POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Standardized annotated dataset generation from raw data collected by lidar sensor

Supervisors

Prof. Angelo BONFITTO

Dott. Alessandro TESSUTI

Dott. Simone MARAGLIULO

Candidate

Michele ANGELETTI

a.y. 2022/2023 December 2023

Abstract

Advanced Driving Assistance Systems and Autonomous Vehicles could significantly reduce road accidents that can be attributed to the driver in more than the 90% of the cases. One of the biggest challenges in driving automation is to ensure the system is safe before placing it on the road since they can perform badly in certain situations such as system fault, inclement weather conditions and complex driving environment, but also cyber security comes to the attention when vehicles are automated. The solution to achieve safety in the context of ADAS/AVs is testing the systems to validate them, in particular driving tests are needed. It has been estimated that vehicles must be driven for hundreds of million kilometers in order to achieve the expected result for the certification process and it would take tens of years. For this reason, virtual testing is the preferred solution by the industry. Virtual testing reduces costs, improves repeatability and raises the scale of the number of tests. The data-driven approach gives a medium fidelity representation of the real world but high quantity of processed data is needed. An important process is the data annotation that is mainly done by hand for its reliability but is slower and more expensive than an automated annotation. As a consequence, collecting and processing a large amount of data needed for the validation is a costly process, both in terms of time and money, therefore the trend is to concentrate on the critical scenarios, but also to standardize databases in order to increase the exchange of data across company and organization and in addition to automate processes. Standardization of databases means data are available for more users, accelerating the automotive development, testing and validation. The aim of this thesis study is to report the process of developing a tool capable of automatically generating standardized labels for an automotive raw dataset.

The research and developing processes have been done as part of an internship in Concept Quality Reply. The developed tool is able to take as an input a structured dataset, collected from the real world by a standardized sensor setup. The dataset's structure is compliant to the one developed for NuScenes by Motional which is a multimodal large-scale dataset for autonomous driving that is now leading the industry. In particular the data under analysis are point clouds generated from a lidar sensor. The developed system recognizes pedestrians and vehicles with a certain accuracy and generating bounding boxes and labels. The 3D object detection process is made through a convolutional neural network model that gives as an output the labels with their confidence level. The pipeline ends with a conversion process of the resulting labels into a standardized format. The standard identified is OpenLabel, defined by ASAM (Association for Standardization of Automation and Measuring Systems) that is the current de-facto standard in the automotive industry. To permit the visualization of the labels, a graphical user interface has been developed together with a web application and an APIs collection, that make possible the communication of the user with the tool hosted by a server and shapes the developed system as a "Software as a Service" (SaaS).

Table of Contents

List of Tables

List of Figures

Acronyms

NHTSA National Highway Traffic Safety Administration

DAS Driving Automation System

ADS Automated Driving Systems

ADAS Advanced Driver Assistance Systems

AVs Autonomous Vehicles

DDT Dynamic Driving Task

ODD Operational Design Domain

LKA Lane Keeping Assistance

AEB Automatic Emergency Braking

ESC Electronic Stability Control

ACC Adaptive Cruise Control

OEDR Object and Event Detection and Response

HARA Hazard Analysis and Risk Assessment

E/E Systems Electrical or electronic systems in an on-board network

ASIL Automotive Safety Integrity Level

ASD Agile Software Development

FDD Feature Driven Development

TDD Test Driven Development

DSDM Dynamic System Development Method

RAD Rapid Application Development

XP Extreme programming

DDDM Data-Driven Decision Making

ML Machine Learning

SVM Support Vector Machine

ASAM e.V. Association for Standardization of Automation and Measuring Systems

OEMs Original equipment manufacturers

NCAP New Car Assessment Program

MiL Model in the Loop

SiL Software in the Loop

HiL Hardware in the Loop

ViL	Vehicle in the Loop
DiL	Driver in the Loop
DuT	Device under Testing
CAV	Connected Automated Vehicles
SaaS	Software as a Service
CNN	Convolutional Neural Network
SSD	Single Shot Detector
IoU	Intersection over Union

Introduction

Achieving a better driving experience is one of the main challenges the automotive industry is facing in the last years. Taking into account the safety side, from the technical report produced by the National Highway Traffic Safety Administration (NHTSA), published in February 2015, it is clear that about 94% of the crashes are driver responsibility [1]. Also European researches, such as the study published in Poland in February 2019 states that almost the same percentage mentioned above of traffic accidents are due to human causes [2]. From the point of view of the sustainability, ADAS/ADS/AVs could realize a reduction on vehicles emissions by helping to perform a more efficient way to drive e reduce fuel or electrical energy consumption, but also improving avoidance of traffic congestion [3]. Autonomous driving functions are very complex systems since they introduce a non-deterministic component that combined with the critical operational design domain results in a very challenging validation effort that aims to ensure functional safety and quality, in order to obtain a failures free system, that does not cause damage to users and environment. The industry is increasingly adopting the Data-Driven approach in order to face the challenges represented by ADAS/AVs functions developing and validation. This innovative and relatively recent method is integrated from the decision making process, to the validation phase passing through the design stage and it consists in learning directly from data and finding paths that would be very difficult or even impossible to identify by modelling systems that are characterized by non-linearities and uncertainties. Data-Driven approach is widely accepted and it is currently in the spotlight of researches and applications. On the other hand, Data-Driven method also means the need of a large volume of data, requirement that in the big-data era is always less difficult to be satisfied thanks to the increasing quantity of sensors and devices that are able to collect information. The objective that this document aims to face is improving data exchange in the automotive industry, by automating and standardizing data-set annotation. The purpose is to obtain labels from the automated annotation process that can be integrated by as many as possible systems and be widely accepted by the industry. Data standardization surely raises the quality of the validation process and makes it faster and more horizontal, it also encourages the definition of an industry benchmark for the validation process.

Chapter 1

AD complexity and industrial needs

1.1 General

1.1.1 Feature variety and expertise domain

The Joint Working Group between ISO and SAE, taken in 2018, released a document called "*Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*", last revised on April 2021, that defines a standard for motor vehicle driving automation description. The aim of this common effort is to unite global experts in driving automation technology and safety for gathering together all their knowledge and expertise in that field [4]. With the terms "*vehicle driving automation systems*" (from now on referred to as "Driving Automation System") the document refers to the systems that perform Dynamic Driving Tasks (DDT) and upon that it has built an important classification of this kind of systems, based on which and how many DDT are automated. The cited *taxonomy* have been fully integrated in the ADS industry's language and are listed below:

1. **Level 0**: No Driving Automation;
2. **Level 1**: Driver Assistance
3. **Level 2**: Partial Driving Automation
4. **Level 3**: Conditional Driving Automation
5. **Level 4**: High Driving Automation
6. **Level 5**: Full Driving Automation

In particular, the assignment of a system to a class is performed accordingly to the expected role the human user, the driving automation system and other components or systems related to the vehicle are expected to play. For this reason active safety systems (e.g. Automatic Emergency Braking and Electronic Stability Control) and some driver assistance systems (e.g. Lane Keeping Assistance) are not included in these definitions because they represent a momentary intervention that does not change the human intervention o substitute it in performing the DDT, while for example Adaptive Cruise Control can be considered a Level 1 driving automation because it actually replaces a DDT. The document also groups these technologies under thirty-two definitions:

1. **Active Safety Systems** are systems that belongs to the vehicle and have the aim to detect potential dangers to the vehicle, occupants and road users and automatically intervene to avoid or mitigate impacts by acting on vehicle subsystems such as suspension, brake, throttle and others or alerting the driver but also performing vehicle system adjustments.
2. **Automated Driving Systems** refers to Level 3, 4 and 5 driving automation system that is composed by the hardware and the software.

They are able to perform continuously the whole DDT also in the case that they belong to a specific Operational Design Domain.

3. **Dispatch in Driverless Operation** means the management through software of the tasks of ADS-equipped vehicles in driverless operations, included the pick-up or drop-off of passengers or goods in a predefined interval of time. The dispatch can involve multiple actors that perform different operations. Therefore, to dispatch an ADS-equipped vehicle, consists in place it into service in driverless operations by engaging the Automated Driving Systems.

4. **Driverless Operation Dispatching Entity** takes care of the dispatch of a vehicle that is equipped with ADS in driverless operations.

5. **Driving Automation** is the action performed by a system of hardware and software to realize a part or all the Dynamic Driving Task in a continuous manner.

6. **Driving Automation System or Technology** is used to refer at any system able to perform from Level 1 to Level 5 driving automation and so a setup of hardware and software that can execute a part or the whole DDT continuously.

7. **Driving Automation System Feature** is a specific function given to the Driving Automation System of any level, in the design phase, that act at a certain level of driving automation and within a particular Operational Design Domain. A DAS can have multiple features that could be *driver support features* or *ADS features*. These feature can be grouped as follows:

    - **Maneuver-Based Feature** is a Driving Automation System feature capable of performing lateral and/or longitudinal vehicle motion control actions to achieve a specific objective that is part of the DDT fulfilled by the driver namely Level 1 and 2 driver support features (e.g. parking maneuver) or the entire DDT with the related Object and Event Detection and Response namely Level 3 and 4 Automated Driving Systems features. Some examples could be the Level 1 (with driver support) or Level 2 (only driver supervision) parking assistance feature but also Level 3 highway overtaking assistance feature.

- **Sub-Trip Feature** indicate the features that help the driver on performing a part of the trip but require the human intervention outside the feature's ODD, An example of Sub-Trip Feature is Level 1 Adaptive Cruise Control feature that helps the driver in preserving the distance between him and the front vehicle in its lane at sustained speed.
- **Full-Trip Feature**, instead, can maneuver the vehicle from the beginning to the end of the trip for example a Level 5 dual-mode vehicle able to pick-up passengers and bring them to the destination.

8. **Driver Support Driving Automation System Feature** is used to describe Level 1 and Level 2 Driving Automation System features.
9. **Driverless operation of an ADS-equipped vehicle** is defined as an operation performed by a ADS-equipped vehicle on the road with or without passengers, in the first case they are not driving.
10. **Dynamic Driving Task** consists of operations, performed in real-time, needed to operate a vehicle in on-road traffic. They do not include the strategic functions for example what concerns the trip planning. As specified in the ISO/SAE document, there have been defined some sub-task, cited below:

- Lateral vehicle motion control via steering (operational).
- Longitudinal vehicle motion control via acceleration and deceleration (operational).
- Monitoring the driving environment via object and event detection, recognition, classification, and response preparation (operational and tactical).
- Object and event response execution (operational and tactical).
- Maneuver planning (tactical).
- Enhancing conspicuity via lighting, sounding the horn, signaling, gesturing, etc. (tactical).

