final *proScenario dataset* is completed. These stages are visualized in Figure 4.1.

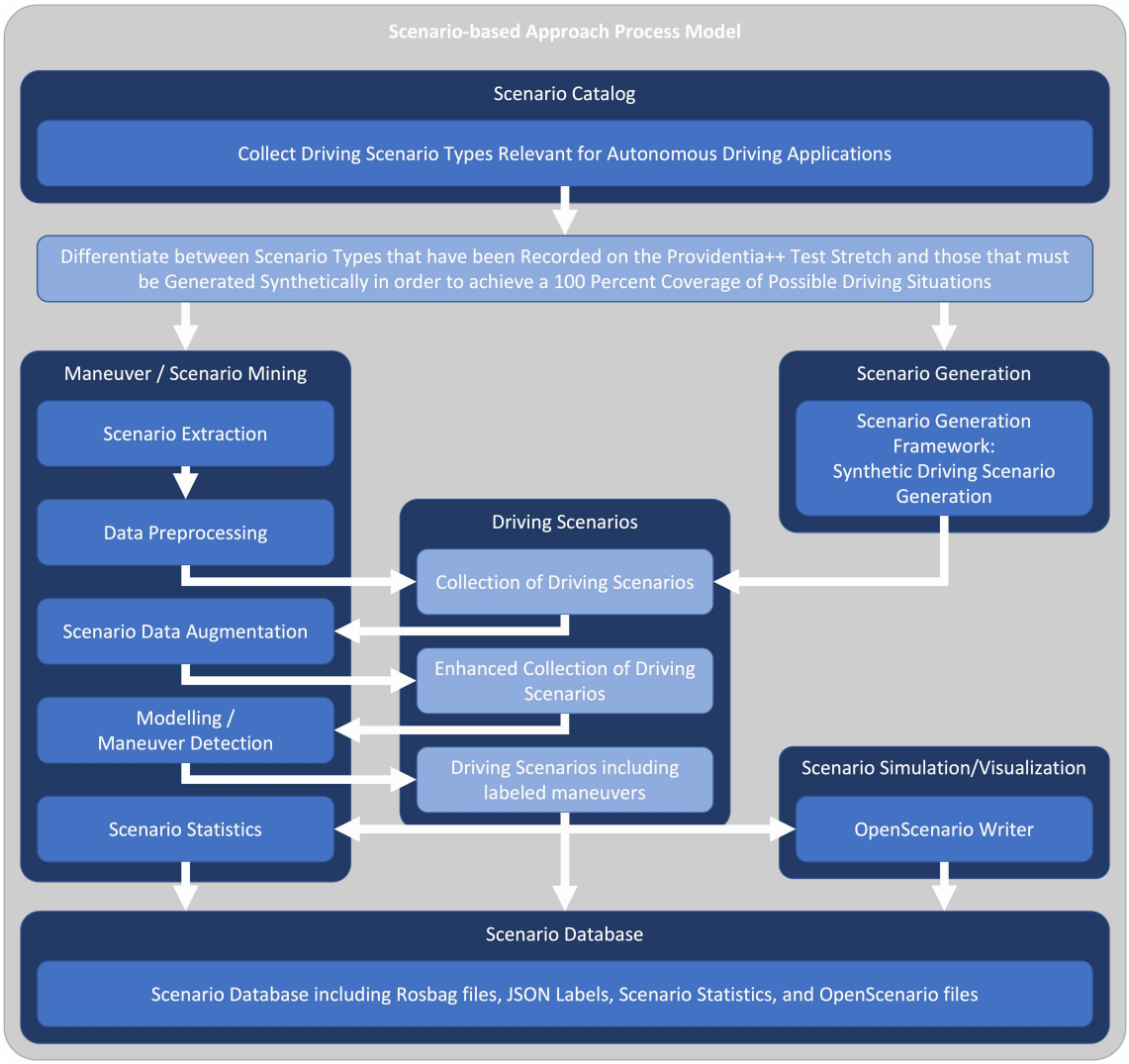

Figure 4.1: Illustration of the stages for the creation of the final *proScenario dataset* (source: own illustration).

1. Scenario Catalog

   In the first stage "Scenario Catalog" various sources are analyzed in order to find traffic scenario types that are safety-critical and relevant for the development of automated driving applications. These sources include documents from the National Highway Traffic Safety Administration (NHTSA), Euro-NCAP, and UNECE. The goal of this analysis is to find an exhaustive list of common driving situations and situations that lead to accidents. These scenario types are then added to a shortlist, from which a catalog specifically for the Providentia++ test stretch is created.

   After the shortlist is finalized, it is determined which of the scenario types occur on the Providentia++ test stretch. This is an important step as many scenario types, such as entering a toll station or exiting a parking lot, cannot occur on the test stretch. Once the scenario types for the test stretch are determined, the catalog is filled with them and provides an extensive overview.

2. Analysis of Recorded Data

   In the next stage it is then analyzed which of the scenario types from the catalog can actually be found in the recorded data. Many of the common scenarios such as a signaled left turn or a lane change occur very frequently in the recorded data, while rare scenarios such as an accident infrequently or never occur.

3. Scenario Generation

   In the "Scenario Generation" step scenarios that cannot be extracted because they do not occur in the data recordings of the Providentia++ test stretch are generated artificially. This is done with the "Scenario Generation Framework", which is introduced in this work. The framework enables an efficient creation of new driving scenarios tailored to the user's needs.

   In practice, typically video footage of such scenario types is used for inspiration. This is then adapted to the environment of the Providentia++ test stretch. Using real-world footage as a basis should ensure a realistic recreation with a very narrow simulation-gap.

4. Scenario Mining

   Since algorithms for autonomous driving must be able to cope with all kinds of situations it is very important to cover all scenario types, including rare ones, in the dataset. To reduce the labeling effort and to categorize various scenario types, easier data mining techniques are used. These are described in Section 4.5. The section describes five steps of data mining that are applied in order to detect and extract interesting types of driving scenarios. The five steps are:

   (a) Raw Recordings of Driving Data

       In this step the data frames at each time step of the recording are extracted. Each data frame represents one point in time and includes details about all vehicles and their position at that specific time.

(b) Data Pre-Processing

The extracted data frames are pre-processed in order to obtain the actors' trajectories, as well as all other scenario specific information.

(c) Scenario Data Augmentation

Here, both extracted and created scenarios are augmented, meaning that multiple variations of one base scenario are created. This way, the number of scenarios is artificially increased.

(d) Modelling / Maneuver Detection

In the next step various maneuvers of interest are "modelled". This means maneuver detection algorithms are written that can automatically recognize these maneuvers from the trajectory data provided in each scenario. The algorithms detect important driving maneuvers directly from the provided driving scenarios and helps to categorize them automatically. Also the detected driving maneuvers are added to the data as labels. This helps reduce the labelling effort drastically.

(e) Scenario Statistics

Finally the driving maneuvers detected in the previous stage are used to calculate an overall statistic for every driving scenario. This way the driving scenarios can be classified and also later be searched for certain events or anomalies.

5. Scenario Simulation / Visualization

In order to simulate and visualize driving scenarios, they are translated to the OpenSCENARIO file format. This is done automatically by the OpenSCENARIO writer created in this work. The standardized OpenSCENARIO files can then be simulated and visualized with compatible tools.

6. Scenario Database

In the last phase of the process a scenario database is created. For each driving scenario in the database a Rosbag file including labeled driving maneuvers, labels in JavaScript object notation (JSON) format, the scenario statistics, and an OpenSCENARIO file for simulation and testing purposes is included.

Before diving into the details of each of these stages, the intermediate data format in which both extracted and generated driving scenarios are stored is presented. This format represents a common basis in which scenarios from both data sources are transformed into. This way, any further processing, augmentation, detection, classification, and simulation tasks can be conducted more efficiently.

4.1 Scenario Description Format

The data used for the scenario mining step is the base data underlying driving scenarios included in the developed scenario catalog. There are two data sources in order to acquire the required data and to cover all test cases listed in the scenario catalog.

The first one is to extract scenarios from the data recorded by the sensor systems on the Providentia++ test stretch. The second source is generating driving scenarios synthetically as described earlier. This is done for scenario types that cannot be extracted from data recordings.

In order to have a basis from which both extracted and generated scenarios can be further processed, a common data format is defined. This format is referred to as the "Scenario Description Format". It includes all information and parameters necessary to fully describe a driving scenario. The format follows a hierarchical order, which can be described as a tree with a depth of four layers.

The first layer consists of the two nodes "scenario meta information" and "actors". The node "scenario meta information" contains all information describing the environment conditions. This node therefore contains the five child nodes "Number of Frame", "Frame Rate", Timestamps seconds", "Timestamp nanoseconds", and "weather conditions". Hereby "Number of Frames" contains the amount of data frames the scenario contains in total and "Frame Rate" describes the frequency at which data frames are sampled. For example a driving scenario that is 60 seconds long and is sampled at a frame rate of 25 frames per second has total number of 1500 frames. Therefore the value stored in the node "Number of Frames" is 1500 and the value stored in the node "Frame Rate" is 25. The node "Timestamps seconds" stores the "unix" timestamp when each of the data frames included in the scenario are recorded. The node "Timestamps nanseconds" additionally stores the exact nanoseconds when each of the data frames are recorded. Finally, the weather conditions at the test stretch at the time of the driving scenario are stored in the node "Weather Conditions".

The node "Actors" contains all actors included in the driving scenario. These can be vehicles, pedestrians, cyclists, and other moving or static objects. Each actor contains child nodes with information describing its properties and dynamics. An actor is divided into the child nodes "ID", "Object Class", "Color", "Path", "Time Stamps", "Velocities", "Length", "Width", and "Height". Here, in the node "ID" the actors unique identification number is stored. Each actor has a unique ID number which is useful to identify an actor in throughout different data processing steps. "Object Class" contains the type information of the actor i.g. if the actor is a car, truck, cyclist, pedestrian, or other type. "Color" contains three integer numbers between 0 and 255. These numbers correspond to RGB values to describe the actor's color. "Offset" describes the time difference between the beginning of the driving scenario and the point in time when the actor first appears in the scenario.

The node "Path" contains the ordered coordinates that define the path of the actor. "Time Stamps" contains the timestamps for each of the data frames the actor appears in. "Velocity" describes the longitudinal speed of the actor at each time step.

Practically, the information stored in the node "trajectory" and one of either nodes "velocity" or "time stamp" is enough to fully describe the actors' movement over time. The reason why both are stored is that there are different ways to describe the basic scenario information in the OpenSCENARIO format. Depending on how this is done, either the information stored in the node "velocity" or in the node "time stamp" can be more beneficial. This is discussed in Section 4.6 in more detail.

Finally, the nodes "Lenght", "Width", and "Height" contain the dimensions of the actor. The data format is visualized in Figure 4.2. This data representation of a driving scenario is the

common base in which both extracted data and synthetically generated data is brought into. From this base any further steps like the generation of OpenSCENARIO files or the automated labeling of maneuvers and scenarios are carried out.

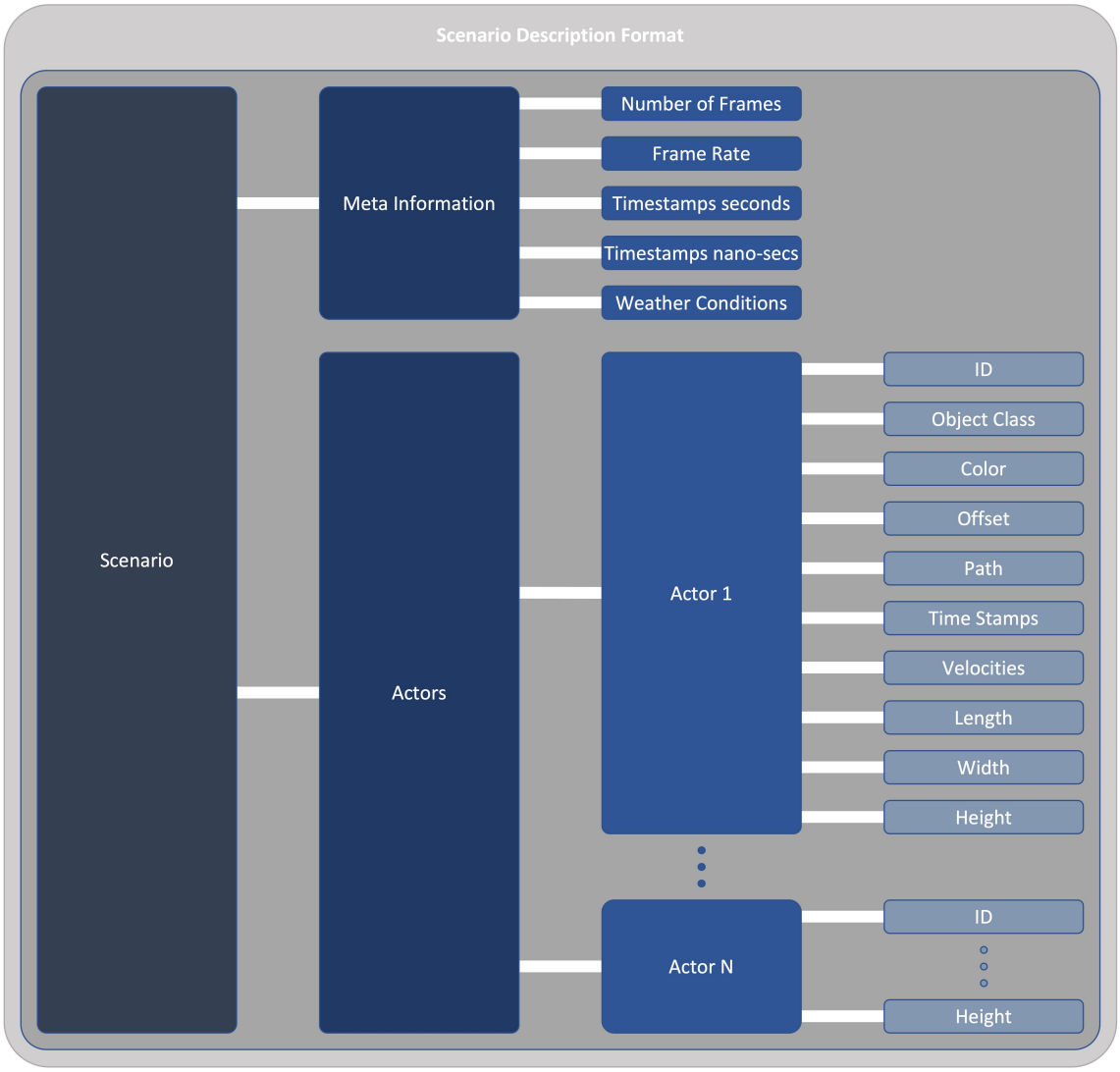

Figure 4.2: Illustration of the data format in which the scenario information is stored (source: own illustration).

4.2 Scenario Catalog

The basis of this work is a scenario catalog that determines which scenario types are included in the dataset. For this reason, it is crucial to define the catalog carefully, as it is mainly responsible for the success of the dataset created in this work. The position of the scenario catalog in the entire process model is highlighted in Figure 4.3.

In order to determine which features and scenario types should be included in the dataset, an analysis of the most relevant and safety-critical traffic scenarios is made. This is done by screening various sources that analyze causes of traffic incidents. These sources include reports from the National Highway Traffic Safety Association (NHTSA), Euro-NCAP, and UN-

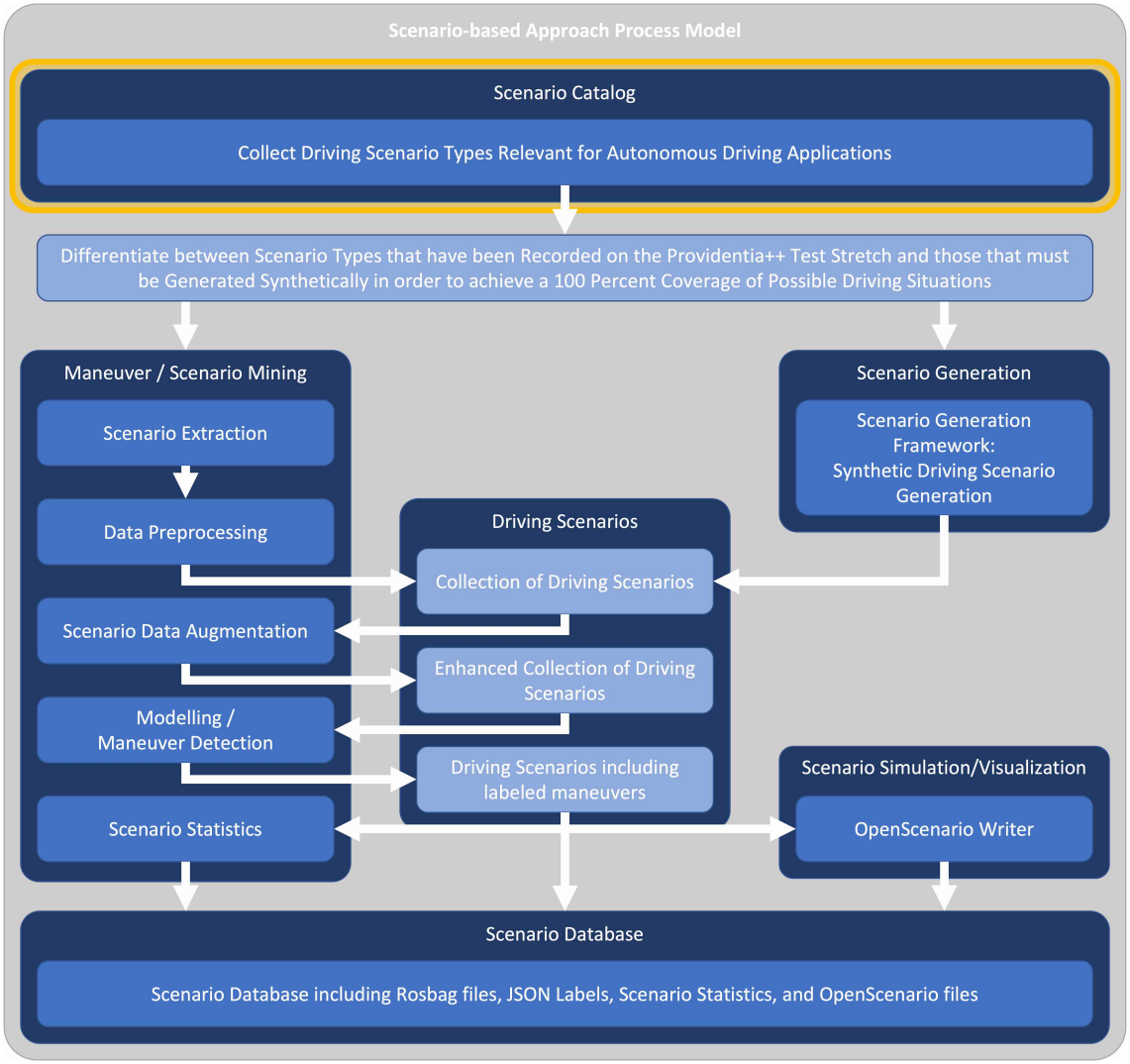

Figure 4.3: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

ECE. In total, from these sources a shortlist of 132 different general scenario types is made, which is then condensed to the final list of 86 scenario types included in the *proScenario dataset*.

First, all scenario types that physically cannot occur on the test stretch are removed. Examples for this are scenarios like entering- and exiting a toll station or a roundabout because both a toll station and a roundabout are not part of the Providentia++ test stretch. Additionally, some scenario types that are listed in the sources analyzed to create the initial shortlist may occur on the test stretch, but are not relevant. These include scenario types like following vehicle changes lane and following vehicle stops. Those scenario types are irrelevant because the situation does not require any further action from the vehicle, they are referring to.

After removing all irrelevant scenario types from the shortlist, the resulting scenario types are added to two different catalogs. One catalog contains all scenario types that occur on highway while the other catalog contains scenarios that occur in urban and rural areas. Figure 4.4 shows some examples of scenario types included in the catalogs. The full catalogs are shown in the appendix under 1 and 2.

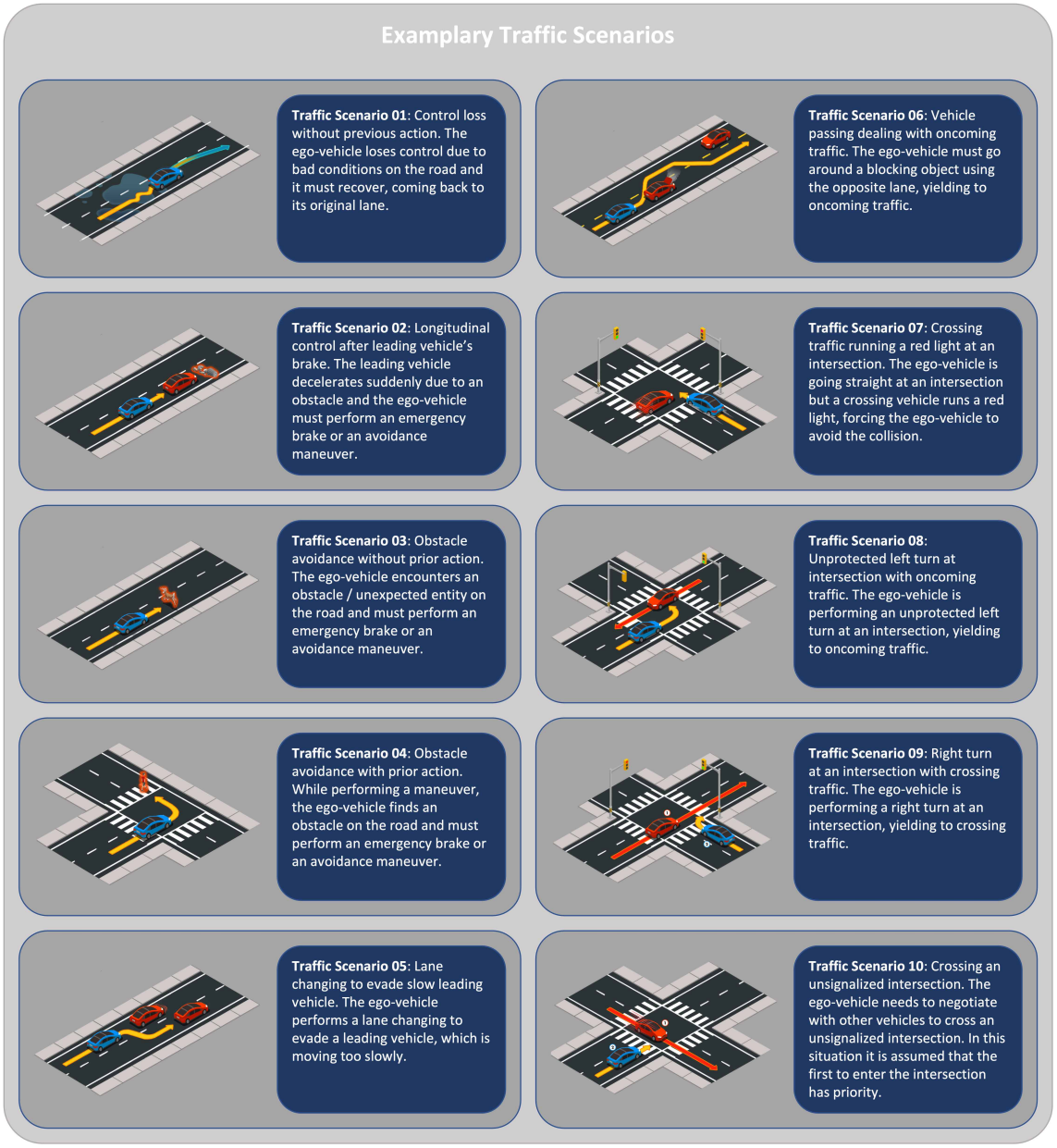

Figure 4.4: Examples of driving scenarios selected from the NHTSA pre-crash typology (source: [175]).

4.3 Analysis of Scenario Types that can be extracted from Data Recordings

In the previous section the creation of a catalog containing all scenario types that can occur on the Providentia++ test stretch has been addressed. With this exhaustive list as a reference, it is observed which of the scenario types can be extracted from data recorded on the test stretch. There are two main motivations for extracting scenarios from the recorded data.

First of all, data recorded from real-world traffic is 100 percent authentic and it can be assured that the data is helpful for perception algorithms to be trained on. In contrast when generating synthetic scenarios, it cannot be fully guaranteed that these reflect a traffic situation in a realistic manner.

The second reason for extracting scenarios is that the synthetic creation of new scenarios is very time consuming and requires manual work. It should therefore be reduced to an absolute minimum.

For these reasons, as many scenarios as possible will be extracted from data recordings of the Providentia++ test stretch using Data Mining techniques. The complete list of scenarios extracted and those created synthetically can be found in the appendix under 1 and 2. Since the Data Mining process is focused on extracting scenarios, it can be referred to as Scenario Mining [190]. The further procedure is explained in the following sections.

4.4 Synthetic Scenario Generation

The scenario catalog developed as part of this work is meant to cover all possible traffic situations. This also means that it includes driving scenarios that generally occur very rarely and have not occurred on the Providentia++ test stretch. However, in order to properly train algorithms for autonomous driving, an extensive suite of driving situations is required. Relevant driving scenarios that have not occurred on the test stretch and can therefore not be extracted are created synthetically. This is done with the "Scenario Generation Framework" developed as part of this work.

This is important as it provides the possibility to train and validate autonomous driving systems on all test cases, including very rare ones. Figure 4.5 highlights the role of synthetic scenario generation in the entire process.

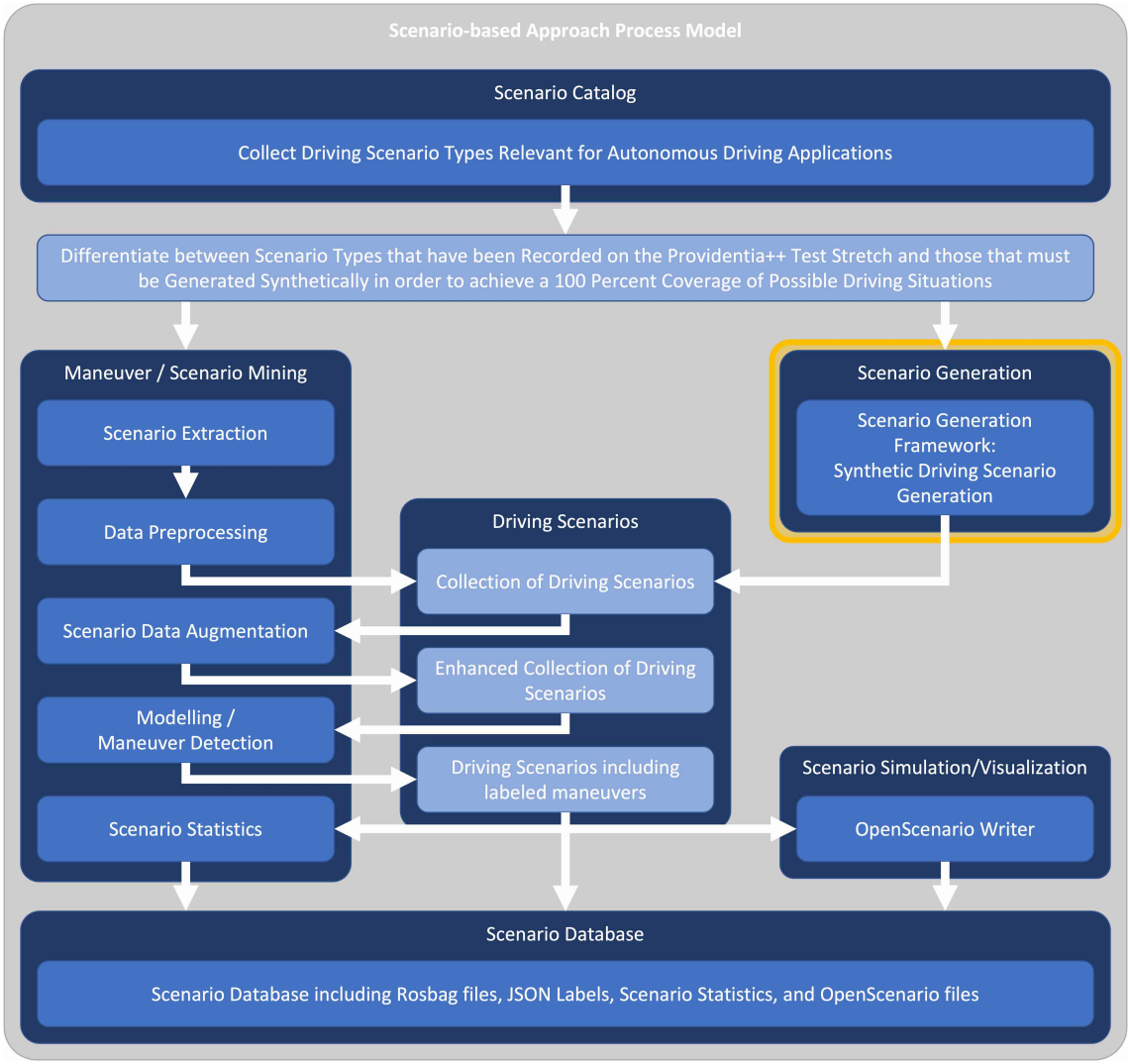

Figure 4.5: Illustration of the Stages for the creation of the final *proScenario* dataset (source: own illustration).

In order to give the user maximum control for defining all details of a scenario, a framework for creating synthetic driving scenarios is introduced as part of this work. The main goal of this framework is to model the dynamics of each actor in the scenario accurately following the user's needs. This includes defining the actors' paths and velocities.

The framework provides a simple yet efficient way to describe driving scenarios while maintaining a high flexibility when describing the actor dynamics. The main building blocks of the framework are a collection of approximate routes, velocity profiles, a trajectory calculator, and a visualization. In the following, these building blocks and the creation of a driving scenario is explained in more detail.

1. Collection of Routes

   When describing the motion of a vehicle or an actor in general, the most information is contained in the actor's path. It describes the positions that are run through, as well as the movement direction and even the orientation when constraining it to the path. Additionally, many maneuvers like turns or lane changes can be detected only by using the path as an information source.

For these reasons, the actors' paths are the most important pieces of information when describing a driving scenario. In the scenario creation framework proposed in this work the user is given full control of designing routes for actors. This is done by creating a set of coordinate points in correct order that define the route. The amount of coordinate points can be chosen freely by the user, since the final trajectory is calculated by an interpolation algorithm, which is described in step three. Due to the nature of the interpolation algorithm, a route must be defined by at least four sets of coordinate points. Depending on the number of coordinate points given by the user, the interpolation algorithm is more or less constrained. All routes defined by the user are collected in independent script, from which they can be imported to a driving scenario depending on the user needs. Figure 4.6 shows an example of how a complex route can be defined by a set of seven coordinate points only.

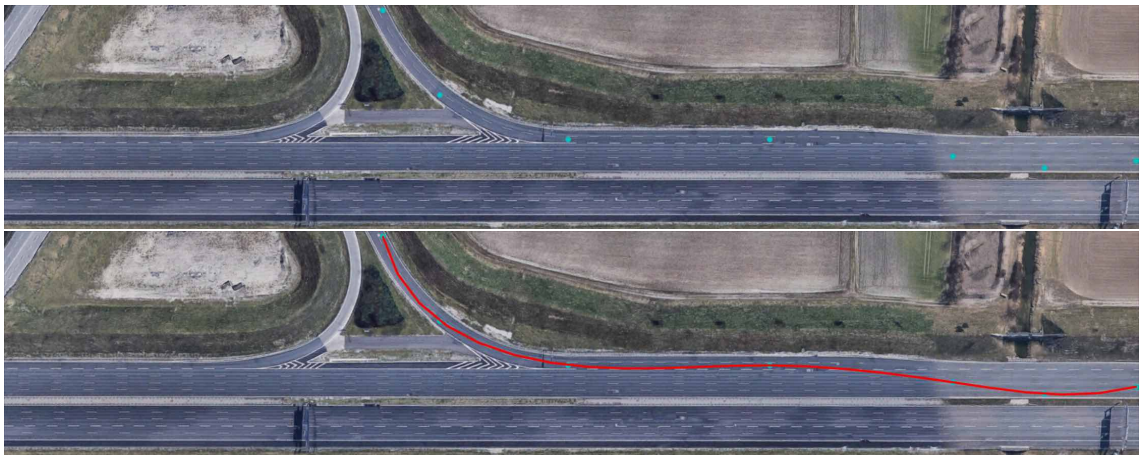

Figure 4.6: Example of how a complex trajectory with multiple lane changes and a highway exit can be defined by a small set of coordinate points in the coordinate system of the Providentia++ test stretch (source: own illustration, [1]).

2. Velocity Profiles

   In order to fully define the position of an actor at each point in time, a path is not sufficient. Additionally, to coordinate points a timestamp must be added when each of the coordinate points along the path is reached by the actor. This combination of positional and temporal information is the actor's trajectory.

   In theory, the trajectory of each actor could be defined by combing the actor's path and a matching timestamp for each coordinate point along the path. This may work well for a computer system that can simply record the current time whenever it tracks the position of a vehicle. However, when creating a trajectory synthetically, it may be more intuitive for the user to provide the current vehicle velocity at each time step.

   For this reason, the user can create an array of velocity values, where each value defines the actor's longitudinal speed at the corresponding point in time. The details of how a complete trajectory is calculated from a simple route and the velocity at each time step are explained in the following paragraph.

3. Trajectory Calculator

The goal of the scenario generation framework is to make the creation of trajectories for any kind of actor as intuitive as possible. Therefore, the user can obtain a complete trajectory by simply passing a small set of coordinate points marking the path as well as a set of velocities resembling the actors speed at each timestamp. These inputs are passed to a module that calculates the trajectory in two main steps.

(a) In the first step, a function describing the path is obtained by performing a cubic interpolation through the set of coordinate points given. The function is parameterized by the distance travelled along the path. This means that it takes the distance travelled as an input and outputs the x- and y-coordinate after travelling that distance. The resulting function can then be sampled at various distances to obtain a set of coordinate points.

(b) Up to this point an interpolation function is obtained that takes a certain distance along the path as an input and outputs the corresponding coordinate points. Depending on the frame rate at which a scenario is sampled the time passed (time step) between two time stamps varies. The relation between the sampling rate and the time passed between two time stamps is inverse. A sampling rate of 25 Hertz equates to a time step of 40 milliseconds whereas a sampling rate of of 0.5 Hertz equates to a time step of 2 seconds. The relation between sampling rate and time step is described by the following equation:

$\Delta t = 1/f$
with $\Delta t = timestep\ f = frequency\ (sampling\ rate)$,

In the course of the Providentia++ project, multiple datasets will be released. They are intended to be sampled at frame rates of 2.5 and 10 or 12.5 Hertz. The sampling rate of synthetic driving scenarios can easily be adapted by the corresponding input parameter of the trajectory calculator. The general sampling procedure however remains the same.

Since the first release of the *proScenario dataset* is intended to have a sampling rate of 2.5 Hertz it is crucial to know what distance the actor has travelled at every time step which occurs every 400 milliseconds. Only if this is known the function describing the actor's path can be sampled appropriately to create a true trajectory.

This is where the velocity profile defined by the user comes into play. It describes the actor's velocity at each point in time. With this information, the distance travelled between two time steps can be calculated through the following relation between distance, time, and velocity:

$d = v * t$
with $d = distance, v = velocity, t = time$

Analogously, since the trajectories are described in discrete time steps the equation can be written as follows:

$\Delta d = v * \Delta t$
with $d = distance, v = velocity, t = time$

This shows that the distance travelled between two timestamps is simply the velocity at this point in time multiplied with the time between the two time steps. Since the data is sampled at 2.5 Hertz the time between two time steps is 400 milliseconds or 0.4 seconds. To get the absolute distance travelled at each time step, the distances calculated are simply accumulated up to the specific time step. An example of the procedure is shown in Figure 4.7.

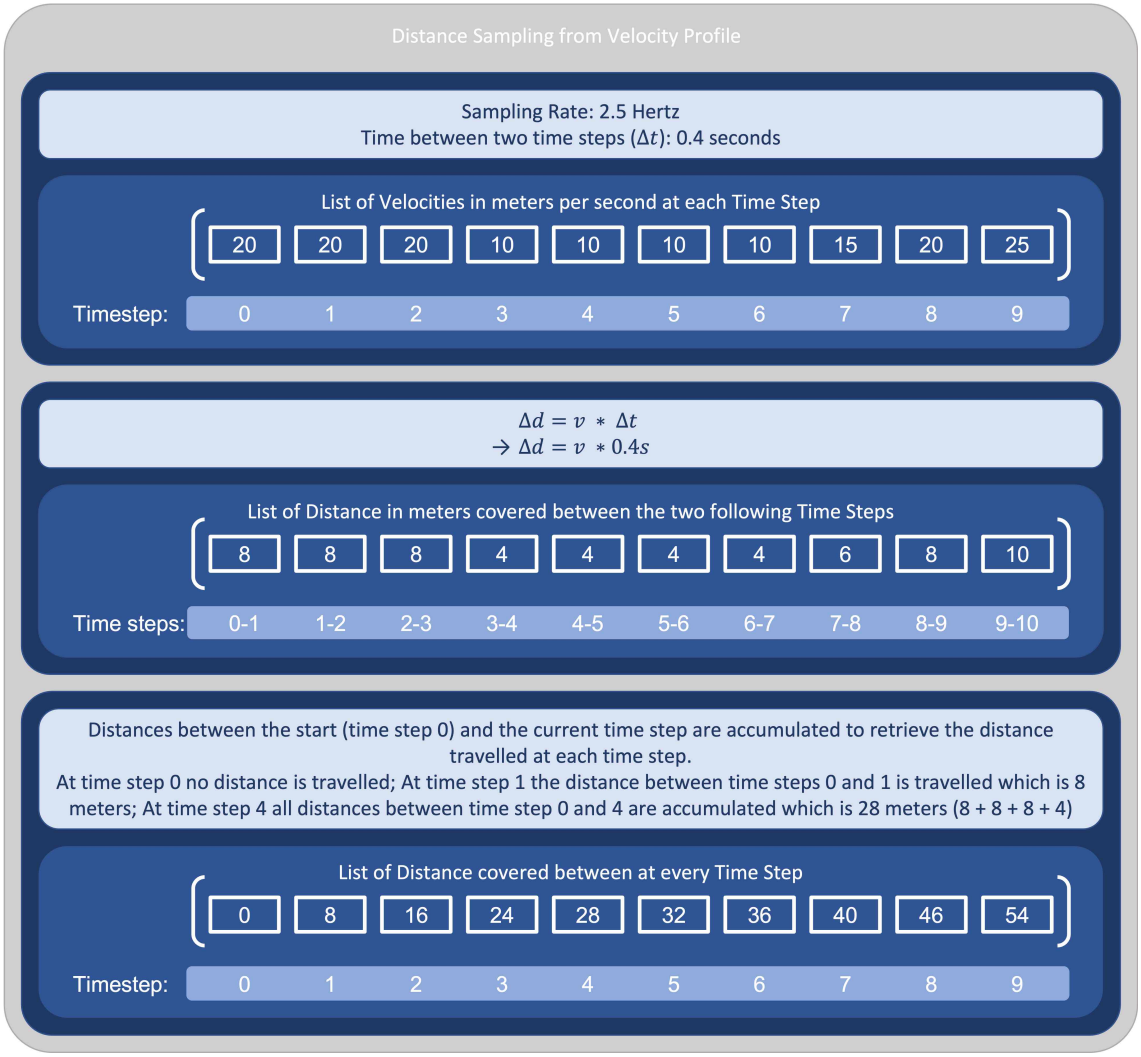

Figure 4.7: Visual representation of the steps to calculate the distance travelled at each time step (source: own illustration).

Once the distance travelled at each time step is calculated, the function describing the actor's path is sampled at each of these distances. This generates a set of coordinate points matching the timestamps, which in combination describe a complete trajectory.

In Figure 4.8 an example of a trajectory with a specific velocity profile is visualized. The profile is defined that the actor moves at a speed of five meters per second for the first fifteen time steps, then moves at 30 meters per second for five time steps before moving at ten meters per second for ten time steps. For the last five time steps, the actor then moves at 20 meters per second. When looking at

the coordinate points along the trajectory at each time step, it becomes clear that the distance between two following points depends on the current longitudinal velocity of the actor.

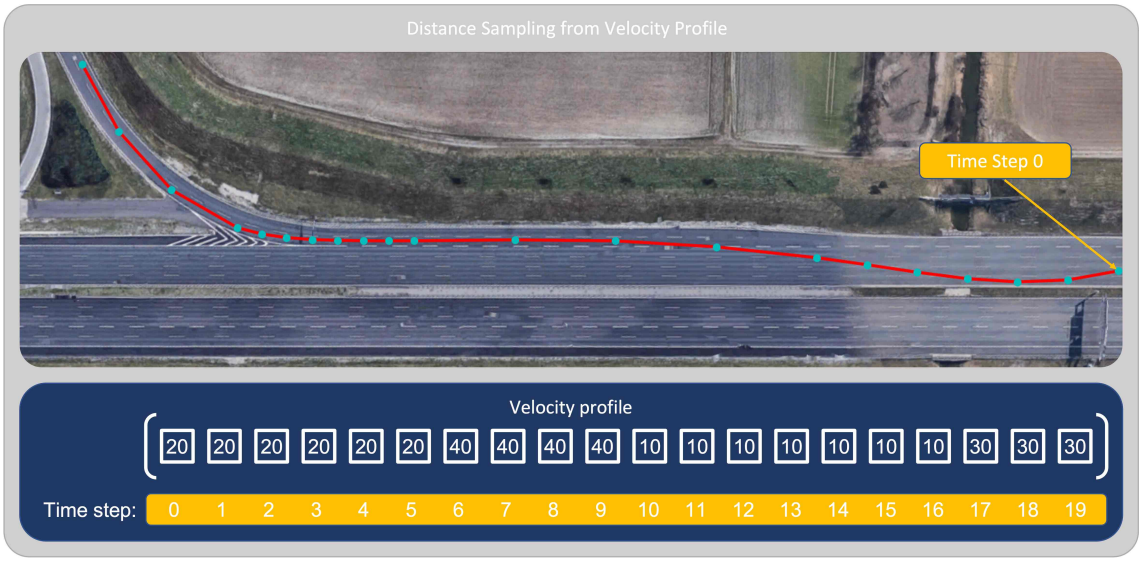

Figure 4.8: Visual representation of how different velocities at each time step affect the distance travelled at that time step (source: own illustration, [1]).

Additional parameters the trajectory calculator takes as inputs are the actor ID and type, the frame rate in which the scenario is sampled, the time offset from the start of the scenario when the actor comes into play, and distance offset from the beginning of a user defined path. This information is used to describe the actor's properties and behavior in more detail.

4. Visualization

   For the user to control whether the defined driving scenario acts as intended, a visualization is included in the framework. The visualization takes the scenario containing the trajectories and creates a plot of all actors at each time step. Only those actors are included in a plot when present in the scenario at that point in time. Actors that are not present at the beginning of the scenario, for example are not included in the first plots. The plots are then animated by playing them according to the sampling rate of the data. In the background of each plot the Providentia++ test stretch is depicted so that the user can also evaluate whether the position of the actors in the scenario is correct at all times. An example showing multiple plots from a scenario is shown in Figure 4.9.

Figure 4.9: Visual representation of a driving scenario generated by the visualization script. The plots shown in this figure are animated and stored as a GIF by the tool. The road network is right-handed (source: own illustration, [1]).

5. Full Example of Scenario Generation

In the previous chapter the main building blocks of which the scenario generation framework is composed of are explained. In this section, all steps involved in the generation of a synthetic driving scenario are explained upon an example scenario. The general process for creating synthetic driving scenarios is pictured in Figure 4.10

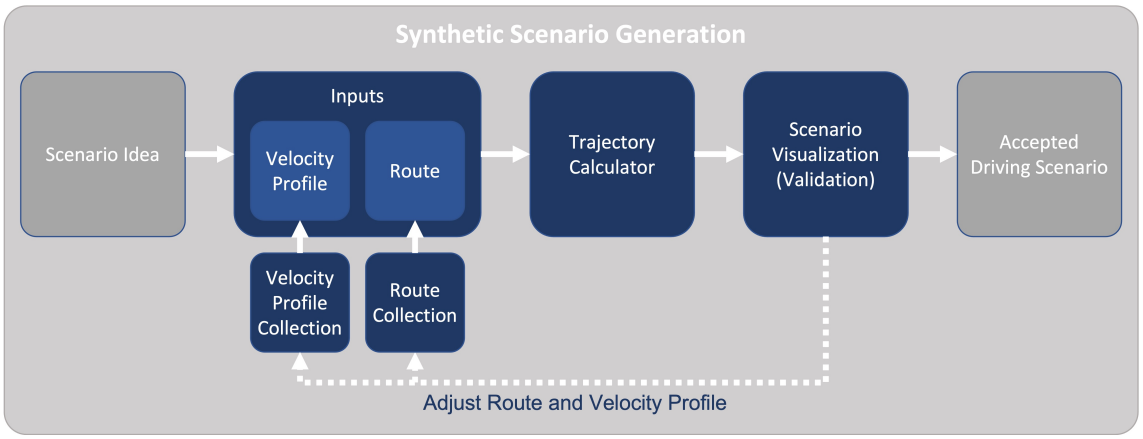

Figure 4.10: General process for creating synthetic driving scenarios. If the user wants to edit the initially created scenario the route and velocity profiles can be adjusted (source: own illustration).

In a first step, the user defines the path for the actor to follow. This can be done by defining a small set of at least four coordinate points. These do not have to match with any of the timestamps, because they only operate as reference points for guiding the path interpolation. In Figure 4.11 a simple lane change maneuver is defined with only four reference points.

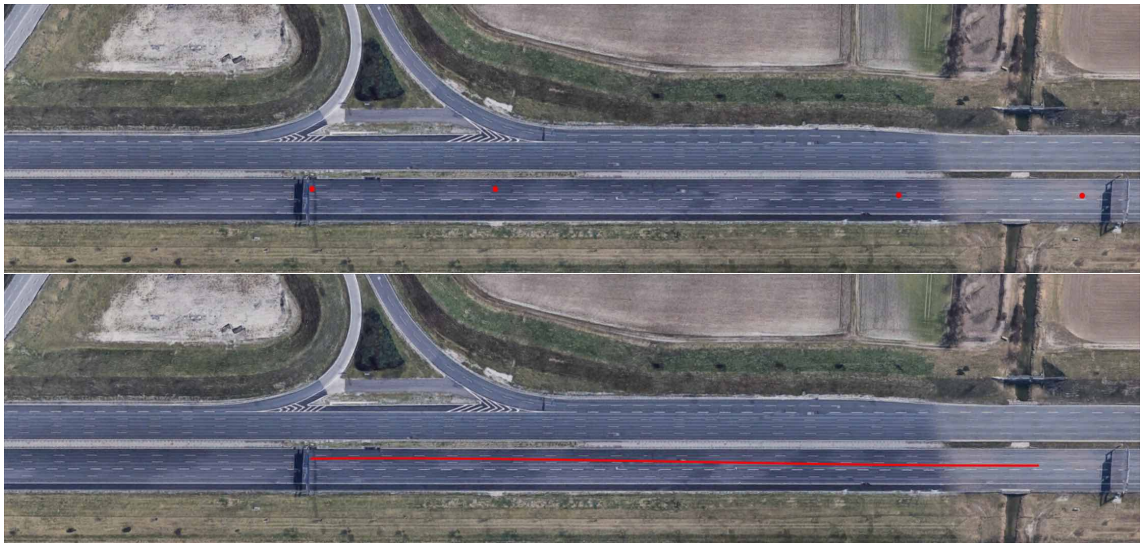

Figure 4.11: Example of how a lane change trajectory can be calculated by using only four reference points to describe the actor's path (source: own illustration, [1]).

Next, the user defines the velocity profile of the actor. In this case, each velocity in the profile must match a time step. This defines the actor's speed at each time step.

With both the reference points for the path and the velocity profile, the trajectory calculator creates the actor's trajectory. To do so, first a spline function is created by performing a cubic interpolation on the reference points passed from the user. Then the distance travelled at each time step is calculated, as described in this chapter under point 2. With this information, the spline function can be sampled correctly, and produces the exact coordinate location of the actor at each time step in the scenario. Finally, all additional information provided is processed to further describe the actor's behavior, as described at the end of point 3. The explained steps are visualized in Figure 4.12.

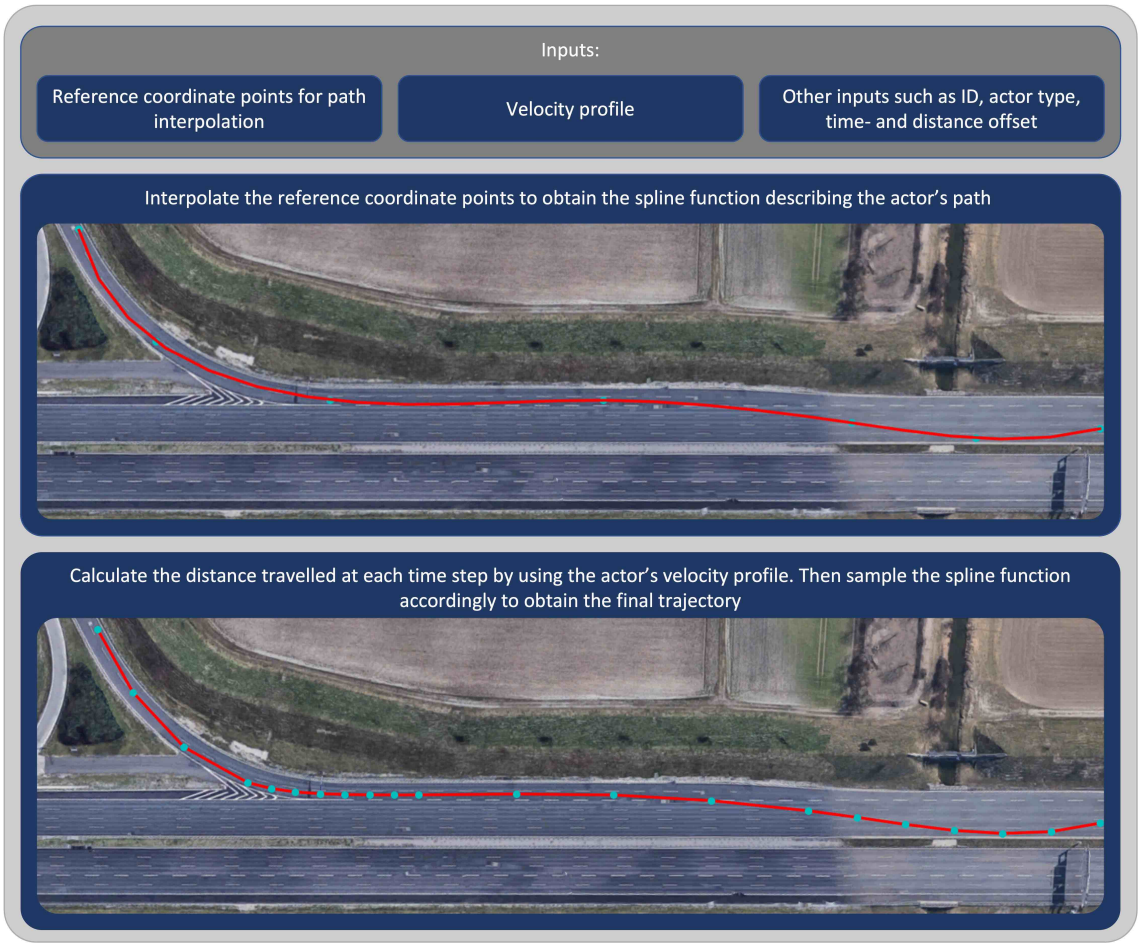

Figure 4.12: Full Example of Scenario Generation (source: own illustration, [1]).

4.5 Maneuver Detection / Scenario Mining

The goal of the scenario mining step is to assign each scenario with appropriate labels in an automated fashion. This is done by first automatically detecting vehicle maneuvers from which conclusions about the scenario type can be drawn. By doing so the labor necessary for manual labeling is reduced dramatically.

The detection and classification of relevant scenario types occurs on the basis of data mining techniques. In order to solve the process of scenario mining in an adequate manner, a

process model is used for guidance. Santos and Azevedo [21] outline that there are two most relevant process models for Data Mining projects, the Knowledge Discovery in Databases (KDD) and the Cross Industry Standard Process for Data Mining (CRISP-DM).

Both models are described in detail in the related work section of this work. Most of the process stages are identical in both models. The main difference lies in the focus of the CRISP-DM on the business case related to a Data Mining project [21, 196]. The KDD process is more focused on the technical solution of the task [21, 62]. The central role of scenario mining including its sub steps in this work is highlighted in Figure 4.13.

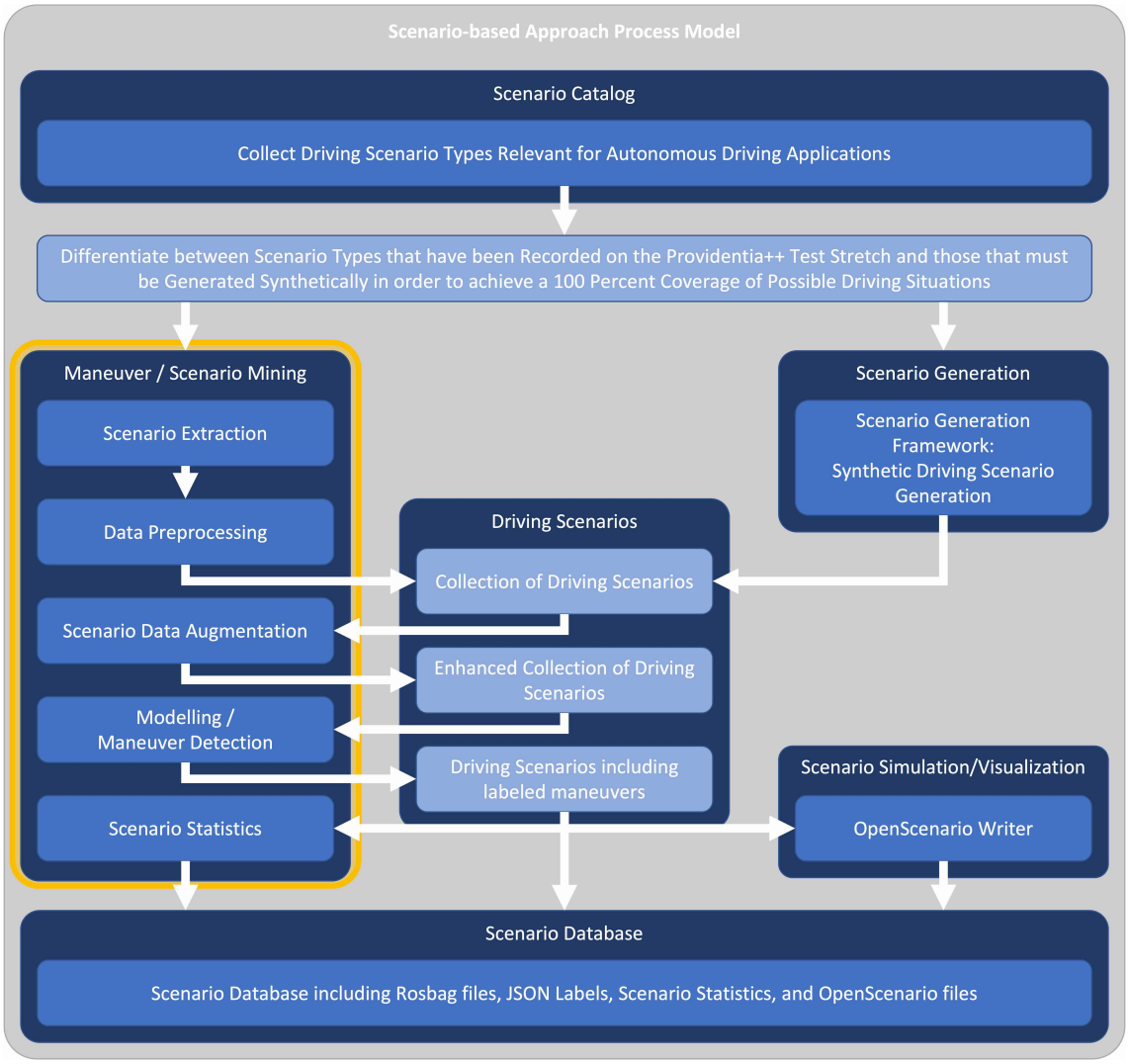

Figure 4.13: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

This work aims to create the basis for generating an open-source dataset for research purposes and there is no underlying business case with the intention to create profits. Therefore, the focus completely lies on the technical implementation. For this reason, the KDD process model is considered as a better fit and is used to guide the Scenario Mining step.

4.5.1 Scenario Extraction

The first step of the scenario mining process is the extraction of driving scenarios from data recordings. Figure 4.14 highlights the position of scenario extraction step in the entire scenario-based approach process model.

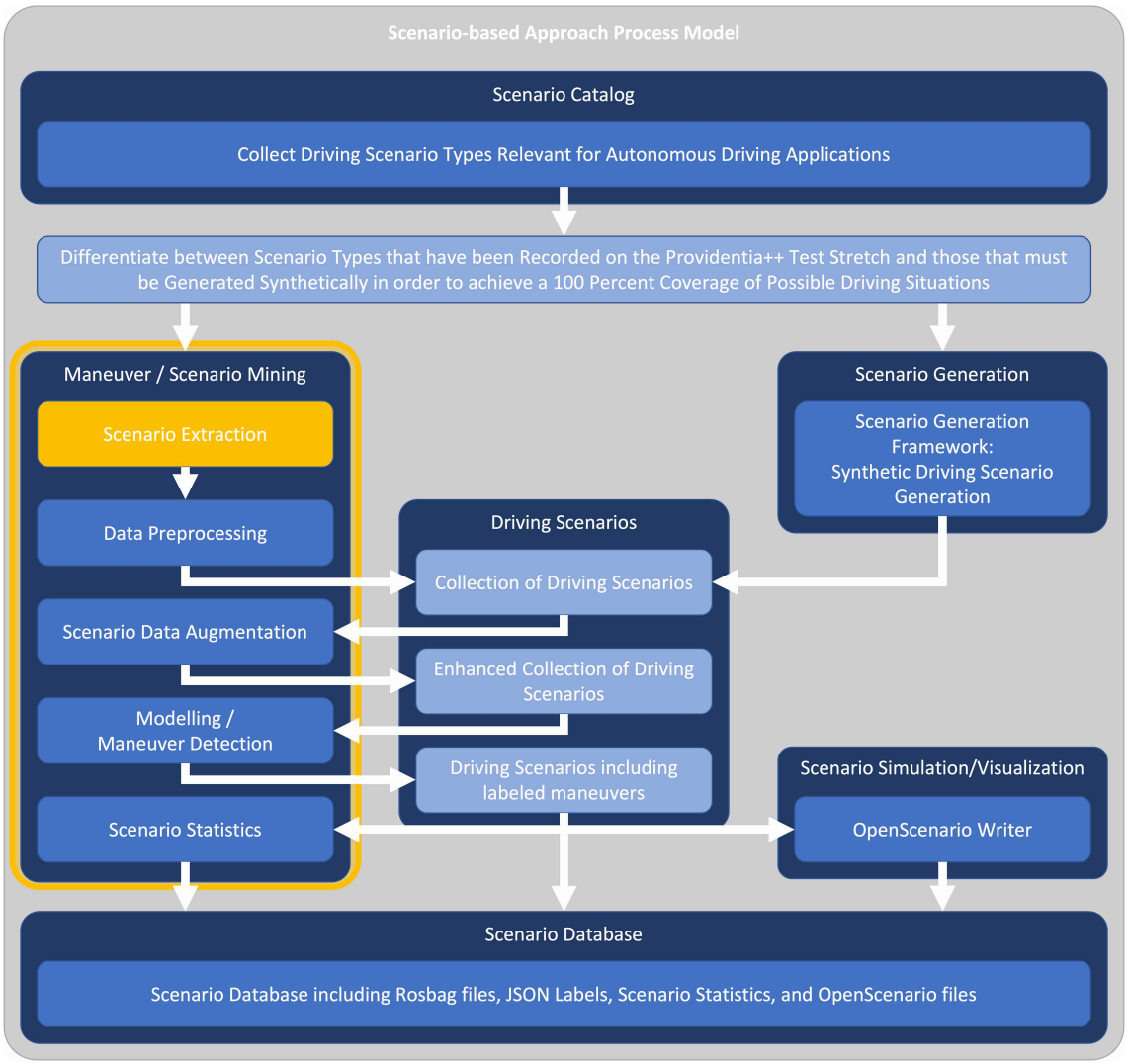

Figure 4.14: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

One method to create driving scenarios is to utilize the data recorded from the Providentia++ test stretch. In multiple steps, the raw sensor data is extracted and transformed so it fully describes a driving scenario.

The sensor recordings from the test stretch include camera, LiDAR, and radar data. In various works part of the Providentia++ project, techniques have been developed to extract information from these data streams like object type, dimensions, position, and velocity. Additionally, a tracking algorithm is applied that recognizes whether the same object (vehicle) is present in multiple data frames and assigns the same unique ID for this object (vehicle) in each of these data frames. Also, the date and time of the recording and a timestamp for each data frame is added. All this information is then stored in a rosbag file.

The extraction pipeline created as part of this work begins with two modules that transform the processed sensor data into the "Scenario Description Format". The first module takes the rosbag file created from the Providentia++ sensor system as an input. The second module takes JSON files containing similar information as the rosbags as an input. The JSON files are used for manually labeling driving scenarios. For this reason it is important to be able to transform both the rosbag files and the JSON files into the "Scenario Description Format".

The two extraction modules are very similar except for the first processing step. In the first step, a Python script extracts the data stored inside the rosbag or JSON files. Since the file format of rosbag and JSON files is different, individual scripts are necessary to extract information from the two data formats. In both cases the extracted information is written into the so called DataFrame format using the Python *pandas* library. After this point the following steps for both extraction modules are identical.

The benefit of the tabular DataFrame structure is that information can be extracted in very targeted manner by simply filtering the DataFrame by values one or more of the columns. Each row of the DataFrame represents all information about a specific object at a certain point in the data recording. This includes the timestamp, unique tracking ID, dimensions, position, type, velocity, and other attributes of the object. In Table 4.1 the structure of the DataFrame is depicted for better understanding.

timestamp	id	category	width	length	height	x	y	z	velocity x	velocity y	velocity z
0.0	7853	CAR	1.9	4.7	1.45	92.930359	-5.265621	0	31.247	0.016	0
1.0	7853	CAR	1.9	4.7	1.45	125.389847	-5.303660	0	31.418	0.022	0
2.0	7853	CAR	1.9	4.7	1.45	158.329147	-5.287288	0	31.473	0.013	0
3.0	7853	CAR	1.9	4.7	1.45	193.811325	-5.048016	0	31.483	0.011	0
4.0	7853	CAR	1.9	4.7	1.45	229.728806	-4.823170	0	31.436	0.012	0
5.0	7853	CAR	1.9	4.7	1.45	263.120880	-5.039521	0	31.218	0.015	0
6.0	7853	CAR	1.9	4.7	1.45	297.389557	-5.200200	0	30.917	0.012	0
7.0	7853	CAR	1.9	4.7	1.45	333.263947	-5.339573	0	30.672	0.014	0
8.0	7853	CAR	1.9	4.7	1.45	367.479401	-5.489977	0	30.456	0.014	0
9.0	7853	CAR	1.9	4.7	1.45	400.318573	-5.680832	0	30.333	0.017	0
10.0	7853	CAR	1.9	4.7	1.45	431.169769	-6.057634	0	30.297	0.015	0
0.0	7859	CAR	1.9	4.7	1.45	257.644379	-16.127937	0	27.343	0.013	0
1.0	7859	CAR	1.9	4.7	1.45	283.190948	-16.041351	0	27.112	0.012	0
2.0	7859	CAR	1.9	4.7	1.45	311.745636	-16.026731	0	26.973	0.011	0
3.0	7859	CAR	1.9	4.7	1.45	340.246307	-16.025745	0	26.427	0.010	0
4.0	7859	CAR	1.9	4.7	1.45	366.813110	-16.133640	0	26.045	0.014	0
5.0	7859	CAR	1.9	4.7	1.45	392.872742	-16.281500	0	26.084	0.018	0
6.0	7859	CAR	1.9	4.7	1.45	416.937653	-16.383415	0	26.114	0.017	0
7.0	7859	CAR	1.9	4.7	1.45	442.191895	-16.558998	0	26.237	0.019	0
6.0	7900	CAR	1.9	4.7	1.45	28.213936	-8.864231	0	26.390	0.014	0
7.0	7900	CAR	1.9	4.7	1.45	63.442867	-9.118527	0	26.516	0.013	0
8.0	7900	CAR	1.9	4.7	1.45	100.112740	-9.298901	0	26.492	0.015	0
9.0	7900	CAR	1.9	4.7	1.45	137.056656	-9.281519	0	26.412	0.012	0
10.0	7900	CAR	1.9	4.7	1.45	177.466614	-8.892739	0	26.378	0.011	0
11.0	7900	CAR	1.9	4.7	1.45	217.548218	-8.453785	0	26.351	0.012	0
12.0	7900	CAR	1.9	4.7	1.45	259.275513	-8.650146	0	26.329	0.014	0
13.0	7900	CAR	1.9	4.7	1.45	297.340485	-8.772322	0	26.296	0.013	0
14.0	7900	CAR	1.9	4.7	1.45	334.646332	-8.907867	0	26.374	0.014	0
15.0	7900	CAR	1.9	4.7	1.45	369.598969	-8.930677	0	26.515	0.012	0
16.0	7900	CAR	1.9	4.7	1.45	404.133057	-8.939241	0	26.746	0.011	0
17.0	7900	CAR	1.9	4.7	1.45	437.150818	-9.254308	0	27.013	0.015	0

Table 4.1: Excerpt of the DataFrame from an extracted scenario (source: own illustration).

To obtain the properties and trajectory for each actor in a first step, all unique actor types are collected in an actor list. Then for each actor type all unique IDs are collected in a list. In a next step, the DataFrame is first filtered by one of the actor types and then by one of the unique IDs.

After these two filters are applied, all information about one specific actor is filtered over the time period it occurs in the recording. This data can then be sorted by the timestamps available in each data row. Once the data is sorted chronologically, the position of the actor at each timestamp is stored in a list in order to describe the actors' path.

Similarly, the timestamps and velocities are stored. The vehicle properties do not change over time and therefore are only stored once.

The various information is then collected in a dictionary and appended to the overall scenario. This is done for all unique actor types and IDs and results in the full description of the scenario in the data format described in the previous chapter. The pseudo code and visual description of the entire extraction process is shown in Figure 4.15.

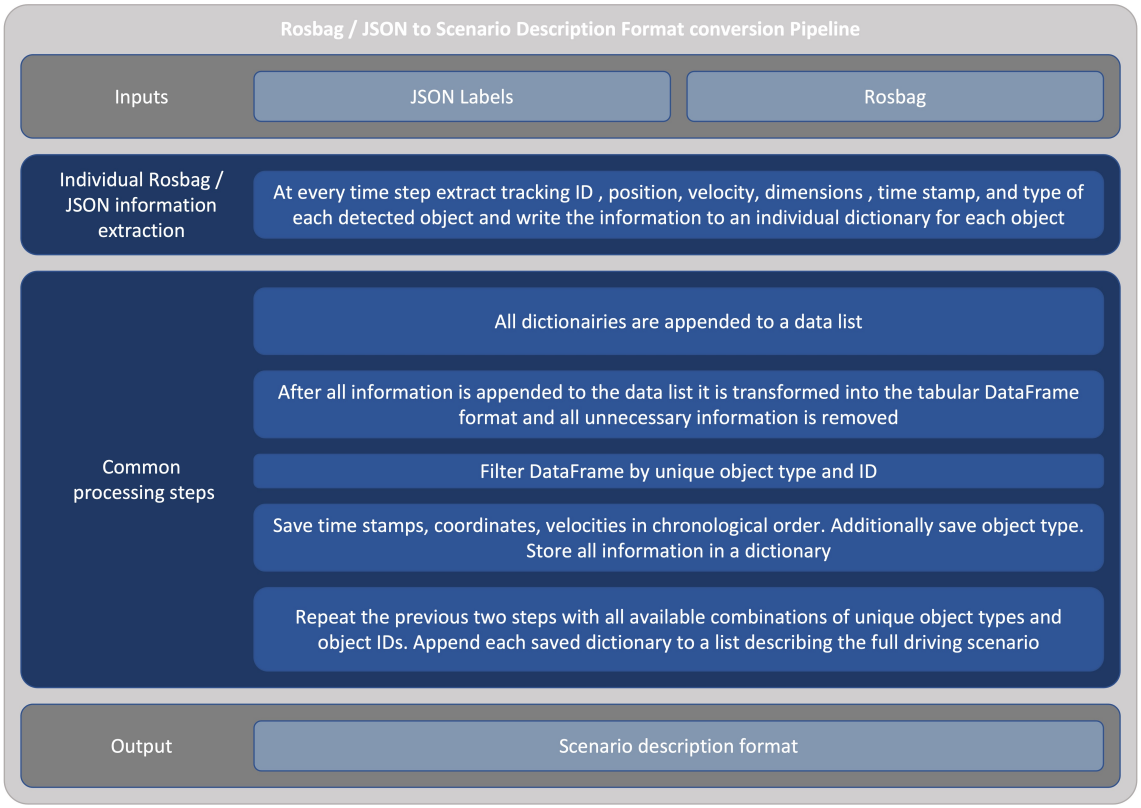

Figure 4.15: Description of the extraction pipeline steps from the raw data stored in rosbag or JSON files to the scenario description data type (source: own illustration).

4.5.2 Data Pre-Processing

The next step of the scenario mining process following the scenario extraction is data pre-processing. In this phase any data errors introduced in the extraction process are corrected. Figure 4.16 highlights the position of scenario extraction step in the entire scenario-based approach process model.

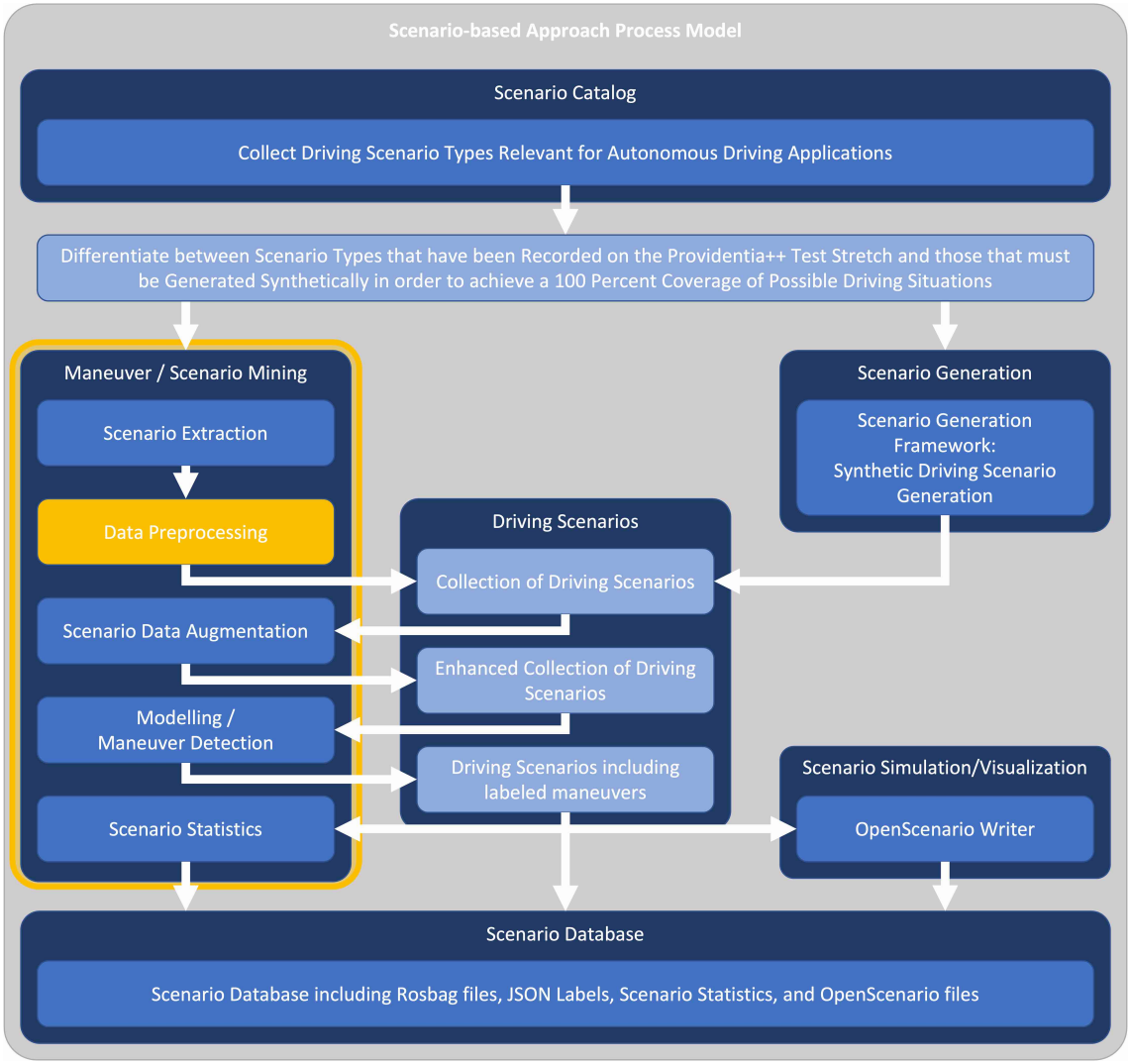

Figure 4.16: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

The raw data recordings from the sensor suite on the Providentia++ test stretch is processed by an object detection and classification model. As a result, traffic participants are assigned with bounding boxes and classified in a category such as car, truck, pedestrian, bicycle, and others. Although the perception models detect a reasonable amount of traffic participants correctly, errors in the exact size and position of the bounding boxes and also in the type classification occur commonly.

To ensure a solid basis for training perception algorithms, the extracted scenarios must be of high quality. For this, the raw data is pre-processed in multiple steps, which are described in the following sections. The steps follow the KDD methodology.

4.5.2.1 Handling of Missing or Faulty Data

The defects that occur in the recorded data are mostly bounding boxes with imprecise positioning and size, missing bounding boxes, inconsistent IDs, and falsely classified traffic participants. These errors all affect the trajectories of the traffic participants and must therefore be eliminated to ensure high quality scenarios are extracted. The following section describes how each of these defects is dealt with.

The unprecise bounding boxes are manually corrected with the labeling tool proAnno. Here the size, position, and orientation of the bounding boxes are adjusted. Analogously for traffic participants that are not detected by the detection model new bounding boxes are created. This step is especially important to ensure that no traffic participants that affect the scenario are left out. Figure 4.17 shows a visualization of the bounding boxes within proAnno.

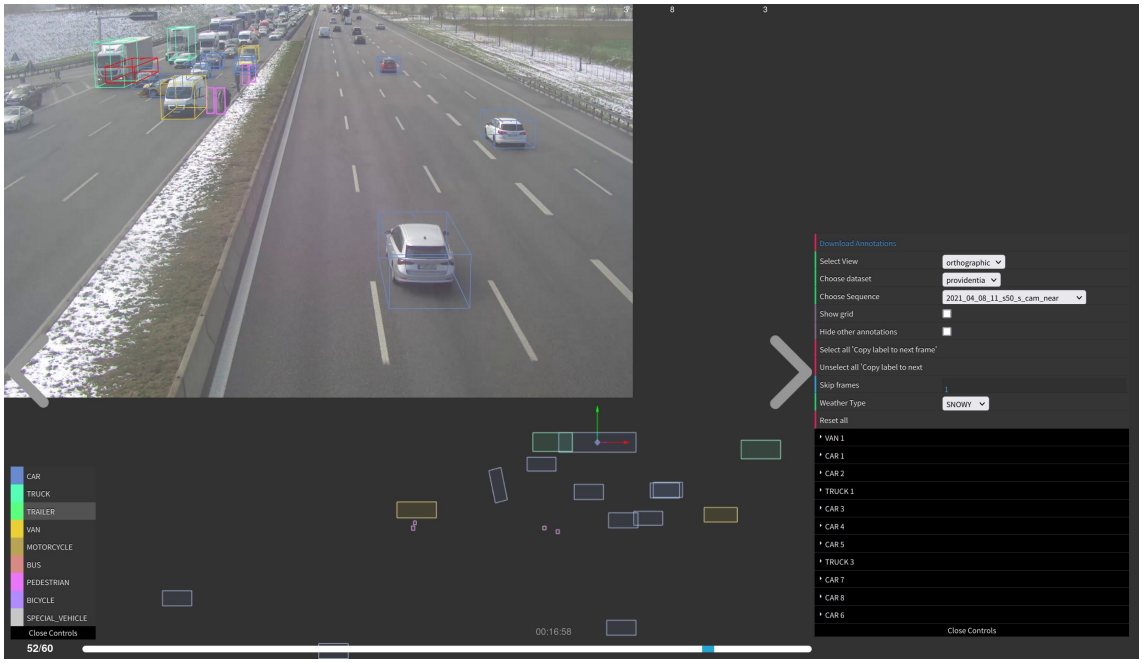

Figure 4.17: Visualization of the bounding boxes within proAnno (source: [136]).

Another important requirement to ensure that a traffic scene is semantically accurate is the correct classification of the traffic participants. If, for instance, a car is falsely classified as an ambulance vehicle, this potentially changes the semantic meaning of a scenario.

In a realistic situation, the surrounding vehicles carry out certain maneuvers to ensure the ambulance vehicle can pass them. In this case since the data is recorded from a real traffic scene where a normal car is driving instead of the detected ambulance vehicle the surrounding vehicles don't make any abnormal maneuvers.

If the scenario is interpreted semantically, this suggests that surrounding vehicles don't make any abnormal maneuvers when an ambulance vehicle is approaching them. In this case the falsely detected ambulance vehicle may have a negative impact on the training process of a trajectory prediction model that learns from this data.

ProAnno offers a selection menu where the type of traffic participants can be selected. This is used to manually correct any misclassifications.

The last defect type occurring in the data is inconsistent or mixed-up traffic participant IDs. The ID should be unique for each traffic participant and is initially determined by a tracking algorithm. Unfortunately, frequently errors occur such as mixing up two vehicles

with each other or not tracking a vehicle for some frames. This leads to the wrong assignment of IDs, which affects the data quality and must be corrected.

Especially when interpreting the data in a temporal matter, a consistent assignment of unique IDs for each traffic participant over all data frames is crucial. The unique IDs are required to allow an independent extraction of the trajectories of each traffic participant. This is a premise in order to be able to extract trajectories from the data recordings, which eventually play a crucial role in detecting vehicle maneuvers and scenario types. Without tracking vehicles over time the movement of each vehicle cannot be described as a trajectory. Without knowledge about the movement over time maneuvers such as lane changes, cut-ins, tailgate and more cannot be detected. Figure 4.18 illustrates the importance of object tracking for the detection of driving maneuvers.

Again, proAnno offers a selection menu where each traffic participant can be assigned an ID. The IDs are then created or corrected manually to ensure their uniqueness.

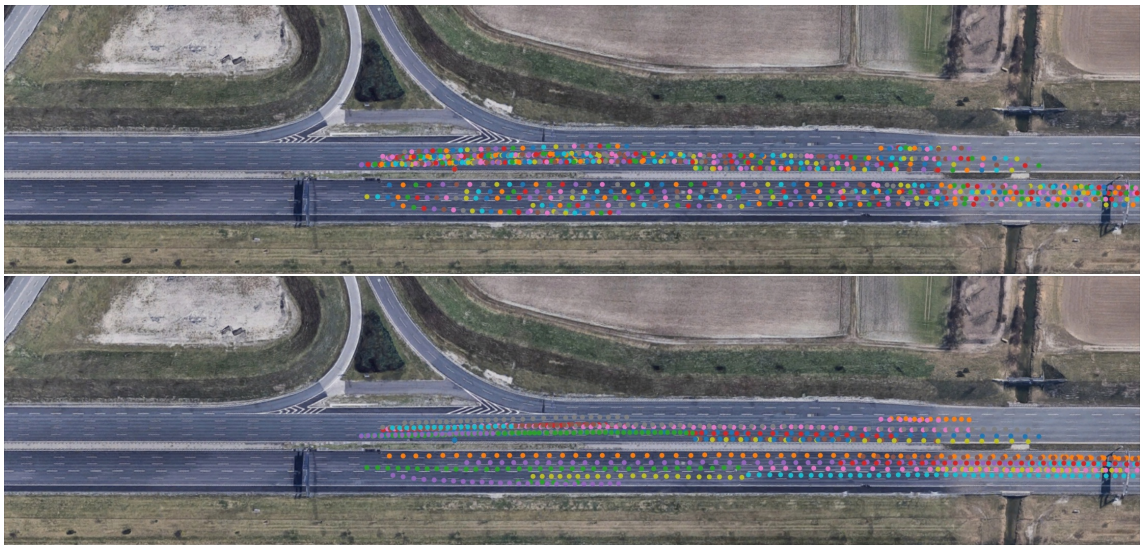

Figure 4.18: Examples illustrating the importance of object tracking for the detection of driving maneuvers. Each of the plotted points stands for a vehicle position at a certain point in time. The color of the point indicates the actors ID. The first plot shows the position of vehicles over time without tracking and therefore randomly sampled track IDs at each point in time. The second plot in contrast shows the position of vehicles over time with tracking. Here the points with same color belong to one specific vehicle. When examining each vehicles positions over time the vehicles trajectory and thus driving maneuvers can clearly be identified (source: own illustration, [1]).