11. **Dynamic Driving Task Fallback** is the response by the user or the ADS to achieve a *minimal risk condition* or in the case of the user also directly to perform the DDT after exiting from the ODD or the failure of a system important to perform the DDT.

12. **Failure Mitigation Strategy** is a vehicle function that can automatically stop the vehicle under the ADS action thanks to a controlled operation, in two cases:

    - When a Level 3 ADS requires a user intervention but there is a prolonged failure to perform the fallback.
    - When a system failure or an extreme external event happens and it make the ADS incapable of controlling vehicle motion. As a result, the ADS cannot execute the fallback to achieve a minimal risk state.

13. **Fleet Operations Functions** are involved in the management of a fleet of vehicles equipped with ADS in driverless operations.

14. **Lateral Vehicle Motion Control** is a DDT subtask that indicate all the operations required for the continuous control of the y-axis component of the vehicle motion in real-time.

15. **Longitudinal Vehicle Motion Control** is the counterpart of the previous subtask for the x-axis component of the motion.

16. **Minimal Risk Condition** is a condition for which the vehicle is stopped and stable. Condition realized by user or ADS to reduce the risk of a crash when it is not possible (or not recommended) to continue the trip.

17. **DDT Performance-Relevant System Failure** is a malfunction of the vehicle system or the ADS that causes the ADS to be unable to perform part or the entire DDT.

18. **Monitor** is a term used to indicate the action of sensing and processing the data used to operate the vehicle.

19. **Object and Event Detection and Response**, for their importance in this paper, their description is reported literally as written in the standard definition "The subtasks of the DDT that include monitoring the driving environment (detecting, recognizing, and classifying objects and events and preparing to respond as needed) and executing an appropriate response to such objects and events (i.e., as needed to complete the DDT and/or DDT fallback)".

20. **Operate a motor vehicle** describes the actions performed by the user (supported or not by ADS) to accomplish the whole DDT.

21. **Operational Design Domain** is the entire set of conditions defined in the design phase of an ADS or feature necessary for them to operate. These conditions can include for example environmental restrictions or traffic characteristics.

22. **Receptivity of the user**

23. **Remote Assistance** is referred to a not present human giving information or advice to a vehicle when the equipped ADS cannot handle the specific situation in driverless operation.

24. **Remote Driving** of a non present driver that performs the DDT or its fallback in real-time, such as braking or steering.

25. **Request to Intervene** made by a Level 3 ADS to the user of resuming manual operation of the vehicle or rich a minimal risk condition, in case of the vehicle cannot be operated.

26. **Routine/Normal ADS Operation** is the action performed by the ADS within its ODD while no DDT performance-relevant system failure occurs.

27. **Supervise Driving Automation System Performance** is performed by the driver during the action of Level 1 or 2 features while the user is operating the vehicle as consequence of the system's wrong behavior end eventually complete the DDT.

28. **Sustained Operation of a Vehicle** means to perform part or the entire DDT both during external events and in the absence of external events, also giving feedback to external events and ensuring the continuous execution of the DDT.

29. **Trip** refers to the whole path of the vehicle that travel from the origin to the destination.

30. **Usage Specification** defines a specific level of driving automation within its ODD.

31. **Human User**.

32. **Motor Vehicle**.

1.1.2 Safety: need for accuracy

The context of Advanced Driver Assistance Systems and Autonomous Vehicles is very complex, in particular a lot of attentions are reasonably given to achieving safety conditions. With the emergence of driver assist and automated driving systems, functional safety has grown its importance in the development of safety-critical automotive systems. To achieve functional safety, the automotive industry has referred to the "*ISO 26262: Functional Safety – Road Vehicles*" since its first release in 2011. In 2018, it has been published the second edition. In the latest release, the standard has been extended to all road vehicles such as motorcycles, trucks, buses, trailers and semi-trailers defining requirements for their safety life-cycle. The terms "safety-critical automotive systems" refer to systems that can cause problems related to the safety while they are not operating as expected or as designed, for this reason it is needed to follow specific rules in a meticulous manner for the whole development phase. In particular the standard focuses in the sector of electrical/electronic systems within road vehicles. The safety goals are defined during the concept phase with the aim of facing the hazards caused by malfunctioning behavior of E/E systems and identified through the hazard and risk assessment process. Those goals are used to defines requirements for the design phase that consists in development of system, hardware and software. The definition of *safety* - in this document - is the absence of unreasonable risk, below some definitions as reported in [5]:

- **Risk**: "*Combination of the probability of occurrence of harm and the severity of that harm.*"
- **Unreasonable Risk**: "*Risk judged to be unacceptable in a certain context according to valid societal, moral concepts.*"
- **Severity**: "*Estimate of the extent of harm to one or more individuals that can occur in a potentially hazardous situation.*"

- **Harm**: "*Physical injury or damage to the health of persons.*"
- **Exposure**: "*state of being in an operational situation that can be if coincident with the failure mode under analysis.*"
- **Controllability**: "*Ability to avoid a specified harm or damage through the timely reactions of the persons involved, possibly with support external measures.*"
- **Safety Goal**: are the highest level safety requirements produced by the processes of hazard analysis and risk assessment.
- **Safe state**: is the way of functioning without an unreasonable level of risk of an item after a failure.
- **Safety Mechanism**: is the technical approach to identify and address faults and failures, either by preventing them or managing them, in order to uphold the intended functionality or ensure a safe state.
- **Work Product**: is the documentation that is produced in order to achieve a requirement defined by ISO 26262.
- **Confirmation Review**: is a verification that a work product has sufficiently reached the functional safety.
- **Safety Case**: a well-structured document, proved by the outcome of the work product, that states a system can be safely operated within a specific context.

Once the potential hazards are identified by analyzing the system, they are classified through their level of severity, probability of exposure and controllability. The classification lead to the assignment of an *Automotive Safety Integrity Level*, both to the potential hazards and to the safety goals while the safety requirements that derive from the goals inherit the same

ASIL. During the *situation analysis and hazard identification* phase the system is observed in the operating modes that are known for triggering the potential hazards if a malfunctioning behavior happen, the effects of this procedure are evaluated and documented. The next step is the *Hazard Classification* that, as said, is based on the estimation of severity, probability of exposure and controllability. The standard define four levels of severity (S), from S0 when no injuries are caused to S3 when the injuries are fatal, while for exposure (E) it goes from E0 when it is an very unusual situation to E4 for a scenario with a high probability to happen and controllability (C) is categorized from C0 for a simply controllable event and C3 for totally uncontrollable. Below Table 1.1 representing all the ASIL Determination.

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Table 1.1: ASIL Determination (Source [5]).

ASILs goes from the lowest safety integrity level "A" to the highest "D". In addition one more class "QM" i.e. Quality Management, is defined for the cases in which there is no requirement from the ISO 26262. For what concerns the **safety goal formulation** the guide lines state that to each hazardous event must correspond at least a safety goal. Safety goals inherit the ASIL from the hazardous event. Two or more hazardous events can be related to the same safety goal and the assigned ASIL is the highest one. Safety goals are not technical solutions but functional objectives to be achieved. From safety goals are derived the "*Functional safety requirements*" that inherit the ASIL. The next phase is the product development at the system level that begin with the definition of the *Technical Safety Concept*, that is where *Technical Safety Requirements* come from, before being allocated to the system elements i.e. hardware and software. These kind of concepts and requirements are more detailed version, from the point of view of the technical aspects such as the specific operations to detect faults and control or mitigate failures. When the technical safety requirements are available, *product development at the hardware and software level* are performed separately while identifying the required coordination between hardware and software. At this point systems and items can be integrated and verified. The last step is the **safety validation**, that is the process of ensuring the safety concepts previously defined are compliant with the functional safety of the item and proving it reached the safety goals identified before. The validation process involves test procedures for each of the safety goals, characterized by a pass/fail criterion.

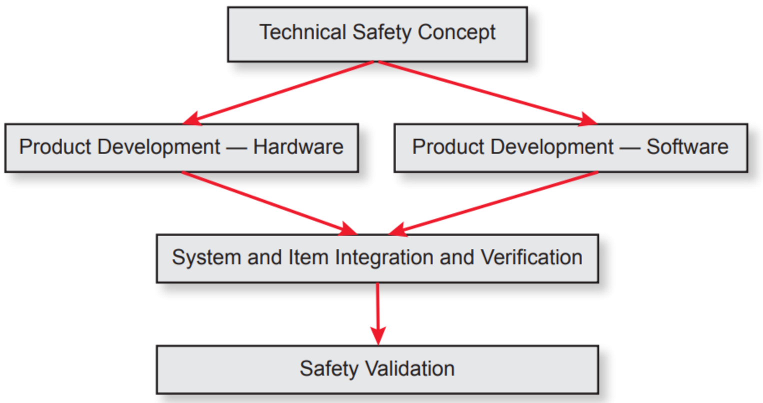
Technical Safety Concept
Product Development — Hardware
Product Development — Software
System and Item Integration and Verification
Safety Validation

Figure 1.1: Product development at the System Level (Source [5]).

1.2 Software development

1.2.1 Data driven approach

Toward agile methodology

Over the past twenty years, the software development industry, gradually increased the adoption of the agile approach until it became the most trusted one. The term *agile* indicate conceptual guidelines to reduce the overhead in the software development process and to face changes while saving the project and keeping small the amount of work to be done due to the variations. The aim is to reach the deploy phase through iterative and incremental interactions for the whole project's life-cycle. Customer satisfaction is the priority but also a flexible process and an improved collaboration among all the actors. According to [6], the entire idea is based on four values:

- **Individuals and interactions over processes and tool**: communication, interactions and the humans as developer are the real key factor of the project and not the instruments nor the processes.
- **Working software over comprehensive documentation**: since producing an up-to-date and exhaustive documentation during the development process requires a huge amount of time and resources due to the big number of changes that are performed, the documentation is realized once the software is in its final version and it meets all the requirements, saving the synchronization time; documentation of software that is not yet meeting the demands, not only it is time consuming but also ambiguous and could lead to misunderstanding.
- **Customer collaboration over contract negotiation**: the customer is involved in the whole development process in order to catch the feedback at every stage and meet the real customer requests and not the

ones defined in contracts, that are still needed but often not sufficient to be sure of how the final result should look like.