4.5.2.2 Trajectory-Data Pre-Processing

The Providentia++ infrastructure provides a data stream containing camera, radar, and LiDAR sensor data. The data recorded by the various sensors is then processed to determine the position, orientation, size, and type of all traffic participants in each of the recorded frames. For the purpose of recognizing maneuvers and classifying scenarios the positional information is the most relevant, as it allows to analyze the movement of each traffic participant. Since the data stream is analyzed by a tracking algorithm, each vehicle has an individual ID which is consistent over all frames in which it appears. Therefore, with the individual ID and positional information a trajectory can be extracted for each vehicle. Concerning the

detection of maneuvers and scenario types, it is essential that the positional information is very accurate in order to achieve good detection results.

As shown in Figure 4.19, trajectories without applying any pre-processing steps can be inaccurate and contain missing values. To obtain trajectories of high quality two main pre-processing steps are applied.

1. Remove Outliers
2. Interpolate Missing Values

In Figure 4.19 the raw trajectories extracted from a 60 seconds long recording are pictured.

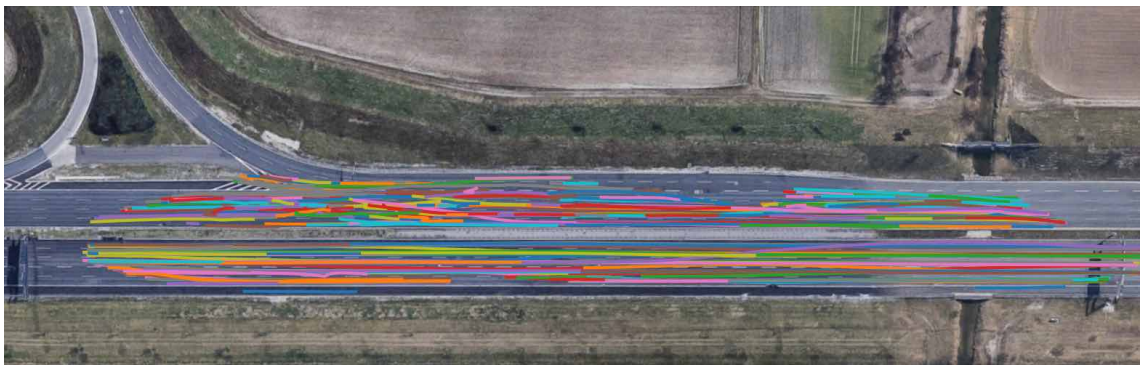

Figure 4.19: Plot showing raw vehicle trajectories on the Providentia++ test stretch over a time span of 60 seconds (source: own illustration, [1]).

4.5.2.3 Outlier Detection

The data recordings from the Providentia++ test stretch include some noise as well as outliers. In the data this is reflected in the recorded trajectories. Both paths and velocity profiles frequently contain outliers which result in unwanted spikes. An example of outliers in the velocity profile of one actor is shown in Figure 4.20.

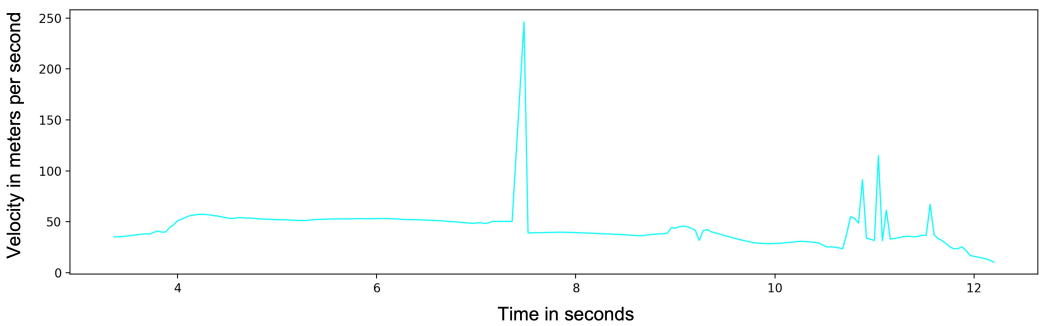

Figure 4.20: Illustration of how outliers affect the velocity profile of a vehicle recorded on the Providentia++ test stretch (source: own illustration).

In order to remove such outliers from a velocity profile a simple two-step outlier detection algorithm is implemented. The first step calculates the so called "moving average" over the entire velocity profile. The "moving average" is a filtering strategy typically used for noise reduction. It recalculates each value of a time series by averaging over N neighboring values. This reduces the fluctuations induced through noisy data. The affect of applying the "moving average" to a noisy velocity profile is shown in Figure 4.21.

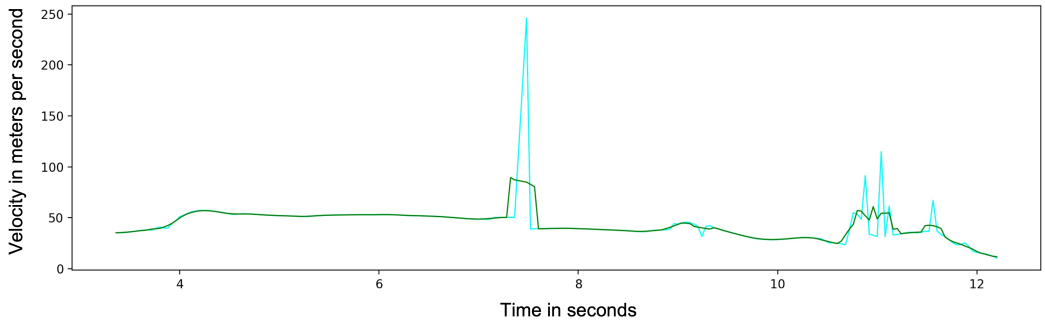

Figure 4.21: Application of the "moving average" filter. The original velocity profile is shown in cyan while the filtered profile is shown in green. (source: own illustration).

As seen in Figure 4.21 applying the "moving average" already leads to quite a good smoothing result. Nonetheless, outliers still substantially influence the correctness of the velocity profile. Therefore, in the second step of the outlier detection, the difference between the filtered result and the raw velocity profile is calculated. The result is shown in Figure 4.22.

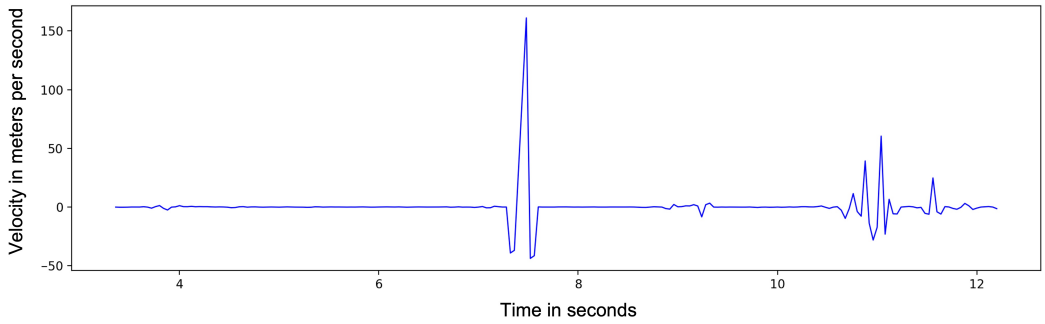

Figure 4.22: Difference between the raw velocity profile and the one filtered with the "moving average" algorithm (source: own illustration).

As seen in Figure 4.22 the filtered trajectory profile follows the original unfiltered profile very tightly, unless a major spike occurs. For this reason, when calculating the difference between filtered and unfiltered velocity profile, the result is mostly close to zero unless an outlier appears. With this information, all points that lie above a defined threshold are detected as outliers and removed from the data. In a last step, the outlier adjusted velocity profile is interpolated to obtain the true velocities in the sections where outliers are detected. Figure 4.23 shows an example of a raw-, filtered-, and final interpolated velocity profile.

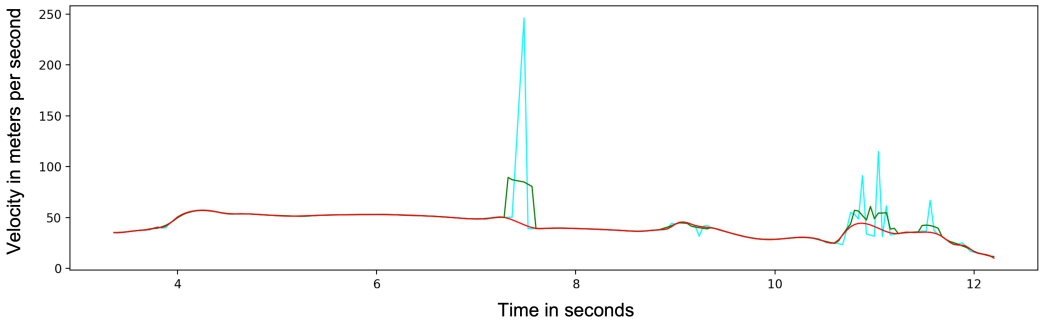

Figure 4.23: Visualization of the transition from the original velocity profile (cyan) to the filtered profile (green) to the final smoothed velocity profile (red) (source: own illustration).

4.5.2.4 Discretization

The positional and temporal data describing each actor's trajectory is given in the recordings and synthetic scenarios as a set of coordinate points and a corresponding set of timestamps. Here, the coordinate points describe the actor's path, whereas the timestamps describe the point in time when the actor reaches each of the coordinate points. Both features are already provided in a discrete format by the data recordings from the Providentia++ test stretch. Therefore, no additional discretization is necessary as part of the data pre-processing step. The synthetically created driving scenarios are already generated in a discrete format. This makes a discretization step obsolete.

4.5.2.5 Normalization

Data normalization is often beneficial when applying various machine learning algorithms [164, 166]. In this work, however, the maneuver detection algorithm is modeled by considering the absolute positions of an actor at each time step and bring these positions into context with several regions of interest along the Providentia++ test stretch. This means that both the vehicle trajectory as well as the position of each region of interest must be known in absolute coordinates. A normalization step is not necessary in this case.

4.5.3 Scenario Data Augmentation

As described in Section 2.15, many success stories have occurred with the use of data augmentation techniques. Usually, the outcome is a higher model performance [163, 193].

Especially in the area of autonomous driving, which is highly safety critical it is crucial to cover as many scenario types and variations as possible. This ensures robust algorithms can be trained on a set of driving scenarios that sufficiently cover the conditions in which the automated vehicle is designed to operate.

In this section, a scenario augmentation framework is introduced and described. Its aim is to enhance the set of available driving scenarios. The basic idea behind the augmentation framework is to create multiple stochastic variations of a base driving scenario. This way only one example of each driving scenario is needed. All variations of this scenario that could occur in reality are then generated by the scenario augmentation framework. The position of the introduced augmentation framework in the entire scenario-based approach process

model is highlighted in Figure 4.24.

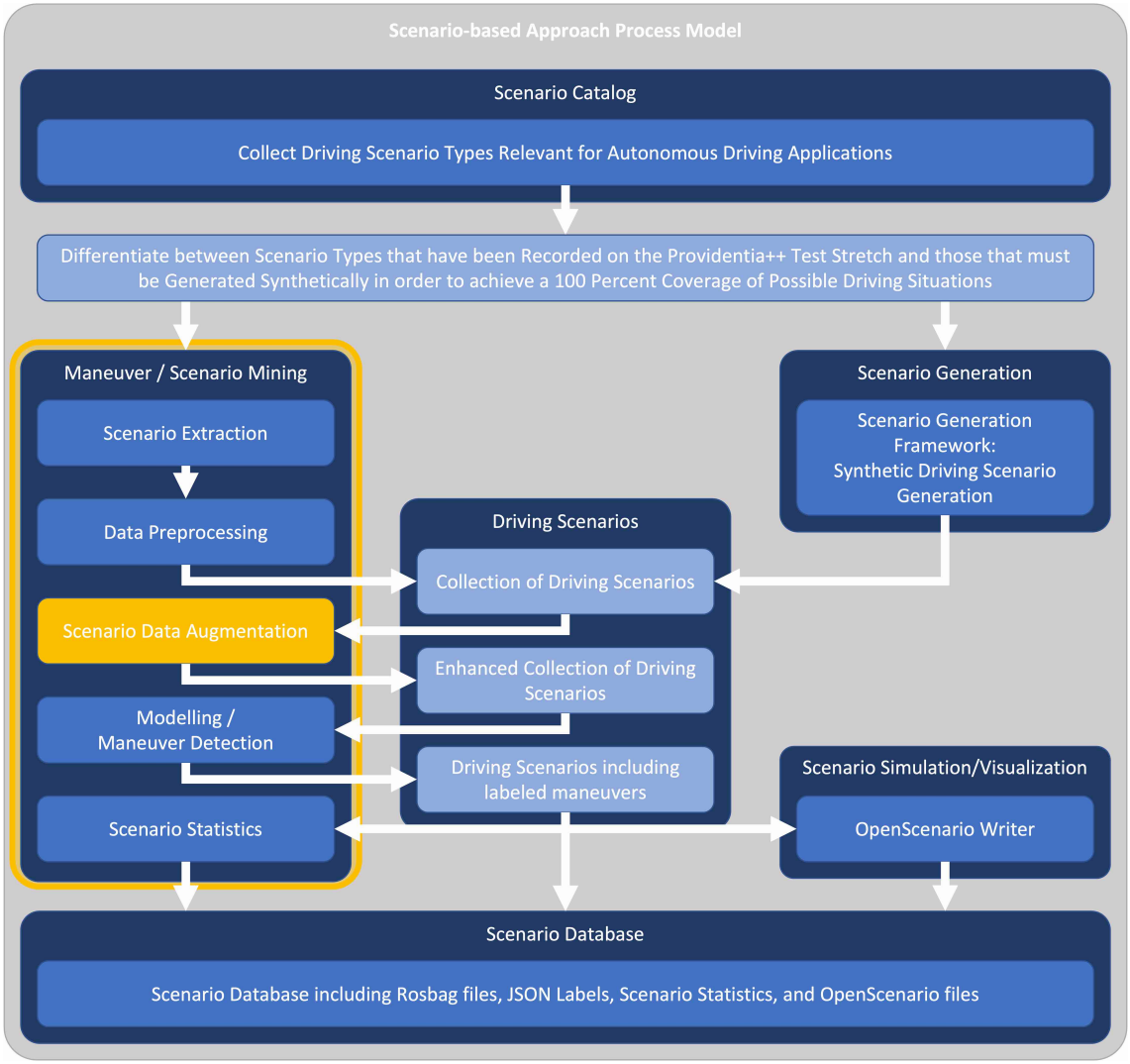

Figure 4.24: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

For the augmentation of vehicle trajectories, a stochastic sampling-based approach is used to achieve nearly unlimited and highly realistic augmentation possibilities. In addition, the stochastic sampling behavior is highly parametrizable and can be adjusted to the user's needs. The trajectory augmentation includes adjustments in the vehicle path and velocity at each time step.

Other aspects of the driving scenarios, such as the vehicle type and color, weather conditions, and time of day are sampled randomly from a discrete pool of combination possibilities. An overview of the sampling pool is given in Table 4.2. In the following sections, both the trajectory augmentation and the sampling of the remaining scenario features are explained. Figure 4.25 illustrates the steps included in the augmentation framework.

Category	Format	Sampling Pool
Vehicle type	Vehicle description in string format Example: 'Tesla Model S'	All vehicle types currently supported by the CARLA simulator
Vehicle color	Three integer values defining a color on the RGB color scale	Three integer values each in the range between zero and 255 Example: color red (255,0,0)
Time of day	Time in the format hour-minute-second (hh-mm-ss)	Three integer values where the value defining the hour must lie in the range zero to 24 whereas the values defining minute and second lie in the range zero to 60
Weather conditions	Weather conditions description in string format Example: 'sunny'	All weather conditions supported by the CARLA simulator

Table 4.2: Overview of sampling pool for the augmentation parameters vehicle type and color, time of day, and weather conditions (source: own illustration).

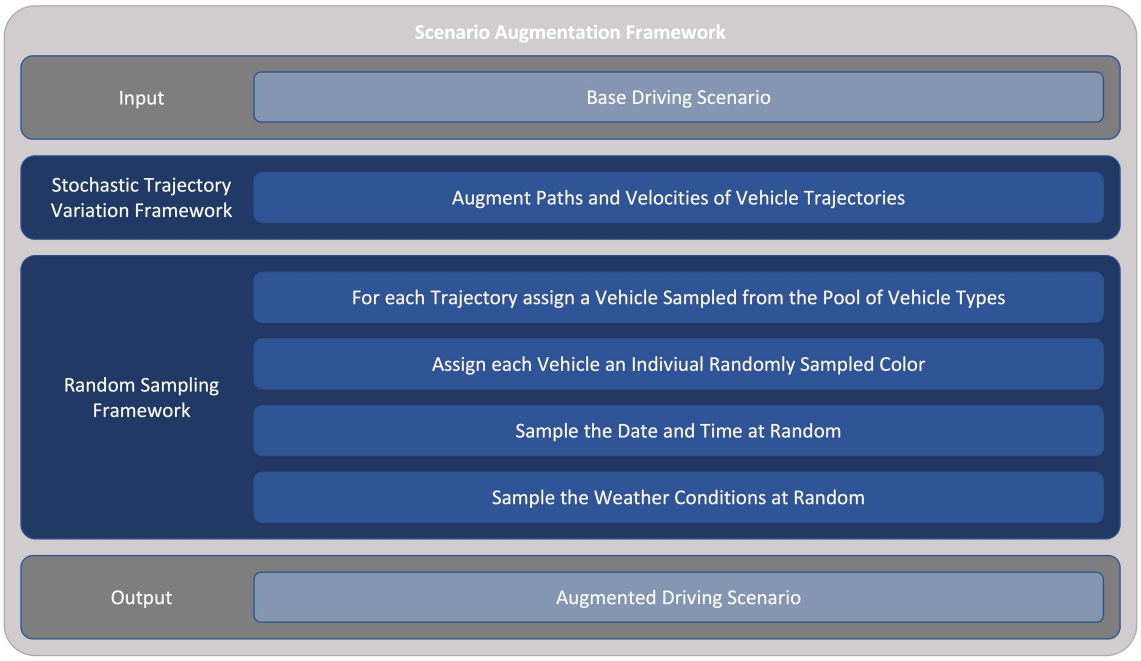

Figure 4.25: Illustration of the steps included in the scenario augmentation process (sources: own illustration).

4.5.3.1 Trajectory Augmentation

The main idea of the trajectory augmentation framework is to take an existing trajectory as an input and variate it inside certain boundaries so that the augmented trajectory still mirrors a realistic driving behavior, but is yet different from the input. The base trajectory used as input for the framework can for example be recorded from a real-world driving scenario or generated synthetically based on realistic assumptions. Additionally, the level of variation can be controlled by user-defined parameters. The results of different parameters are shown

in Figures 4.26 and 4.27 where the same three trajectories are augmented with a high and low degree of variation.

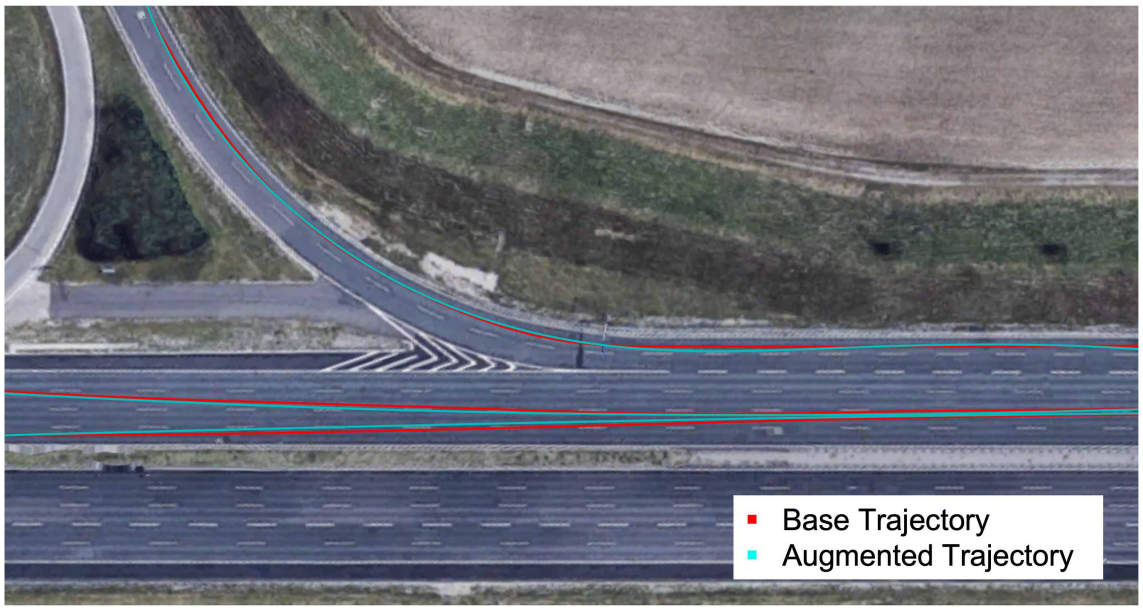

Figure 4.26: Example of three trajectories augmented with Gaussian noise of zero mean and 0.2 meters standard-deviation (sources: own illustration, [1]).

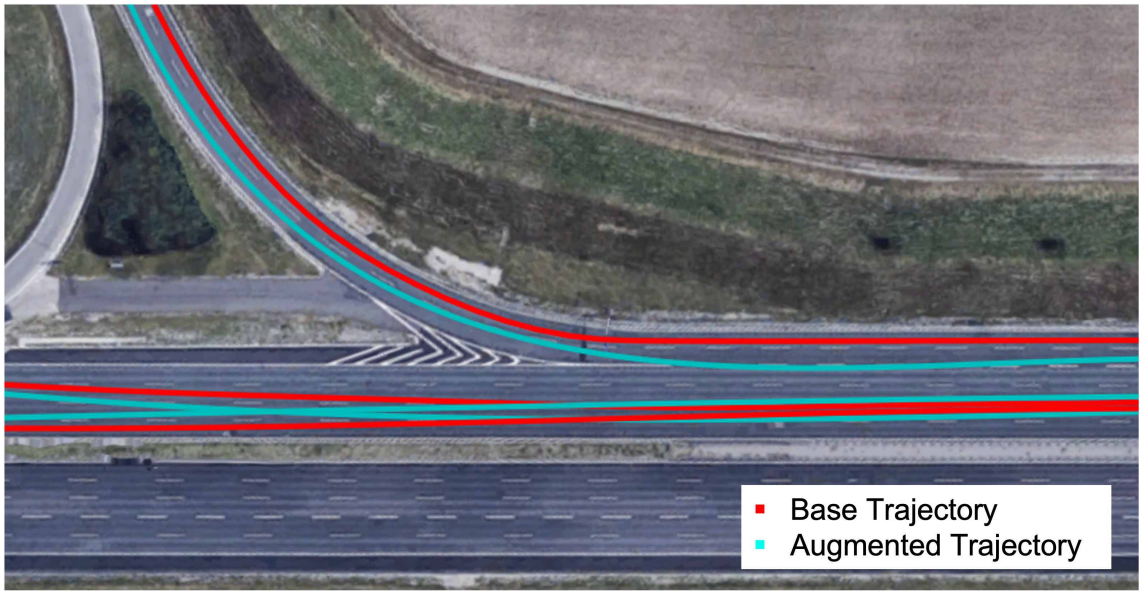

Figure 4.27: Example of three trajectories augmented with Gaussian noise of zero mean and 2 meters standard-deviation (sources: own illustration, [1]).

Looking at the framework in detail it consists of four major variation steps:

1. Create Offset from Path

   In the first variation step, the new trajectory is offset parallel to the base trajectory by a value sampled from a normal distribution. The intuition behind this step is that, while most drivers are likely to drive in the center of a lane, some may tend to drive closer to

left or right bound of the lane. By sampling the offset value from a normal distribution, this behavior is reflected since most trajectories are sampled close to the base trajectory and fewer are sampled with large offsets. In other words, it is more likely to sample a new trajectory with small offset to the base trajectory than one with a larger offset.

2. Add Gaussian Noise to Path

   To further increase the realism of the path variations, in the second step normal-distributed noise is added to the way points of the trajectory at each time step. This way, the new trajectory does not have a constant offset from the base trajectory anymore and, additionally, some swaying movement is added to the path. The mean and standard deviation is parameterized and can be adjusted to the users demands. Since the noise is sampled individually for each way point, interpolating the calculated way points results in a zig-zag path with many sharp turns which is depicted in Figures 4.28 and 4.29.

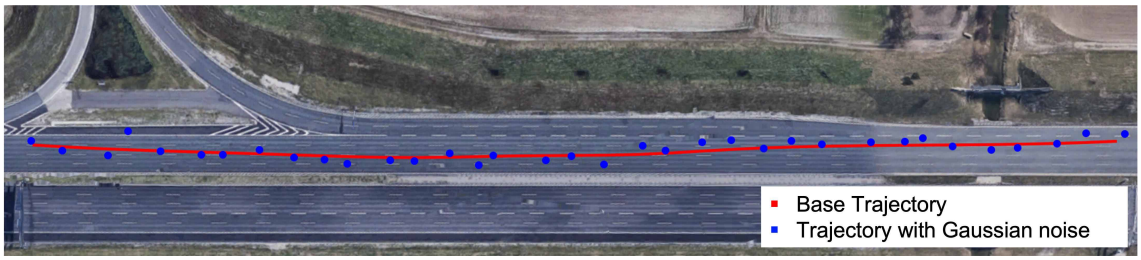

Figure 4.28: Example of Gaussian noise with zero mean and 2 meters standard-deviation added to the way points of the base trajectory (sources: own illustration, [1]).

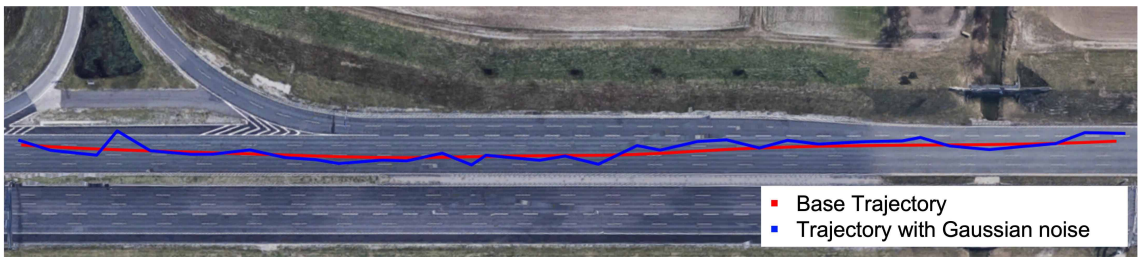

Figure 4.29: Example of a path with many sharp turns resulting from the addition of Gaussian noise to the base trajectory (sources: own illustration, [1]).

This behavior is not intended simply because it does not mimic a natural driving behavior. More than this, it leads to a dilemma since adding normal distributed noise is essential for creating some swaying movement but on the other side the new path could contain sharp turns that physically cannot be completed by a vehicle. This problem is addressed and solved in the next step.

3. Smoothing

   In order to remove the sharp turns that occur after adding noise to the path, a smoothing algorithm is applied. The result is a path with gentle curves that mimics a more realistic driving behavior. The benefit of this method is that the swaying movement generated through the application of normal-distributed noise to the path is kept, while removing all physically impossible sharp turns. Figure 4.30 shows an example of how the smoothing algorithm affects the noisy path.

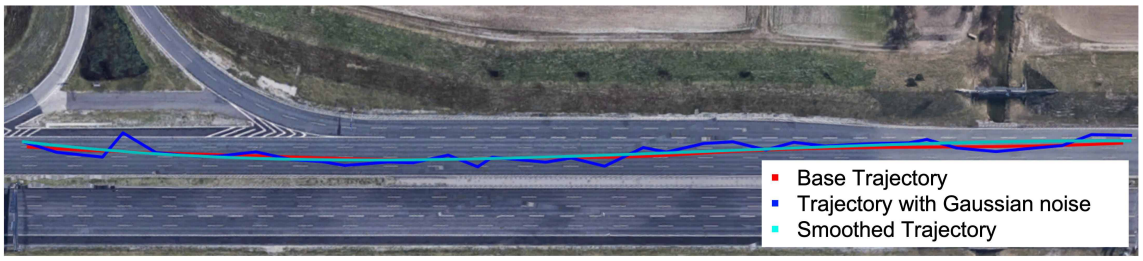

Figure 4.30: Example of how the smoothing algorithm removes sharp turns from the noisy path but maintains the swaying motion (sources: own illustration, [1]).

4. Velocity Augmentation

   In the last step the velocity of the vehicle is varied at each time step. This is done by once more adding normal-distributed noise to the velocities defining the base trajectory. Again, a smoothing algorithm is applied to remove abrupt changes in vehicle speed and maintain a realistic velocity profile.

4.5.3.2 Vehicle Type and -Color, Time of Day, and Weather Conditions Augmentation

Next to the vehicle trajectories, other factors with a strong influence on the scenario are the vehicle type and color, time of day, as well as the weather conditions. To ensure a large diversity of these factors within the collection of driving scenarios, they are all sampled at random.

1. Vehicle Type

   In practice, different vehicle types show different driving behaviors. This can be due to the physics connected to the vehicle, but also due to the purpose of the vehicle. For instance, a firetruck is usually a very heavy vehicle that generally drives slower than a much more dynamic sports car. In certain situations however, like a fire emergency the firetruck will speed up and possibly perform some abnormal maneuvers. For this reason, an important task in autonomous driving is not only object detection, but also classification so that the behavior can be predicted more accurately. To train an algorithm appropriately for this task, all vehicle types must be represented in the training data. How this is ensured in the augmenting driving scenarios, is explained as follows.

   To select a random vehicle in a first step, the vehicle types included in Carla are all stored in a list. A random integer is sampled in the range of the lists indices. The sampled integer is then used as an index to select a vehicle from the list. For example, if the list includes 10 vehicles, then an integer in the range zero to nine is sampled. Let's assume, for example, the number 7 is sampled. Then the vehicle in the list at position 7 is selected and returned as the output of the sampling process. The procedure is visualized in Figure 4.31.

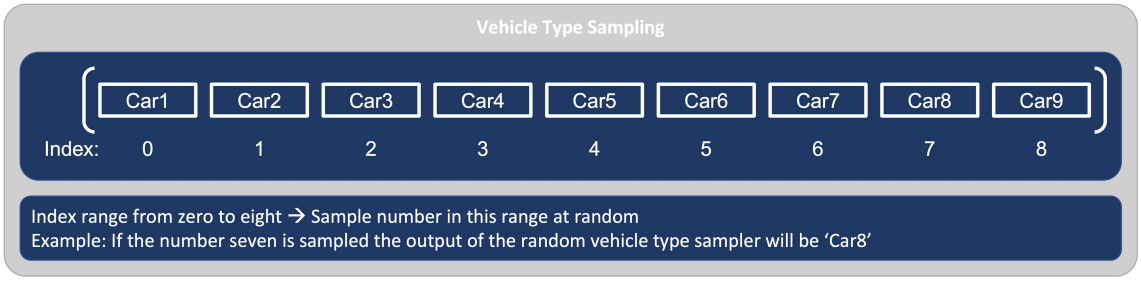

Figure 4.31: Example of how a vehicle type is sampled at random (sources: own illustration).

2. Vehicle Color

   For the computer vision task of object detection or in more specific vehicle detection, it is beneficial to train an algorithm on vehicles with a large range of different colors. This way, the algorithm learns to detect vehicles based on geometric features that may appear differently depending on the vehicle color.

   The vehicle color displayed by the Carla simulator is defined by an RGB color scale. RGB colors are created by adding various intensities of the base colors red, blue, and green to obtain the desired color. The intensity scale for each base color ranges from zero to 255.

   To ensure that all color combinations of the RGB scale can be achieved, three integer numbers are sampled individually at random inside the range zero to 255. Each of the values represents the intensity of either the r, g, or b value. These values are then passed on to the Carla simulator and define the vehicle color. The process is illustrated in Figure 4.32.

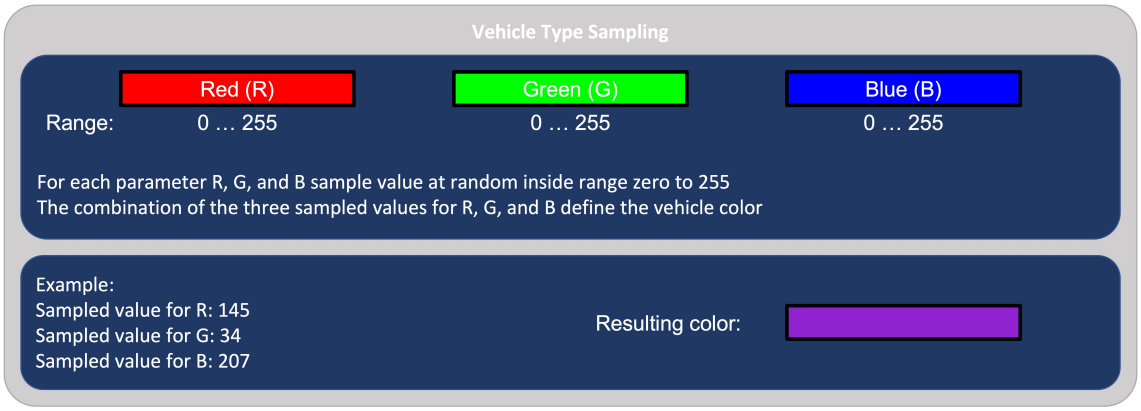

Figure 4.32: Example of how a vehicle color is sampled at random (sources: own illustration).

3. Sample Time-of-Day

   In the real world, the perception of the environment can be very different depending on the time-of-day. For instance, objects, lane markings, and other traffic participants can be perceived differently during night time, dusk, or dawn. This is mostly due to the different lighting conditions at these times compared to broad daylight.

To reflect these conditions in the *proScenario dataset*, the date and time at which a scenario takes place is sampled randomly. The sampling range lies between January 1st, 2020 at midnight and the current date and time when the sampling algorithm is used.

4. Weather

   Similar to the time-of-day, the weather conditions can have a strong influence on the behavior of the actors in a scenario. For example, making a sharp turn at 30 kilometers per hour may be possible in warm and dry conditions, but could lead to drifting off the intended path if it is cold and the road is icy.

   Weather conditions are therefore sampled randomly from the pool of weather types supported by Carla and OpenSCENARIO, which are stored in a list. Analogously to the sampling of the vehicle types, an integer is sampled in the range of list indices. The list element at the index position of the sampled integer is then returned. This way the weather conditions are sampled randomly. Figure 4.33 shows a driving scenario simulated with the CARLA Simulator in four different weather conditions.

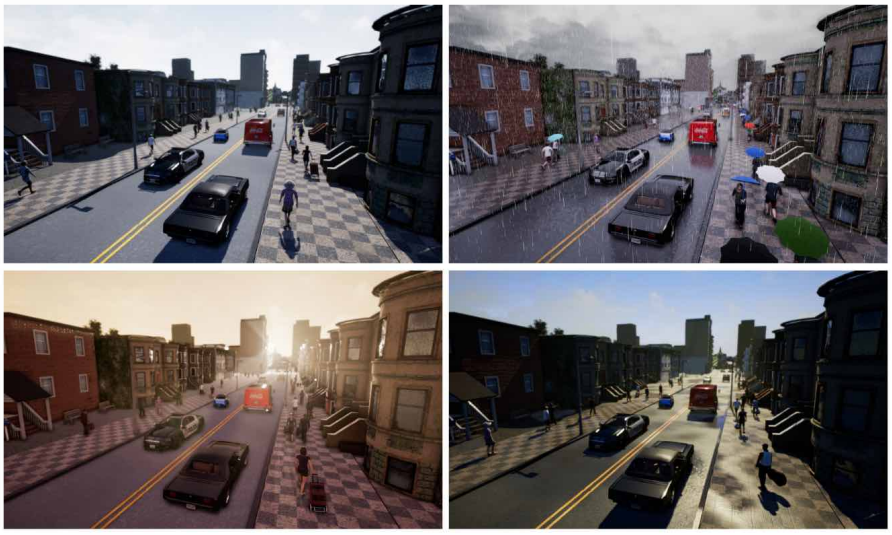

Figure 4.33: Driving scenario simulated with the CARLA Simulator in four different weather conditions (sources: [53]).

4.5.4 Feature Selection

In this step the features that contain information relevant for the detection of maneuvers are selected from all of the features that are included in the data recordings. Specifically those which determine the actor's positions and the timestamps. They define the trajectory as well as the velocity profile of the actor while following his trajectory. Additionally, the feature ID, containing each actor's unique ID is selected for the purpose of matching each detected maneuver to the corresponding actor.

Other features such as the actor's orientation as well as the weather conditions are not selected as the information stored inside these features is not relevant for detecting maneuvers.

4.5.5 Feature Extraction

In order to detect certain driving maneuvers – for instance, a lane change – it is beneficial - if not necessary - to extract additional features. A detailed description of which features are extracted and how this is done is given in the following sections. The extraction of new features is the last pre-processing step and fundamental for the automated detection of driving maneuvers.

4.5.5.1 Feature: Lane ID

For the feature “Lane ID”, the position of an object in the traffic scene is labeled. This means, that depending on where an object is located, it is assigned a label such as “lane 1”, “lane 2”, etc. if it is positioned inside one of the traffic lanes. If the object is not located on a road area, the lane ID is labeled as "None". The bounding box of a traffic participant could lie either on a single road segment or on multiple road segments, for example, “lane 1” and “lane 2” if it is changing lanes. This would lead to unwanted ambiguity. Since it is desirable for the label to be unambiguous, the center point of the object bounding box is used as the object position. In general, this means that the lane ID of the road area where the object center is located on is assigned.

To obtain the correct road position labels the different segments such as lanes or sidewalks are defined by polygons. The polygons create a hull which is an exact outline of the lane or sidewalk. This works for both straight and curved road segments. After the polygons are defined correctly, for each object it is tested whether the center point lies in one of the polygons. This is done by an algorithm which tests whether a point lies inside the hull of a polygon. Depending on which road segment the point falls into, it is labeled accordingly. The procedure is visualized in Figure 4.34.

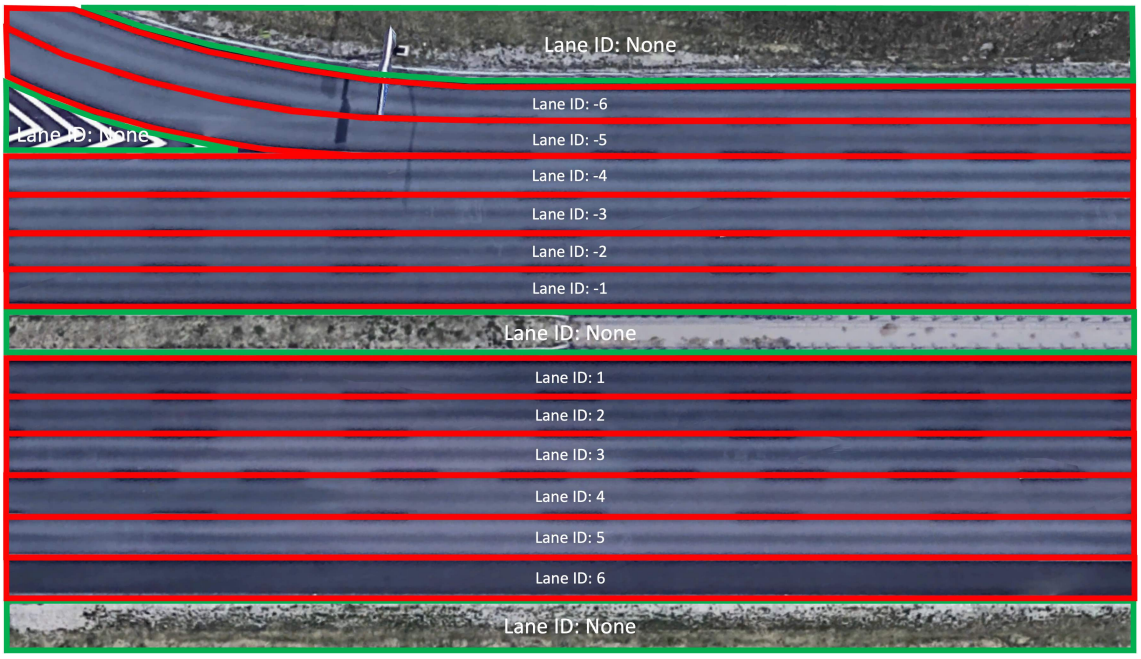

Figure 4.34: Description of road lanes with polygons (sources: own illustration, [1]).