- **Responding to change over following a plan**: following a predefined path is not a priority for agile because during a developing process both the developer and the customer will gain a better understanding of the project as well as more knowledge and changes will be made in relation to that since the objective is always the customer satisfaction.

Also twelve principle are given in order to guide the agile developing process implementation:

1. The trust between customer and developers is built through "*early and continuous delivery of valuable software*" since customers will be satisfied by the product at each stage because they are shaping it with their feedback that are coming from an improved understanding and the developers team will perfectly know, step by step, which are the requirements reducing the lack of understanding.

2. Changes are not feared but embraced as they are the reflection of turbulence in business and in technology and they can help not to being stuck at old necessities, but at the same time it is more effective to facilitate the changes than attempting to prevent it.

3. the time between working software delivery should go from a couple of week to a couple of month (the less the better), delivery doesn't mean release. The first one is for the internal use while the second one refers to the production phase.

4. Reducing distances between business people and developers helps clarify doubts and fills the gap between the set of functionalities that people want to buy and the different way the software development works.

5. Team's feeling reflects how the job will be done, so the project will benefit of giving trust and make developers feel comfortable and motivated.
6. Direct human communication is more efficient and effective than written plans.
7. Divide the product into small pieces to be delivered is a better progress measurement method.
8. The developing process should maintain a certain rhythm, delivering small high quality pieces and not aim to the ultimate error free delivery.
9. High quality code is a priority, even in presence of necessary refactoring.
10. The software should be as simple as possible to reduce the amount of work to be done while fulfilling the customer requirements.
11. Agile teams use their knowledge to self-organize themselves and to optimize their structure.
12. On a regular base the team should work on its behavior to improve effectiveness.

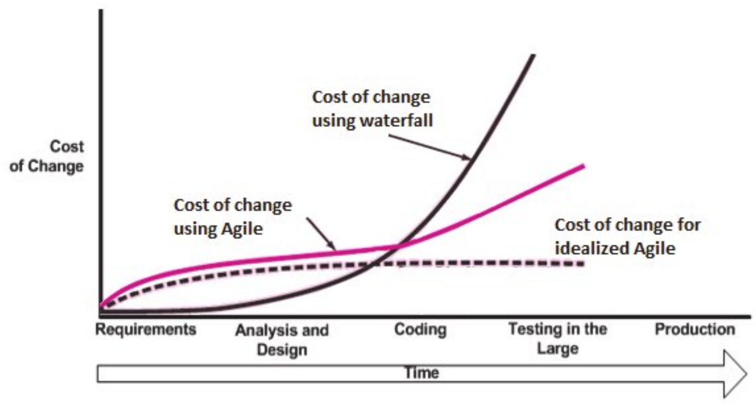

Figure 1.2: Traditional agile methodology cost-change curve (from [7]).

Before the advent of agile methodologies, one of the most popular approaches since 1970 was the waterfall model that was inherited from the hardware manufacture and construction strategies. As the name suggests, the philosophy of the model is to execute task in a sequential way, the succession of phases has only one direction [8]. The project has the following phases: requirement specification, conception, analysis, design, coding, testing and debugging, installation and finally, maintenance. The flow starts with the first phase and does not enter the next one until the phase is totally completed. This kind of approach consumes a lot of time because every problem that could affect the project in the future is solved in the conceptual phase, therefore every requirement must be accomplished, every difficulty is cleared and no changes can be made in the next phases. In addition, an

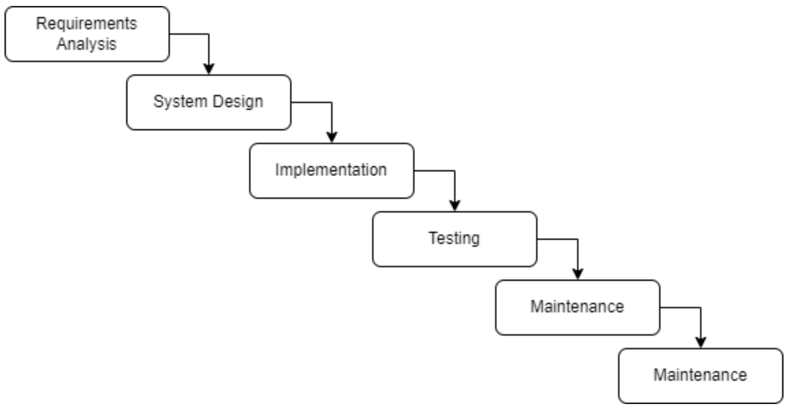

Figure 1.3: Waterfall model.

extensive documentation is needed in order to link each stage because teams do not communicate since they are often separated also in terms of time as well as the phases, it results in a very rigid structure, not able to embrace changes nor to learn from evolving project. In comparison with the agile approach, the Waterfall one in less efficient because the quality of the final software has more probability to be low since new information give a smaller contribute. Waterfall would be suitable in that case where the requirements are very clear and bug are very predictable because it could reduce costs but that level of predictability is very rare and it also need a longer deadline. From the point of view of the validation, also testing is a procedure that can be performed in a agile compliant manner [8]. Agile testing consists in testing iteratively the new software component in parallel to the regression tests. The whole team become composed by testers because of how dynamic this task is. Testers must be ready to change test suits in order to validate all new deliveries, they have to adapt themselves to the rapid development cycles and assume the point of view of the customer. Involving testers in the project some phases before allow them to give earlier the necessary feedback, useful for the development that means less time and more quality for the software. For the Waterfall method a similar approach can be found in one of its extensions that is the V-Model. This model implements a test design phase parallel to each stage, in such a way testing is part of the project for the entire software development life-cycle.

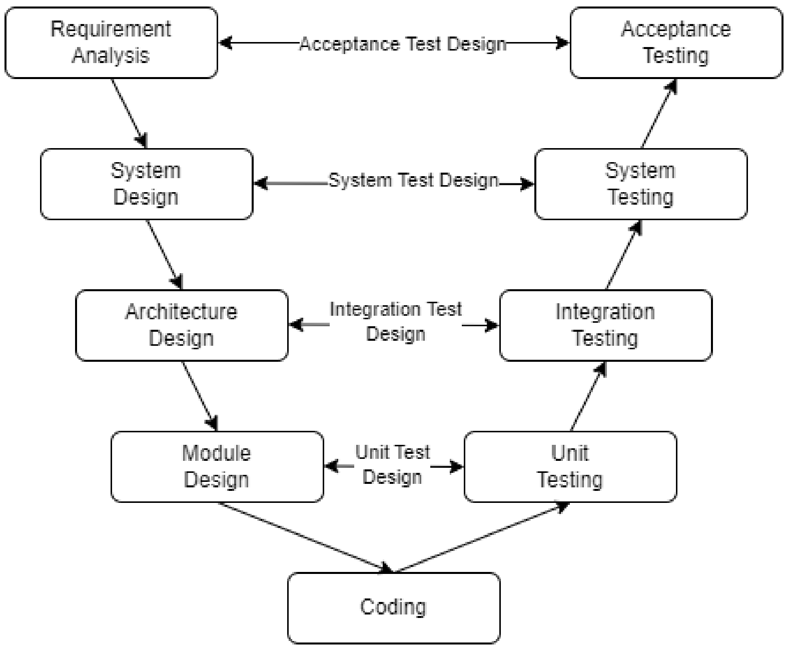

Figure 1.4: V-Model.

In the last years, many agile methods have been ideated and below are listed and shortly described some of them accordingly to [9]:

- **Feature Driven Development (FDD) method**, thought for large project, consists in five short incremental iterations: model development based on the requirements, features' list creation with the customer help, a high level plan based on the features' list is prepared to schedule each feature with its priority, features are finally designed and built, the life cycle begin again for another feature following the plan. It provide a high number of deliveries and accepts changes but there are no guidelines about requirement handling and it requires expert team, not to mention the lack of attention for the project problems.

- **Test Driven Development (TDD) method** is based on build a small automated testing programs from the requirements and then write the code that can pass that test. Some test will be failed and so the next step is to solve the problems that cause the test to fail. In this step performances are not taken into account but lately the software will be refined and stress or performance tests will be done. When the code will pass all the test it will satisfy the requirements and changes can be made but it must still pass the tests. Errors are found early in the process and therefore are easier to be solved, software's quality is improved but developers must have also tester's skills. TDD can be time consuming when repeated failures happened and poor documentation for maintenance is produced by this method.

- **Dynamic System Development Method (DSDM)**, uses Rapid Application Development (RAD) and it is an incremental method where the amount of functionality is decided based on time and resources available instead of the opposite. It passes through the feasibility and business study phases before analyzing the model by producing prototypes that permit to define the subjects, the timing and the methodologies of development. Once the software is designed and coded it is published and tested directly by the users that will give feedback used to enhance the system iteratively. When the software is ready it is released with training sessions and user's manuals. DSDM follows many of the Agile principles and is a very rapid software development but it needs many roles that make the administration of the project difficult, it does not take into account project criticality and the team size can result in issues.

- **Extreme programming XP method** includes six phases: in the first one, the exploration phase, in which user stories are used to identify requirements and features, the team get introduced to the future working environment and a sample prototype of the system is built in order to better understand the future system; The planning phase consists in scheduling, prioritizing and estimating future; in the Iterations to Release phase, iterations are done from the first overall system architecture to the production system, passing through the testing directly developed by the customer and executed at the end of each phase; during the Productionizing phase performances are tested one last time and the system is released to the customer but it must be kept running while new iterations are made; The Maintenance phase may require to add people to the team for the customer support; the last phase is the Death one, where no more changes are needed since there are no more user stories but also when going on with the development is too expensive.
- **SCRUM method** is the most popular one due to its simplicity and to the fact that it concentrates more on the management issues rather than on the technical ones that makes it applicable to many fields. Firstly objectives, tools and resources are identified, after that it passes

through a sprint phase, where iterations are done thanks to sprint cycles with fixed duration and each time an incremental value is given to the system with the same stages that goes from the sprint planning, where requirements are analyzed, to the delivery one after the design phase and the sprint review. The project is closed when objectives are reached and the documentation is ready. This kind of approach leads to divide the whole software in small components that can be easy managed and understood, improving the shareability among the teams and therefore also the communication that takes place during the frequent scheduled meetings. Also continuous testing is implemented since achieved goals can be self assessed and productivity increases while changes are not considering before the end of the sprint and the teams does not get distracted during the completion of current functionality. However since responsibilities are not well defined some violation can occur as well as the lack of technical guidelines can introduce some complications.

Surely, Big Data revolution characterized the last decade as discussed in [10], in fact the amount of collected data is exponentially increasing due to the growing number and diffusion of devices capable of collecting them (e.g. IoT) as well as the sharing channels, storage capacity and applications. The Big Data era led to the emergence of a relatively new approach to the ASD, the Data-Driven one. Industry started to experiment the use of data in decision making, since if quality of data and processing techniques, but also the right interpretation of them can result in better choices. An important step that leading-edge firms have done is to perform customer experiments in order to collect data on new products instead of passively make use of old data, we can define it a "test and learn" strategy. In [11] they provided evidence that Data-Driven Decision Making (DDDM) can strongly help to improve productivity, return on assets, return on equity and market value, that relates DDDM to firms performance.

Development

The development of ADAS and ADS functions is a very challenging task since these systems must interact with real world traffic scenario that is a very complex environment in addition to the serious safety issues that the user have to face. Vehicles have a highly nonlinear dynamics, especially in dangerous situations that may occur due to inclement weather conditions, high speed etc., that makes these systems very difficult to be controlled. From [12] emerges that the classical approach is to perform a model-based control analysis as well as synthesis techniques such as linear robust and optimal control, Nonlinear Parameter Varying (NLPV), Model Predictive Control (MPC), polynomial or Lyapunov control methods. However identifying and modelling nonlinearities and uncertainties - introduced also by the numerous external sources - is a very challenging task, for this reason the data-driven approach i.e. machine learning methods, in particular deep neural network, is growing in popularity to accomplish the perception and scene interpretation tasks.

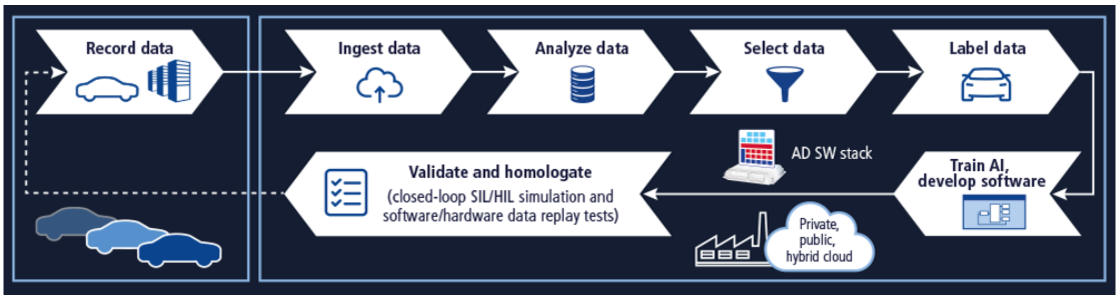
Record data
Ingest data
Analyze data
Select data
Label data
AD SW stack
Train AI, develop software
Validate and homologate
(closed-loop SIL/HIL simulation and software/hardware data replay tests)
Private, public, hybrid cloud

Figure 1.5: "*Data-driven development for developing autonomous vehicles leverages a data loop. This consists of data collection on the road, high-bandwidth data ingestion in a data center, analysis of data for relevant traffic situations, and labeling for AI training and for generation of ground truth in the data replay testing.*" (Source [13]).

Machine learning is used to automatically detect patterns in data collected from the real world. There are two types of machine learning methods:

Supervised and unsupervised learning. **Supervised Learning** uses methods such as Linear Regression, Bayes Theorem, Gaussian distributions, Euclidean dince etc. in order to extract a mathematical model from a labelled data set given as an input in the training phase. Naïve Bayes classifiers, k-Nearest Neighbours, Support Vector Machines and Artificial Neural Networks are just some of the most used supervised learning methods. **Unsupervised learning** is used to classify unlabelled data through a "trial and error" procedure that divides the data set into homogeneous groups, or clustering, and it is applied in data mining and market stocks analysis. The main models are Maximum Likelihood, Maximum a Posteriori, K-means, mixture and hierarchical models. One of the main challenges of machine learning is the large volume of data needed. As explained in [14] amount, size and scale of data are crucial parameters to be accounted. For machine learning size of data can be expressed either vertically by the number of records and samples in a data set, and horizontally by the number of features or attributes that are included. Volume is about the data typology, since quantity of data may compete with complexity of data, a smaller data set could be comparable with a bigger one but composed by simpler data. Naturally, the scale of the data introduces computational challenges because computational time will increase exponentially with the size of the data set. For example SVM has a training time complexity of $O(m^3)$ and space complexity of $O(m^2)$ or logistic regression $O(mn^2 + n^3)$ where m is the number of training samples and n the number of features. These algorithms are very sensitive to the imbalance of classes occurrence probabilities, in particular as data sets grows larger, uniformity of data distribution across all classes goes reducing itself. Also dimensionality, i.g. the number of features or attributes, represents a challenging issue since as dimensionality grows, the performance and accuracy of the model decreases, because of the similarity-based reasoning on which many of these algorithms rely on in parallel to the time complexity of many of them. These volume issues lead to lose the assumption of holding all the processing data in memory or in a single file on the disk and most popular solution is parallel execution on a large number of nodes. Another extremely time consuming task in the field of ML is the feature selection based on their relevance, it can also reduce dimensionality and hence time but become challenging for high dimensions due to spurious correlations and incidental endogeneity that is the correlation of the variable that explains variations in the response variable. Furthermore non-linearity of the system the algorithm is learning from, negatively affects model such as neural networks and logistic regression since even the correlation coefficient loses its meaning. Linearity is often exploited through graphical techniques but with large cloud of points, observation of relationships comes out to be more complicated. The learning process passes through the analysis of data to generate a prediction and therefore this necessarily introduces a generalization error, that can be divided in two components: variance and bias 1.6. Variance measures how consistent is the learner in predicting random things (overfitting) while bias is the tendency of the learner to learn the wrong thing (underfitting). A point of optima should be found in order to produce a good model. Regularization techniques are used to improve generalization and reduce overfitting but on the other hand, approaches such as cross-validation are needed in order to fit the model on unseen data. A less impactful but still important ML challenge, is the variety of the data set adopted for the training process, that means its sources, what the data represents, their semantic interpretation, the structural data set variation, data type and content. Other aspects of data variety is physical data location, as previously explained, as well as its syntactic and semantic heterogeneity not to mention that data are often dirty and noisy. After discussed the quality of data problems are not over because they need to be produced and analyzed with a high rate, it means that in some cases collect a huge amount of data is very complex, expensive

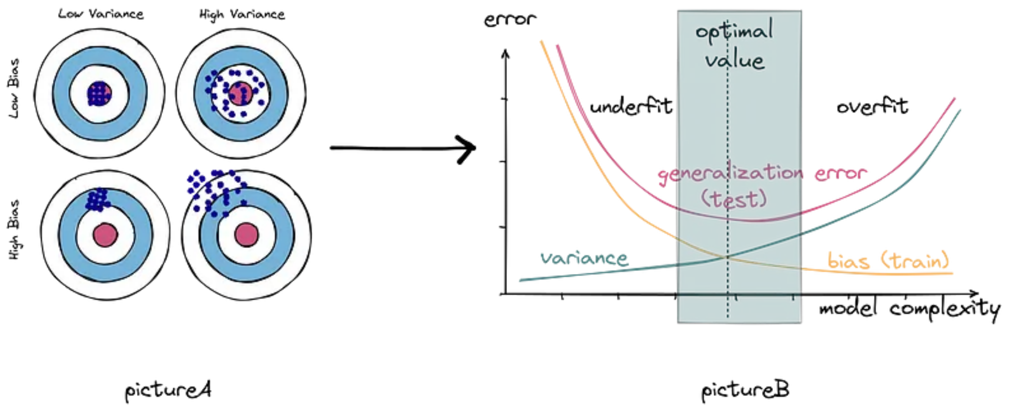

Figure 1.6: bias-variance trade-off (Source).

and requires time but even if availability is sidelined just for a moment, train again the model is a slow, costly and complex task. Following this path is easy to arrive at data veracity, another huge challenge for the industry. Veracity refers to the quality of the data for its provenance, i.e. how data was collected, from where the data is taken and how it will be analyzed, in this context data sources become a central topic. Knowing the source of data means also knowing the source of processing error, identifying invalid data thanks to the knowledge of contextual information. Thus, must be taken into account the data uncertainty introduced by the gathering process and procedures. For example some method can rely only upon subjective criteria, that makes very difficult for the learner to find a path. Such a subjectivity can influence also data processing resulting, for example inaccurate labels are very dangerous for these models because from their point of view there is no difference between the wrong image and an unclear one.