4.5.5.2 Feature: Offset from Lane Center

Another extracted feature is the distance of a vehicle to the center of its current lane. The distance is calculated for each vehicle at every time step. This information is valuable as it allows to detect lane change maneuvers quite simply. Two main steps are taken to calculate the lateral offset from the lane center.

1. First, it is checked on which lane each vehicle is driving at the current time step. This is done by analyzing the result of the lane ID extraction, which already assigns the correct lane ID for each vehicle.
2. With the information of the correct lane ID the corresponding lane center line can be selected. Then simply the shortest distance between the vehicles center of mass and the center line is calculated using the euclidean distance metric.

4.5.5.3 Feature: Distance to Leading / Following Vehicle

The last feature extracted is the distance of a vehicle to the closest leading vehicle and the closest following vehicle. This is a quite challenging task as it requires two main prerequisites:

1. First, it must be determined on which lane each vehicle is driving and which other vehicles are also driving on this lane. This is an important step as only vehicles driving on the same lane can be considered as leading vehicles or following vehicles.

   Algorithmically, this is done by iterating through all vehicles present in the scenario at the specific time step and sorting them by their lane ID. Now, all vehicles driving on the same lane are grouped.
2. Next, for each group of vehicles driving on the same lane, all pairwise combinations are determined. For example, if vehicle (1,2,3) are driving on the same lane, the pairwise combinations resulting are (1,2), (1,3), and (2,3). For each of these pairs, the distance along the lane center line from the start of the lane to the corresponding vehicle is calculated. Then the difference between the calculated distance of the first vehicle and that of the second vehicle is calculated. If this distance is positive, the first vehicle of the pair is driving in front of the second vehicle. If the calculated distance is negative, the opposite is the case - the second vehicle leads the first vehicle. For the leading vehicle the absolute value of the calculated distance is added to a list containing the distances to all following vehicles. For the following vehicle, the absolute value of the calculated distance is added to a list containing the distances to all leading vehicles. This process is repeated for all pairwise combinations. As a result, each vehicle is assigned with a list containing the distances to all leading vehicles and another list containing the distances to all following vehicles. From each of these lists the minimum value is determined which marks the distance to the closest leading and the closest following vehicle. If there is no leading or following vehicle this is marked by setting the distance to -1.

The entire process is repeated for each driving lane and calculated at each time step of the driving scenario. An Example is shown in Figure 4.35.

Figure 4.35: Example of how the distances between vehicles along the road is calculated (own illustration).

4.5.6 Modeling

The modeling phase deals with the detection of driving maneuvers. In this phase algorithms are modeled that can detect driving maneuvers based on their trajectories as well as other available information such as the road network. The position of the modeling phase in the entire scenario-based approach process model is highlighted in Figure 4.36.

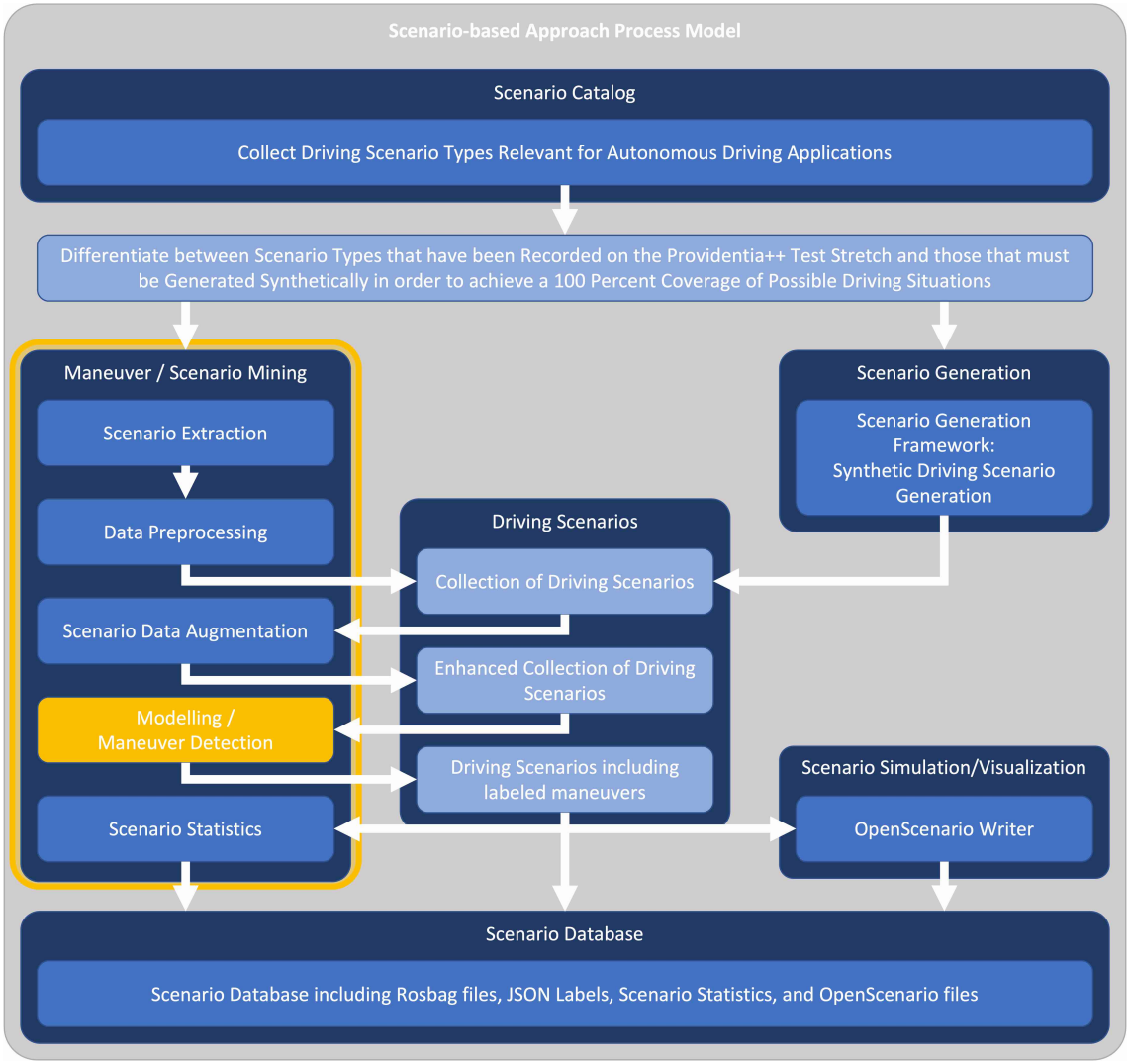

Figure 4.36: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

To classify driving scenarios in a first step driving maneuvers occurring in the scenario are detected. Based on detected maneuvers solely or in combination with additional factors, such as traffic lights or signs, a traffic scenario can be determined. For this various detection models are implemented to recognize different maneuvers. The details concerning the functionality and implementation of these models is explained in the following sections.

4.5.6.1 Overview of Detected Driving Maneuvers

In this section, an overview is provided over the driving maneuvers and scenario types that are detected and labeled (Fig. 4.37). The reasoning for the inclusion and how driving maneuvers are extracted is explained independently for each of the maneuvers and scenario

types in Sections 4.5.6.2 to 4.5.6.9.

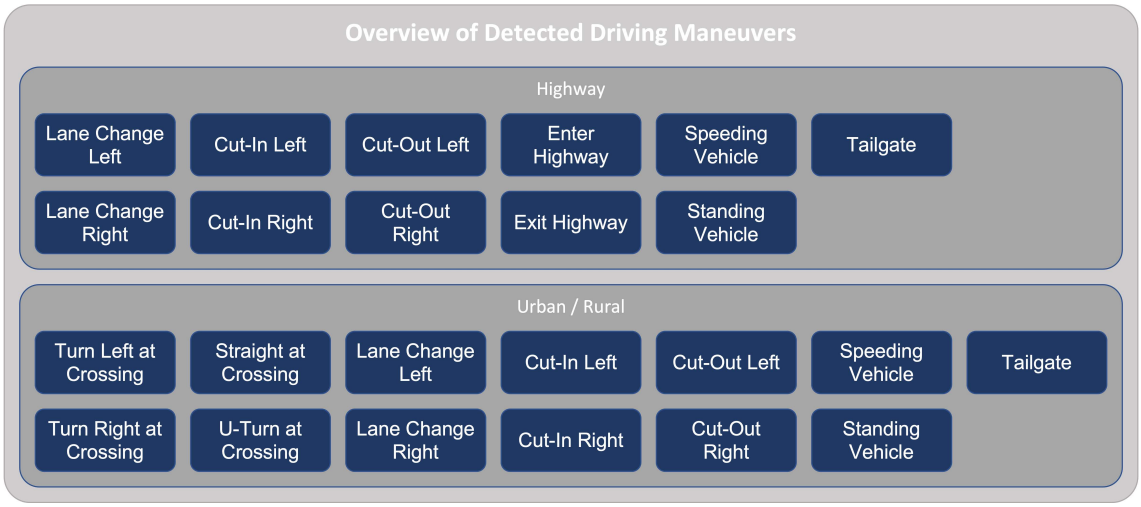

Figure 4.37: Overview of the detected driving maneuvers (source: own illustration).

4.5.6.2 Label: Lane Change

Lane changes are one of the most frequent traffic maneuvers and a large contributor to traffic accidents [143]. This means it is very important for intelligent algorithms to predict lane changes of surrounding vehicles as soon as possible in order to avoid potentially dangerous situations. When performing a lane change, the vehicles trajectory typically has the shape of an s-curve. Existing approaches to detect lane changes therefore take the s-curve of a standard lane change as a basis [108]. Vehicle trajectories are then compared to this s-curve and if the deviation to the base curve is below a certain threshold, a lane change is detected [108]. This procedure works well on straight roads like highways, but tends to produce false positives on roads that can have s-shape segments. This problem is depicted in Figure 4.38.

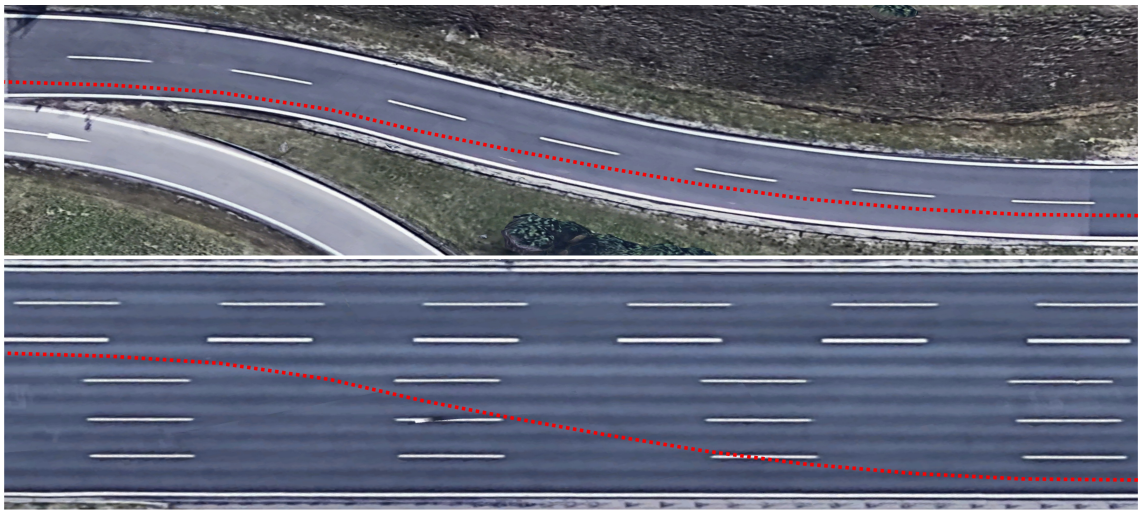

Figure 4.38: Example of two identical s-curve trajectories. On the top picture the trajectory simply follows the road while on the lower picture three consecutive lane changes are completed (source: own illustration, [1]).

In order to account for this problem, a method is developed that identifies lane changes based on the offset to the center line of the lanes a vehicle is changing between. The details of this method are explained in the following.

To label when a vehicle changes lanes a detection algorithm consisting of three main steps is implemented:

1. In the first step, it is detected when the vehicle crosses a lane marking based on the lane ID of a vehicle. This information can easily be extracted by iterating through the data frames and checking if the lane ID of a vehicle changes in the following frame. This is graphically illustrated in Figure 4.39. Since a vehicle must cross a lane marking in order to perform a lane change this is already a fairly good indicator for detecting lane changes. Using lane crossings as an indicator for lane changes all true positives and true negatives and no false negatives are detected. Nevertheless, quite frequently false positives are detected. Some reasons for this are vehicles crossing the lane marking but not performing a lane change while swaying in their current lane or avoiding small obstacles. An example of a vehicle crossing lanes but not performing a lane change is given in Figure 4.40.

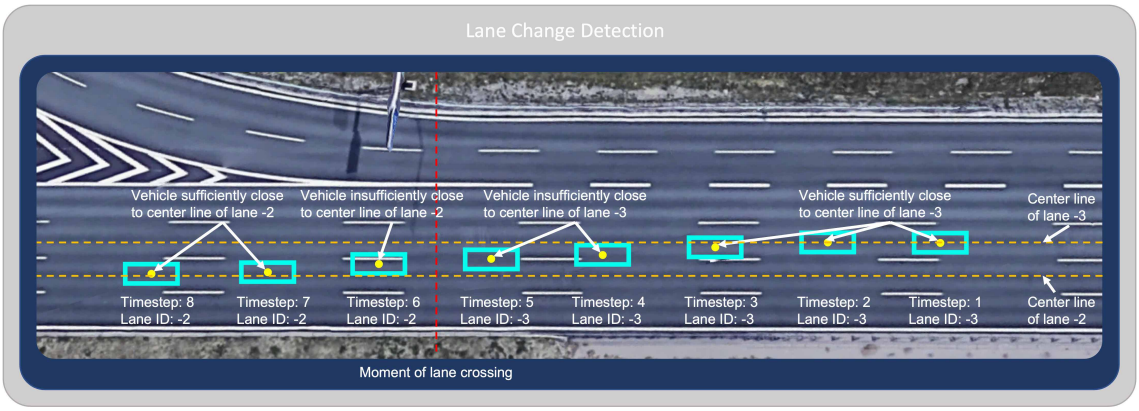

Figure 4.39: Visualization of the lane crossing detection (sources: own illustration, [1]).

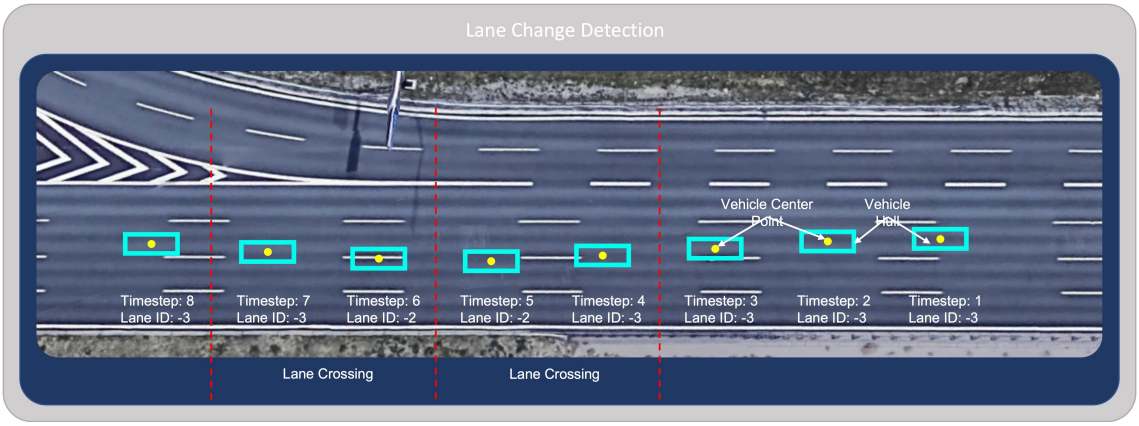

Figure 4.40: Example of a vehicle crossing lanes multiple times but not performing a lane change (sources: own illustration, [1]).

2. In order to eliminate the false positive detection of lane changes, a measure must be

found that can differentiate whether or not a lane change is completed after a vehicle crosses a lane marking.

To find such a measure first the characteristics of a lane change must be determined. It can be observed that at the beginning and end of a lane change maneuver the vehicle is located close to the middle line of the according lanes it is switching between. For the lane change detection algorithm developed in this work a vehicle is defined as close to the lane center if its lateral distance to the lane center is below one meter.

In order to perform a lane change the vehicle at some point by definition must lie outside the region that is considered close to the center line of a lane. This is the case when crossing lane markings. Therefore, whenever a lane marking is crossed the last time the vehicle was close to the lane center and the next time the vehicle is close to the lane center is determined. In the following the lane id of the vehicle is checked at both of these points. If the lane id is the same at both points then no lane change has happened. If the lane id is different at both points this clearly indicates a lane change. Also it can be determined if a lane change to the left or right is made depending on the lane id at the beginning and end of the lane change. The explained procedure is graphically illustrated in Figure 4.41.

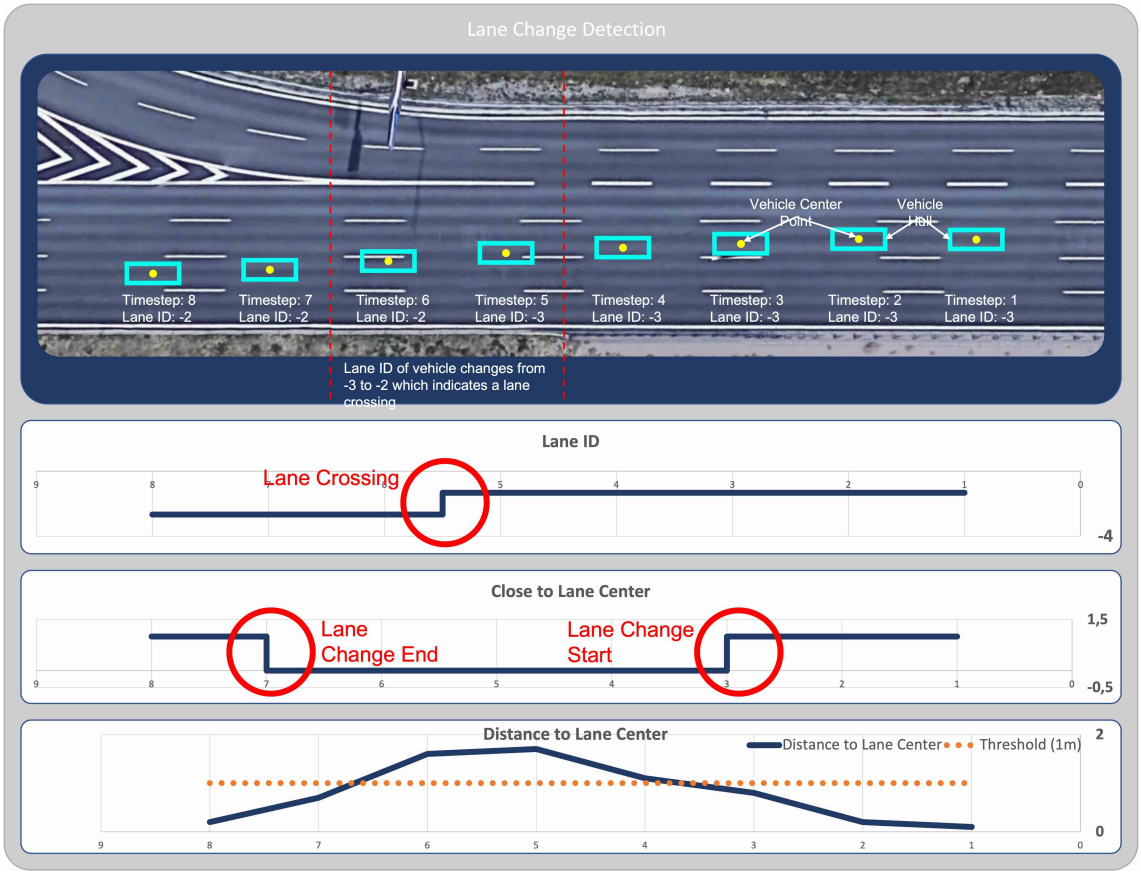

Figure 4.41: Example of a lane change detection (sources: own illustration, [1]).

4.5.6.3 Label: Cut-In / Cut-Out

One of the most dangerous traffic maneuvers according to [57, 143] are cutting-in and cutting-out. These maneuvers are basically the same as lane change maneuvers, but with some additional specific characteristics.

A cut-in occurs when a vehicle performs a lane change in front of another vehicle, while having a time-to-collision of less than two seconds between the following vehicle and itself. A cut-out maneuver occurs when a vehicle already has a time-to-collision of less than two seconds to the leading vehicle and then performs a lane change [57].

Time-to-collision (TTC) is a metric used to quantify the level of risk colliding with another vehicle or object. It is defined by [140] as follows:

$TTC = \Delta v / \Delta d$
with $\Delta v = velocity\ difference\ between\ following\ and\ leading\ vehicle$ and $\Delta d = absolute\ distance\ between\ following\ and\ leading\ vehicle$

Examples of a cut-in and cut-out maneuver are shown in Figures 4.42 and 4.43.

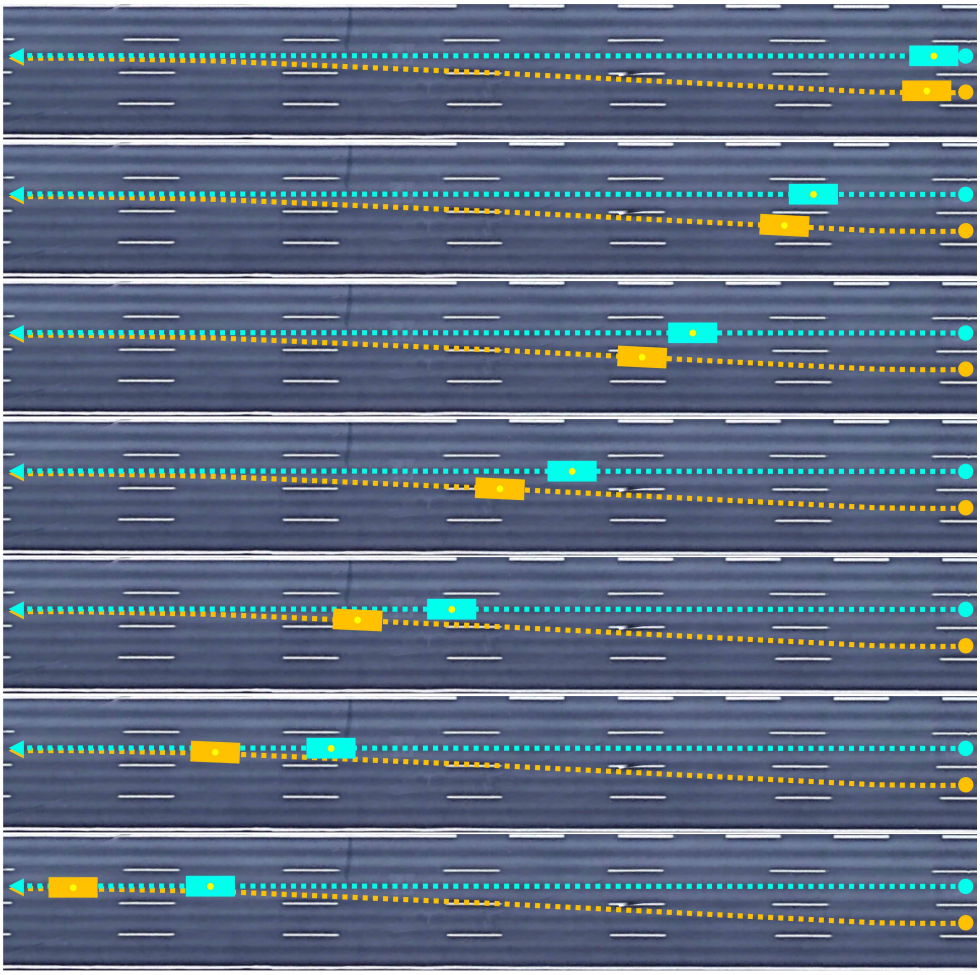

Figure 4.42: Example of a typical cut-in maneuver performed by the orange vehicle (sources: own illustration, [1]).

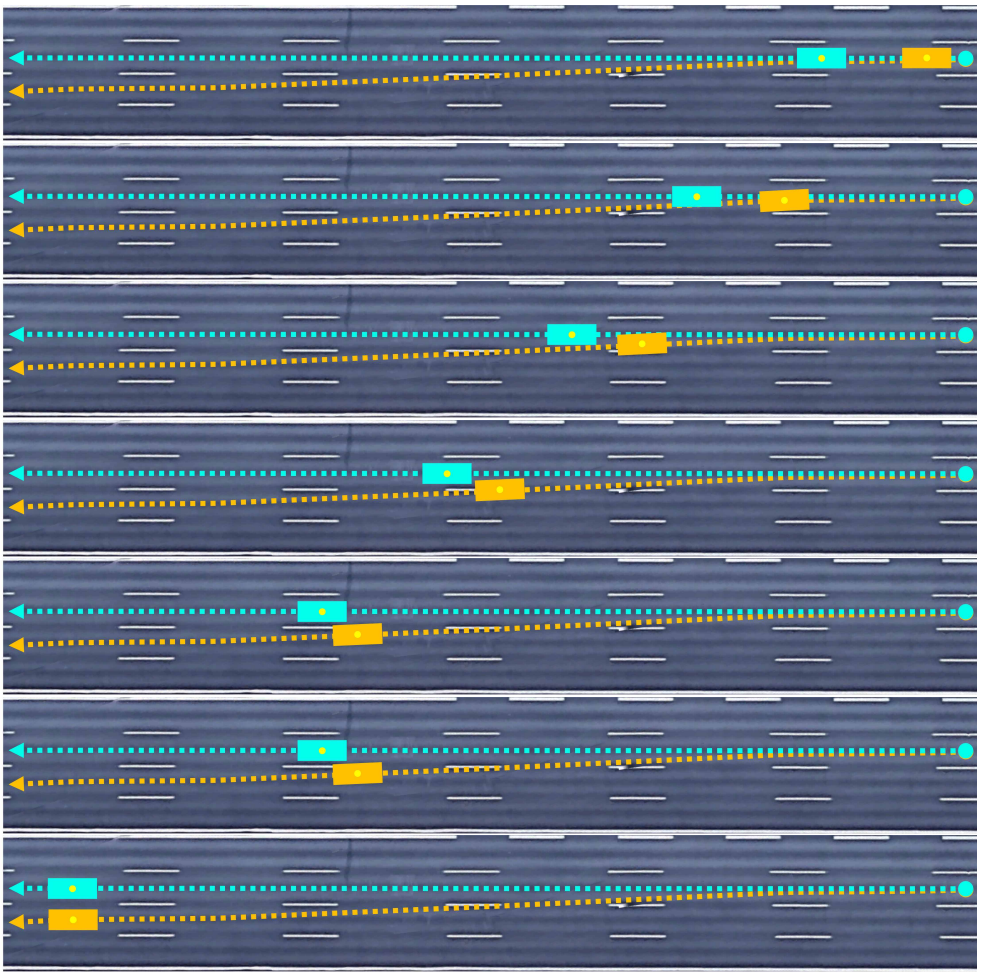

Figure 4.43: Example of a typical cut-out maneuver performed by the orange vehicle (sources: own illustration, [1]).

Both cut-in and cut-out maneuvers are modeled and detected in this work. The detection of these maneuvers occurs alongside to the detection of regular lane changes. Therefore, the lane change information is combined with the information concerning the distance of a vehicle to the closest leading and following vehicle.

Whenever the lane change detection algorithm described in the previous chapter recognizes a lane change it is also checked if a cut-in or cut-out has occurred.

For a cut-in, the time span when the lane changing vehicle crosses the lane marking and enters the new lane until the moment when the lane change is completed is analyzed. If the TTC to the following vehicle is less than two seconds, a cut-in is detected.

Similarly, cut-outs are detected. Here, the time span between the lane change begins until the moment when the lane changing vehicle crosses the lane marking and enters the new lane is analyzed. If the TTC to the leading vehicle is less than two seconds before leaving the lane, a cut-out is detected. Figures 4.44 and 4.45 provide an overview of the mentioned metrics to detect cut-in and cut-out maneuvers.

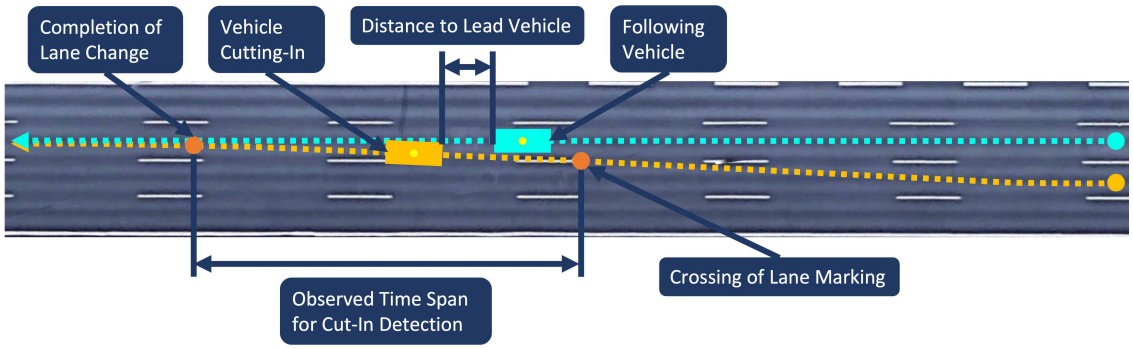

Figure 4.44: Overview of the metrics used to detect cut-in maneuvers (sources: own illustration, [1]).

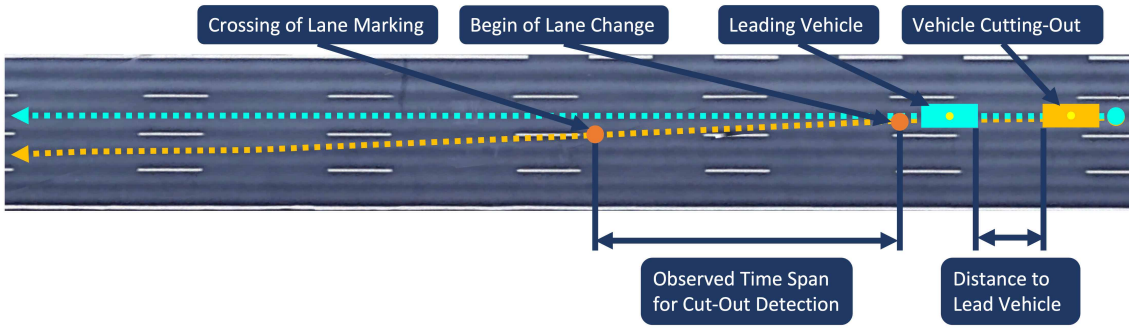

Figure 4.45: Overview of the metrics used to detect cut-out maneuvers (sources: own illustration, [1]).

4.5.6.4 Label: Tailgate

When a vehicle is keeping an insufficient distance to its leading vehicle for a longer period of time and does not immediately increase the distance, this is considered as tailgating [158]. Since this driving behavior is dangerous and is a cause of many accidents [143], it is modeled and labeled in the *proScenario dataset* so prediction algorithms can be later trained on this data to avoid such situations.

In the German traffic law specific rules exist that define when a vehicle is not keeping a sufficient distance to its predecessor and when tailgating occurs [158].

In urban areas the distance in meters that must be kept to a leading vehicle is at least the distance travelled in one second. This distance can be calculated by dividing the vehicle's velocity in kilometers per hour by 3.6. For example, if a vehicle is driving through an urban area at a velocity of 36 kilometers per hour it must keep a distance of at least ten meters to any leading vehicle [158].

Outside of urban areas, the distance in meters that must be kept to a leading vehicle is at least the velocity in kilometers per hour divided by two. For example, if a vehicle is driving through a non-urban area at a velocity of 120 kilometers per hour it must keep a distance of at least 60 meters to any leading vehicle [158].

Additionally, if the minimum required distance is not held because another vehicle is entering the driving lane ahead, the driver has three seconds time to restore the required distance. Above speeds of 160 kilometers per hour, this time period is reduced to one second

[158].

These rules are defined as follows:

$$Minimum\ required\ Distance\ to\ leading\ vehicle = \begin{cases} v/2, & in\ non-urban\ areas. \\ v/3.6, & in\ urban\ areas. \end{cases} \quad (4.1)$$

The detection of tailgates is based on two pieces of information. These are the vehicles velocity and its distance to the leading vehicle. The vehicle velocity is measured by the Providentia++ sensor system while the distance to the leading vehicle is calculated as shown in Section 4.5.5.3. The minimum required distance to the leading vehicle is calculated by the corresponding equations mentioned previously. The actual distance to the leading vehicle is then subtracted from the calculated minimum required distance. If the resulting value is negative a tailgate occurs otherwise the vehicle is keeping the required distance.

The detected tailgates are divided into the three categories minor tailgate, moderate tailgate, and severe tailgate. The category is determined by the degree to which the minimum required distance is violated. Table 4.3 illustrates how the degree of violation is determined. Figure 4.46 provides an example for each tailgate category at a velocity of 120 kilometers per hour.

Tailgate Severity	Definition
Minor Tailgate	• Violation of the minimum required distance to the lead vehicle at speeds of less than 80 km/h • Keeping a distance to the lead vehicle of less than 100 percent and a minimum of 50 percent of the minimum required distance at speeds of 80 km/h and above
Moderate Tailgate	• Keeping a distance to the lead vehicle of less than 50 percent of the minimum required distance at speeds between 80 km/h and 100 km/h • Keeping a distance to the lead vehicle of less than 50 percent and a minimum of 30 percent of the minimum required distance at speeds of 100 km/h and above
Severe Tailgate	• Keeping a distance to the lead vehicle of less than 30 percent of the minimum required distance at speeds between over 100 km/h

Table 4.3: Definition of minor, moderate, and sever tailgates (source: [158]).

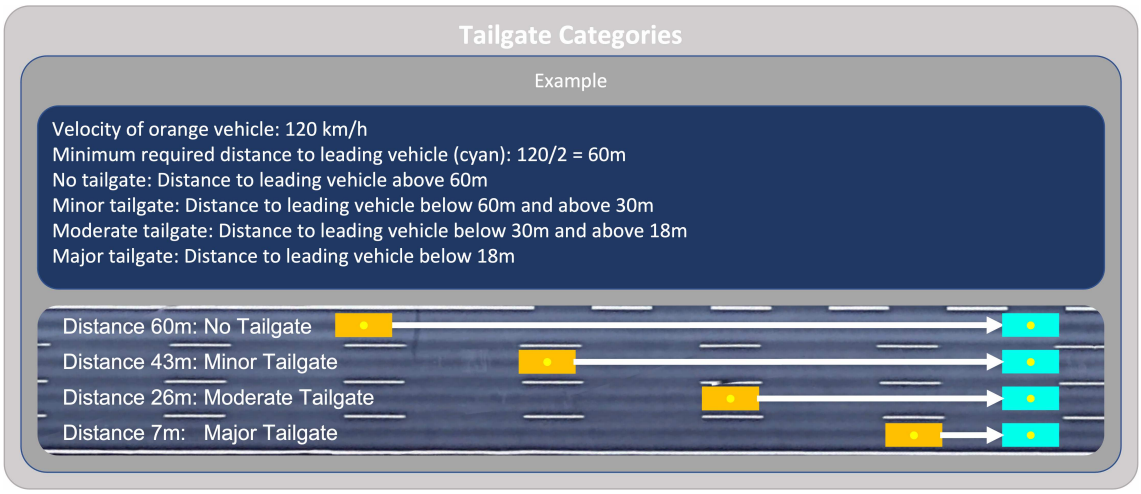

Figure 4.46: Example for each tailgate category at a velocity of 120 kilometers per hour (sources: own illustration, [1]).

4.5.6.5 Label: Speeding

The detection of speeding vehicles is done quite simply. All fixed speed-limits along the Providentia++ test stretch are known. Depending on which section of the test stretch a vehicle is driving on, its velocity profile is compared with the applicable speed limit. For each time step it is checked if the vehicle velocity lies above or below the speed limit. If it lies above the limit, the vehicle is labeled as speeding for this time step.

On the highway section of the test stretch the speed limit is controlled electronically and therefore subject to change. Since at the time of this work no live access to the speed limit is available some approximations must be made. From experience it is known that the speed limit is mostly 130 kilometers per hour in the south direction and unlimited in the northern direction of the highway. Therefore these speed limits are assumed for detecting speeding vehicles on the highway stretch.

4.5.6.6 Label: Standing Vehicle

The detection of standing vehicles is done quite similarly to that of speeding vehicles. At each time step it is checked if the vehicle velocity is zero or not. If it is zero the vehicle is labeled as standing for this time step.

4.5.6.7 Label: Weather

Certain weather conditions can lead to an increase of traffic incidents such as crashes or dangerous situations [55, 128]. For the vast majority of the time the weather condition – from a traffic safety perspective – does not have a large impact on the driveability of a vehicle or on the traction the road surface provides. However, this implicitly means that the vehicle operators are not used to the new driving situation that occurs due to bad weather conditions and are therefore more prone to cause an accident [55]. A human driving a vehicle in traffic already knows that in different weather conditions the vehicle and road will behave differently. In other words, the same driving behavior can be safe or unsafe depending on the weather conditions. The logical consequence is to take safety precautions when critical weather situations arise, such as reducing speed or paying higher attention to the situation

in general [128].

Thinking one step further, the ultimate goal of the dataset created in this work is to be able to train Deep Learning algorithms that can detect and predict dangerous driving situations and act upon these in an intelligent manner. Various sources have shown that certain weather conditions have lead to a higher risk of traffic accidents [55, 66, 128].

For this reason, information regarding weather conditions is regarded as relevant to improve the prediction power of Deep Learning models trained for autonomous driving applications and will therefore be included in the *proScenario dataset*.

OpenWeatherMap [48] offers a global weather database including live weather information. This information is also available for Garching-Hochbrück where the Providentia++ test stretch is located. As part of this work a script is created that calls the live weather data at the test stretch location in 10 second intervals and adds the information to the sensor recordings. The information includes the weather type (sunny, cloudy, thunderstorm, etc.), temperature, windspeed, visibility, and cloudiness in percent.

4.5.6.8 Label: Right Turn / Left Turn /Straight at Crossing / U-Turn

According to the NHTSA 13 percent of traffic accidents occur while performing turn maneuvers at crossings [143]. Such maneuvers include protected and unprotected left- and right turns as well as turns that cross pedestrian walkways and cycling lanes. Turn maneuvers occur very frequently in urban- and rural areas and are highly relevant for training automated driving algorithms and ensuring their safe functioning [143].

Labeling turn maneuvers by hand is very time consuming due to the fact that they occur very frequently and require a label for each vehicle in each data frame. In order to massively reduce the resources necessary to label turn maneuvers, a framework is developed that creates the appropriate labels in an automated fashion. The functionality of this framework is explained in the following section.

Classical sensor systems used in autonomous driving applications are mounted directly on the vehicle they control [182, 183, 206]. Differently to these systems the sensor systems used in the Providetia++ project are mounted either on or alongside the road infrastructure. The main difference between the two types of sensor systems is the location of the sensors over time. Classical sensor systems will move with the vehicle they are mounted on, meaning that at each time step a different road stretch is perceived. In contrast, the statically mounted Providentia++ sensor systems perceive the same road stretch at all times. This key property necessary for the implementation is illustrated in Figure 4.47.

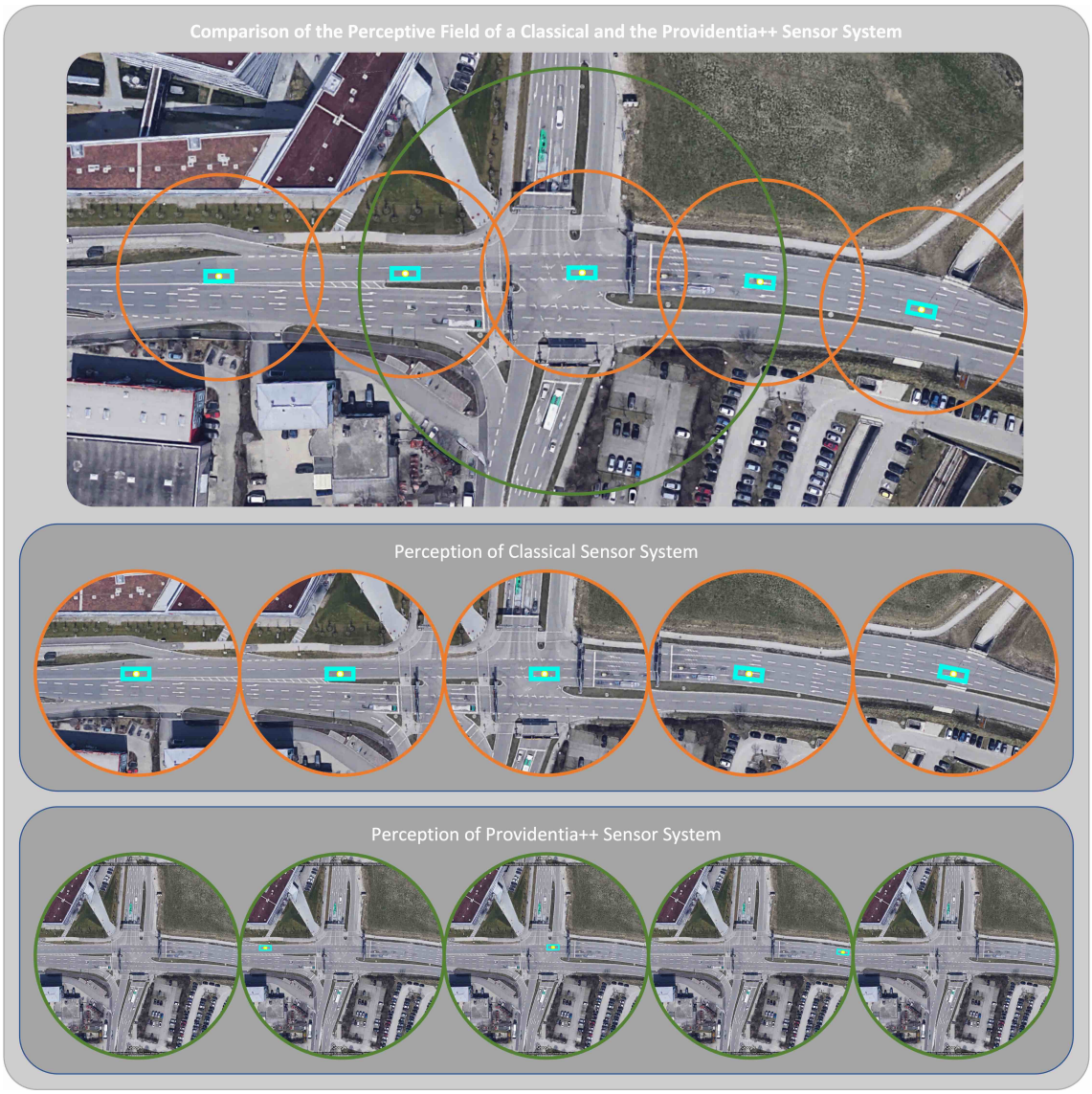

Figure 4.47: Perceptive field of a vehicle with classical sensor system following a trajectory compared to the perceptive field of the Providentia++ sensor system(sources: own illustration, [2]).

Because the sensor system always observes the same road section, an approach using start and goal boxes is utilized. The main idea behind this approach is that multiple regions of interest inside the road area are marked with boxes. Depending on which boxes a vehicle drives through, it becomes clear what kind of maneuver the vehicle performs. In Figure 4.48 the detection mechanism is illustrated. Here, different vehicle trajectories cross different goal boxes, which results in different maneuvers being detected. For example when looking at the turquoise trajectory in Figure 4.48, it becomes obvious for the observer that the vehicle is performing a right turn at the crossing. While making a right turn, the vehicle passes through boxes 1 and 2. Algorithmically, the right turn can be detected by applying a simple if clause asking if the vehicle trajectory first passes through box 1 and at a later point in time passes through box 2. If this is the case, the vehicle has clearly performed a right turn. In order for the algorithm to know if and when a vehicle passes through a goal box, for each point of the trajectory a test is performed whether it lies inside any of the goal boxes.

Analogously, the green trajectory passes through boxes 1 and 4 indicating that it is going

straight and the red trajectory passes through boxes 1 and 8 indicating a left turn. Additionally, U-turns are also detected if an actor passes through the goal box pairs 1 and 10, 3 and 2, 5 and 4, or 9 and 8.

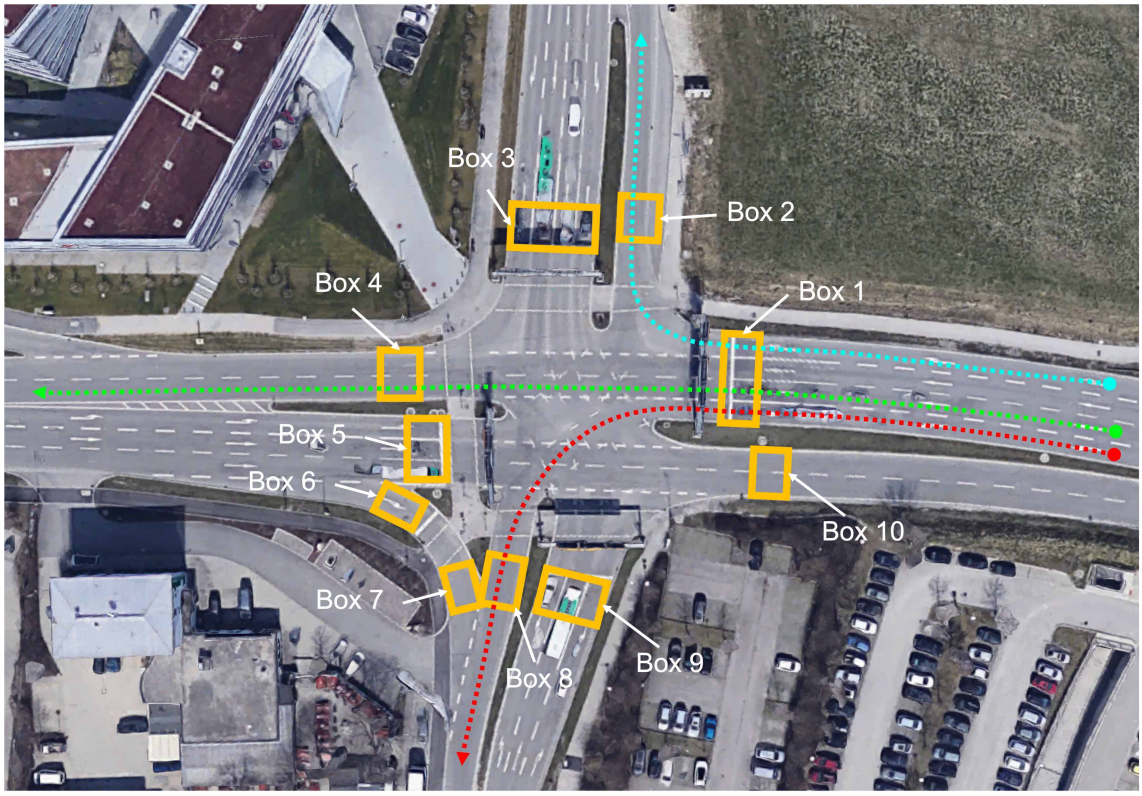

Figure 4.48: Illustration of how turns are detected by analyzing the goal boxes passed by a vehicle (sources: own illustration, [2]).

The data frames that lie between the point in time when the start box is exited until the goal box is entered are labeled accordingly to the completed maneuver. For example when looking at the turquoise trajectory, all frames starting from the point in time when box 1 is exited until box 2 is entered are labeled as “turn right”. For the red trajectory all data frames between the point in time when box 1 is exited and box 8 is entered are labeled as “turn left”.

This procedure can easily be exploited to other maneuver types which is shown in the following sections.

4.5.6.9 Label: Enter Highway / Exit Highway

In the previous section the detection of turns, U-turns, and going straight at an intersection with the goal box principle is explained. Similarly, vehicles entering or exiting the highway on the Providentia++ test stretch are detected. In this case, two goal polygons are used for each of the enter or exit ramps. These are both larger and function in a slightly different way as compared to the goal boxes used at intersections.

When detecting an exit, the first goal polygon covers the exit ramp from the beginning of the ramp until the point where a vehicle can no longer return to the highway and is forced to exit. The second goal polygon begins where the first goal polygon ends and goes on until the end of the exit ramp. The placement of the two goal polygons is illustrated in Figure 4.49.

Here goal polygon one detects the start and goal polygon two detects the end of the highway exit maneuver. Because of the clever placement of the two goal polygons, a detected exit maneuver starts as soon as the exiting vehicle enters the first goal polygon. This corresponds to entering the exit ramp. The end of the maneuver is the last trajectory point of the vehicle that lies inside the second goal polygon. The choice not to place the second goal polygon at the end of the exit ramp, but rather extend it over a longer stretch is a large benefit. It means that in case the driving scenario ends before the vehicle reaches the end of the exit ramp the exit maneuver is still recognized, as long as the vehicle has reached the second goal polygon.

Highway entering maneuvers are detected quite similarly to the exit maneuvers. In this case, the first goal polygon starts at the beginning of the on ramp and ends as soon as the vehicle has no other option as to enter the highway. At this point, the second goal polygon begins. It continues until the end of the on ramp when the ramp merges into the highway. An example is shown in Figure 4.49. Here, goal polygon three detects the start and goal polygon four detects the end of the highway enter maneuver.

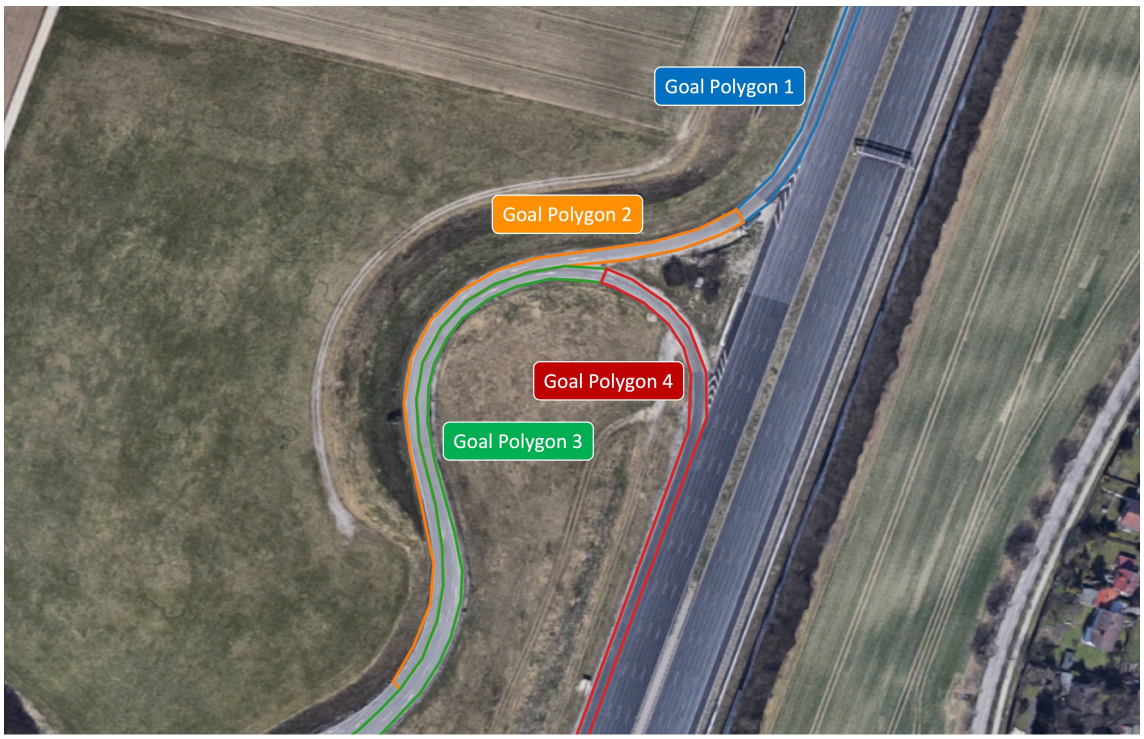

Figure 4.49: Illustration of how vehicles entering or exiting the highway are detected by analyzing the goal polygons passed by the vehicles (sources: own illustration, [3]).

4.5.7 Scenario Statistics

After all driving maneuvers are detected a statistical summary of each driving scenario is calculated. The statistics include the total number of each different driving maneuver as well as the maximum number of lane changes performed by a single actor and the top speed achieved by an actor throughout the scenario. The complete list of information included in the statistics is shown in Table 4.4.

Statistics Type	Description
Total lane changes left	Sum of lane changes to the left committed by all vehicles
Total lane changes right	Sum of lane changes to the right committed by all vehicles
Total lane changes	Sum of all types of lane changes committed by all vehicles
Maximum lane changes	Maximum amount of lane changes committed by a single vehicle
Total cut-ins left	Sum of cut-ins to the left committed by all vehicles
Total cut-ins right	Sum of cut-ins to the right committed by all vehicles
Total cut-ins	Sum of all types of cut-ins committed by all vehicles
Total cut-outs left	Sum of cut-outs to the left committed by all vehicles
Total cut-outs right	Sum of cut-outs to the right committed by all vehicles
Total cut-outs	Sum of all types of cut-outs committed by all vehicles
Total minor tail-gates	Sum of minor tail-gates committed by all vehicles
Total moderate tail-gates	Sum of moderate tail-gates committed by all vehicles
Total severe tail-gates	Sum of severe tail-gates committed by all vehicles for the duration of the driving scenario
Total speeding vehicles	Sum of vehicles exceeding the allowed speed limit
Total standing vehicles	Sum of vehicles standing

Table 4.4: Overview of the statistics provided with every driving scenario (source: own illustration).

Additionally, a graphical overview of various driving maneuvers occurring throughout a scenario is included in the dataset. The graphical overview contains plots over time showing exactly when a vehicle performs certain driving maneuvers. The features included in the graphical statistics overview are explained in Table 4.5. Figure 4.50 shows an example of the graphical visualization of the statistics.

Statistics Type	Description
Trajectories	All vehicle trajectories in the driving scenario
Velocity profiles	All vehicle velocity profiles over time throughout the driving scenario
Lane IDs	The driving lane each vehicle is using at every point in time throughout the driving scenario
Lane changes left	Point in time when a lane change to the left is committed by a vehicle
Lane changes right	Point in time when a lane change to the right is committed by a vehicle
Cut-ins left	Point in time when a cut-in to the left is committed by a vehicle
Cut-ins right	Point in time when a cut-in to the right is committed by a vehicle
Cut-outs left	Point in time when a cut-out to the left is committed by a vehicle
Cut-outs right	Point in time when a cut-out to the right is committed by a vehicle
Tail-gates	Point in time when a tail-gate is committed by a vehicle. The numbers 1, 2, and 3 on the Y-axis indicate the severity of the tail-gate maneuver. 1 indicates a minor tail-gate, 2 a moderate tail-gate, and 3 a severe tail-gate.
Speeding	Vehicles exceeding the allowed speed limit at each point in time
Total vehicles speeding	Total number of vehicles speeding at each point in time during the driving scenario
Standing	Standing vehicles at each point in time
Total vehicles standing	Total number of vehicles standing at each point in time during the driving scenario

Table 4.5: Overview of the graphical statistics provided with every driving scenario (source: own illustration).

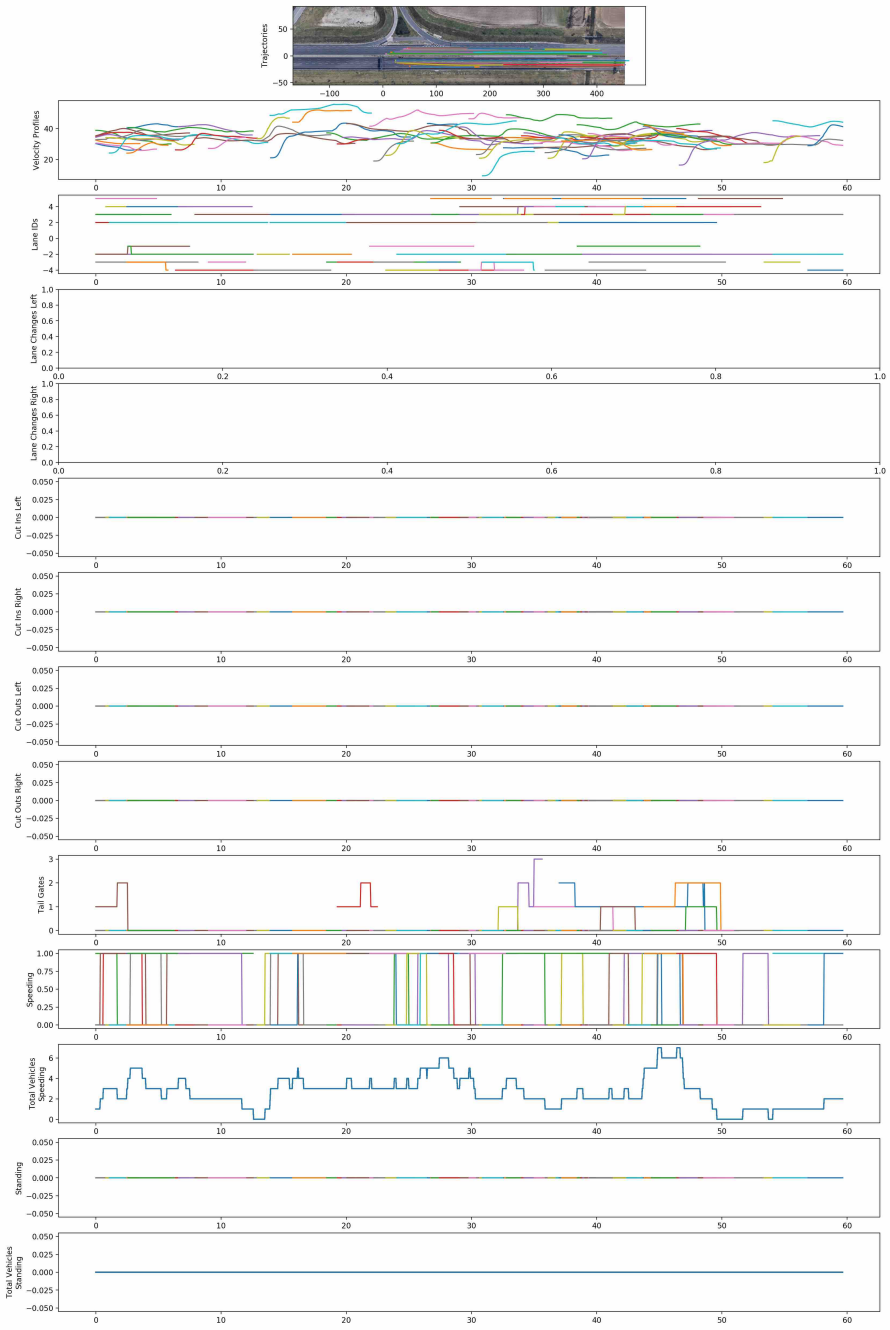

Figure 4.50: Graphical statistics overview showing the occurrence of various driving maneuvers over time (source: own illustration).

4.6 Creating OpenSCENARIO Files for Simulation and Visualization

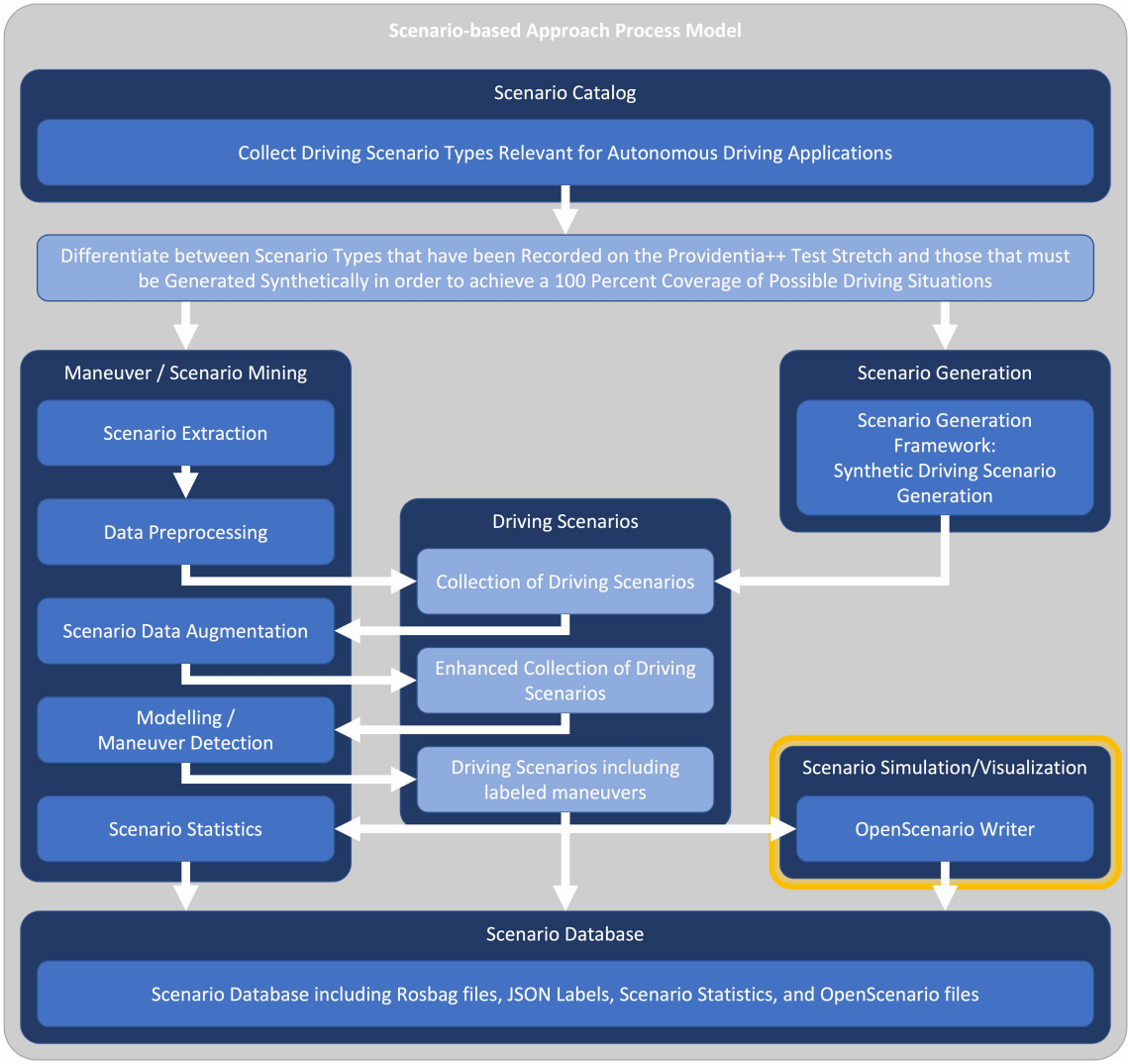

Figure 4.51: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

The main focus of the scenario-based approach is to extensively test AVs in a simulation environment. For this a detailed representation of realistic driving scenarios is necessary. The driving scenarios can then be executed in the simulation environment allowing the safety of AVs to be assessed. The assessment process is based on two tasks - creating files that describe the driving scenarios and executing the scenarios in a simulation environment.

1. In the first step, the vehicle and trajectory data is written into an OpenSCENARIO file. This is beneficial as the OpenSCENARIO file format provides an industry standard and is compatible with many simulation environments. It also provides a simple syntax that is easy to understand.

2. In the second step the OpenSCENARIO files are simulated and visualized in CARLA using the ScenarioRunner. The simulation allows to assess the safe operation of AVs.

4.6.1 Creating OpenSCENARIO Files

In order to create OpenSCENARIO files in an automated fashion in this work, a scenario writer is created. This file parser takes driving scenarios provided in the scenario description format which is explained in Section 4.1 and automatically creates a corresponding OpenSCENARIO file.

To do so the Python library "Scenario Generation" [192] is used. In total two scripts are created that both define the OpenSCENARIO file in a different way due to the different requirements of the simulators "ScenarioRunner" and "Esmini". In the following, first the definition parameters in the OpenSCENARIO standard that are defined identically in both scripts are described. Afterwards, the differences in describing the actors' trajectories in each of the two scripts are explained.

The initialization of the road network, actors, and environment conditions is identical in both scripts. Here the "Scenario Generation" library provides various functionalities to define these parameters. The road network is connected to the OpenSCENARIO file by adding the relative path to the OpenDRIVE file corresponding to the road network. Listing 4.1 shows the definition of the road network in an OpenSCENARIO file.

```
<RoadNetwork>
  <LogicFile filepath="providentia_plus_plus_map_0_4_offset" />
</RoadNetwork>
```

Listing 4.1: Example of how road network is defined with the "RoadNetwork" feature of OpenSCENARIO (source: own illustration).

Actors are defined by passing information about the type, dimensions, performance, kinematics, and color. Here, the type defines whether the actor is a pedestrian or what type of vehicle it is. Performance measures indicate the maximum speed as well as the maximum acceleration and deceleration capabilities of the actor. Finally, the kinematics define the position and dimensions of the driving axles as well as the maximum steering angle they are capable of. The definition of a vehicle actor is shown in Listing 4.2.

```
<Entities>
  <ScenarioObject name="CAR_8140">
    <Vehicle name="vehicle.tesla.model3" vehicleCategory="car">
      <ParameterDeclarations />
        <BoundingBox>
          <Center x="1.5" y="0" z="0.9" />
            <Dimensions height="1.8" length="4.5" width="2.1" />
        </BoundingBox>
        <Performance maxAcceleration="10.0" maxDeceleration="10.0"
        maxSpeed="69.444" />
        <Axles>
          <FrontAxle maxSteering="0.5" positionX="3.1" positionZ="0.3"
          trackWidth="1.8" wheelDiameter="0.6" />
          <RearAxle maxSteering="0" positionX="0" positionZ="0.3"
          trackWidth="1.8" wheelDiameter="0.6" />
        </Axles>
          <Properties>
```

```
            <Property name="type" value="simulation" />
            <Property name="color" value="0,0,255" />
          </Properties>
    </Vehicle>
  </ScenarioObject>
</Entities>
```

Listing 4.2: Example of how the environment conditions are defined with the "EnvironmentAction" feature of OpenSCENARIO (source: own illustration).

The environment conditions are defined using the "EnvironmentAction" feature of the "Scenario Generation" package. Here the date and time of day as well as the weather conditions and road conditions are set. The weather conditions are composed of the cloud state, sun, fog, and precipitation. The cloud state can either be cloudy or clear, whereas the sun is defined by setting parameters for the intensity, azimuth, and elevation. Fog is defined by the visual range, which is set in meters. The road conditions are simply defined by a friction value ranging from zero to one. A friction value of zero means the road provides no grip at all while a value of one relates to the maximum grip which typically occurs on a dry sunny day. An example of how the environment conditions are defined in an OpenSCENARIO file is shown in Listing 4.3.

```
<GlobalAction>
  <EnvironmentAction>
    <Environment name="Environment">
      <TimeOfDay animation="false" dateTime="2020-07-25T12:00:00" />
      <Weather cloudState="free">
        <Sun azimuth="0" elevation="1.31" intensity="0.85" />
        <Fog visualRange="10000" />
        <Precipitation intensity="0.0" precipitationType="dry" />
      </Weather>
      <RoadCondition frictionScaleFactor="1" />
    </Environment>
  </EnvironmentAction>
</GlobalAction>
```

Listing 4.3: Example of how the environment conditions are defined with the "EnvironmentAction" feature of OpenSCENARIO (source: own illustration).

The last common step is the initialization of each actor. When initializing an actor, first the speed and spawn location is defined. These parameters describe the actor's state at the start of the scenario. Here, the speed defines the actor's longitudinal velocity whereas the spawn location defines the initial position and orientation in coordinates and yaw angle. An example for the initialization of an actor is shown in Listing 4.4.

```
<Init>
  <Actions>
    <Private entityRef="CAR_8140">
      <PrivateAction>
        <LongitudinalAction>
          <SpeedAction>
            <SpeedActionDynamics dynamicsDimension="time"
            dynamicsShape="step" value="1000" />
```

```
          <SpeedActionTarget>
            <AbsoluteTargetSpeed value="0" />
          </SpeedActionTarget>
        </SpeedAction>
      </LongitudinalAction>
    </PrivateAction>
    <PrivateAction>
      <TeleportAction>
        <Position>
          <WorldPosition h="-1.27" x="-138.72" y="1874.51" z="486.5" />
        </Position>
      </TeleportAction>
    </PrivateAction>
  </Private>
```

Listing 4.4: Example of how an actor is initialized in an OpenSCENARIO (source: own illustration).

For the Esmini player the OpenSCENARIO feature "FollowTrajectory" is used as a main building block for creating the OpenSCENARIO file. This feature defines the complete movement of an actor, by taking the trajectory information including positions and corresponding timestamps as an input. The trajectory is then added to the story-line of the scenario. This is done for each of the actors by iterating through the scenario provided in the intermediate data format until all actors are included in the scenario. In Listing 4.5 an excerpt from an OpenSCENARIO file showing the definition of an actor's trajectory with the "FollowTrajectory" feature is shown.

```
<Maneuver name="CAR_7850maneuver">
  <Event maximumExecutionCount="1" name="CAR_7850event"
  priority="overwrite">
    <Action name="newspeed">
      <PrivateAction>
        <RoutingAction>
          <FollowTrajectoryAction>
            <TrajectoryRef>
              <Trajectory closed="false" name="CAR_7850traj">
                <ParameterDeclarations/>
                <Shape>
                  <Polyline>
                    <Vertex time="0.0">
                      <Position>
                        <WorldPosition x="102.082" y="-8.901"/>
                      </Position>
                    </Vertex>
                    <Vertex time="0.04">
                      <Position>
                        <WorldPosition x="103.255" y="-8.907"/>
                      </Position>
                    </Vertex>
                    <Vertex time="0.08">
                      <Position>
                        <WorldPosition x="104.431" y="-8.912"/>
                      </Position>
```

```
                </Vertex>
                ...
                <Vertex time="10.96">
                  <Position>
                    <WorldPosition x="450.958" y="-9.615"/>
                  </Position>
                </Vertex>
                <Vertex time="11.0">
                  <Position>
                    <WorldPosition x="452.150" y="-9.620"/>
                  </Position>
                </Vertex>
              </Polyline>
            </Shape>
          </Trajectory>
        </TrajectoryRef>
        <TimeReference>
          <Timing domainAbsoluteRelative="absolute" offset="0"
          scale="1"/>
        </TimeReference>
        <TrajectoryFollowingMode followingMode="follow"/>
      </FollowTrajectoryAction>
    </RoutingAction>
  </PrivateAction>
</Action>
<StartTrigger>
  <ConditionGroup>
    <Condition conditionEdge="rising" delay="0"
    name="starttrigger7850">
      <ByValueCondition>
        <SimulationTimeCondition rule="greaterThan" value="0.1"/>
      </ByValueCondition>
    </Condition>
  </ConditionGroup>
</StartTrigger>
</Event>
```

Listing 4.5: Example of how an actor's trajectory is defined with the "FollowTrajectory" feature of OpenSCENARIO (source: own illustration).

When creating scenarios for use with the "ScenarioRunner", an actor's trajectory must be defined differently. This is due to lack of support for the "FollowTrajectory" feature of OpenSCENARIO. Therefore, the actors' trajectories are modeled by combining the two OpenSCENARIO features "Route" and "SpeedAction". With the feature "Route" an actor's path is defined by passing a set of coordinate points. When running the final scenario, the actor then follows these coordinate points in chronological order. In this work, the actor's positions which are included in the trajectory data are used to define the route. An example for a route in the OpenSCENARIO format is shown in Listing 4.6.

```
<Action name="CAR_8140route">
  <PrivateAction>
    <RoutingAction>
```

```
      <AssignRouteAction>
        <Route closed="false" name="CAR_8140route">
          <ParameterDeclarations />
          <Waypoint routeStrategy="shortest">
            <Position>
              <WorldPosition x="-138.72" y="1874.51" />
            </Position>
          </Waypoint>
          <Waypoint routeStrategy="shortest">
            <Position>
              <WorldPosition x="-138.30" y="1873.09" />
            </Position>
          </Waypoint>
          <Waypoint routeStrategy="shortest">
            <Position>
              <WorldPosition x="-137.87" y="1871.66" />
            </Position>
          </Waypoint>
          <Waypoint routeStrategy="shortest">
            <Position>
              <WorldPosition x="-137.43" y="1870.22" />
            </Position>
          </Waypoint>
          ...
          <Waypoint routeStrategy="shortest">
            <Position>
              <WorldPosition x="-36.91" y="1532.83" />
            </Position>
          </Waypoint>
          <Waypoint routeStrategy="shortest">
            <Position>
              <WorldPosition x="-36.91" y="1532.83" />
            </Position>
          </Waypoint>
        </Route>
      </AssignRouteAction>
    </RoutingAction>
  </PrivateAction>
</Action>
```

Listing 4.6: Example of how an actor's route is defined with the "Route" feature of OpenSCENARIO (source: own illustration).

The defined route is followed by the actor with the velocity defined when initializing the actor. In order to reflect a velocity profile that is not constant, the changes in velocity must be reflected in the OpenSCENARIO file. This is done by defining a "SpeedAction" feature at each time step. Here the actor's speed is adjusted according to the velocity profile provided with the scenario description format. An example of the defined "SpeedAction" feature for a single actor is shown in Figure 4.7.

```
<Event maximumExecutionCount="1" name="CAR_81400" priority="overwrite">
  <Action name="CAR_81400speed">
```

```
    <PrivateAction>
      <LongitudinalAction>
        <SpeedAction>
        <SpeedActionDynamics dynamicsDimension="time"
        dynamicsShape="step" value="1000" />
          <SpeedActionTarget>
            <AbsoluteTargetSpeed value="37.15" />
          </SpeedActionTarget>
        </SpeedAction>
      </LongitudinalAction>
    </PrivateAction>
  </Action>
  <StartTrigger>
    <ConditionGroup>
      <Condition conditionEdge="rising" delay="0.0"
      name="CAR_8140starttrigger2">
        <ByValueCondition>
          <SimulationTimeCondition rule="greaterThan" value="0.0" />
        </ByValueCondition>
      </Condition>
    </ConditionGroup>
  </StartTrigger>
</Event>
<Event maximumExecutionCount="1" name="CAR_81401" priority="overwrite">
  <Action name="CAR_81401speed">
    <PrivateAction>
      <LongitudinalAction>
        <SpeedAction>
          <SpeedActionDynamics dynamicsDimension="time"
          dynamicsShape="step" value="1000" />
          <SpeedActionTarget>
            <AbsoluteTargetSpeed value="37.15" />
          </SpeedActionTarget>
        </SpeedAction>
      </LongitudinalAction>
    </PrivateAction>
  </Action>
  <StartTrigger>
    <ConditionGroup>
      <Condition conditionEdge="rising" delay="0.04"
      name="CAR_8140starttrigger2">
        <ByValueCondition>
          <SimulationTimeCondition rule="greaterThan" value="0.0" />
        </ByValueCondition>
      </Condition>
    </ConditionGroup>
  </StartTrigger>
</Event>
    ...
<Event maximumExecutionCount="1" name="CAR_8140317"
```

```
priority="overwrite">
  <Action name="CAR_8140317speed">
    <PrivateAction>
      <LongitudinalAction>
        <SpeedAction>
          <SpeedActionDynamics dynamicsDimension="time"
          dynamicsShape="step" value="1000" />
            <SpeedActionTarget>
              <AbsoluteTargetSpeed value="11.85" />
            </SpeedActionTarget>
          </SpeedAction>
        </LongitudinalAction>
      </PrivateAction>
  </Action>
  <StartTrigger>
    <ConditionGroup>
      <Condition conditionEdge="rising" delay="12.84"
      name="CAR_8140starttrigger2">
        <ByValueCondition>
          <SimulationTimeCondition rule="greaterThan" value="0.0" />
        </ByValueCondition>
      </Condition>
    </ConditionGroup>
  </StartTrigger>
</Event>
```

Listing 4.7: Example of how an actor's velocity is defined with the "SpeedAction" feature of OpenSCENARIO (source: own illustration).

4.6.2 Scenario Database

The final phase in the process model deals with the creation of a driving scenario database. (Figure 4.52) The database includes the achievements made throughout the previous process phases. This includes a rosbag file of each driving scenario including labeled driving maneuvers, JSON labels describing the driving scenario and maneuvers, a graphical and non-graphical statistical overview, as well as an OpenSCENARIO file for simulation purposes. The data included for each driving scenario is shown in Figure 4.53.

These files are provided with each driving scenario added to the database. The database has the potential to grow very quickly due to the automated scenario processing pipeline introduced in this work. The statistics provided with each driving scenario provide a detailed characterization and allow to filter the database efficiently for specific types of scenarios. Currently the database contains driving scenarios from 32 evaluated rosbags which are also included in the *proScenario dataset* dataset. However, the database can easily be extended by adding new rosbag files and processing them with the scenario-based approach pipeline. Since the processing is fully automated hardly any manual labor is necessary to extend the database.

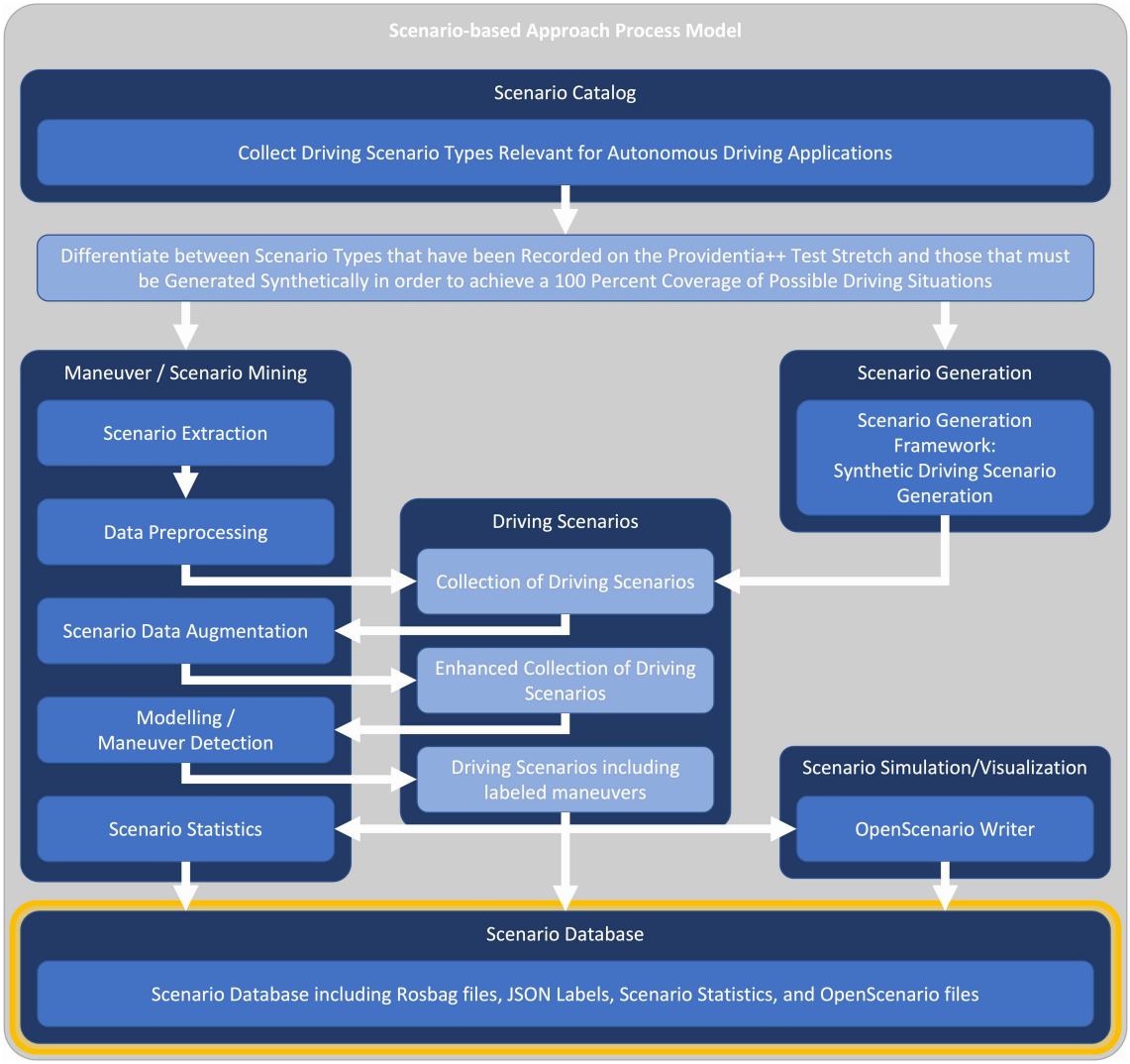

Figure 4.52: Illustration of the Stages for the creation of the final *proScenario dataset* (source: own illustration).