Validation

Data driven approach, as seen, is revolutionizing the world of decision making and management, it has been also a disruptive news for software development introducing Artificial Intelligence in developing functions and it has its huge effect also in validation processes. Ensuring safety of ADAS and AVs is one of the most challenging automotive industry's objectives of our time. As stated in [15], software testing, very often turns to be a bug hunt instead of an entire and well performed quality ensuring process. When the objective is deploying a safe autonomous vehicles at scale, the safety assessment must be a more methodical approach. Dealing with autonomous driving functions means encountering a unique testing challenge that the ISO 26262 tries to face with a V process that establishes a structured framework that connects various types of testing to specific design or requirement documents. According to the standard, five major challenge areas have been found for what concerns the testing process, which are: driver out of the loop, complex requirements, non-deterministic algorithms, inductive learning algorithms, and fail-operational systems. To this day autonomous driving systems are considered safe when they are designed and validated within the ISO 26262, i.e. must be shown that these system may conform or map to the standard. The first concept that have to be taken into account is the infeasibility of complete testing and vehicle-level testing is not enough to achieve an ensured safety. According to [16], in order to demonstrate that failure rate of Autonomous Vehicles is lower than some benchmark, modelling the problem as a binomial distribution, the number of miles AVs must be driven n is given by:

$$C = 1 - R^n \rightarrow n = \frac{\ln(1-C)}{\ln(R)} \tag{1.1}$$

where R is the reliability, that can be also expressed as $R = 1 - F$ where F is the per-mile failure rate of a vehicle and C is the desired confidence. For example, in order to have 1.09 fatalities per 100 million miles (R =

99.9999989%) with a C = 95% confidence level the vehicle should be driven on the road for 275 million **failure-free** miles, if the task is split among one hundred vehicles, that are driven for 24 hours a day, 365 days at an average speed of 25 miles per hour, it would take about 12.5 years. But if it is necessary to estimate the distance to obtain a failure rate with a specific degree of precision things become even more costly in terms of miles. In fact, it must be taken into account the real autonomous failure rate that is the number of failure that occur for a given distance driven with its probability, in [16], a normal approximation to the Poisson distribution have been used to calculate such a number. With the same data previously used but estimating the fatality rate of the fleet to be within 20% of the assumed rate using a 95% CI, it gives 8.8 billion miles that have to be driven, it would take 400 years in the conditions of previous example. The paper also includes a calculation where the concept of "*power*" is added, the power of the test is the probability that the test will reject the tested hypothesis when a specific alternative hypothesis is true. The calculation are made with a power of 80% and the result is that in order to demonstrate failure rate of AVs is 20% better than the human driver fatality rate, they have to be driven for more than 11 billion miles, that means 518 years. An interesting observation is that Advanced Driver Assistance Systems can still rely upon the driver to over-ride the software function that is faulting. For highest automation level the driver cannot take any corrective action and there is a lack if controllability, that is a popular automotive safety arguments for low-integrity devices, that means they have to be designed to a higher Automotive Safety Integrity Level. Potentially high ASIL can be handled through keeping severity and exposure low while controllability is set to C3 but there is also another approach that consists in designing a redundant monitor/actuator system. In this architecture the primary function is performed by the actuator while the monitor has a behavior validation role, shutting the entire function down when detecting actuator's misbehavior. If it is done properly, the actuator can be designed to a low ASIL when the monitor has a sufficiently high ASIL, requirements also take care of detecting latent faults in the monitor that could cause it to fail in detecting an actuator fault. The action of the monitor could cause a loss of the actuator function and in the case of operational fail it is a huge problem, for this reason it is required more redundancy as well as design diversity because same design leads to the same failure across primary and backup system, which become a systematic fault. The absence of the driver in handling unpredictable situations that can occur in a very complex scenario as the road traffic leave to the autonomous system the responsibilities of reacting to exceptions such as bad weather, traffic rule violations, local drive conventions, animal hazards and many more. High quantity and many different type of these exception make classical requirements impossible for the use case. A solution is to plan a phased expansion of the requirement and limit them to precise operational concepts, the latter can be scaled in many direction for example:

- Limitations in road access such as urban street, highways, rural roads, suburbs, HOV lanes, closed campuses, etc.
- Visibility conditions e.g. : day, night, fog, haze, smoke, rain, snow, etc.
- The interactions with the external environment like infrastructure support, pre-mapped roads, convoying with human-driven cars.
- for what may concern speed it is know that reducing speed can decrease the damage caused by failures and larger recovery margins.

Obviously many combinations of these degrees of freedom can be made in order to reduce the complexity of requirements. In the next phases the operational scenario can be extended adding degrees of freedom. Another huge challenge in validating Autonomous Vehicles comes from the fact that in this field some technologies are inherently statistical for their nature, means that the responses have to be taken with their probability (when it is available) therefore they are non-deterministic and so non-repeatable, some vehicle-level test could give a different outcome in each attempts of executing the same test case. For example, this is the case of probabilistic roadmap planners or perception algorithms. Some sources of non-determinism are the modeling geometry of surroundings through sensors acquisitions or extraction of labels from that data in order to perform the object detection, for example shadows and reflective surfaces in the case of vision system or the tradeoff between false negatives and false positives that any classification process must face. It implicates that in the testing phase, the results of such algorithms have to be taken with a statistical approach. One of the main consequences of non-determinism in testing is that it become very difficult to reach a specific edge-case situation since the system under test could require a very specific input sequence from the world in order to trigger such edge-case, small changes of input can cause a very different response each time. A problematic testing situation, for example, is to test the behavior of the worst choice between two roadways but the system would never choose the most unattractive one and it must be manually forced to do such choice. This leads to an even more difficult testing scenario which validate the vehicle's ability to consistently choose the better of two equally bad paths without vacillating. Also stating the correctness of the test result for a stochastic system, is not a simple task due to the presence of more than one correct behavior, moreover, there could be no certainty that a passed test will always success, in fact validating probabilistic systems often means make sure that the statistical characteristics of the behavior are accurately specified, but it requires a large number of tests. Some of the solutions to the cited issues in validating high automation level systems are to target the tests on edge cases and specific data-driven scenarios but also replace or partially replace physical testing with virtual testing. As explained in [17] Scenario-based testing and virtual testing can bring many advantages such as tests repeatability across different combinations of autonomous systems, the ability of scaling up the amount of tests, a safer environment for the validation and shrinking the costs that arise from the validation process. When the process is executed with the system in a simulated environment it is referred to as X-in-the-loop. The latter is a very recent approach and still has its challenges, first of all the huge amount of data needed in order to create the simulation environments and to structure the training, testing and validation data sets. In the previously discussed context where a lot of data are produced and consumed, they must be consistently exchanged as well as the certification processes must be scalable and repeatable not only in the same organization but also across different stakeholders, in order to permit also a comparison between certifications from different subjects due to requirements and results that can be shared across the industry. A standardized data structure means it can be better processed, analyzed, transferred and stored.

Chapter 2

The Standard

2.1 ASAM: The Organization

ASAM e.V. (Association for Standardization of Automation and Measuring Systems) is an organization that was born in 1998 from the previous project "*ASAM*" by AUDI, BMW, Daimler-Benz, Porsche and Volkswagen gathered in a cooperative effort to promote the standardization in the automotive industry, trying to reverse the direction of the standardization process in fact, until that moment, OEMs were developing standards in a unilateral manner and then imposing them to the suppliers. The initiative was taken because in the area of measurement and testing for vehicle development, the founders identified as a source of costs that could be optimized, the lack of interconnectivity and exchange of data due to the presence of incompatible interfaces and data formats. The proposed solution consisted in including the suppliers in the standard developing process from the beginning as equal partners, in such a way also their technological know-how was added to the project improving standards feasibility and cost-efficiency of the products. Today ASAM e.V. is a no-profit association that encourages and gives its contributes to the standardization of tool chains in automotive development and testing, it distributes the owned standards, developed with the help of experts who are its members, in fact, standards are result of the stakeholders collaboration which are OEMs, suppliers, tool vendors, service providers and research institutes. Common challenges are identified in a wide range of use cases in automotive development, test and validation, then members try to find a solution all together. The organization respects the EU competition law by defining no product or taking no business decisions, there are no relations with the regulatory framework and everyone can download or purchase the standards. The process of standardization takes place on a neutral platform where members can cooperate in a non-competitive manner. ASAM e.V. can count on more than 400 members companies worldwide that are OEMs such as AUDI AG, BMW AG, VolksWagen AG, PSA Group etc. or tool vendors such as Amazon, MathWorks, National Instruments, Vector or academics such as Politectnico di Torino, Research Institute of Sweden, Shangai Artificial Intelligence Innovation Center and so on. These members provide experts that make possible the standardization through the exchange of technical ideas [18].

2.2 The ASAM Domains

The ASAM standards cover seven domains of the automotive sector and they are categorized within the domain each standard belongs:

1. **Measurement & Calibration**: inside this class there are the standards for working with ECU variables and parameters that includes read-write access to the data in ECU memory, meta-description of the data, storing the data in files and describing the Calibration process. The standards are: ARTI, CDF, CMP, CPX, HMS, MCD-1 CCP, MCD-1 POD, MCD-1 XCP, MCD-2 CERP, MCD-2 MC and MDF.

2. **Diagnostics**: contains the standards for describing and testing the diagnostic subsystems of devices, which are MCD-2 D, MCD-3 D and SOVD.

3. **ECU Networks**: this domain includes the standards for describing and testing ECU networks, in particular ASAM MCD-2 NET.

4. **Software Development**: it groups all the standards that help the ECU software development and the development of functional safety features such as the formal description and documentation of ECU software or the description of change requests, but also blocks sets for model-based engineering and a notation for graphically represent the interactions between safety design and system architecture. The standards in this domain are: CC, FSX, ISSUE, LXF, MBFS, MDX and SCDL.

5. **Test Automation**: all the standards that refer to the test systems such as in APIs for programmatic access to sensor and actuator devices, Measurement and Calibration systems, HIL systems, DoE systems and formats for test descriptions, are included in this domain. ASAM standards for test systems are ACI, ASAP 3, ATX, GDI, iLinkRT, MCD-3 MC, OTX Extensions, XI and LXIL-MA.

6. **Data Management & Analysis**: these are the standards used when it comes to store, retrieve and analyze mass data captured during simulation, testing, production and the operation of vehicles. In particular they are CEA and ODS.

7. **Simulation**: The ASAM standards inside the domain Simulation are called ASAM OpenX® standards and are thought to provide a complete set of standards for simulation-based testing of automated driving functions. They span over a wide range of use cases for virtual development, for example hybrid testing approaches that combine virtual and physical components. The standards that are part of ASAM OpenX are OpenCRG, OpenDRIVE, OpenLABEL, OpenODD, OpenSCENARIO and OSI.