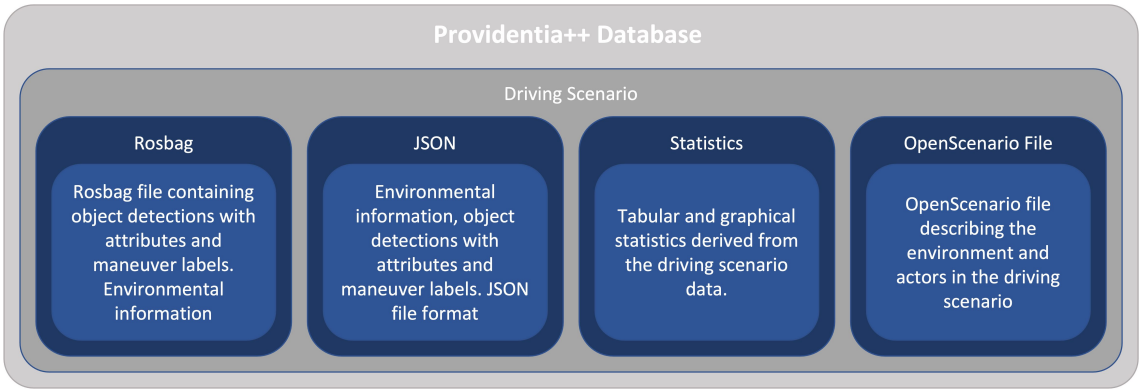

Figure 4.53: Overview of the components included in the Providentia++ database (source: own illustration).

4.6.3 Summary of Codebase and Modules

In the previous sections of this chapter the implementation and execution of a pipeline to automate the scenario based approach is explained in detail. For each step in the scenario based approach a module is created and computer programs are written that automate the

process. Figure 4.54 shows the modules and scripts included in the implementation of the scenario based approach and illustrates their interactions within the processing pipeline.

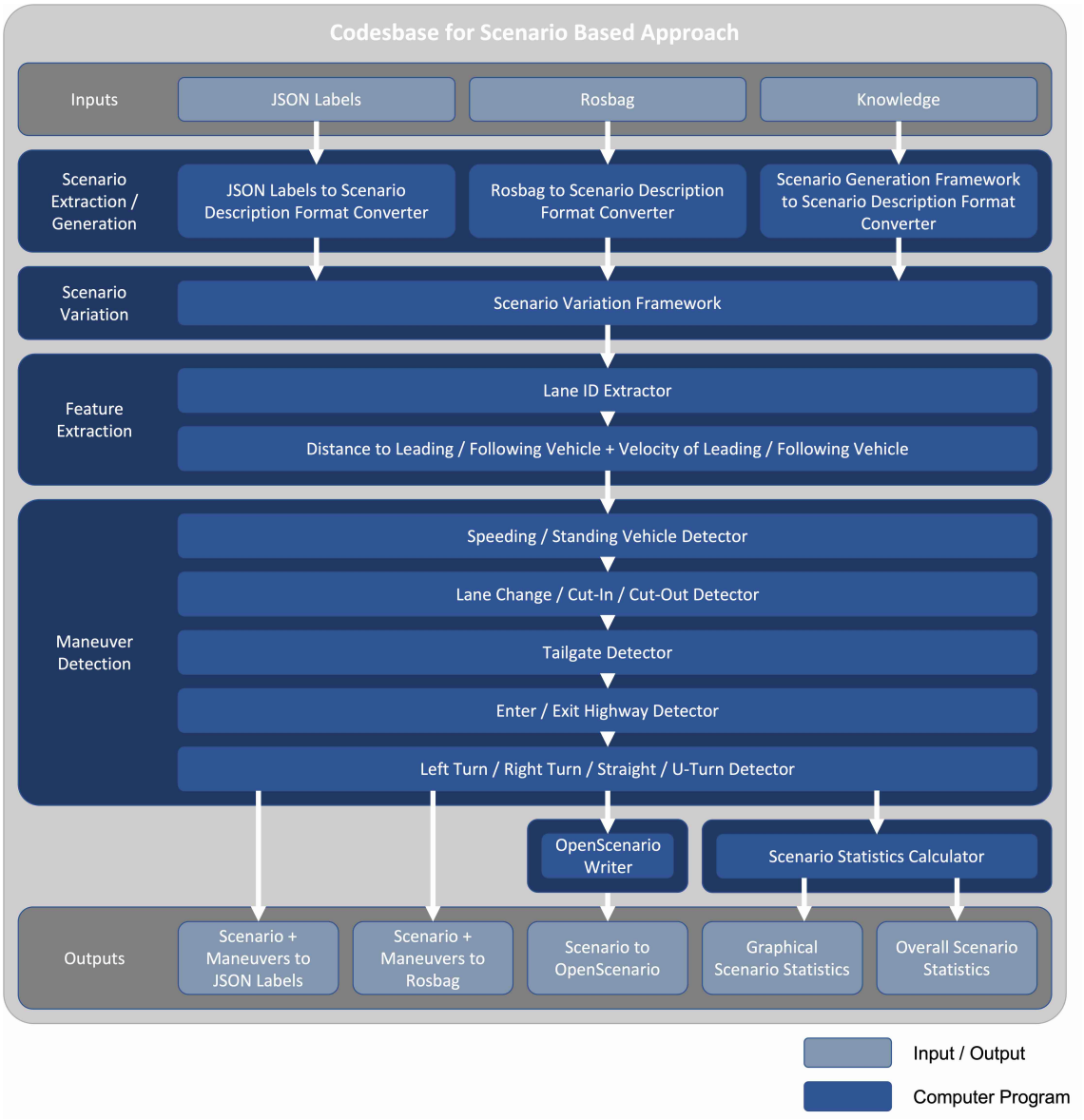

Figure 4.54: Overview of the codebase and modules included in the implementation of the scenario based approach (source: own illustration).

The pipeline can process three types of inputs which all describe driving scenarios in a different format. The input types include JSON labels, rosbags, and knowledge of driving scenarios. JSON labels and rosbags are transformed to the "Scenario Description Format" by the corresponding converter scripts. Knowledge is transformed into the "Scenario Description Format" by using the "Scenario Generation Framework".

After the conversion step driving scenarios are available in the scenario description format. The scenarios are then augmented with the "Scenario Variation Framework". Afterwards two scripts extract features from the driving scenarios. At this point the driving scenarios contain all features necessary to detect driving maneuvers. The maneuvers are then detected by five different detection scripts. Finally with all maneuvers detected a graphical and tabular

scenario statistics is calculated. Additionally, OpenSCENARIO files describing the driving scenarios are generated by the "OpenSCENARIO Writer".

The outputs are added to the scenario database. They include detected scenario and driving maneuvers which are added to JSON labels and optionally to a rosbag file. The OpenSCENARIO file as well as graphical and tabular scenario statistics are also included.

All process steps included in the described pipeline are created as separate modules that can easily be modified or exchanged. This allows for a high flexibility for enhancing or adapting the pipeline in the future.

Chapter 5

Evaluation / Analysis

In this work, six main accomplishments are achieved. These include the creation of a scenario catalog containing crucial driving scenarios, a scenario extraction pipeline, a scenario generation framework, a scenario variation framework, a maneuver detection framework, as well as the simulation and analysis of a scenario leading to an accident.

In this chapter, each accomplishment is analyzed and evaluated upon its potential to automate or accelerate the process of training and evaluating algorithms for autonomous driving.

5.1 Scenario Catalog

The scenario catalog created in this work is based on driving maneuvers described in publications of the National Highway Traffic Safety Association, Euro NCAP, and UNECE. The list is meant to create an awareness of which driving scenarios can possibly occur on the Providentia++ test stretch. During the process of creating the catalog, more than 120 different driving situations are collected. They are categorized in the road categories urban, rural, and highway. With the categorization it is easier to determine, which driving scenarios are relevant for a specific application.

For example, all three types of road areas – urban, rural, and highway – occur on the Providentia++ test stretch. With the categorization of driving scenarios, the training data for training the algorithms on each of the sensor stations can be made more efficiently. This is due to the fact that maneuvers or driving situations that cannot occur on the highway stretch are left out of the training data for all highway sensor stations. This can both speed up the training process and also lead to higher performance results as less “noise” is contained in the data.

5.2 Scenario Extraction

Due to the large amount of driving scenarios needed for the meaningful evaluation of AVs an efficient method for creating the test cases is needed. Therefore a method for the automated extraction of driving scenarios from sensor recordings is introduced in this work. The scenario extraction process is designed as a modular pipeline that can handle various input formats and generate two types of output formats.

There are two types of input that can be processed by the extraction pipeline. The first type is the processed sensor recording which includes information about the detected vehicles such as position, velocity, vehicle type, size, and more. It is passed as a ROS object. The second input type is labeled data in form of JSON files. These files are used by the proAnno framework to correct or add new labels to the data generated through sensor recordings.

For each of the two input types an extraction module is created that translates the input into the scenario description format. From here on various tasks such as automated maneuver labeling, scenario variation, and statistics calculations can be conducted automatically.

5.3 Scenario Generation Framework

The scenario generation framework introduced in this work is designed to efficiently construct new user defined driving scenarios. This is achieved by strongly reducing the complexity of creating trajectories for each actor in a scenario, while still maintaining full control of the scenario design. For this, a trajectory calculator is created that takes very few highly intuitive inputs to automatically form a fully defined actor. In this section the time savings potential by using the introduced trajectory calculator is explained and quantified. The results are based on experiments where the time effort of creating driving scenarios completely manually is compared to creating them with the "Scenario Generation Framework".

Additionally, the path defined for an actor can be reused very efficiently. This is due to the fact that the waypoints defining each path are created and collected in a Python file from which they can be loaded into the trajectory calculator. Additionally, the trajectory calculator offers the option to set both a time and distance offset for a path. This way, the starting position along the path as well as the point in time when the actor enters the scenario, can be determined. This means new trajectories can be calculated very quickly once the base path to follow is defined.

Another factor for increasing efficiency is the available helper functions for defining the velocity profile of an actor. The functions include a constant speed profile as well as an acceleration profile calculator. These types of speed profiles can easily be combined in any way that suits the users demands.

A constant speed profile is created by passing the desired velocity in meters per second and the desired number of data frames for which the actor shall maintain this velocity as inputs.

An acceleration profile is calculated by passing the velocities before and after the acceleration as well as the acceleration rate in meters per seconds squared. With these inputs a velocity profile with constant acceleration or deceleration is calculated. Both helper functions drastically reduce the effort of creating velocity profiles compared to defining them manually. Another factor to consider is the more intuitive and understandable nature of using these helper functions. Finally, changes to the velocity profile can be made very efficiently by changing very few parameters.

Lastly, the re-usability of the paths and easy modification lead to an extremely efficient creation of many different driving scenarios.

An overview of the savings potential of the scenario generation framework is given in Figure 5.1.

Figure 5.1: Overview of the time savings potential when using the Scenario Generation Framework compared to manually creating trajectories (source: own illustration).

5.4 Scenario Variation

Naturally, some driving scenarios occur more frequently than others. For example, vehicles changing lanes is a very common driving scenario, while the avoidance of a construction zone is a rather rare event. Having underrepresented driving scenario types is a large drawback when using this data to train machine learning algorithms for autonomous driving. This is due to the fact that an imbalanced data set results in the learning algorithm paying higher attention to scenario types that occur more frequently in the data. This also means that maneuvering through frequently occurring scenarios is learned better and the algorithm may have trouble mastering rare events such as avoiding a construction zone.

To overcome this problem, a scenario variation framework is created as part of this work. It gives the user the possibility to augment underrepresented scenario types in order to achieve a balanced data set. The augmentation is achieved by varying the trajectory of each actor in a scenario. The trajectory variation is done by multiple Gaussian distributed sam-

pling algorithms. Each of these algorithms augments the trajectory in different ways. This includes augmenting the path and velocity profile. The mean and standard deviation of the Gaussian distribution can be set by the user to control the sampling behavior of the augmentation algorithms.

Other aspects that are addressed, are the weather conditions, time of day, and vehicle parameters such as color and type.

The number of augmented scenarios is defined by the user and is unlimited. Given the stochastic nature of the framework, each of the augmented scenarios is unique. With an increase of the number of augmentations, the variations become complete stochastically. This implies all possible configurations inside the sampling range are achieved. Figures 5.2 and 5.3 illustrate how a simple scenario is augmented 500 times to achieve an extremely high coverage of possible variations.

Figure 5.2: Trajectories of a non-augmented rural driving scenario (source: own illustration, [4]).

Figure 5.3: Trajectories of 500 augmented variations of a rural driving scenario (source: own illustration, [4]).

5.5 Maneuver Detection

An important feature of the Providentia++ dataset is the labeling of maneuvers. This is generally a manual process done by experts in the field. Since this process is very labor intensive, one of the main goals of this work is to drastically reduce the effort needed for labeling maneuvers. To accomplish this, a maneuver detection framework is created and introduced in this work. The exact details of how this framework functions are explained in Section 4.5 of this work.

With the use of the highly adjustable maneuver detection framework, the following maneuvers can be detected automatically:

1. Turn left
2. Turn right
3. U-turn
4. Enter highway
5. Exit highway
6. Lane change left
7. Lane change right
8. Cut-in left

9. Cut-in right
10. Cut-out left
11. Cut-out right
12. Tailgate
13. Speeding vehicle
14. Standing vehicle

The detection accuracy of these maneuvers highly depends on the results of the tracking algorithm. This is due to the fact that tracking errors can lead to the extraction of incomplete trajectories of the actors. Only if the actor's trajectory passes through all of the required boxes, a maneuver can be detected. When tracking errors occur, the trajectory often ends before all of the detection boxes are reached, resulting in no maneuver being recognized.

However, when using synthetic data, all trajectories are complete. This is due to the nature of how they are generated.

Since the maneuver detection framework highly depends on the quality of the trajectories, the accuracy is determined in two independent experiments.

The first experiment determines the detection rate based on synthetic data, while the second experiment uses data extracted from the test stretch.

Both experiments are additionally carried out with augmented data. In this case, the base scenarios are augmented by the scenario augmentation framework before the maneuver detection algorithms are applied. The augmentation leads to a higher amount of test data and also reduces the statistical variance in the determined detection rates. The results of the experiments are shown and explained in more detail in the following two sections.

5.5.1 Maneuver Detection with Synthetic Data

In this section the achieved detection results based on augmented and non-augmented synthetically generated driving scenarios are presented and explained.

When using synthetically created driving scenarios with no further augmentation, the detection rate lies at 100 percent for all vehicle maneuvers. These include "turn left", "turn right", "straight at intersection", "U-turn", "lane change left", "lane change right", "enter highway", "exit highway", and "speeding". The results are achieved based on 17 different driving scenarios containing 2345 trajectories. Additionally to the 100 percent maneuver detection rate, no false positives occur. This means not a single detection is made when no maneuver occurs and also no maneuvers are misclassified.

The faultless detection rate is the result of two main factors.
One is the very robust nature and carefully chosen design of the detection algorithms.

Another factor that potentially could be beneficial is the synthetically generated data.

The generated trajectories are based on realistic driving maneuvers, but potentially may not contain driving maneuvers that are carried out in a non-typical way. This could be, for example, a vehicle taking an exceptionally wide right turn.

To reduce the chances of missing some of these special cases, the stochastic scenario augmentation framework is applied to the previously tested scenarios. This way, a high number of scenario variations is achieved, including special cases as mentioned above.

An overview of the maneuver detection rate based on synthetically created driving scenarios is given in Tables 5.1 and 5.2. Here a distinction between augmented and non-augmented test data is made.

Maneuver Type	Ground Truth Occurrences	Detected Occurrences	Precision	Recall
Turn Left	15	15	100	100
Turn Right	18	18	100	100
Straight	30	30	100	100
U-turn	12	12	100	100
Lane Chang Left	11	11	100	100
Lane Change Right	17	17	100	100
Cut-in Left	10	10	100	100
Cut-in Right	6	6	100	100
Cut-out Left	8	8	100	100
Cut-out Right	6	6	100	100
Enter Highway	15	15	100	100
Exit Highway	10	10	100	100
Minor Tailgate	39	39	100	100
Moderate Tailgate	21	21	100	100
Severe Tailgate	15	15	100	100
Speeding	106	106	100	100
Standing	7	7	100	100

Table 5.1: Maneuver detection results categorized by maneuver type based on synthetically created driving trajectories without augmentation (source: own illustration).

Maneuver Type	Ground Truth Occurrences	Detected Occurrences	Precision	Recall
Turn Left	303	303	100	100
Turn Right	303	303	100	100
Straight	404	404	100	100
U-turn	202	202	100	100
Lane Chang Left	121	121	100	100
Lane Change Right	185	185	100	100
Cut-in Left	110	110	100	100
Cut-in Right	47	47	100	100
Cut-out Left	71	71	100	100
Cut-out Right	68	68	100	100
Enter Highway	165	165	100	100
Exit Highway	110	110	100	100
Minor Tailgate	394	394	100	100
Moderate Tailgate	213	213	100	100
Severe Tailgate	138	138	100	100
Speeding	1156	1156	100	100
Standing	7	7	100	100

Table 5.2: Maneuver detection results categorized by maneuver type based on synthetically created driving trajectories with augmentation (source: own illustration).

5.5.2 Maneuver Detection with recorded real-world Data

In a second test series, the detection of driving maneuvers is tested with real-world driving scenarios recorded on the Providentia++ test stretch. Before going into further detail, it must be stated that at the time this work is created real-world driving data can only be extracted from the highway section of the Providentia++ test stretch. This consequently implies that only highway maneuvers can be detected from the real-world data recordings. These include lane changes, exiting the highway, as well as detecting standing vehicles.

In order to obtain driving scenarios from the recorded sensor data, some pre-processing steps are necessary. One of the main steps is forming trajectories and assigning them to the correct actor.

All vehicles and other actors recorded on the test stretch are tracked by an algorithm. This algorithm recognizes if the same actor appears in multiple data frames and assigns the same unique ID for this actor in each of these frames. The ID is then stored together with all the other information describing the actor, such as position, orientation, timestamp, and more. To create the trajectory for each actor, the data is first filtered by an ID. This way, all information describing a single actor at multiple time steps is obtained. Next, the positional information of the actor, which is provided in each data frame is sorted by the time steps, which are also provided in each frame. This way, the positional information of the actor at different points in time is brought into a chronological order and now represents a true trajectory.

In theory, with this strategy the trajectory of each actor can be obtained quite simply. In practice, however, a few problems occur. For example, if the tracking algorithm fails to recognize the same vehicle over multiple data frames, it assigns a new ID in each of these data

frames. In this case the trajectory of the actor cannot be obtained. Due to the varying IDs the positional information is mapped to multiple actors although in reality it may belong to one actor. This behavior is illustrated in Figures 5.4 and 5.5.

Figure 5.4: Ground truth trajectory of a single vehicle driving on the Providentia++ test stretch (source: own illustration, [1]).

Figure 5.5: Tracking result of a single vehicle driving on the Providentia++ test stretch. Instead of one complete trajectory the tracking algorithm detects five independent ones (source: own illustration, [1]).

Another tracking error that occurs, is not detecting a vehicle or actor at all. This can have multiple reasons, although two are most common. These are either the actor is occluded by other actors, or he is carrying out very abrupt movements. The latter error especially often occurs when the actor's orientation changes significantly in a very short period of time.

The results of the previously described tracking errors are either an incomplete trajectory or the detection of multiple actors instead of a single one. The maneuver detection algorithms developed in this work, however, heavily rely on trajectory data. To ensure all maneuvers of an actor can be detected, the trajectory must be complete over the entire test stretch without any gaps. Additionally, the trajectory must be correctly assigned to a single actor. This means that previously described incomplete trajectories are not acceptable. The trajectory of an actor must cannot be split into multiple actors as it currently can occur. An actor must completely be tracked as the same object throughout the entire test field leading to a complete trajectory. Unfortunately, this is only true for roughly 73.8 percent of the vehicles and actors tracked in the data recordings.

Since the tracking algorithm is not part of this work, the main goal is to test the performance of the maneuver detection algorithm. Therefore, the experiment is split into two parts. One experiment is based on all trajectories extracted from the recordings. This includes both complete and incomplete trajectories as well as other previously explained tracking errors. A trajectory is considered as complete if it starts at one of the entry areas of the Providentia++ test stretch and ends at one of the exit areas. Trajectories that end before one of the exit areas of the test stretch are reached or begin after one of the entry areas of the test stretch are considered as incomplete. This is illustrated in Figures 5.6 and 5.7.

The second experiment only uses trajectories that are complete and correctly assigned to an actor. Similarly to the previously described tests with synthetic data, both experiments are carried out once without and once with augmented data.

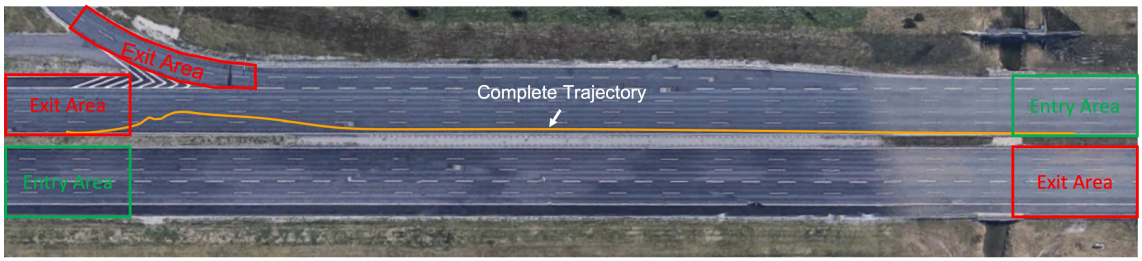

Figure 5.6: Example of a vehicle trajectory considered as complete based on the entry and exit areas of the Providentia++ test stretch. The complete trajectory starts in one of the entry areas of the test stretch and ends in one of the exit areas of the test stretch. (source: own illustration, [1]).

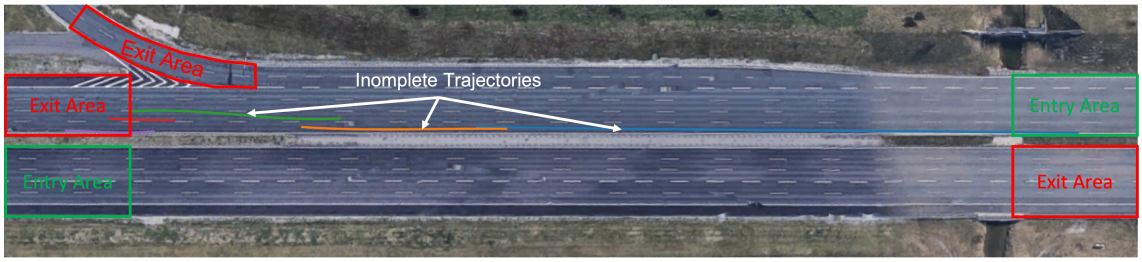

Figure 5.7: Example of multiple vehicle trajectories considered as incomplete based on the entry and exit areas of the Providentia++ test stretch. The incomplete trajectories either start outside the entry areas of the test stretch or end outside of the exit areas of the test stretch or both. (source: own illustration, [1]).

The distinction between testing with all trajectories versus testing with only complete and correctly assigned trajectories provides the opportunity to draw two main conclusions. First, when testing only with complete and correctly assigned trajectories, the correct detection rate of the maneuver recognition algorithms is obtained. This test procedure solely focuses on determining the quality of the detection algorithm because it only uses high-quality trajectory data.

In contrast, when testing with all trajectories the imperfections of the Providentia++ detection and tracking system are reflected in the data. The resulting maneuver detection rate cannot be higher compared to the maneuver detection rate when only using complete trajectories. Therefore, in this test procedure the detection rate of the maneuver detection algorithms is tested based on data including errors that are introduced through other components of the Providentia++ system. In other words the overall maneuver detection rate of the entire Providentia++ system is tested.

First, the results of the maneuver detection with complete trajectories are presented. Not much surprisingly, lane changes are detected with an accuracy of 100 percent. Also standing vehicles are recognized with 100 percent accuracy. Only the detection of vehicles exiting the highway fails completely, achieving a zero percent detection rate. However, this result must be brought into the right context. As seen in the tests with synthetic data, the detection of vehicles exiting the highway works very effectively. The reason for the poor results with real-world data is also very simple. The exit ramp of the highway lies mostly outside the view field

of the Providentia++ sensor station. This means that vehicles exiting the highway exit the sensor stations field of view before completing the exit maneuver. Therefore, it is impossible to detect exit maneuvers until further improvements on the sensor station are made.

When augmenting the real-world driving data the exact same results are achieved. Lane changes and standing vehicles are detected 100 percent, while exit maneuvers are again not detected. This lies in the nature of the augmentation framework, which only augments each driving trajectory within a small stochastic range. Tables 5.3 and 5.4 show the maneuver detection results for corrected and uncorrected trajectories without applying data augmentation.

Maneuver Type	Ground Truth Occurrences	Detected Occurrences	Precision	Recall
Lane Change Left	14	14	100	100
Lane Change Right	55	55	100	100
Cut-in Left	6	6	100	100
Cut-in Right	2	2	100	100
Cut-out Left	1	1	100	100
Cut-out Right	2	2	100	100
Exit Highway	17	0	0	0
Minor Tailgate	139	139	100	100
Moderate Tailgate	26	26	100	100
Severe Tailgate	10	10	100	100
Speeding Vehicle	103	103	100	100
Standing Vehicle	1	1	100	100

Table 5.3: Maneuver detection results categorized by maneuver type based on corrected real-world driving trajectories without augmentation (source: own illustration).

Maneuver Type	Ground Truth Occurrences	Detected Occurrences	Precision	Recall
Lane Change Left	14	16	87.5	100
Lane Change Right	55	77	71.4	100
Cut-in Left	6	13	46.2	100
Cut-in Right	2	5	40.0	100
Cut-out Left	1	1	100	100
Cut-out Right	2	5	40.0	100
Exit Highway	17	0	0	0
Minor Tailgate	139	229	60.7	100
Moderate Tailgate	26	64	40.6	100
Severe Tailgate	10	45	22.2	100
Speeding Vehicle	103	144	71.5	100
Standing Vehicle	1	1	100	100

Table 5.4: Maneuver detection results categorized by maneuver type based on non-corrected real-world driving trajectories without augmentation (source: own illustration).

Finally, the same tests are carried out with all available trajectories from the data recordings. This includes both complete and incomplete trajectories. Without data augmentation lane changes are detected with 74.2 percent precision and 100 percent recall whereas standing vehicles are detected with 100 percent precision and 100 percent recall. Again vehicles exiting the highway are not detected at all.

With augmented data, the results are very similar. Again, standing vehicles are detected with 100 percent. Lane changes are detected with 74.2 percent accuracy and 100 percent recall, whereas exit maneuvers are not detected at all. Tables 5.5 and 5.6 show the maneuver detection results for corrected and uncorrected trajectories with data augmentation.

Maneuver Type	Ground Truth Occurrences	Detected Occurrences	Precision	Recall
Lane Change Left	135	135	100	100
Lane Change Right	521	521	100	100
Cut-in Left	60	60	100	100
Cut-in Right	20	20	100	100
Cut-out Left	10	10	100	100
Cut-out Right	20	20	100	100
Exit Highway	170	0	0	0
Minor Tailgate	1431	1431	100	100
Moderate Tailgate	263	263	100	100
Severe Tailgate	105	105	100	100
Speeding Vehicle	1030	1030	100	100
Standing Vehicle	10	10	100	100

Table 5.5: Maneuver detection results categorized by maneuver type based on corrected real-world driving trajectories with augmentation (source: own illustration).

Maneuver Type	Ground Truth Occurrences	Detected Occurrences	Precision	Recall
Lane Change Left	135	156	86.5	100
Lane Change Right	521	687	75.8	100
Cut-in Left	60	163	36.8	100
Cut-in Right	20	46	43.5	100
Cut-out Left	10	17	58.8	100
Cut-out Right	20	31	64.5	100
Exit Highway	170	0	0	0
Minor Tailgate	1431	2529	56.6	100
Moderate Tailgate	263	639	41.2	100
Severe Tailgate	105	423	24.8	100
Speeding Vehicle	1030	1030	100	100
Standing Vehicle	10	10	100	100

Table 5.6: Maneuver detection results categorized by maneuver type based on non-corrected real-world driving trajectories with augmentation (source: own illustration).

5.6 OpenSCENARIO Writer

Another important accomplishment of this work is the visualization and simulation of driving scenarios. For simulation purposes, a script is created that automatically generates OpenSCENARIO files based on the scenario parameters given as an input. The OpenSCENARIO files can then again be run using the open source tools ScenarioRunner or Esmini player. The OpenSCENARIO writer is compatible with scenarios given in the scenario description format, which is explained in Section 4.6.

The scenario description format is created when extracting real world scenarios or generating synthetic scenarios. This implies both, outputs from the scenario extraction pipeline and the scenario generation framework can be used as an input for the OpenSCENARIO writer. This is highly beneficial because it allows for both real world and synthetic driving scenarios to be simulated and provides a solid basis for evaluating autonomous agents.

The main benefit of the OpenSCENARIO writer is the automation of the generation of OpenSCENARIO files. Of course, these files can also be written manually, but with an increase of actors and maneuvers this becomes quite cumbersome. Through the automation of this process large time savings are made.

5.7 Dataset

The final contribution made in this work is a complete dataset for conducting further research in the field of autonomous driving. In this section the components, format, and overall characteristics of the dataset are presented.

5.7.1 Dataset Components

The dataset includes numerous driving scenarios. Each of these scenarios is composed of four different data sources. These include image data from each of the cameras installed on the Providentia++ test stretch, sensor recordings including labels in the rosbag format, scenario environment and actor data including labels in JSON format, as well as a statistical overview of the driving scenario. The components included in the dataset are visualized in Figure 5.8.

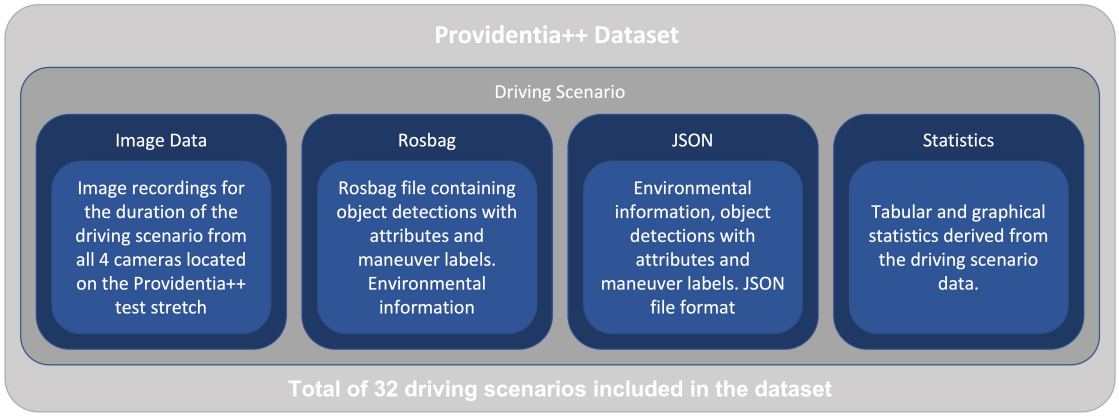

Figure 5.8: Overview of the components included in the Providentia++ dataset (source: own illustration).

1. Image data

   The image data contains the images recorded from each of the four cameras on the test stretch at the time of the driving scenario. The images are recorded at a frame rate of 25 Hz. This means for a 60 seconds long driving scenario 1500 images are recorded by each of the four cameras and included in the dataset.

2. Rosbag data

   Before the data recorded on the test stretch is stored, it is first preprocessed. This is done by multiple preprocessing scripts. Additionally, objects and their attributes (position, velocity, size, etc.) are identified. The processed data is then stored in rosbag files. The information stored in these files contain environmental information such as the time of day and the weather conditions, as well as object specific information including the position, velocity, size, type, and more.

3. JSON data

   Another data format found in the Providentia++ dataset is JSON. For each data frame of a driving scenario a JSON file is generated. This JSON file contains the same information as the rosbag file at each time step of the scenario. The JSON files are created at a frequency of 25 Hz - the same frequency as the sensor recordings.

4. Statistical overview

   The final element included in the dataset is a statistical overview of each driving scenario. This includes the number of speeding or standing vehicles at each time step, as well as the number of maneuvers of lane changes, cut-ins, cut-outs, tailgates, and more. The complete list of information included in the statistics is shown in Table 5.7.

   Additionally, a graphical overview of various driving maneuvers occurring throughout a scenario is included in the dataset. The graphical overview contains plots over time showing exactly when a vehicle performs certain driving maneuvers. The features included in the graphical statistics overview are explained in Table 5.8. Figure 5.9 shows an example of the graphical visualization of the statistics.

Statistics Type	Description
Total lane changes left	Sum of lane changes to the left committed by all vehicles throughout the duration of the driving scenario
Total lane changes right	Sum of lane changes to the right committed by all vehicles throughout the duration of the driving scenario
Total lane changes	Sum of all types of lane changes committed by all vehicles throughout the duration of the driving scenario
Maximum lane changes	Maximum amount of lane changes committed by a single vehicle throughout the duration of the driving scenario
Total cut-ins left	Sum of cut-ins to the left committed by all vehicles throughout the duration of the driving scenario
Total cut-ins right	Sum of cut-ins to the right committed by all vehicles throughout the duration of the driving scenario
Total cut-ins	Sum of all types of cut-ins committed by all vehicles throughout the duration of the driving scenario
Total cut-outs left	Sum of cut-outs to the left committed by all vehicles throughout the duration of the driving scenario
Total cut-outs right	Sum of cut-outs to the right committed by all vehicles throughout the duration of the driving scenario
Total cut-outs	Sum of all types of cut-outs committed by all vehicles throughout the duration of the driving scenario
Total minor tailgates	Sum of minor tailgates committed by all vehicles throughout the duration of the driving scenario
Total moderate tailgates	Sum of moderate tailgates committed by all vehicles throughout the duration of the driving scenario
Total severe tailgates	Sum of severe tailgates committed by all vehicles for the duration of the driving scenario
Total speeding vehicles	Sum of vehicles exceeding the allowed speed limit throughout the duration of the driving scenario
Total standing vehicles	Sum of vehicles standing throughout the duration of the driving scenario

Table 5.7: Overview of the statistics provided with every driving scenario (source: own illustration).

Statistics Type	Description
Trajectories	Visualization of all vehicle trajectories in the driving scenario
Velocity profiles	Visualization of all vehicle velocity profiles over time throughout the driving scenario
Lane IDs	Visualization of the driving lane each vehicle is using at every point in time throughout the driving scenario
Lane changes left	Visualization of point in time when a lane change to the left is committed by a vehicle throughout the duration of the driving scenario
Lane changes right	Visualization of point in time when a lane change to the right is committed by a vehicle throughout the duration of the driving scenario
Cut-ins left	Visualization of point in time when a cut-in to the left is committed by a vehicle throughout the duration of the driving scenario
Cut-ins right	Visualization of point in time when a cut-in to the right is committed by a vehicle throughout the duration of the driving scenario
Cut-outs left	Visualization of point in time when a cut-out to the left is committed by a vehicle throughout the duration of the driving scenario
Cut-outs right	Visualization of point in time when a cut-out to the right is committed by a vehicle throughout the duration of the driving scenario
Tailgates	Visualization of point in time when a tailgate is committed by a vehicle throughout the duration of the driving scenario. The numbers 1, 2, and 3 on the Y-axis indicate the severity of the tailgate maneuver. 1 indicates a minor tailgate, 2 a moderate tailgate, and 3 a severe tailgate.
Speeding	Visualization of vehicles exceeding the allowed speed limit at each point in time throughout the duration of the driving scenario
Total vehicles speeding	Total number of vehicles speeding at each point in time during the driving scenario
Standing	Visualization of standing vehicles at each point in time throughout the duration of the driving scenario
Total vehicles standing	Total number of vehicles standing at each point in time during the driving scenario

Table 5.8: Overview of the graphical statistics provided with every driving scenario (source: own illustration).

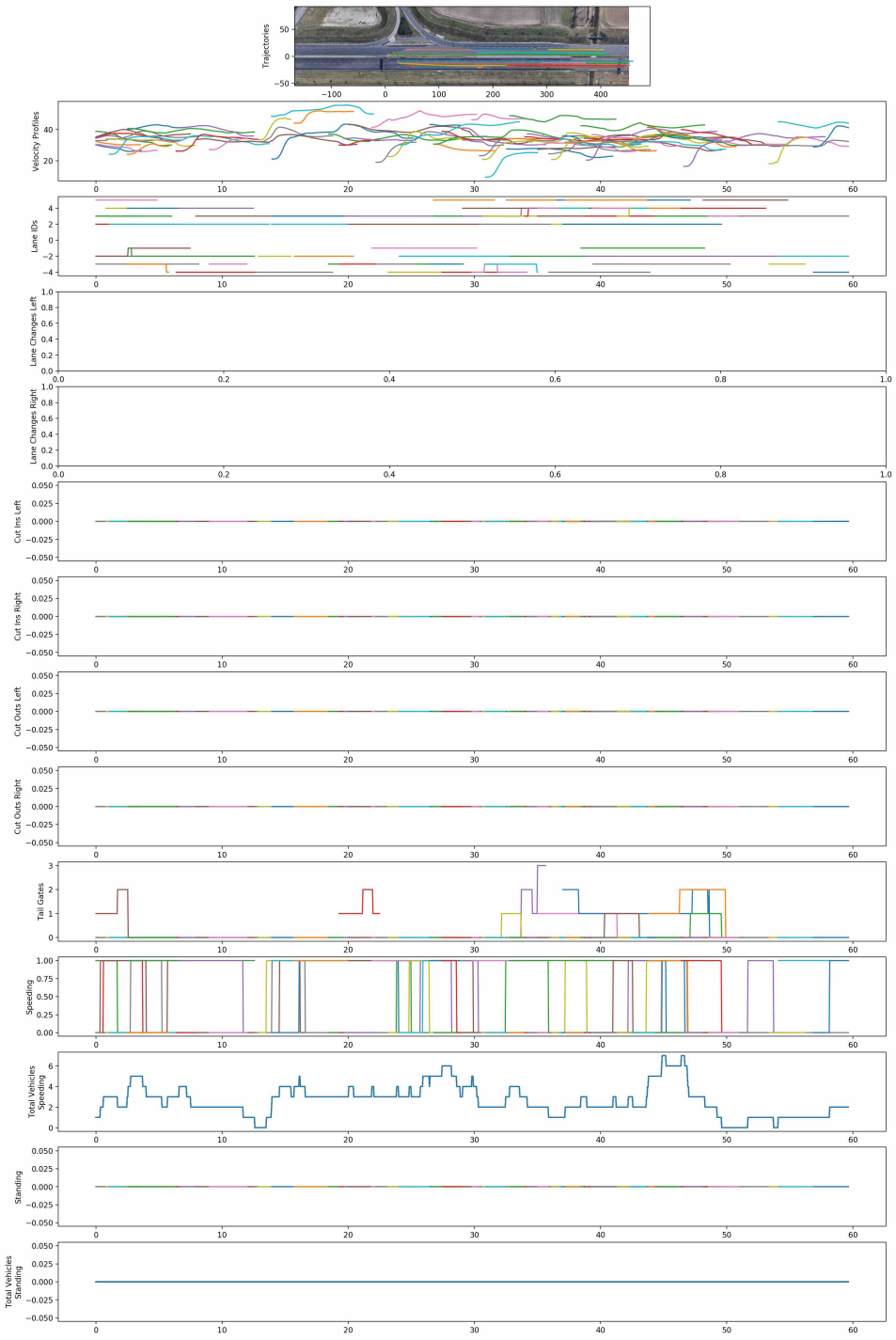

Figure 5.9: Graphical statistics overview showing the occurrence of various driving maneuvers over time (source: own illustration).

5.7.2 Dataset Characteristics

In this section the overall characteristics of the Providentia++ dataset are described. In total 32 different data recordings each with a length of one minute are included in the dataset. The dataset characteristics are composed of the driving maneuvers detected in the scenarios. These include the total number of lane changes, cut-ins, cut-outs, tailgates, speeding, and standing vehicles. Additionally the maximum number of lane changes committed by a single vehicle, the top speed, and the total number of vehicle trajectories are included in the statistics. Figure 5.10 shows the exact details of the dataset characteristics.