2.3 ASAM OpenX: The Simulation Domain

As noticed before the technological progresses allowed the vehicles to be equipped with a growing number of sensors and an increasing computational power while keeping the costs reasonably low. The affordability of such a system led to the development of new driving assistance functions and the ADAS swiftly became standard installation in vehicles sold in the consumer market, from being just optional, mainly because of the 5-star requirements provided by the New Car Assessment Program, therefore the potential of Automated Driving Systems is estimated to rapidly grow in the next years. Development and validation of these systems, as evident from the previous discussion, cannot rely only on the physical tests due to many reasons such as the long distance they should be driven and the not availability of the target hardware, where the software will run, during the development phase since it is being produced in parallel. In this context, simulation plays a crucial role because it allows software testing to commence at a very early stage. This early testing phase enables the detection of bugs and missing capabilities, which can then be rectified in the development phase and as simulations become more realistic in mimicking stimuli for the target system under test, their importance in the automotive industry continues to escalate. Simulations can be used for challenging virtual models (Model in the Loop) or software containers (SiL) that represent ADAS or ADS systems. As the development progresses, simulations also interface with tangible components, individual Electronic Control Units (ECUs), or groups of ECUs (HiL) within the final product. These physical components of the vehicle will already run the software on the intended target hardware. When the complete set of ECUs and software can be assembled to form the vehicle control system (ViL/DiL/proving ground), simulations can continue to interact with the system. What makes SiL and MiL testing exceptionally advantageous is the possibility to perform tests and validation without relying on a physical Device under Testing (DuT). This approach offers significant benefits, notably in terms of time and cost efficiency. In the realm of simulations, one second corresponds to about 20 real-world seconds, this allows for the execution of a higher number of test cases, enabling larger validation among the necessary scenarios. Consequently, this accelerated testing process not only saves time and costs but also ensures that the system undergoes rigorous scrutiny across a wide array of situations, enhancing its overall reliability and functionality. Physical testing needs test drivers, test vehicles, large spaces for proving ground, is expensive and relies on external factors to reach specific use cases, such as weather conditions that are not controllable. On the other hand, virtual testing in simulations permits parallel testing, allows to create scenarios that focus on the test objectives, enables to perform a larger number of tests in the same amount of time, tests can be replicated and helps to define the test focus for physical test sites. Creating the necessary test cases for validating ADAS or ADS systems is a daunting and time-consuming task. It's practically impossible to anticipate and cover every conceivable scenario a vehicle might encounter throughout its life cycle. The challenge becomes even more challenging when developing level 3 or level 4 automation, even for major OEMs. Collaboration is vital in this context to prevent unnecessary expenditures, such as those incurred for format conversion or incompatible tooling. The demand for exchange formats and description formats that support the development of these systems is consistently increasing. Standardized approaches are not just helpful but essential: employing universally accepted exchange formats enables the industry to collaborate effectively, fostering an environment where automated driving can truly become a reality. The standards inside the ASAM Simulation Domain pursue precisely this goal since they have been conceived from the original owner to the ASAM release.

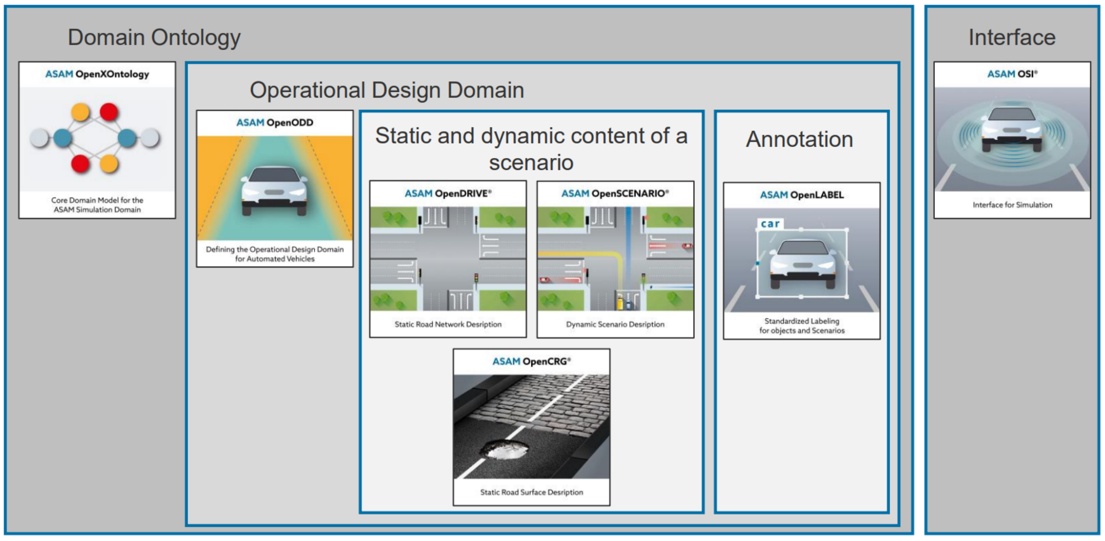
Domain Ontology
ASAM OpenXOntology
Core Domain Model for the ASAM Simulation Domain
Operational Design Domain
ASAM OpenODD
Defining the Operational Design Domain for Automated Vehicles
Static and dynamic content of a scenario
ASAM OpenDRIVE®
Static Road Network Desription
ASAM OpenSCENARIO®
Dynamic Scenario Desription
ASAM OpenCRG®
Static Road Surface Desription
Annotation
ASAM OpenLABEL
car
Standardized Labeling for objects and Scenarios
Interface
ASAM OSI®
Interface for Simulation

Figure 2.1: ASAM OpenX Standards (Source [19]).

The Simulation Domain contains the following standards:

- **ASAM OpenXOntology**: it is currently in a early stage where the first concept have been released. The aim of this project is to provide a foundation of common definitions, properties and relations for main concepts of the ASAM OpenX standards, in fact, despite all of them refer to the same domain i.e. road traffic, they are based on different domain models. This results in redundant definitions and descriptions of concepts that sometimes also contradict themselves. For example exist two overlapping definitions of lane references one in OpenSCENARIO and one in OpenDRIVE. OpenXOntology aims to solve this issue in order to allow the linkage of data stored in the different formats of the OpenX standards. For example the concepts inside the scenario labels for driving simulations, their properties and relations are not standardized so it is difficult to extract scenarios form logged data in a standardized manner or automatically generate synthetic scenarios in a traceable way. It could also prevent the use of common labels for object and scenes and therefore retrieve scenarios from common queries. An ASAM ontology, which encompasses definitions related to traffic infrastructure, interactions among traffic participants, and environmental conditions, addresses a crucial gap by establishing a standardized foundation for vocabulary and domain models within the OpenX standards. This standardization not only offers a common language but also streamlines the integration of artificial intelligence into OpenX applications. It also allows a more efficient and effective implementation of artificial intelligence for OpenX applications.

- **ASAM OpenODD**: (Operational Design Domain) it is in the last stages of the development phase and its objective is to represent a specific Operational Design Domain for Connected Automated Vehicles through a machine-interpretable format. It should be valid throughout the entire operating life of a vehicle and it is part of the safety and operational concept of the system, furthermore, the ODD is used for the functional specification of the CAV and which environment parameters (both static and dynamic) the vehicle must be able to manage i.e. everything that compose the driving situation such as traffic participants, the weather conditions, the infrastructure, the location, the time of the day etc. ASAM OpenODD represents the vehicle ODD and it can be used for description, simulation and post-processing purposes but the format must be searchable, exchangeable, extensible, readable by machine and by humans (constrained natural language), measurable and verifiable. ASAM OpenODD is composed by a base set of *attributes*, semantics and syntax of its description language includes different ontologies and taxonomies, it also enables the possibility for any application to perform analysis on the ODD making use of the measurable metrics, the ODD format is capable of handling rare events and misuse throughout the representation of uncertainty. The ASAM OpenODD development consider to enrich the activities of BSI (BSI PAS 1883 provides a taxonomy for ODD) and ISO (ISO 34503 uses the taxonomy to provide a high-level definition format for ODD), these projects communicate to avoid contradictions.

- **ASAM OpenDRIVE**: have been developed multiple versions and there is another one in the conceptual phase but their goal is always to give a common base for describing road networks with the syntax of extensible markup language using the file extension xodr. Within an ASAM OpenDRIVE file, stored data outlines the geometry of roads, lanes and objects like roadmarks and features such as signals. Described road networks can either be artificial constructed or based on real-world data. The ASAM OpenDRIVE primarily functions is to provide a detailed description of road networks, essential for simulations used in the development and validation of ADAS and Automated Driving Systems features. Thanks to the ASAM OpenDRIVE standard, these descriptions of road networks can be seamlessly shared between different simulators, promoting interoperability in the industry. By establishing a standardized format for road descriptions, the automotive sector benefits from reduced costs associated with the creation and conversion of these files for various development and testing purposes. These road datasets may be generated manually using road network editors, derived from map data conversions, or even originated from scanned real-world roads, further illustrating the versatility and practicality of the ASAM OpenDRIVE format. One of the main characteristics of ASAM OpenDRIVE is that it is organized in nodes that the user can extend with its own defined data, allowing a high level of specialization of the single application (often simulation) but still preserve the capability of interoperate that permits the exchange of data. Another important aspect is that the whole road network is modelled along the reference line. Road and lanes with their elevation profile are attached to the reference line that represents a s/t-coordinate system. Objects, representing features, refers either to the reference line which is the core of the road or to the global coordinate system. The road network is composed by segments of road and junctions that interconnect the sections of road and the lanes. The junction snippets tell which is the predecessor and the successor of the piece of road it refers to. The only roads with overlapping surfaces are the roads that connect entry roads, they are called "*connecting-roads*". However, to fully represent the entire environment, supplementary formats are necessary. These additional formats detail static 3D objects found along roadsides, such as trees and buildings. Road surface profiles, crucial for accurate representation, are integrated from the ASAM OpenCRG file format. When it comes to the dynamic aspects of driving simulations, including vehicle maneuvers and behaviors, ASAM OpenSCENARIO steps in to provide a detailed description. These three standards are complementary and together describe all the static and dynamic content of in-the-loop vehicle simulation applications.

- **ASAM OpenCRG**: establishes a specific file format tailored for describing road surfaces. Initially designed to store high-precision elevation data collected from scans of road surfaces, its main applications are in tire, vibration, and driving simulations. This highly accurate elevation data is used in conducting realistic endurance simulations, whether for the individual vehicle components or the entire vehicle undergoing analysis. In fact, for driving simulators, OpenCRG enables the creation of a realistic 3D representation of the road surface. Moreover, the versatility of the OpenCRG file format is evident as it can also accommodate other essential road surface properties these include factors like the friction coefficient or gray values. The standard outlines a technique for organizing data in a specialized format known as "curved regular grid" (abbreviated as CRG). This method offers several key advantages. Firstly, it ensures optimal memory usage, maximizing efficiency in storing vast datasets. Secondly, it significantly reduces the time required for both generating files and processing data within simulation tools. Lastly, the CRG layout guarantees exceptional accuracy when positioning data onto intricate road networks, ensuring precise and reliable representation of the road surface in simulations. The fundamental approach to depicting the road surface involves organizing data within a grid aligned along the road reference line, the segments of the latter are described by a start position and a heading angle. This grid is constructed through longitudinal cuts (columns) and lateral cuts (rows) made along these consecutive line segments. Within each grid cell, there is a specific value assigned, usually denoting the elevation of that particular point on the road. The end position provides crucial information that can be utilized to identify and rectify any potential discrepancies or shifts in the placement of data along the roads. ASAM OpenCRG establishes both ASCII and binary file formats, each incorporating clear-text headers. Within these headers there are road parameters for the reference line as well as the overall configuration of the longitudinal sections, modifiers and option parameters. They also outline the data format (ASCII and binary) and describe the expected sequence of data in the trailing data block. The format includes the possibility of referencing to external files that often contains the actual data, in order to manage different parameters for the same dataset.

- **ASAM OpenSCENARIO**: defines a file format specifically tailored for describing the dynamic aspects of driving and traffic simulations. Its main purpose is to depict intricate and synchronized maneuvers involving various entities like vehicles, pedestrians and other traffic participants. These maneuvers can be described based on driver actions, such as lane changes, or on trajectories derived for example, from recorded driving maneuvers. Additionally, OpenSCENARIO includes details about the ego vehicle, driver appearance, pedestrians, traffic, and environmental conditions. The standard defines vehicle maneuvers in a structured manner, organized into storyboards containing stories, acts, and sequences. A story can either outline the maneuvers of an individual vehicle or describe the dynamic behavior of multiple entities, such as vehicles performing a lane change when reaching specific positions. Stories are composed by acts that specific conditions trigger for example when a specific distance to a vehicle ahead is reached, but also exceeding a fixed speed or when the vehicle is going off-road and with the concept of sequences, the standard permit to respond with the definition of maneuvers of multiple vehicles, for example overtaking another car with a lane change or creating a corridor for emergency vehicles. The driving behavior of the vehicle is detailed through events and actions, the latter can also be related to the environment and such a traffic light change. Thanks to the standard can be defined also the routes and trajectories that the vehicle must follow. All these element organizes in catalogs as well as the complete scenario description can be parameterized to improve test automation without large amount of scenario files. ASAM OpenSCENARIO organizes data in a hierarchical structure and serializes them in an XML file format with a precise schema, since it is technology and vendor independent it can be easily validated, edited, imported and exported by content editors and simulation tools. Since maneuver description is essential for the safety certification process the industry, certification agencies and government authorities are working together on the definition of maneuver libraries.

- **ASAM OpenLABEL**: it provides the annotation format and the labeling methods for objects and scenarios. ASAM OpenLABEL defines a set of rules to guide the process of labeling. The categorization and description of the objects that populate the driving environment are fundamental parts of the Automated Driving Systems perception stack and this format represents a huge effort to standardize these processes in order to solve many issues that affect the industry, for this reason it will be better discussed in the next paragraph.

- **ASAM OSI**: ASAM Open Simulation Interface ensures intuitive and consistent compatibility between automated driving functions and different driving simulation frameworks. It permits users to connect sensors to automated driving functions and various simulator tools through a standardized interface. This simplifies integration, significantly enhancing the accessibility and usefulness of virtual testing. ASAM OSI was initially thought to be a generic data exchange interface compliant with the ISO 23150 logic interface for the environmental perception of automated driving functions in virtual scenarios. In collaboration with packaging specifications like ASAM OSI Sensor Model Packaging (OSMP), this standard offers solutions for exchanging simulation model data efficiently across various implementations. The environment description inside ASAM OSI is object-based and uses the message format of the protocol buffer library, developed and maintained by Google. Top-level messages are defined in order to exchange data between separate models and they represent the GroundTruth interface, the SensorData interface, the SensorView/Sensor-View configuration interfaces and the FeatureData interface. While the GroundTruth interface provides en exact representation of the simulated object in a global coordinate system, the FeatureData interface gives a list of of simple features in the reference frame of the respective sensor of a vehicle for environmental perception. The list is generated by a GroundTruth message and could be an input for a sensor model that simulates object detection or feature fusion of multiple sensors. Also traffic participant models have their own interfaces, in fact, it is possible to send them commands through the TrafficCommand interface and through the TrafficUpdate interface it is possible to get the updated state of the traffic participant models. The latter can also make use of other ASI OSI interfaces internally to model autonomous vehicles.

2.4 ASAM OpenLABEL Standard

The ASAM OpenLABEL is a widely adopted standard in the automotive industry. The current and below described version is the 1.0.0. It provides a standardized annotation format and gives precise labeling methods for multi-sensor data streams and scenario files. This standardization process helps stakeholders in cutting costs and avoiding the processes of creating, converting and transferring annotated and tagged data and as a consequence also saving resources. ASAM OpenLABEL provides many methods for labeling multi-sensor data streams such as images or point clouds that are for example 2D or 3D bounding boxes. The standard also focuses on scenario tagging that allows the scenarios to be categorized and searched inside large data sets but also gives information about the individual scenario such as the scenario creator, the setup used to capture the scenario and so on. Essentially it is a common data structure for organizing annotations for labeling multi-sensor data streams and perform simulation and test scenarios tagging. As already demonstrated, for developing, testing, and validating highly automated driving functions, the industry heavily relies on Machine Learning (ML), particularly for tasks involving perception and prediction. Machine learning demands substantial training data, which not only needs to be abundant but also meticulously annotated and enriched with metadata to serve its purpose effectively during the training and validation processes. The lack of an industry standard for defining the structure and the organization of these annotations results in a limited reuse of annotated datasets, the annotations maintenance and updating become difficult, the sharing of datasets across the industry and between industry and academia is necessarily hindered and annotations surely experience a worsening in quality. On the other hand the presence of a multi-sensor data labeling standard such as ASAM OpenLABEL can lead to an efficient sharing of annotated perception datasets and object lists, an increase in annotations quality and finally improve the maintainability and reuse of annotated datasets. The main sectors to which ASAM OpenLABEL is targeted are: Perception/computer-vision engineers, Machine-learning engineers, Perception/computer-vision research scientists, Machine-learning research scientists, Data-annotation engineers, Data-annotation analysts and Test engineers. The need for a scenario tagging standard comes from the fact that scenario databases storing multi-sensor data, annotated multi-sensor data, simulation scenarios and test scenarios can be very extensive and must be organized and tagged using semantic, meaningful tags that refer to the content of the data, its ODD, the high-level behavior of the dynamic agents and administrative information. The process of information extraction from scenario artifacts, required for the tags, is difficult and sometimes impossible due to the used scenario definition language but ASAM OpenLABEL based scenario tagging aims to solve this problem. It permits to group test scenarios in scenario databases, facilitate the scenario storage systems, make more navigable the scenario databases throutgh search and filtering, improve the scenario data sharing, gain easier maintainability and reuse of test scenario and scenario data, increase quantity and quality of machine-learning training and validation datasets but also Enable specific machine-learning classification tasks to be performed on scenario data. The main target groups of the standard are systems engineer, validation and verification engineers, functional-safety engineers, simulation specialists. ASAM OpenLABEL consists in a JSON format and for such reason, it can be easily parsed by tools and applications. The structure, sequence, elements and values within the JSON file are also specified by ASAM OpenLABEL. The elements of the standard are actions, objects, events, contexts, relations, frames and tags and also relationships between them are defined in the standard. Quality of the annotations is no taken into consideration between the features that can be described in the format but it includes geometries, coordinate systems and transforms, and other concepts relevant to spatio-temporal annotations for multi-sensor data labeling. For what may concern the taxonomy it allows the integration of external knowledge repositories/ontologies and in particular of ASAM OpenXOntology as the ontology of reference. The standard provides naming conventions, units, guidelines for timestamps, date and time formats. Data annotation means enhancing raw data, like sensor streams from cameras, LiDAR, radar, or test scenario artifacts, with extra metadata. This metadata provides context, such as identifying static or dynamic objects in a video, detailing their actions, or describing the surrounding environmental conditions. Raw data can be enriched by adding other pertinent additional information. Raw data can take many forms, for example, individual files, file streams, or test scenario artifacts. For ASAM OpenLABEL, important example of raw data are png images, frames in a video sequence, pcd point clouds, or OpenSCENARIO files. ASAM OpenLABEL provides a very well defined annotation schema that represents a data model.

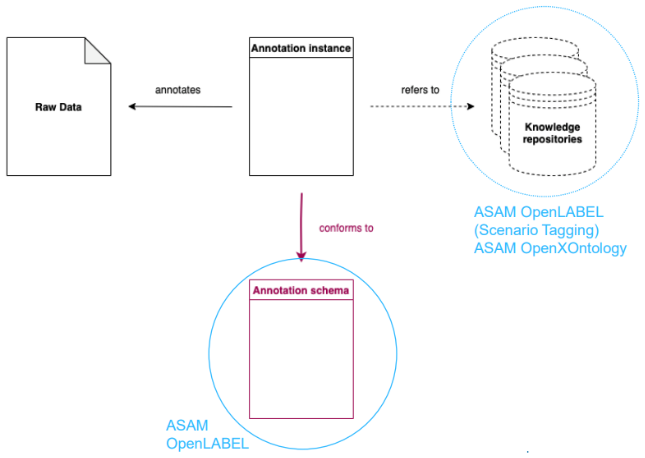

Figure 2.2: Multi-sensor data labeling concept (Source [20]).

The annotation format includes geometries, such as, bounding boxes, polygons or other primitives to isolate and localize relevant semantic concepts inside the raw data. Labels usually represent a semantic reference to agents type identification, their relations, actions they are performing and contexts in which these actions or agents take place or exist. Moreover, the standard gives space for details about spatial calibration across sensors, temporal synchronizations, coordinate transforms and consistent entity IDs across frames and sensor streams. In figure 2.3 an example of the relations between files that the annotations refer to.