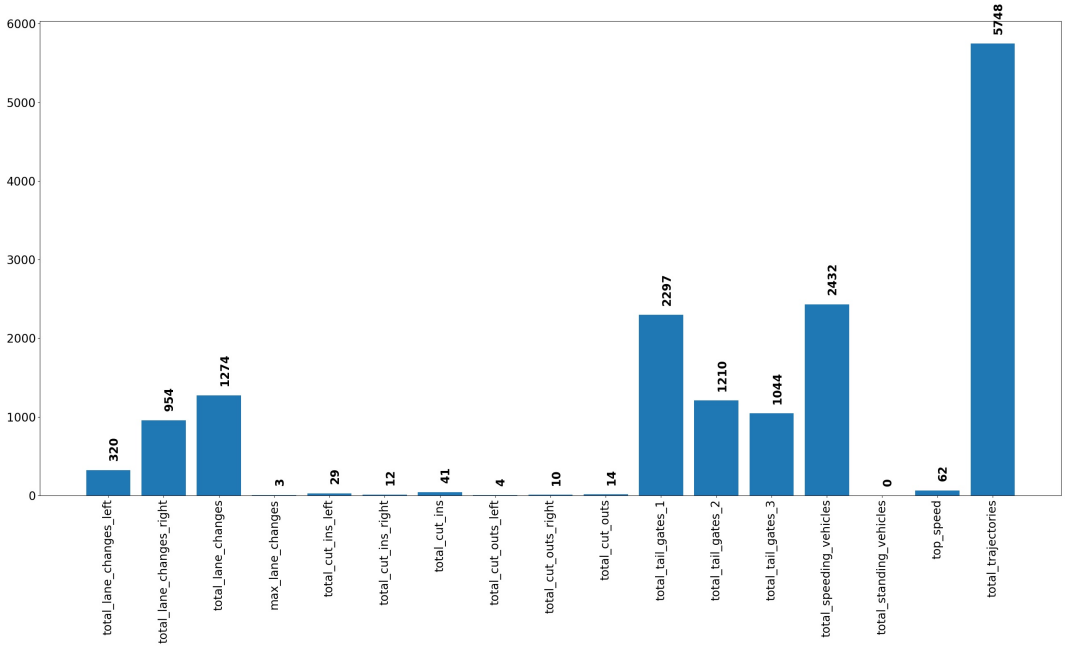

Figure 5.10: Overview of the Providentia++ dataset characteristics (source: own illustration).

Chapter 6

Summary

The efforts made in this work deal with parts of the scenario-based approach for the assessment of automated vehicles. This approach is chosen due to the fact that it is regarded as the most promising approach for guaranteeing an extensive safety assessment while maintaining sufficient efficiency. Hereby the following research questions are answered:

1. “Which driving scenarios are relevant for the assessment of automated vehicles capabilities?”

2. “How can an extensive collection of driving scenarios be created?”

3. “How can driving scenarios be carried out to assess the safety of automated vehicles?”

Each of the three research questions addresses a different task within the scenario-based approach. The research questions are first solved independently and later integrated into a processing pipeline covering the full scenario-based approach. In total seven main contributions are made. These include a scenario catalog as well as software frameworks for creating, augmenting, labeling, detecting, classifying, and visualizing driving scenarios. With the integration of these accomplishments into a processing pipeline the scenario-based approach is fully automated. The output of the processing pipeline is a database composed of driving scenarios with labeled driving maneuvers in rosbag and JSON format, statistics characterizing the scenario, and corresponding OpenSCENARIO files for simulation purposes.

In a first step a novel process model for the scenario-based approach is derived. Inspiration is taken from [148], however in this work the process structure is significantly different. In the newly derived process different elements play a key role and the overall model is condensed to reduce complexity. Additional efforts are made into designing the process highly modular, providing the opportunity of enhancing or exchanging process steps with new approaches.

The focus of this work lies in automating and simplifying the scenario-based approach by automating the process steps where possible. Tasks inside the process that necessarily require manual work are facilitated by introducing various software frameworks. All components are then integrated into a processing pipeline to streamline the process of creating driving scenarios, datasets, and simulation-based test scripts.

Due to the high variety of data generated with the scenario-based processing pipeline numerous problems can be solved. The generated test scripts are used for the safety assessment of automated vehicles. The datasets are used for the training of deep learning algorithms, for tasks such as trajectory prediction, maneuver classification and more. Additionally, traffic, or accident studies can be conducted based on the datasets.

In the following it is first described how each of the research questions is solved independently. Afterwards the functionality of the resulting processing pipeline for automating the scenario-based approach is outlined.

1. “Which driving scenarios are relevant for the assessment of automated vehicles capabilities?”

   Various sources are investigated in order to discover traffic scenario types that are both safety critical and relevant for the development of autonomous driving applications. Among these sources are documents from government agencies, such as the National Highway Traffic Safety Administration (NHTSA), Euro-NCAP, UNECE, and others. The goal of this inquiry is to find a reasonable balance between regular driving practices and situations that often cause accidents.

   These scenario types are then added to a shortlist, and a catalog specific to the Providentia++ test stretch is created from there. Following the conclusion of the selection, the scenario types that will appear on the Providentia++ test stretch are determined. Because some scenario types, such as entering a toll station or exiting a parking lot, are not conceivable on the test stretch because they do not exist, this is a vital step. As a consequence, they may go unnoticed. Once the Providentia++ test stretch’s required scenario types have been established, they are added to the catalog, which provides a thorough overview of scenarios.

2. How can an extensive collection of driving scenarios be created?

   To achieve an extensive collection of driving scenarios, two accomplishments are made. First, the driving scenarios are created as a representation of objects and their position over time within the road network. Secondly, it must be ensured that all driving scenarios listed in the scenario catalog are also included in the collection of concrete driving scenarios.

   For creating driving scenarios that cannot be extracted from the Providentia++ test stretch, the “Scenario Generation Framework" is introduced in this work. This framework enables the creation of synthetic driving scenarios. This applies frequently in the case of unusual occurrences, such as car accidents or vehicle breakdowns. Thanks to the generation framework, new driving scenarios may be designed rapidly and accurately to meet the demands of the user. Using the “Scenario Generation Framework" a total of 26 different driving scenarios are created and added to the database.

   For the automated creation of driving scenarios, a scenario extraction pipeline is implemented in this work. As an input it takes the data recordings from the Providentia++ test stretch which contain information about all detected vehicles and objects at each

point in time. The pipeline is referred to as the "Scenario Extraction Pipeline" in this work. It utilizes the sensor recordings to characterize the actors and their trajectories as well as the environment conditions. This information is then transferred into a format that formally describes concrete driving scenarios. Using the scenario extraction pipeline a total of 31 different driving scenarios are created and added to the database.

To accommodate for underrepresented scenario types the "Scenario Augmentation Framework" is introduced. This framework allows for a simple yet very effective variation of driving scenarios to increase the number of scenarios. The variations are created in a stochastic sampling-based approach. Hereby the paths and velocity profiles of all actors are varied. The user can control the stochastic sampling behavior by defining the sampling ranges. In this work the scenario augmentation framework is used for increasing the total amount of driving scenarios.

At this point driving scenarios are created either by extracting them from real-world data or synthetically generating them. The scenarios are additionally augmented to increase the total number of driving scenarios. Finally, the scenarios are classified depending on the driving patterns found in each scenario. With the classification of all extracted and generated scenarios it can be ensured that every scenario type listed in the scenario catalog is also included in the data collection of driving scenarios. The classification is done in two steps:

(a) Maneuver Detector

Many driving maneuvers that follow a deterministic behavior are modeled by rule-based algorithms. These driving maneuvers are then automatically be detected in driving scenarios and labeled in the data accordingly. Among the deterministic driving maneuvers for example are lane changes, cut-ins, cut-outs, speeding, among others.

More complex driving maneuvers, such as ones causing an accident, are labeled manually. All labels are then combined yielding a set of driving scenarios with fully labeled driving maneuvers.

(b) Scenario Statistics

Finally on the basis of the detected driving maneuvers, statistics describing the entire scenario are calculated. These include a summation of each maneuver type and extreme values such as the top speed or maximum number of lane changes committed by a single vehicle. Based on the statistics driving scenarios can be classified into a category.

3. How can driving scenarios be carried out to assess the safety of automated vehicles?

The assessment of automated vehicles in a simulation environment has benefits concerning reducing costs and avoiding safety hazards. Therefor this approach is chosen to execute driving scenarios for testing purposes.

To make use of various simulation software, a tool called "Scenario Writer" is created. It parses both extracted or synthetically generated driving scenarios into the OpenSCENARIO format. The OpenSCENARIO files can then be read by many simulation and

visualization tools for testing purposes. With driving scenarios in the OpenSCENARIO format test-series can then be automated.

After addressing and solving all research questions, the developed software components are integrated into pipeline. With this pipeline the scenario-based approach is streamlined and automated.

Chapter 7

Outlook

Six major milestones have been realized as a result of this work. These include the establishment of a scenario catalog with critical driving situations, a scenario extraction pipeline, a scenario generation framework, a scenario variation framework, a maneuver detection framework, and the automatic creation of OpenSCENARIO files for the simulation and analysis of scenarios. All these achievements are integrated into a pipeline that handles the scenario-based approach.

In this chapter an outlook is given on how the software components and pipeline created in this work can be used productively for introducing the scenario-based approach on the Providentia++ test stretch. First, the use cases for each individual achievement are pointed out. Finally, an additional use case concerning the pipeline resulting from these achievements is presented.

7.1 Scenario Catalog

The scenario catalog created in this work contains 86 different driving situations. These are categorized in the road categories urban, rural, and highway. With the categorization it is easier to determine which driving scenarios are relevant for a specific application.

In general, the scenario catalog can be used as an orientation which driving scenarios must be extracted or generated in order to cover all test cases required for a meaningful assessment of automated vehicles.

In the future this catalog can be continuously improved and expanded with additional driving situations which will make automated vehicles even more safe.

7.2 Scenario-Based Approach Pipeline

The pipeline created by combining the scenario extraction pipeline, scenario generation framework, scenario variation framework, maneuver detection framework, and the OpenSCENARIO writer has two main use cases. These use cases are outlined in the following.

1. Database Creation

The first use case is the automated creation of a scenario database. For this both data recordings from the Providentia++ test stretch, or generated driving scenarios are taken as input. Extracted driving scenarios are first pre-processed, while synthetic driving scenarios are created with the scenario generation framework. Depending on the user needs they can then be augmented in an automated procedure. Finally, maneuvers are detected, and statistics summarizing the driving scenarios are calculated. This means that an arbitrary driving scenario can be given as an input. Automatically it is then characterized and an OpenSCENARIO file is created for simulation purposes. This way a database for carrying out the scenario-based approach can be created very efficiently. Based on their statistics the driving scenarios can be classified and selected for testing purposes.

2. Driving Scenario Detection

   The second use case is the automated detection of relevant driving scenarios from the Providentia++ test stretch. For this the scenario extraction pipeline and the scenario classification algorithm only are used.

   Data recordings of the live traffic from the test stretch are generated by the sensor infrastructure continuously. Hereby each of the recordings covers a time span of one minute. For the detection of driving scenarios every produced data recording is extracted, maneuvers are then extracted, and finally the driving situation is classified. This way it is continuously determined what kind of traffic situation is occurring and if the situation is classified as relevant for the scenario-based approach. If a traffic situation is of interest, it is then stored into the database. The process is illustrated in Figure 7.1.

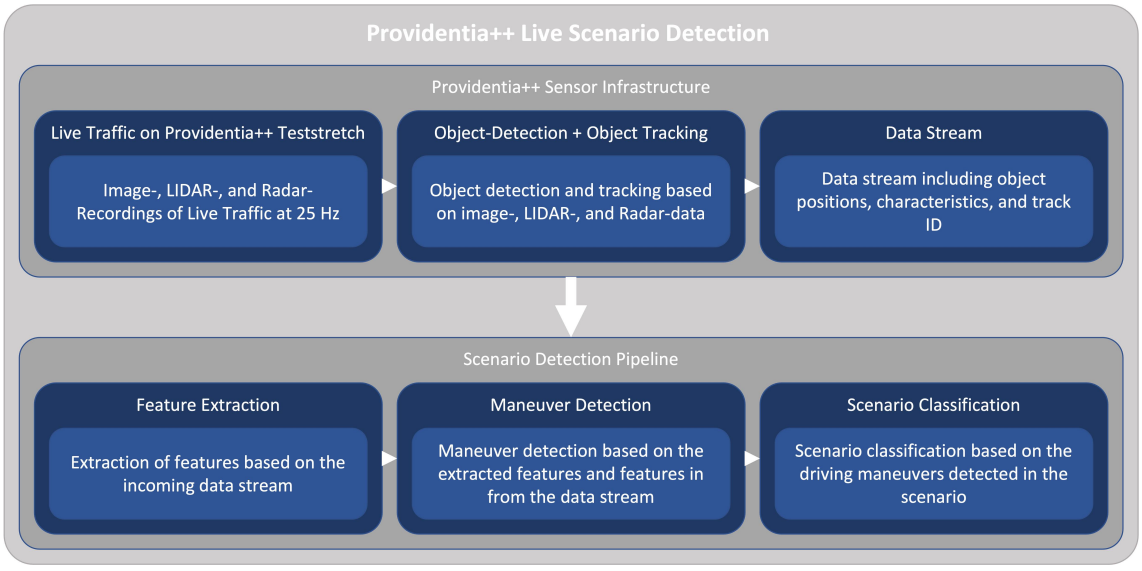

Figure 7.1: Illustration of the live detection of driving scenarios on the Providentia++ test stretch (source: own illustration).

In the past the driving scenario types that are included in the data recordings from the test stretch have been determined manually. This is done by re-watching each of the recordings and manually classifying the scenario type. With the ability of detecting rel-

evant driving scenarios automatically all manual labor becomes abundant.

Additionally, the labeling of driving maneuvers is automated to a large extent. Under the premise of 100 percent correctly tracked vehicles a maneuver detection rate of 100 percent including correct labeling is achieved. In practice due to the non-flawless vehicle tracking not all driving maneuvers are detected correctly. A maneuver detection rate of 100 percent is achieved excluding highway exit maneuvers. However, many false positives are generated due to errors in the vehicle tracking data. Therefore, a precision of only 60.72 percent on average over all maneuver classes can be achieved. This means that 60.72 percent of the detected maneuvers are labeled correctly. This reduces the effort of manual labeling for these maneuvers by roughly 40 percent. Data labeling is considered as labor intensive and the reduction of manual labeling efforts therefore has a high potential for cost-savings [72]. This indicates a development that the cost to provide automated vehicles will decrease in the future.

Chapter 8

Conclusion

The main goal of this work is to enable the assessment of AVs concerning their safety inside the environment of the Providentia++ project. For this inspiration is drawn from the SBA with a high focus on automating and simplifying each of the process components. Thereby the following tasks are solved:

1. Determination of relevant driving scenarios
2. Extraction of driving scenarios
3. Synthetic generation of driving scenarios
4. Variation of driving scenarios
5. Classification of driving scenarios
6. Simulation of driving scenarios
7. Creation of a driving scenario database

In the following the additional value, as well as room for improvements, is highlighted for each of the solutions developed for the above mentioned tasks.

1. Determination of relevant driving scenarios

   The goal of the scenario catalog created in this work is to produce an extensive list of driving scenarios describing the operational design domain of AVs. The catalog is heavily based on the accident report published by the NHTSA and enhanced by a number of scenarios from other sources. The resulting catalog is meant to include all driving scenarios that can occur on the Providentia++ test stretch. However, although the catalog is believed to be complete this can not be guaranteed for the future as new types of driving scenarios may arise. This can be due to changes in the vehicle operation, traffic laws, or road network. Therefore the scenario catalog must be consistently updated in the future.

2. Extraction of driving scenarios

   The extraction of driving scenarios is completely automated by the introduction of the scenario extraction pipeline. This means no manual work is necessary for the extraction process. In the future changes or updates to the pipeline may be necessary if the format of the sensor recordings changes. Also further efforts could be invested in the efficiency of the feature extraction. To achieve faster execution times of the feature extraction scripts a switch to an efficiency oriented programming language like C++ should be

considered. Also the effort of creating a lane ID extractor or creating goal boxes for the detection of driving maneuvers could be optimized for new road segments. Instead of manually defining polygonal hulls for each driving lane and goal box or goal polygon the features of the OpenDRIVE HD-map could be used. The information of each lane's boundaries and lane ID could be used to automatically create polygons for the extraction of the actor's lane ID. Similarly polygons defining goal boxes could be created. With the lane linkage information in intersections the detection rules for the goal box principle could also be assigned automatically. This would further simplify the feature extraction and make it scalable to new road segments.

3. Synthetic generation of driving scenarios

   The scenario generation framework introduced in this work drastically simplifies the process of generating synthetic driving scenarios. In the future new functionalities such as speed limits or generative scenario types could be added to further increase efficiency when creating new driving scenarios. Additionally, a graphical user interface could be created to provide a more detailed visual feedback while defining the actor's trajectories. This would make the process of generating and editing synthetic driving scenarios even more intuitive and efficient.

4. Variation of driving scenarios

   The scenario variation framework created as part of this work creates stochastic variations of existing driving scenarios. It can be used to easily create a stochastic complete variations of a driving scenario. In the future the framework could be extended to automatically sample completely new driving scenarios instead of stochastic variations of existing driving scenarios. This would further increase the test case coverage in the operational driving domain and account for the lack of recorded driving scenarios.

5. Classification of driving scenarios

   The classification of driving scenarios is based on the maneuvers detected inside the scenario. This can be useful for quickly identifying scenarios relevant for specific test suites. In the future the maneuver types detected in the driving scenarios can be further enhanced. This would allow for a more precise classification and increase the selection possibilities.

Additionally, the current maneuver detection algorithms have a detection rate of 100 percent. In practice, a maneuver detection rate of 100 percent is achieved excluding highway exit maneuvers. However, many false positives are generated due to errors in the vehicle tracking data. Therefore, a precision of only 60.72 percent on average over all maneuver classes can be achieved. With the increase of tracking accuracy to 100 percent an effective maneuver detection rate of 100 percent is achieved. Therefore, in the future, further efforts should be invested in the quality of the tracking algorithm.

Another way to increase the amount of detected driving scenarios in the urban and rural environment is to enhance the goal box detection. The existing framework could be used to detect ghost drivers, violations of traffic signs or rules, illegal turning maneuvers, or pedestrians crossing the road. These and more scenarios can be detected by using the existing or adding new goal boxes as well as defining new detection rules for each of the added scenarios. Also the placement of the goal polygons for detecting highway enter and exit maneuvers should be reconsidered. With a better positioning

the detection rate of zero percent for highway enter and exit maneuvers could be drastically increased.

To further enhance the number of driving maneuvers that can be detected and labeled the addition of new methods to the existing approach is highly recommended. The current approach deals very well with the detection of maneuvers that can be modeled by rule-based algorithms. However, it is limited concerning the detection of highly complex maneuvers such as an accident trajectory. In such a case the use of an Long-Short-Term Memory network could potentially bring good detection results. Reasons for this are the capability of the LSTM network to learn how to classify driving maneuvers correctly based on sequential data such as trajectories. This means detection models that are too complex to be modeled explicitly can simply be learned by the LSTM network. Finally, the introduction of a detection algorithm based on a Convolutional Neural Network could be well suited for detecting driving scenarios that are visually distinctive. Throughout this work it has been observed that most driving scenarios are prominent either due to their visuals or their trajectory. Therefore, a combination of both LSTM networks and Convolutional Neural Networks could be a promising approach to increase the number of detected maneuvers and driving scenarios.

Besides detecting new driving scenarios some enhancements can be made to existing ones. Similar to the detection of tailgates, cut-ins and cut-outs could be split into categories depending on the risk induced by executing the maneuver.

6. Creation of a driving scenario database

   The driving scenario database established in this work enables a fast identification of relevant test cases for various applications. Every driving scenario includes a statistic file summarizing the main characteristics of the scenario. This can be used to easily filter the driving scenarios necessary for certain test suites. In the future the statistics can be further enhanced by adding more parameters, such as new maneuver types or actor attributes. Additionally the database could be created with SQL [137], Influx [49], or MongoDB [132]. This would make it easier to scale the database or retrieve analytics from the stored data.

List of Figures

List of Tables

Bibliography

[1] (1), G. M. *Google Maps (1) Satellite Photograph Providentia++ Test Stretch Highway*. de. URL: https://www.google.de/maps/@48.2399411,11.6392164,522a,35y,108.76h/data=!3m1!1e3 (visited on 02/08/2022).

[2] (2), G. M. *Google Maps (2) Satellite Photograph Providentia++ Test Stretch Urban*. de. URL: https://www.google.de/maps/@48.2493125,11.6309106,232a,35y,7.45h/data=!3m1!1e3 (visited on 02/08/2022).

[3] (3), G. M. *Google Maps (3) Satellite Photograph Providentia++ Test Stretch Highway*. de. URL: https://www.google.de/maps/@48.2465063,11.6415044,389m/data=!3m1!1e3 (visited on 02/08/2022).

[4] (4), G. M. *Google Maps (4) Satellite Photograph Providentia++ Test Stretch Rural*. de. URL: https://www.google.de/maps/@48.248195,11.6350177,234m/data=!3m1!1e3 (visited on 02/08/2022).

[5] @MattRowe18. *CARLA 0.9.13 Release*. Nov. 2021. URL: http://carla.org//2021/11/16/release-0.9.13/ (visited on 12/10/2021).

[6] Abbas, H., O'Kelly, M., Rodionova, A., and Mangharam, R. "Safe At Any Speed: A Simulation-Based Test Harness for Autonomous Vehicles". In: (2017).

[7] *About ASAM*. en. URL: https://www.asam.net/about-asam/ (visited on 12/10/2021).

[8] Adriaans, P. and Zantinge, D. *Data mining*. English. OCLC: 35917402. Harlow, England; Reading, Mass.: Addison-Wesley, 1996. ISBN: 978-0-201-40380-0.

[9] Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., Homm, F., Huber, W., and Kaempchen, N. "Experience, Results and Lessons Learned from Automated Driving on Germany's Highways". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (2015). Conference Name: IEEE Intelligent Transportation Systems Magazine, pp. 42–57. ISSN: 1941-1197. DOI: 10.1109/MITS.2014.2360306.

[10] Ahmed, S. A., Dogra, D. P., Kar, S., and Roy, P. P. "Trajectory-Based Surveillance Analysis: A Survey". en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.7 (July 2019), pp. 1985–1997. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2018.2857489. URL: https://ieeexplore.ieee.org/document/8413165/ (visited on 11/22/2021).

[11] Alpaydin, E. *Introduction to machine learning*. en. Third edition. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2014. ISBN: 978-0-262-02818-9.

[12] Altché, F. and La Fortelle, A. de. "An LSTM network for highway trajectory prediction". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Oct. 2017, pp. 353–359. DOI: 10.1109/ITSC.2017.8317913.

[13] Althoff, M., Koschi, M., and Manzinger, S. “CommonRoad: Composable benchmarks for motion planning on roads”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 719–726. DOI: 10.1109/IVS.2017.7995802.

[14] Althoff, M. and Lutz, S. “Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587. June 2018, pp. 1326–1333. DOI: 10.1109/IVS.2018.8500374.

[15] Amini, A., Gilitschenski, I., Phillips, J., Moseyko, J., Banerjee, R., Karaman, S., and Rus, D. “Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation”. In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020). Conference Name: IEEE Robotics and Automation Letters, pp. 1143–1150. ISSN: 2377-3766. DOI: 10.1109/LRA.2020.2966414.

[16] Antona-Makoshi, J., Uchida, N., Kitahara, E., and Ozawa, K. “A Safety Assurance Process for Automated Driving Systems”. en. In: (2019), p. 6.

[17] Arief, M., Glynn, P., and Zhao, D. “An Accelerated Approach to Safely and Efficiently Test Pre-Production Autonomous Vehicles on Public Streets”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 2006–2011. DOI: 10.1109/ITSC.2018.8569371.

[18] *ASAM Members*. en. URL: https://www.asam.net/members/ (visited on 12/10/2021).

[19] *ASAM OpenDrive*. en. URL: https://www.asam.net/standards/detail/opendrive/ (visited on 12/10/2021).

[20] *ASAM OpenScenario*. en. URL: https://www.asam.net/standards/detail/openscenario/ (visited on 12/10/2021).

[21] Azevedo, A. and Santos, M. “KDD, semma and CRISP-DM: A parallel overview”. In: Jan. 2008, pp. 182–185.

[22] Bach, J., Langner, J., Otten, S., Holzäpfel, M., and Sax, E. “Test Scenario Selection for System-Level Verification and Validation of Geolocation-Dependent Automotive Control Systems”. In: June 2017. DOI: 10.1109/ICE.2017.8279890.

[23] Bagschik, G., Menzel, T., and Maurer, M. “Ontology based Scene Creation for the Development of Automated Vehicles”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587. June 2018, pp. 1813–1820. DOI: 10.1109/IVS.2018.8500632.

[24] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. “A study of the behavior of several methods for balancing machine learning training data”. en. In: *ACM SIGKDD Explorations Newsletter* 6.1 (June 2004), pp. 20–29. ISSN: 1931-0145, 1931-0153. DOI: 10.1145/1007730.1007735. URL: https://dl.acm.org/doi/10.1145/1007730.1007735 (visited on 12/07/2021).

[25] Batsch, F., Kanarachos, S., Cheah, M., Ponticelli, R., and Blundell, M. “A taxonomy of validation strategies to ensure the safe operation of highly automated vehicles”. en. In: *Journal of Intelligent Transportation Systems* (Mar. 2020), pp. 1–20. ISSN: 1547-2450, 1547-2442. DOI: 10.1080/15472450.2020.1738231. URL: https://www.tandfonline.com/doi/full/10.1080/15472450.2020.1738231 (visited on 12/10/2021).

[26] Beglerovic, H., Stolz, M., and Horn, M. “Testing of autonomous vehicles using surrogate models and stochastic optimization”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Oct. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317768.

[27] Beglerović, H., Schlömicher, T., Metzner, S., and Horn, M. "Deep Learning Applied to Scenario Classification for Lane-Keep-Assist Systems". In: *Applied Sciences* 8 (Dec. 2018), p. 1. DOI: 10.3390/app8122590.

[28] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". en. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 9296–9306. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00939. URL: https://ieeexplore.ieee.org/document/9010727/ (visited on 12/03/2021).

[29] Ben Abdessalem, R., Nejati, S., Briand, L. C., and Stifter, T. "Testing advanced driver assistance systems using multi-objective search and neural networks". In: *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Sept. 2016, pp. 63–74.

[30] Biever, W., Angell, L., and Seaman, S. "Automated Driving System Collisions: Early Lessons". en. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 62.2 (Mar. 2020), pp. 249–259. ISSN: 0018-7208, 1547-8181. DOI: 10.1177/0018720819872034. URL: http://journals.sagepub.com/doi/10.1177/0018720819872034 (visited on 12/06/2021).

[31] Bolte, J.-A., Bar, A., Lipinski, D., and Fingscheidt, T. "Towards Corner Case Detection for Autonomous Driving". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 438–445. DOI: 10.1109/IVS.2019.8813817.

[32] Brachman, R. J. and Anand, T. "The process of knowledge discovery in databases". In: *Advances in knowledge discovery and data mining*. USA: American Association for Artificial Intelligence, Feb. 1996, pp. 37–57. ISBN: 978-0-262-56097-9. (Visited on 12/10/2021).

[33] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. "nuScenes: A Multimodal Dataset for Autonomous Driving". en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 11618–11628. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.01164. URL: https://ieeexplore.ieee.org/document/9156412/ (visited on 12/06/2021).

[34] Campbell, M., Egerstedt, M., How, J. P., and Murray, R. M. "Autonomous driving in urban environments: approaches, lessons and challenges". en. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1928 (Oct. 2010), pp. 4649–4672. ISSN: 1364-503X, 1471-2962. DOI: 10.1098/rsta.2010.0110. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2010.0110 (visited on 11/22/2021).

[35] *CARLA ScenarioRunner*. URL: https://carla-scenariorunner.readthedocs.io/en/latest/ (visited on 12/10/2021).

[36] *CETRAN - Centre of Excellence for Testing and Research of Autonomous Vehicles - NTU*. en-GB. URL: https://cetran.sg/ (visited on 02/13/2022).

[37] Chang, M.-F., Ramanan, D., Hays, J., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., and Lucey, S. "Argoverse: 3D Tracking and Forecasting With Rich Maps". en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 8740–8749. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00895. URL: https://ieeexplore.ieee.org/document/8953693/ (visited on 12/06/2021).

[38] Chawla, N. V., Japkowicz, N., and Kotcz, A. “Editorial: special issue on learning from imbalanced data sets”. en. In: *ACM SIGKDD Explorations Newsletter* 6.1 (June 2004), pp. 1–6. ISSN: 1931-0145, 1931-0153. DOI: 10.1145/1007730.1007733. URL: https://dl.acm.org/doi/10.1145/1007730.1007733 (visited on 12/07/2021).

[39] Chen, L., Zhan, W., Tian, W., He, Y., and Zou, Q. “Deep Integration: A Multi-Label Architecture for Road Scene Recognition”. en. In: *IEEE Transactions on Image Processing* 28.10 (Oct. 2019), pp. 4883–4898. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2019.2913079. URL: https://ieeexplore.ieee.org/document/8708965/ (visited on 11/22/2021).

[40] Chen, W. and Kloul, L. “An Ontology-based Approach to Generate the Advanced Driver Assistance Use Cases of Highway Traffic:” en. In: *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Seville, Spain: SCITEPRESS - Science and Technology Publications, 2018, pp. 75–83. ISBN: 978-989-758-330-8. DOI: 10.5220/0006931700750083. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006931700750083 (visited on 12/09/2021).

[41] Chu, H., Guo, L., Gao, B., Chen, H., Bian, N., and Zhou, J. “Predictive Cruise Control Using High-Definition Map and Real Vehicle Implementation”. en. In: *IEEE Transactions on Vehicular Technology* 67.12 (Dec. 2018), pp. 11377–11389. ISSN: 0018-9545, 1939-9359. DOI: 10.1109/TVT.2018.2871202. URL: https://ieeexplore.ieee.org/document/8468109/ (visited on 11/22/2021).

[42] Cohen, G., Afshar, S., Tapson, J., and Schaik, A. van. “EMNIST: Extending MNIST to handwritten letters”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407. May 2017, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217.

[43] Collins, J., Howard, D., and Leitner, J. “Quantifying the Reality Gap in Robotic Manipulation Tasks”. en. In: *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 6706–6712. ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8793591. URL: https://ieeexplore.ieee.org/document/8793591/ (visited on 12/07/2021).

[44] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 3213–3223. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.350. URL: http://ieeexplore.ieee.org/document/7780719/ (visited on 12/06/2021).

[45] Corso, A., Du, P., Driggs-Campbell, K., and Kochenderfer, M. J. “Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validation”. In: *arXiv:1908.01046 [cs, eess, stat]* (Aug. 2019). arXiv: 1908.01046. URL: http://arxiv.org/abs/1908.01046 (visited on 01/25/2022).

[46] *CRISP-DM: Ein Standard-Prozess-Modell für Data Mining | Statistik Dresden*. de-DE. Apr. 2012. URL: https://statistik-dresden.de/archives/1128 (visited on 12/10/2021).

[47] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. “AutoAugment: Learning Augmentation Strategies From Data”. en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 113–123. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00020. URL: https://ieeexplore.ieee.org/document/8953317/ (visited on 11/22/2021).

[48] *Current weather and forecast - OpenWeatherMap*. URL: https://openweathermap.org/ (visited on 01/26/2022).

[49] *Customer Experience & Customer Support Agency*. en. URL: https://influx.com/ (visited on 02/14/2022).

[50] Dash, M. and Liu, H. "Feature selection for classification". en. In: *Intelligent Data Analysis* 1.1 (Jan. 1997), pp. 131–156. ISSN: 1088-467X. DOI: 10.1016/S1088-467X(97)00008-5. URL: https://www.sciencedirect.com/science/article/pii/S1088467X97000085 (visited on 01/24/2022).

[51] Dávid, B., Láncz, G., and Hunyady, G. "Highway Situation Analysis with Scenario Classification and Neural Network based Risk Estimation for Autonomous Vehicles". In: *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. Jan. 2019, pp. 375–380. DOI: 10.1109/SAMI.2019.8782729.

[52] Dixit, V. V., Chand, S., and Nair, D. J. "Autonomous Vehicles: Disengagements, Accidents and Reaction Times". en. In: *PLOS ONE* 11.12 (Dec. 2016). Ed. by Xu, J., e0168054. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0168054. URL: https://dx.plos.org/10.1371/journal.pone.0168054 (visited on 12/06/2021).

[53] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Levine, S., Vanhoucke, V., and Goldberg, K. Vol. 78. Proceedings of Machine Learning Research. PMLR, Nov. 2017, pp. 1–16. URL: https://proceedings.mlr.press/v78/dosovitskiy17a.html.

[54] Dupuis, M., Strobl, M., and Grezlikowski, H. "OpenDRIVE 2010 and Beyond – Status and Future of the de facto Standard for the Description of Road Networks". en. In: *Trends in driving simulation design and experiments: proceedings of the driving simulation conference Europe 2010, Arts et métiers ParisTech, France, September 9-10, 2010*. Actes INRETS A 126. Bron: INRETS, 2010, pp. 233–242. ISBN: 978-2-85782-685-9.

[55] Edwards, J. B. "The Relationship Between Road Accident Severity and Recorded Weather". en. In: *Journal of Safety Research* 29.4 (1998), pp. 249–262. ISSN: 0022-4375. DOI: 10.1016/S0022-4375(98)00051-6. URL: https://www.sciencedirect.com/science/article/pii/S0022437598000516 (visited on 01/25/2022).

[56] Erdogan, A., Ugranli, B., Adali, E., Sentas, A., Mungan, E., Kaplan, E., and Leitner, A. "Real- World Maneuver Extraction for Autonomous Vehicle Validation: A Comparative Study". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 267–272. DOI: 10.1109/IVS.2019.8814254.

[57] *Euro NCAP | 2018 Automatisiertes Fahren*. URL: https://www.euroncap.com/de/fahrzeugsicherheit/sicherheitskampagnen/2018-automatisiertes-fahren/ (visited on 11/17/2021).

[58] Europe, U. N. E. C. for. "Regelung Nr. 131 der Wirtschaftskommission der Vereinten Nationen für Europa (UNECE) — Einheitliche Bedingungen für die Genehmigung von Kraftfahrzeugen hinsichtlich des Notbremsassistenzsystems (AEBS)". de. In: (2014), p. 16.

[59] Fahrenkrog, F., Wang, L., Platzer, T., Fries, A., and Raisch, F. "PROSPECTIVE SAFETY EFFECTIVENESS ASSESSMENT OF AUTOMATED DRIVING FUNCTIONS – FROM THE METHODS TO THE RESULTS". en. In: (2019), p. 11.

[60] Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. "Data augmentation using synthetic data for time series classification with deep residual networks". en. In: *arXiv:1808.02455 [cs]* (Aug. 2018). arXiv: 1808.02455. URL: http://arxiv.org/abs/1808.02455 (visited on 11/22/2021).

[61] Fayyad, U. and Irani, K. "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning". en US. In: (Sept. 1993). Accepted: 2004-10-06T02:12:48Z. URL: https://trs.jpl.nasa.gov/handle/2014/35171 (visited on 12/10/2021).

[62] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. "From Data Mining to Knowledge Discovery in Databases". en. In: *AI Magazine* 17.3 (Mar. 1996). Number: 3, pp. 37–37. ISSN: 2371-9621. DOI: 10.1609/aimag.v17i3.1230. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/1230 (visited on 12/10/2021).

[63] Feig, P., Schatz, J., Audi, L., and Leonhardt, T. "Assessment of Technical Requirements for Level 3 and Beyond Automated Driving Systems Based on Naturalistic Driving and Accident Data Analysis". In: June 2019.

[64] Gangopadhyay, B., Khastgir, S., Dey, S., Dasgupta, P., Montana, G., and Jennings, P. "Identification of Test Cases for Automated Driving Systems Using Bayesian Optimization". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Oct. 2019, pp. 1961–1967. DOI: 10.1109/ITSC.2019.8917103.

[65] Gao, F., Duan, J., He, Y., and Wang, Z. "A Test Scenario Automatic Generation Strategy for Intelligent Driving Systems". In: *Mathematical Problems in Engineering* 2019 (Jan. 2019), pp. 1–10. DOI: 10.1155/2019/3737486.

[66] Gao, K., Tu, H., Sun, L., Sze, N., Song, Z., and Shi, H. "Impacts of reduced visibility under hazy weather condition on collision risk and car-following behavior: Implications for traffic control and management". In: *International Journal of Sustainable Transportation* 14.8 (June 2020). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15568318.2019.1597226, pp. 635–642. ISSN: 1556-8318. DOI: 10.1080/15568318.2019.1597226. URL: https://doi.org/10.1080/15568318.2019.1597226 (visited on 01/25/2022).

[67] García, S., Luengo, J., and Herrera, F. *Data Preprocessing in Data Mining*. en. Vol. 72. Intelligent Systems Reference Library. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-10246-7 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4. URL: http://link.springer.com/10.1007/978-3-319-10247-4 (visited on 12/07/2021).

[68] Gelder, E. de and Paardekooper, J.-P. "Assessment of Automated Driving Systems using real-life scenarios". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 589–594. DOI: 10.1109/IVS.2017.7995782.

[69] Gelder, E. d., Manders, J., Grappiolo, C., Paardekooper, J.-P., Camp, O. O. d., and Schutter, B. D. "Real-World Scenario Mining for the Assessment of Automated Vehicles". en. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, Sept. 2020, pp. 1–8. ISBN: 978-1-72814-149-7. DOI: 10.1109/ITSC45102.2020.9294652. URL: https://ieeexplore.ieee.org/document/9294652/ (visited on 11/22/2021).

[70] Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A. S., Hauswald, L., Pham, V. H., Mühlegg, M., Dorn, S., Fernandez, T., Jänicke, M., Mirashi, S., Savani, C., Sturm, M., Vorobiov, O., Oelker, M., Garreis, S., and Schuberth, P. "A2D2: Audi Autonomous Driving Dataset". en. In: *arXiv:2004.06320 [cs, eess]* (Apr. 2020). arXiv: 2004.06320. URL: http://arxiv.org/abs/2004.06320 (visited on 12/07/2021).

[71] Geyer, S., Baltzer, M., Franz, B., Hakuli, S., Kauer, M., Kienle, M., Meier, S., Weißgerber, T., Bengler, K., Bruder, R., Flemisch, F., and Winner, H. "Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance". en. In: *IET Intelligent Transport Systems* 8.3 (2014). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-its.2012.0188, pp. 183–189. ISSN: 1751-9578. DOI: 10.1049/iet-its.2012.0188. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-its.2012.0188 (visited on 12/09/2021).

[72] Gilyazev, R. A. and Turdakov, D. Y. "Active Learning and Crowdsourcing: A Survey of Optimization Methods for Data Labeling". en. In: *Programming and Computer Software* 44.6 (Nov. 2018), pp. 476–491. ISSN: 0361-7688, 1608-3261. DOI: 10.1134/S0361768818060142. URL: http://link.springer.com/10.1134/S0361768818060142 (visited on 01/25/2022).

[73] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. en. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 978-0-262-03561-3.

[74] Graves, A. "Long Short-Term Memory". In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45. ISBN: 978-3-642-24797-2. DOI: 10.1007/978-3-642-24797-2_4. URL: https://doi.org/10.1007/978-3-642-24797-2_4.

[75] Gruner, R., Henzler, P., Hinz, G., Eckstein, C., and Knoll, A. "Spatiotemporal representation of driving scenarios and classification using neural networks". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 1782–1788. DOI: 10.1109/IVS.2017.7995965.

[76] Guo, J., Kurup, U., and Shah, M. "Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 21.8 (Aug. 2020). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 3135–3151. ISSN: 1558-0016. DOI: 10.1109/TITS.2019.2926042.

[77] Hallerbach, S., Xia, Y., Eberle, U., and Köster, F. "Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles". In: *SAE Technical Papers* 2018-01-1066 (Apr. 2018). DOI: 10.4271/2018-01-1066.

[78] Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*. en. Morgan Kaufmann, 2011. ISBN: 978-0-12-381479-1. (Visited on 12/07/2021).

[79] Hand, D. J., Mannila, H., and Smyth, P. *Principles of data mining*. en. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2001. ISBN: 978-0-262-08290-7.

[80] Hauer, F., Schmidt, T., Holzmüller, B., and Pretschner, A. "Did We Test All Scenarios for Automated and Autonomous Driving Systems?" In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Oct. 2019, pp. 2950–2955. DOI: 10.1109/ITSC.2019.8917326.

[81] Hayward, J. "NEAR-MISS DETERMINATION THROUGH USE OF A SCALE OF DANGER". en. In: *undefined* (1972). URL: https://www.semanticscholar.org/paper/NEAR-MISS-DETERMINATION-THROUGH-USE-OF-A-SCALE-OF-Hayward/d7dc3003871814c9ac84d9c75456aa8089fc70fd (visited on 01/25/2022).

[82] He, H. and Garcia, E. A. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1263–1284. ISSN: 1558-2191. DOI: 10.1109/TKDE.2008.239.

[83] Hochreiter, S. and Schmidhuber, J. “Long Short-Term Memory”. en. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[84] *Home - pegasus-EN*. URL: https://www.pegasusprojekt.de/en/home (visited on 01/24/2022).

[85] Huang, W., Wang, K., Lv, Y., and Zhu, F. “Autonomous vehicles testing methods review”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2016, pp. 163–168. DOI: 10.1109/ITSC.2016.7795548.

[86] Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., and Yang, R. “The ApolloScape Open Dataset for Autonomous Driving and Its Application”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (Oct. 2020). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 2702–2719. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2926463.

[87] Jakobi, N., Husbands, P., and Harvey, I. “Noise and the reality gap: The use of simulation in evolutionary robotics”. en. In: *Advances in Artificial Life*. Ed. by Goos, G., Hartmanis, J., Leeuwen, J., Carbonell, J. G., Siekmann, J., Morán, F., Moreno, A., Merelo, J. J., and Chacón, P. Vol. 929. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720. ISBN: 978-3-540-59496-3 978-3-540-49286-3. DOI: 10.1007/3-540-59496-5_337. URL: http://link.springer.com/10.1007/3-540-59496-5_337 (visited on 12/07/2021).

[88] Jenkins, I. R., Gee, L. O., Knauss, A., Yin, H., and Schroeder, J. “Accident Scenario Generation with Recurrent Neural Networks”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 3340–3345. DOI: 10.1109/ITSC.2018.8569661.

[89] Jesenski, S., Stellet, J. E., Schiegg, F., and Zöllner, J. M. “Generation of Scenes in Intersections for the Validation of Highly Automated Driving Functions”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 502–509. DOI: 10.1109/IVS.2019.8813776.

[90] Jiao, J. “Machine Learning Assisted High-Definition Map Creation”. en. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Tokyo, Japan: IEEE, July 2018, pp. 367–373. ISBN: 978-1-5386-2666-5. DOI: 10.1109/COMPSAC.2018.00058. URL: https://ieeexplore.ieee.org/document/8377682/ (visited on 11/22/2021).

[91] Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. “A Survey of Deep Learning-Based Object Detection”. In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 128837–128868. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2939201.

[92] Jo, K., Kim, C., and Sunwoo, M. “Simultaneous Localization and Map Change Update for the High Definition Map-Based Autonomous Driving Car”. en. In: *Sensors* 18.9 (Sept. 2018), p. 3145. ISSN: 1424-8220. DOI: 10.3390/s18093145. URL: http://www.mdpi.com/1424-8220/18/9/3145 (visited on 11/22/2021).

[93] Jullien, J.-M., Martel, C., Vignollet, L., and Wentland, M. "OpenScenario: A Flexible Integrated Environment to Develop Educational Activities Based on Pedagogical Scenarios". In: *2009 Ninth IEEE International Conference on Advanced Learning Technologies*. ISSN: 2161-377X. July 2009, pp. 509–513. DOI: 10.1109/ICALT.2009.24.

[94] Junietz, P. "Microscopic and Macroscopic Risk Metrics for the Safety Validation of Automated Driving". In: 2019. DOI: 10.25534/tuprints-00009282.

[95] Junietz, P., Bonakdar, F., Klamann, B., and Winner, H. "Criticality Metric for the Safety Validation of Automated Driving using Model Predictive Trajectory Optimization". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 60–65. DOI: 10.1109/ITSC.2018.8569326.

[96] Junietz, P., Wachenfeld, W., Klonecki, K., and Winner, H. "Evaluation of Different Approaches to Address Safety Validation of Automated Driving". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 491–496. DOI: 10.1109/ITSC.2018.8569959.

[97] Kalra, N. and Paddock, S. M. "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" In: *Transportation Research Part A: Policy and Practice* 94 (2016), pp. 182–193. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2016.09.010. URL: https://www.sciencedirect.com/science/article/pii/S0965856416302129.

[98] Kang, Y., Yin, H., and Berger, C. "Test Your Self-Driving Algorithm: An Overview of Publicly Available Driving Datasets and Virtual Testing Environments". In: *IEEE Transactions on Intelligent Vehicles* 4.2 (June 2019). Conference Name: IEEE Transactions on Intelligent Vehicles, pp. 171–185. ISSN: 2379-8904. DOI: 10.1109/TIV.2018.2886678.

[99] Kirchhof, J. C., Kusmenko, E., Rumpe, B., and Zhang, H. "Simulation as a Service for Cooperative Vehicles". In: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. Sept. 2019, pp. 28–37. DOI: 10.1109/MODELS-C.2019.00011.

[100] Kitajima, S., Shimono, K., Tajima, J., Antona-Makoshi, J., and Uchida, N. "Multi-agent traffic simulations to estimate the impact of automated technologies on safety". In: *Traffic Injury Prevention* 20 (June 2019), S58–S64. DOI: 10.1080/15389588.2019.1625335.

[101] Klischat, M. and Althoff, M. "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 2352–2358. DOI: 10.1109/IVS.2019.8814230.

[102] Klueck, F., Li, Y., Nica, M., Tao, J., and Wotawa, F. "Using Ontologies for Test Suites Generation for Automated and Autonomous Driving Functions". In: *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. Oct. 2018, pp. 118–123. DOI: 10.1109/ISSREW.2018.00-20.

[103] Koopman, P. and Fratrik, F. "How Many Operational Design Domains, Objects, and Events?" en. In: (2019), p. 4.

[104] Koren, M., Alsaif, S., Lee, R., and Kochenderfer, M. J. "Adaptive Stress Testing for Autonomous Vehicles". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587. June 2018, pp. 1–7. DOI: 10.1109/IVS.2018.8500400.

[105] Koschi, M., Pek, C., Maierhofer, S., and Althoff, M. “Computationally Efficient Safety Falsification of Adaptive Cruise Control Systems”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Oct. 2019, pp. 2879–2886. DOI: 10.1109/ITSC.2019.8917287.

[106] Koschuch, M., Sebron, W., Szalay, Z., Török, Á., Tschiürtz, H., and Wahl, I. “Safety amp; Security in the Context of Autonomous Driving”. In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. ISSN: 2378-1297. Nov. 2019, pp. 1–7. DOI: 10.1109/ICCVE45908.2019.8965092.

[107] Kotsiantis, S. B., Kanellopoulos, D., and Pintelas, P. E. “Data Preprocessing for Supervised Leaning”. en. In: 1.1 (2006), p. 7.

[108] Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. “The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems”. en. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 2118–2125. ISBN: 978-1-72810-321-1 978-1-72810-323-5. DOI: 10.1109/ITSC.2018.8569552. URL: https://ieeexplore.ieee.org/document/8569552/ (visited on 12/09/2021).

[109] Krajewski, R., Moers, T., Nerger, D., and Eckstein, L. “Data-Driven Maneuver Modeling using Generative Adversarial Networks and Variational Autoencoders for Safety Validation of Highly Automated Vehicles”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 2383–2390. DOI: 10.1109/ITSC.2018.8569971.

[110] Krizhevsky, A. and Hinton, G. “Convolutional Deep Belief Networks on CIFAR-10”. en. 2010.

[111] Kruber, F., Wurst, J., and Botsch, M. “An Unsupervised Random Forest Clustering Technique for Automatic Traffic Scenario Categorization”. en. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 2811–2818. ISBN: 978-1-72810-321-1 978-1-72810-323-5. DOI: 10.1109/ITSC.2018.8569682. URL: https://ieeexplore.ieee.org/document/8569682/ (visited on 11/22/2021).

[112] Kruber, F., Wurst, J., Morales, E. S., Chakraborty, S., and Botsch, M. “Unsupervised and Supervised Learning with the Random Forest Algorithm for Traffic Scenario Clustering and Classification”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 2463–2470. DOI: 10.1109/IVS.2019.8813994.

[113] Langner, J., Bach, J., Ries, L., Otten, S., Holzäpfel, M., and Sax, E. “Estimating the Uniqueness of Test Scenarios derived from Recorded Real-World-Driving-Data using Autoencoders”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587. June 2018, pp. 1860–1866. DOI: 10.1109/IVS.2018.8500464.

[114] Laptev, A., Korostik, R., Svischev, A., Andrusenko, A., Medennikov, I., and Rybin, S. “You Do Not Need More Data: Improving End-To-End Speech Recognition by Text-To-Speech Data Augmentation”. In: *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. Oct. 2020, pp. 439–444. DOI: 10.1109/CISP-BMEI51763.2020.9263564.

[115] LeCun, Y. and Bengio, Y. “Convolutional Networks for Images, Speech, and Time-Series”. en. In: *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1998, pp. 255–258. ISBN: 0-262-51102-9.

[116] LeCun, Y., Bengio, Y., and Hinton, G. "Deep learning". en. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14539. URL: http://www.nature.com/articles/nature14539 (visited on 11/22/2021).

[117] LeCun, Y., Kavukcuoglu, K., and Farabet, C. "Convolutional networks and applications in vision". en. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. Paris, France: IEEE, May 2010, pp. 253–256. ISBN: 978-1-4244-5308-5. DOI: 10.1109/ISCAS.2010.5537907. URL: http://ieeexplore.ieee.org/document/5537907/ (visited on 11/22/2021).

[118] Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., and Owen, M. P. "Adaptive stress testing of airborne collision avoidance systems". In: *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. ISSN: 2155-7209. Sept. 2015, pp. 6C2–1–6C2–13. DOI: 10.1109/DASC.2015.7311450.

[119] Lee, R., Mengshoel, O., Saksena, A., Gardner, R., Genin, D., Brush, J., and Kochenderfer, M. "Differential Adaptive Stress Testing of Airborne Collision Avoidance Systems". In: Jan. 2018. DOI: 10.2514/6.2018-1923.

[120] Lee, Y., Jeon, J., Yu, J., and Jeon, M. "Context-Aware Multi-Task Learning for Traffic Scene Recognition in Autonomous Vehicles". en. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 723–730. ISBN: 978-1-72816-673-5. DOI: 10.1109/IV47402.2020.9304708. URL: https://ieeexplore.ieee.org/document/9304708/ (visited on 11/22/2021).

[121] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., and Thrun, S. "Towards fully autonomous driving: Systems and algorithms". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587. June 2011, pp. 163–168. DOI: 10.1109/IVS.2011.5940562.

[122] Li, W., Pan, C. W., Zhang, R., Ren, J. P., Ma, Y. X., Fang, J., Yan, F. L., Geng, Q. C., Huang, X. Y., Gong, H. J., Xu, W. W., Wang, G. P., Manocha, D., and Yang, R. G. "AADS: Augmented autonomous driving simulation using data-driven algorithms". EN. In: *Science Robotics* (Mar. 2019). Publisher: American Association for the Advancement of Science. DOI: 10.1126/scirobotics.aaw0863. URL: https://www.science.org/doi/abs/10.1126/scirobotics.aaw0863 (visited on 01/24/2022).

[123] Li, Y., Tao, J., and Wotawa, F. "Ontology-based test generation for automated and autonomous driving functions". In: *Inf. Softw. Technol.* (2020). DOI: 10.1016/j.infsof.2019.106200.

[124] Liu, R., Wang, J., and Zhang, B. "High Definition Map for Automated Driving: Overview and Analysis". en. In: *Journal of Navigation* 73.2 (Mar. 2020), pp. 324–341. ISSN: 0373-4633, 1469-7785. DOI: 10.1017/S0373463319000638. URL: https://www.cambridge.org/core/product/identifier/S0373463319000638/type/journal_article (visited on 11/22/2021).

[125] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. "SSD: Single Shot MultiBox Detector". en. In: *Computer Vision – ECCV 2016*. Ed. by Leibe, B., Matas, J., Sebe, N., and Welling, M. Vol. 9905. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46447-3 978-3-319-46448-0. DOI: 10.1007/978-3-319-46448-0_2. URL: http://link.springer.com/10.1007/978-3-319-46448-0_2 (visited on 12/06/2021).

[126] Luengo, J., García, S., and Herrera, F. “On the choice of the best imputation methods for missing values considering three groups of classification methods”. en. In: *Knowledge and Information Systems* 32.1 (July 2012), pp. 77–108. ISSN: 0219-1377, 0219-3116. DOI: 10.1007/s10115-011-0424-2. URL: http://link.springer.com/10.1007/s10115-011-0424-2 (visited on 12/07/2021).

[127] Mahmud, S. M., Ferreira, L., Hoque, M., and Hojati, A. “Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs”. In: *IATSS Research* 41 (2017). DOI: 10.1016/j.iatssr.2017.02.001.

[128] Malin, F., Norros, I., and Innamaa, S. “Accident risk of road and weather conditions on different road types”. en. In: *Accident Analysis & Prevention* 122 (Jan. 2019), pp. 181–188. ISSN: 0001-4575. DOI: 10.1016/j.aap.2018.10.014. URL: https://www.sciencedirect.com/science/article/pii/S0001457518308455 (visited on 01/25/2022).

[129] Marsland, S. *Machine Learning: An Algorithmic Perspective*. en. 2nd ed. Chapman and Hall/CRC, Oct. 2014. ISBN: 978-0-429-10250-9. DOI: 10.1201/b17476. URL: https://www.taylorfrancis.com/books/9781466583337 (visited on 12/07/2021).

[130] Menzel, T., Bagschik, G., Isensee, L., Schomburg, A., and Maurer, M. “From Functional to Logical Scenarios: Detailing a Keyword-Based Scenario Description for Execution in a Simulation Environment”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 2383–2390. DOI: 10.1109/IVS.2019.8814099.

[131] *Mobilität der Zukunft: Aktuelle Forschung am Robotik-Lehrstuhl der TUM*. de-DE. URL: https://innovation-mobility.com/ (visited on 11/23/2021).

[132] *MongoDB: The Application Data Platform*. en-us. URL: https://www.mongodb.com (visited on 02/14/2022).

[133] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S. “Junior: The Stanford entry in the Urban Challenge”. en. In: *Journal of Field Robotics* 25.9 (Sept. 2008), pp. 569–597. ISSN: 15564959, 15564967. DOI: 10.1002/rob.20258. URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.20258 (visited on 12/08/2021).

[134] Mullins, G. E., Stankiewicz, P. G., and Gupta, S. K. “Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 1443–1450. DOI: 10.1109/ICRA.2017.7989173.

[135] Mullins, G. E. “ADAPTIVE SAMPLING METHODS FOR TESTING AUTONOMOUS SYSTEMS”. en. In: (2018). Accepted: 2018-09-12T05:38:36Z. DOI: 10.13016/M2PZ51Q5N. URL: https://drum.lib.umd.edu/handle/1903/21225 (visited on 01/25/2022).

[136] Munich, T. *Internal source*. 2022.

[137] *MySQL*. URL: https://www.mysql.com/de/ (visited on 02/14/2022).

[138] Nabhan, M., Schoenauer, M., Tourbier, Y., and Hage, H. “Optimizing coverage of simulated driving scenarios for the autonomous vehicle”. In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. ISSN: 2378-1297. Nov. 2019, pp. 1–5. DOI: 10.1109/ICCVE45908.2019.8965211.

[139] Nalic, D., Mihalj, T., Bäumler, M., Lehmann, M., and Bernsteiner, S. "SCENARIO BASED TESTING OF AUTOMATED DRIVING SYSTEMS: A LITERATURE SURVEY". en. In: (2020), p. 11.

[140] Naseralavi, S., Nadimi, N., Saffarzadeh, M., and Mamdoohi, A. R. "A general formulation for time-to-collision safety indicator". In: *Proceedings of the ICE - Transport* 166 (2013), pp. 294–304. DOI: 10.1680/tran.11.00031.

[141] Nastjuk, I., Herrenkind, B., Marrone, M., Brendel, A. B., and Kolbe, L. M. "What drives the acceptance of autonomous driving? An investigation of acceptance factors from an end-user's perspective". en. In: *Technological Forecasting and Social Change* 161 (Dec. 2020), p. 120319. ISSN: 00401625. DOI: 10.1016/j.techfore.2020.120319. URL: https://linkinghub.elsevier.com/retrieve/pii/S0040162520311458 (visited on 11/22/2021).

[142] *New Assessment/Test Method for Automated Driving*. en. Apr. 2021.

[143] *NHTSA pre-crash scenario report 2007*. en. 2007.

[144] Panagiotopoulos, I. and Dimitrakopoulos, G. "An empirical investigation on consumers' intentions towards autonomous driving". en. In: *Transportation Research Part C: Emerging Technologies* 95 (Oct. 2018), pp. 773–784. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.08.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0968090X1830086X (visited on 11/22/2021).

[145] Pierson, A., Schwarting, W., Karaman, S., and Rus, D. "Learning Risk Level Set Parameters from Data Sets for Safer Driving". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 273–280. DOI: 10.1109/IVS.2019.8813842.

[146] Piwowar, H. A. and Chapman, W. W. "Public sharing of research datasets: A pilot study of associations". en. In: *Journal of Informetrics* 4.2 (Apr. 2010), pp. 148–156. ISSN: 17511577. DOI: 10.1016/j.joi.2009.11.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S1751157709000881 (visited on 01/24/2022).

[147] Poggenhans, F., Pauls, J.-H., Janosovits, J., Orf, S., Naumann, M., Kuhnt, F., and Mayr, M. "Lanelet2: A high-definition map framework for the future of automated driving". en. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 1672–1679. ISBN: 978-1-72810-321-1 978-1-72810-323-5. DOI: 10.1109/ITSC.2018.8569929. URL: https://ieeexplore.ieee.org/document/8569929/ (visited on 11/22/2021).

[148] Ponn, T. "How to Define System-Specific Corner Cases for the Type Approval of Automated Vehicles". en. PhD thesis. 2020.

[149] *Projekt Shadow Mode*. de-DE. URL: https://www.akka-technologies.com/case-study/projekt-shadow-mode/?lang=de (visited on 12/10/2021).

[150] Pütz, A., Bock, J., Zlocki, A., Küfen, J., and Eckstein, L. "Database approach for the sign-off process of highly automated vehicles". en. In: *undefined* (2017). URL: https://www.semanticscholar.org/paper/Database-approach-for-the-sign-off-process-of-P%C3%BCtz-Bock/1fe96ad63d894970fbf1fff3aed238ae55c9fe1f (visited on 12/10/2021).

[151] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. "ROS: an open-source Robot Operating System". en. In: *Workshops at the {IEEE} International Conference on Robotics and Automation*. Jan. 2009, p. 6.

[152] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. "You Only Look Once: Unified, Real-Time Object Detection". en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 779–788. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91. URL: http://ieeexplore.ieee.org/document/7780460/ (visited on 12/06/2021).

[153] Reinartz, T. and Wirth, R. "The need for a task model for knowledge discovery in databases. in: Workshop notes Statistics, Machine Learning, and Knowledge Discovery in Databases". In: *Workshop notes Statistics, Machine Learning, and Know ledge Discovery in Databases*. 1995, pp. 19–24.

[154] Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., and Diermeyer, F. "Survey on Scenario-Based Safety Assessment of Automated Vehicles". en. In: *IEEE Access* 8 (2020), pp. 87456–87477. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2993730. URL: https://ieeexplore.ieee.org/document/9090897/ (visited on 11/23/2021).

[155] Roesener, C., Fahrenkrog, F., Uhlig, A., and Eckstein, L. "A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2016, pp. 1360–1365. DOI: 10.1109/ITSC.2016.7795734.

[156] Roesener, C., Harth, M., Weber, H., Josten, J., and Eckstein, L. "Modelling Human Driver Performance for Safety Assessment of Road Vehicle Automation". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 735–741. DOI: 10.1109/ITSC.2018.8569669.

[157] *SAE Levels of Driving Automation™ Refined for Clarity and International Audience*. en. URL: https://www.sae.org/site/blog/sae-j3016-update (visited on 01/24/2022).

[158] sagt, H. S. *Sicherheitsabstand einhalten - Verkehrsregeln 2021*. de. URL: https://www.bussgeldkatalog.org/sicherheitsabstand/ (visited on 11/17/2021).

[159] Saraoglu, M., Morozov, A., and Janschek, K. "MOBATSim: MOdel-Based Autonomous Traffic Simulation Framework for Fault-Error-Failure Chain Analysis". en. In: *IFAC-PapersOnLine*. 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019 52.8 (Jan. 2019), pp. 239–244. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.08.077. URL: https://www.sciencedirect.com/science/article/pii/S2405896319304100 (visited on 02/13/2022).

[160] Schreier, M., Willert, V., and Adamy, J. "An Integrated Approach to Maneuver-Based Trajectory Prediction and Criticality Assessment in Arbitrary Road Environments". In: *IEEE Transactions on Intelligent Transportation Systems* 17 (2016), pp. 2751–2766. DOI: 10.1109/TITS.2016.2522507.

[161] Shalev-Shwartz, S., Shammah, S., and Shashua, A. "On a Formal Model of Safe and Scalable Self-driving Cars". en. In: *arXiv:1708.06374 [cs, stat]* (Oct. 2018). arXiv: 1708.06374. URL: http://arxiv.org/abs/1708.06374 (visited on 12/08/2021).

[162] Sheng, S., Pakdamanian, E., Han, K., Kim, B., Tiwari, P., Kim, I., and Feng, L. "A Case Study of Trust on Autonomous Driving*". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Oct. 2019, pp. 4368–4373. DOI: 10.1109/ITSC.2019.8917251.

[163] Shorten, C. and Khoshgoftaar, T. M. "A survey on Image Data Augmentation for Deep Learning". en. In: *Journal of Big Data* 6.1 (Dec. 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0 (visited on 11/22/2021).

[164] Singh, D. and Singh, B. "Investigating the impact of data normalization on classification performance". en. In: *Applied Soft Computing* 97 (2020), p. 105524. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2019.105524. URL: https://www.sciencedirect.com/science/article/pii/S1568494619302947 (visited on 01/25/2022).

[165] So, J. J., Park, I., Wee, J., Park, S., and Yun, I. "Generating Traffic Safety Test Scenarios for Automated Vehicles using a Big Data Technique". In: *KSCE Journal of Civil Engineering* 23.6 (June 2019). ISSN: 1226-7988. URL: https://trid.trb.org/view/1607030 (visited on 01/25/2022).

[166] Sola, J. and Sevilla, J. "Importance of input data normalization for the application of neural networks to complex industrial problems". In: *IEEE Transactions on Nuclear Science* 44.3 (June 1997). Conference Name: IEEE Transactions on Nuclear Science, pp. 1464–1468. ISSN: 1558-1578. DOI: 10.1109/23.589532.

[167] Song, X., Chen, K., Li, X., Sun, J., Hou, B., Cui, Y., Zhang, B., Xiong, G., and Wang, Z. "Pedestrian Trajectory Prediction Based on Deep Convolutional LSTM Network". en. In: *IEEE Transactions on Intelligent Transportation Systems* 22.6 (June 2021), pp. 3285–3302. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2020.2981118. URL: https://ieeexplore.ieee.org/document/9043898/ (visited on 11/22/2021).

[168] Standardization, I. O. for. "ISO 15622 Transport information and control systems – Adaptive Cruise Control systems – Performance requirements and test procedures". In: (2018).

[169] Stark, L., Düring, M., Schoenawa, S., Maschke, J., and Do, C. "Quantifying Vision Zero: Crash avoidance in rural and motorway accident scenarios by combination of ACC, AEB, and LKS projected to German accident occurrence". In: *Traffic Injury Prevention* 20 (June 2019), S126–S132. DOI: 10.1080/15389588.2019.1605167.

[170] Stark, L., Obst, S., Schoenawa, S., and During, M. "Towards Vision Zero: Addressing White Spots by Accident Data based ADAS Design and Evaluation". In: Sept. 2019, pp. 1–6. DOI: 10.1109/ICVES.2019.8906409.

[171] Stellet, J. E., Zofka, M. R., Schumacher, J., Schamm, T., Niewels, F., and Zöllner, J. M. "Testing of Advanced Driver Assistance Towards Automated Driving: A Survey and Taxonomy on Existing Approaches and Open Questions". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. ISSN: 2153-0017. Sept. 2015, pp. 1455–1462. DOI: 10.1109/ITSC.2015.236.

[172] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 2443–2451. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.00252. URL: https://ieeexplore.ieee.org/document/9156973/ (visited on 12/06/2021).

[173] Sun, Y., Xue, B., Zhang, M., and Yen, G. G. "Evolving Deep Convolutional Neural Networks for Image Classification". en. In: *arXiv:1710.10741 [cs]* (Mar. 2019). arXiv: 1710.10741. URL: http://arxiv.org/abs/1710.10741 (visited on 12/07/2021).

[174] Taylor, L. and Nitschke, G. "Improving Deep Learning using Generic Data Augmentation". en. In: *arXiv:1708.06020 [cs, stat]* (Aug. 2017). arXiv: 1708.06020. URL: http://arxiv.org/abs/1708.06020 (visited on 11/22/2021).

[175] *Traffic Scenarios*. en. URL: http://leaderboard.carla.org/scenarios/ (visited on 01/25/2022).

[176] Tuncali, C. E. "Search-based Test Generation for Automated Driving Systems: From Perception to Control Logic". en. In: *undefined* (2019). URL: https://www.semanticscholar.org/paper/Search-based-Test-Generation-for-Automated-Driving-Tuncali/0ea7e0e4f3442c2b5409b8530662d3ba77a24cd2 (visited on 01/25/2022).

[177] Tuncali, C. E. and Fainekos, G. "Rapidly-exploring Random Trees for Testing Automated Vehicles". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Oct. 2019, pp. 661–666. DOI: 10.1109/ITSC.2019.8917375.

[178] Tuncali, C. E., Fainekos, G., Prokhorov, D., Ito, H., and Kapinski, J. "Requirements-Driven Test Generation for Autonomous Vehicles With Machine Learning Components". In: *IEEE Transactions on Intelligent Vehicles* 5.2 (June 2020). Conference Name: IEEE Transactions on Intelligent Vehicles, pp. 265–280. ISSN: 2379-8904. DOI: 10.1109/TIV.2019.2955903.

[179] Tuncali, C. E., Pavlic, T. P., and Fainekos, G. "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2016, pp. 1470–1475. DOI: 10.1109/ITSC.2016.7795751.

[180] Tuncali, C. E., Yaghoubi, S., Pavlic, T. P., and Fainekos, G. "Functional gradient descent optimization for automatic test case generation for vehicle controllers". In: *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. ISSN: 2161-8089. Aug. 2017, pp. 1059–1064. DOI: 10.1109/COASE.2017.8256245.

[181] Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., and Maurer, M. "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. ISSN: 2153-0017. Sept. 2015, pp. 982–988. DOI: 10.1109/ITSC.2015.164.

[182] Van Brummelen, J., O'Brien, M., Gruyer, D., and Najjaran, H. "Autonomous vehicle perception: The technology of today and tomorrow". en. In: *Transportation Research Part C: Emerging Technologies* 89 (Apr. 2018), pp. 384–406. ISSN: 0968-090X. DOI: 10.1016/j.trc.2018.02.012. URL: https://www.sciencedirect.com/science/article/pii/S0968090X18302134 (visited on 01/25/2022).

[183] Varghese, J. Z. "Overview of Autonomous Vehicle Sensors and Systems". en. In: (), p. 14.

[184] Wachenfeld, W., Junietz, P., Wenzel, R., and Winner, H. "The worst-time-to-collision metric for situation identification". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. June 2016, pp. 729–734. DOI: 10.1109/IVS.2016.7535468.

[185] Wachenfeld, W. and Winner, H. "The Release of Autonomous Vehicles". en. In: *Autonomous Driving*. Ed. by Maurer, M., Gerdes, J. C., Lenz, B., and Winner, H. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 425–449. ISBN: 978-3-662-48845-4 978-3-662-48847-8. DOI: 10.1007/978-3-662-48847-8_21. URL: http://link.springer.com/10.1007/978-3-662-48847-8_21 (visited on 12/08/2021).

[186] Wang, C. and Winner, H. "Overcoming Challenges of Validation Automated Driving and Identification of Critical Scenarios". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Oct. 2019, pp. 2639–2644. DOI: 10.1109/ITSC.2019.8917045.

[187] Wang, C., Ma, L., Li, R., Durrani, T. S., and Zhang, H. "Exploring Trajectory Prediction Through Machine Learning Methods". In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 101441–101452. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2929430.

[188] Wang, W. and Zhao, D. "Extracting Traffic Primitives Directly from Naturalistically Logged Data for Self-Driving Applications". In: *IEEE Robotics and Automation Letters* 03 (Sept. 2018), pp. 1223–1229. DOI: 10.1109/LRA.2018.2794604.

[189] Watanabe, H., Malý, T., Wallner, J., Dirndorfer, T., Mai, M., and Prokop, G. "Methodology of Scenario Clustering for Predictive Safety Functions". en. In: (2019), p. 8.

[190] Watanabe, H., Tobisch, L., Rost, J., Wallner, J., and Prokop, G. "Scenario Mining for Development of Predictive Safety Functions". In: *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. ISSN: 2643-9751. Sept. 2019, pp. 1–7. DOI: 10.1109/ICVES.2019.8906293.

[191] Weber, H., Bock, J., Klimke, J., Rösener, C., Hiller, J., Krajewski, R., Zlocki, A., and Eckstein, L. "A framework for definition of logical scenarios for safety assurance of automated driving". In: *Traffic Injury Prevention* 20 (June 2019), S65–S70. DOI: 10.1080/15389588.2019.1630827.

[192] *Welcome to scenariogeneration*. URL: https://pyoscx.github.io/#useful-links (visited on 12/10/2021).

[193] Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., and Xu, H. "Time Series Data Augmentation for Deep Learning: A Survey". en. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence* (Aug. 2021). arXiv: 2002.12478, pp. 4653–4660. DOI: 10.24963/ijcai.2021/631. URL: http://arxiv.org/abs/2002.12478 (visited on 11/22/2021).

[194] Wick, C. "Deep Learning". de. In: *Informatik-Spektrum* 40.1 (Feb. 2017), pp. 103–107. ISSN: 0170-6012, 1432-122X. DOI: 10.1007/s00287-016-1013-2. URL: http://link.springer.com/10.1007/s00287-016-1013-2 (visited on 11/22/2021).

[195] Winner, H., Wachenfeld, W., and Junietz, P. "Validation and Introduction of Automated Driving". In: *Automotive Systems Engineering II*. Ed. by Winner, H., Prokop, G., and Maurer, M. Cham: Springer International Publishing, 2018, pp. 177–196. ISBN: 978-3-319-61607-0. DOI: 10.1007/978-3-319-61607-0_8. URL: https://doi.org/10.1007/978-3-319-61607-0_8.

[196] Wirth, R. and Hipp, J. "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining* (Jan. 2000).

[197] Wotawa, F. and Li, Y. "From Ontologies to Input Models for Combinatorial Testing". In: *ICTSS*. 2018. DOI: 10.1007/978-3-319-99927-2_14.

[198] Wu, F.-Y., Yan, S.-Y., Smith, J. S., and Zhang, B.-L. "Traffic scene recognition based on deep CNN and VLAD spatial pyramids". en. In: *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*. Ningbo, China: IEEE, July 2017, pp. 156–161. ISBN: 978-1-5386-0406-9 978-1-5386-0408-3. DOI: 10.1109/ICMLC.2017.8107758. URL: http://ieeexplore.ieee.org/document/8107758/ (visited on 11/22/2021).

[199] Xia, Q., Duan, J., Gao, F., Chen, T., and Yang, C. "Automatic Generation Method of Test Scenario for ADAS Based on Complexity". In: Sept. 2017. DOI: 10.4271/2017-01-1992.

[200] Xia, Q., Duan, J., Gao, F., Hu, Q., and He, Y. “Test Scenario Design for Intelligent Driving System Ensuring Coverage and Effectiveness”. en. In: *International Journal of Automotive Technology* 19.4 (Aug. 2018), pp. 751–758. ISSN: 1229-9138, 1976-3832. DOI: 10.1007/s12239-018-0072-6. URL: http://link.springer.com/10.1007/s12239-018-0072-6 (visited on 01/25/2022).

[201] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. “A Survey of Autonomous Driving: Common Practices and Emerging Technologies”. In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 58443–58469. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2983149.

[202] Zhao, D., Lam, H., Peng, H., Bao, S., LeBlanc, D. J., Nobukawa, K., and Pan, C. S. “Accelerated Evaluation of Automated Vehicles Safety in Lane-Change Scenarios Based on Importance Sampling Techniques”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.3 (Mar. 2017). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 595–607. ISSN: 1558-0016. DOI: 10.1109/TITS.2016.2582208.

[203] Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X. “Object Detection With Deep Learning: A Review”. en. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (Nov. 2019), pp. 3212–3232. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2018.2876865. URL: https://ieeexplore.ieee.org/document/8627998/ (visited on 12/04/2021).

[204] Zhu, J., Wang, W., and Zhao, D. “A Tempt to Unify Heterogeneous Driving Databases using Traffic Primitives”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 2052–2057. DOI: 10.1109/ITSC.2018.8569940.

[205] Zimmer, W., Rangesh, A., and Trivedi, M. “3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1816–1821.

[206] Zong, W., Zhang, C., Wang, Z., Zhu, J., and Chen, Q. “Architecture Design and Implementation of an Autonomous Vehicle”. In: *IEEE Access* 6 (2018). Conference Name: IEEE Access, pp. 21956–21970. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2828260.

Appendix 1

Scenario Catalogs

Scenario Catalog Highway

Scenario Catalog Highway	
Lane change	Lane change: 2 vehicles going straight and 1 vehicle encroaching in same lane
	Lane change: 2 vehicles going straight and 1 vehicle encroaching into another lane
	Lane change: 1 vehicle going straight and another changing lane
	Lane change: 1 vehicle going straight and another passing
	Lane change: vehicle changing lane in absence of other vehicles
Strong acceleration / deceleration	Strong acceleration
	Strong deceleration
Road geometry	Widening road
	Merging road
	Exit highway
	Merge into highway
Lead vehicle maneuver	Lead vehicle strong acceleration
	Lead vehicle strong deceleration
	Lead vehicle moving at much lower constant speed
	Lead vehicle stopped
	Lead vehicle changing lanes
	Lead vehicle cutting-in
	Lead vehicle cutting-out
Violating traffic laws	Speeding
	Overtaking right side
	Irrational driving (swerving etc.)
	Insufficient distance to leading vehicle (tailgate)
	Ghost driver
	Emergency vehicle on duty passing
Avoiding maneuvers	Avoiding road user blocking road
	Avoiding obstacles blocking road
	Avoiding construction zone
	Dynamic object avoidance

Continued on next page

Table 1 – continued from previous page

Scenario Catalog Highway	
	Animal: vehicle going straight and animal in road
	Animal: vehicle negotiating a curve and animal in road
	Evasive action with prior vehicle maneuver
	Evasive action without prior vehicle maneuver
Accident scenarios	Accident
	Animal crash with prior vehicle maneuver
	Animal crash without prior vehicle maneuver
	Vehicle failure
	Control loss with prior vehicle action
	Control loss Without prior vehicle action
	Road edge departure with prior vehicle maneuver
	Road edge departure without prior vehicle maneuver
	Vehicle(s) drifting – same direction
	Object crash with prior vehicle maneuver
	Object crash without prior vehicle maneuver
Cyclist	Cyclist crossing road
	Cyclist: vehicle going straight on crossing paths
	Cyclist: vehicle going straight on parallel paths
	Cyclist: vehicle starting in traffic lane on crossing paths
	Cyclist crash with prior vehicle maneuver
	Cyclist crash without prior vehicle maneuver
Pedestrian	Pedestrian crossing road
	Driving through pedestrian crowd
	Pedestrian: vehicle backing
	Pedestrian: vehicle going straight and pedestrian crossing road
	Pedestrian: vehicle going straight and pedestrian darting onto road
	Pedestrian: vehicle going straight and pedestrian playing / working on Road
	Pedestrian: vehicle going straight and pedestrian walking along road
	Pedestrian crash with prior vehicle maneuver
	Pedestrian crash without prior vehicle maneuver

Table 1: Scenario catalog highway (source: own illustration)

Scenario Catalog Urban / Rural

Scenario Catalog Urban / Rural	
Lane change	Lane change: 2 vehicles going straight and 1 vehicle encroaching in same lane
	Lane change: 2 vehicles going straight and 1 vehicle encroaching into another lane
	Lane change: 1 vehicle going straight and another changing lane
	Lane change: 1 vehicle going straight and another passing
	Lane change: 1 vehicle going straight and another entering or leaving parking position
	Lane change: 1 vehicle going straight and another turning
	Lane change: 1 vehicle passing and another turning
	Lane change: Vehicle changing lane in absence of other vehicles
Strong acceleration / deceleration	Strong acceleration
	Strong deceleration
Protected turn / road crossing	Protected left turn
	Protected right turn
	Protected road crossing
Unprotected turn / road crossing	Unprotected left turn
	Unprotected right turn
	Unprotected road crossing
Protected turn while pedestrian / cyclist crossing road	Protected left turn with pedestrian / cyclist crossing road
	Protected right turn with pedestrian / cyclist crossing road
Unprotected turn while pedestrian / cyclist crossing road	Unprotected left turn with pedestrian / cyclist crossing road
	Unprotected right turn with pedestrian / cyclist crossing road
Pedestrian / cyclist / animal / etc. crossing road	Pedestrian / cyclist / animal / etc. crossing road
Road geometry	Widening road
	Merging road
	Exit highway
	Merge into highway
Lead vehicle maneuver	Lead vehicle strong acceleration
	Lead vehicle strong deceleration
	Lead vehicle moving at much lower constant speed
	Lead vehicle stopped
	Lead vehicle changing lanes
	Lead vehicle cutting-in
	Lead vehicle cutting-out
Violating traffic laws	Speeding
	Running stop sign

Continued on next page

Table 2 – continued from previous page

Scenario Catalog Urban / Rural	
	Hit red light
	Violating traffic signs
	Irrational driving (swerving etc.)
	Insufficient distance to leading vehicle (tailgate)
	Ghost driver
	Emergency vehicle on duty passing
Avoiding maneuvers	Avoiding road user blocking road
	Avoiding obstacles blocking road
	Avoiding construction zone
	Dynamic object avoidance
	Animal: vehicle going straight and animal in road
	Animal: vehicle negotiating a curve and animal in road
	Evasive action with prior vehicle maneuver
	Evasive action without prior vehicle maneuver
	Avoiding pedestrian: vehicle backing
	Avoiding pedestrian: vehicle going straight and pedestrian crossing road
	Avoiding pedestrian: vehicle going straight and pedestrian darting onto road
	Avoiding pedestrian: vehicle going straight and pedestrian playing / working on Road
	Avoiding pedestrian: vehicle going straight and pedestrian walking along road
	Driving through pedestrian crowd
	Avoiding cyclist: vehicle going straight on crossing paths
	Avoiding cyclist: vehicle going straight on parallel paths
	Avoiding cyclist: vehicle starting in traffic lane on crossing paths
	Avoiding cyclist: vehicle turning left on crossing paths
	Avoiding cyclist: vehicle turning left on parallel paths
	Avoiding cyclist: vehicle turning right on crossing paths
	Avoiding cyclist: vehicle turning right on parallel paths
Accident scenarios	Accident
	Animal crash with prior vehicle maneuver
	Animal crash without prior vehicle maneuver
	Pedestrian crash with prior vehicle maneuver
	Pedestrian crash without prior vehicle maneuver
	Cyclist crash with prior vehicle maneuver
	Cyclist crash without prior vehicle maneuver
	Object crash with prior vehicle maneuver
	Object crash without prior vehicle maneuver
	Backing up into another vehicle
	Vehicle failure
	Control loss with prior vehicle action
	Control loss without prior vehicle action
	Road edge departure with prior vehicle maneuver
	Road edge departure without prior vehicle maneuver

Continued on next page

Table 2 – continued from previous page

Scenario Catalog Urban / Rural	
	Vehicle(s) drifting – same direction
	Backing: at intersections
	Crossing paths: left turn across path from lateral direction
	Crossing paths: left turn across path from opposite direction
	Crossing paths: left turn into path
	Crossing paths: right turn across path from lateral direction
	Crossing paths: right turn into path
	Crossing paths: straight crossing paths
	Crossing paths: other/unknown

Table 2: Scenario catalog urban / rural (source: own illustration)