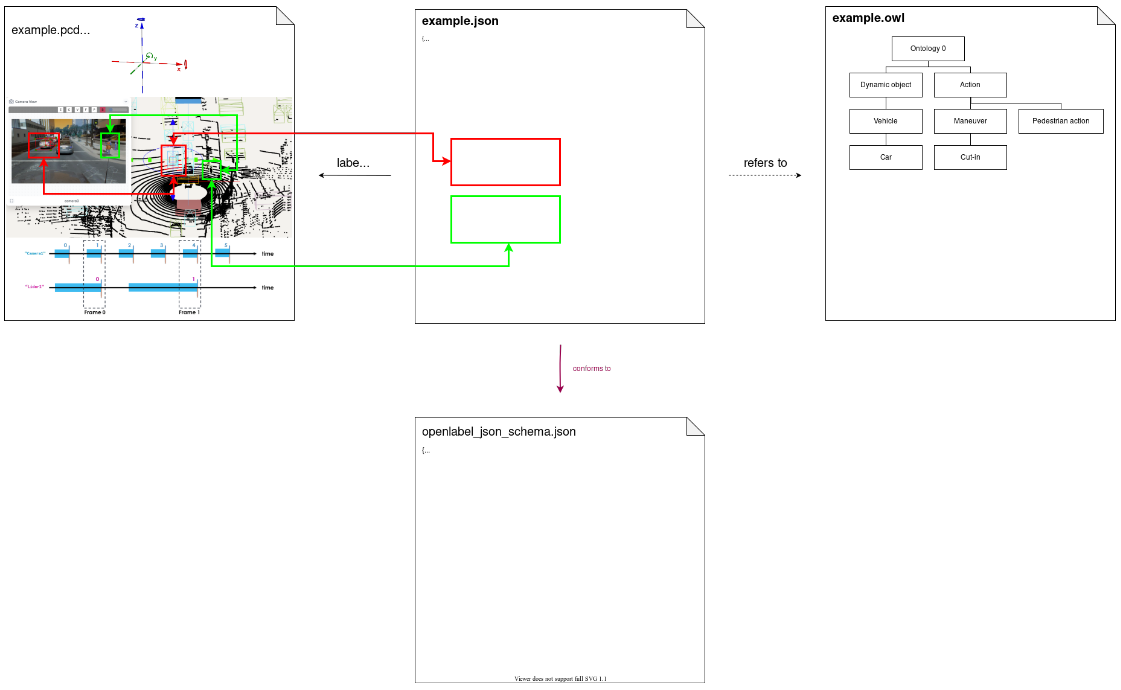

Figure 2.3: Multi-sensor data labeling example (Source [20]).

Annotations of multiple raw sensor data streams inside example.pcd, example.png and example.json files are contained in the example.json file that is compliant to the schema present in the openlabel_json_schema.json file that can be used to validate the annotations. There is also represented an external ontology in the example.owl and is referenced by the file example.json in order to semantically enrich the annotations.

The annotation schema outlines the organization of annotations, including data types and clear conventions, ensuring unambiguous interpretation. It also dictates how annotation data is encoded for storage in computer files. ASAM OpenLABEL's annotation schema is versatile, accommodating tasks such as basic object labeling in single images, with techniques like bounding boxes or semantic segmentation, but also intricate multi-sensor data labeling involving elements like cuboids, odometry, coordinate systems, and transforms. Its format (JSON schema) allows for easy serialization of labels in files or messages, making them readable for both computers and humans. This flexibility ensures communication and understanding between systems and users. The annotation schema is defined using JavaScript Object Notation (JSON) schema, specifying the format that valid JSON annotation instances must adhere to. This schema is serialized in the ASAM OpenLABEL JSON schema file, outlining the structure. The schema itself conforms to the JSON schema Draft 7 specification. The keys in the JSON schema can be either predefined keywords as strings or identifiers that can be numerical, strings or unique identifiers. The schema tells which pattern keys shall follow for different types of items. ASAM OpenLABEL annotations can be represented as JSON string payloads, where the data, organized as key-value pairs, is encapsulated within a string format. As encoding format of characters shall be used UTF-8. Labels are nothing but spatiotemporal descriptive information of data such as images, through objects, actions, events, contexts and relations. It is possible to add simple o complex tags to any content such as images, data files or scenarios and through additional structures can be added details for metadata, ontologies, frames and coordinate system. For the purposes of this thesis work the focus will be mainly on the process of multi-sensor data labeling that consists in adding information to data streams about the location and the characteristics of labeled objects or even the entire scenario at a given point in time. Labeling means taking relevant semantic entities within the raw data and locating them in a spatiotemporal manner relatively to the other data using spatiotemporal constructs such as many labeling geometries, each one suitable for a specific use like computer vision or machine learning. Raw data can be images, videos, point clouds.

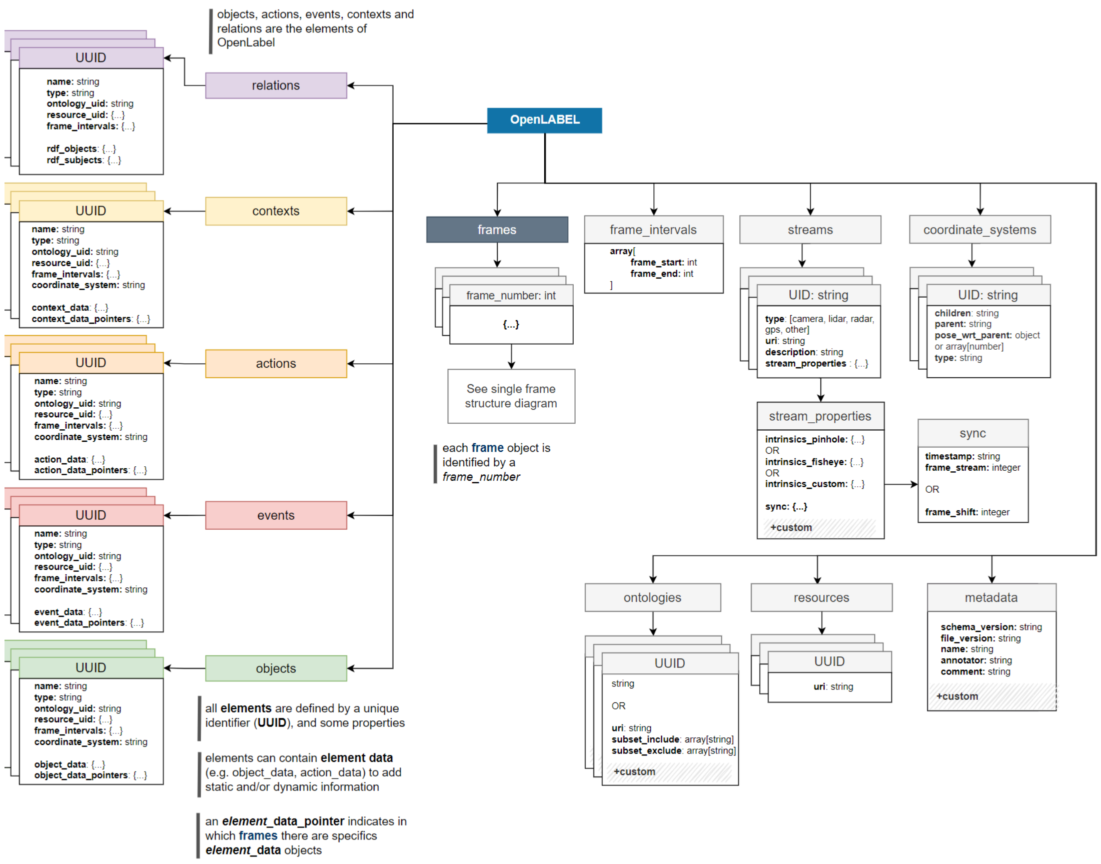
objects, actions, events, contexts and relations are the elements of OpenLabel
OpenLABEL
relations
contexts
actions
events
objects
UUID
name: string
type: string
ontology_uid: string
resource_uid: {...}
frame_intervals: {...}
rdf_objects: {...}
rdf_subjects: {...}
coordinate_system: string
context_data: {...}
context_data_pointers: {...}
action_data: {...}
action_data_pointers: {...}
event_data: {...}
event_data_pointers: {...}
object_data: {...}
object_data_pointers: {...}
all elements are defined by a unique identifier (UUID), and some properties
elements can contain element data (e.g. object_data, action_data) to add static and/or dynamic information
an element_data_pointer indicates in which frames there are specifics element_data objects
frames
frame_number: int
{...}
See single frame structure diagram
each frame object is identified by a frame_number
frame_intervals
array[frame_start: int frame_end: int]
streams
UID: string
type: [camera, lidar, radar, gps, other]
uri: string
description: string
stream_properties: {...}
stream_properties
intrinsics_pinhole: {...}
OR
intrinsics_fisheye: {...}
OR
intrinsics_custom: {...}
sync: {...}
+custom
sync
timestamp: string
frame_stream: integer
OR
frame_shift: integer
coordinate_systems
UID: string
children: string
parent: string
pose_wrt_parent: object or array[number]
type: string
ontologies
UUID
string
OR
uri: string
subset_include: array[string]
subset_exclude: array[string]
+custom
resources
UUID
uri: string
metadata
schema_version: string
file_version: string
name: string
annotator: string
comment: string
+custom

Figure 2.4: ASAM OpenLABEL schema (Source [20]).

The annotation schema is based on three main characteristic aspects of the annotation data:

- **Structure**: the organization of the data such as hierarchies and key-value dictionaries.
- **Types**: Primitive data types for key-value items.

- **Conventions**: Documented interpretation of data values.

For what may concern the **structure**, ASAM OpenLABEL JSON file is a dictionary with a root key named *openlabel.* The value of the root key is an object containing all the other keys, such keys are supporting structures and are the followings: *ontologies* that groups all the terminologies that shall be used to conform to the standard, *resources* external items to link data and *coordinate_systems* in order to explicitly specify the transformation of the data and finally *streams* for other information on the collection of data they refer to, for example, sensor information, such as intrinsic calibration parameters of cameras. Other keys are the **elements** listed below:

- *objects*: contains information about physical entities in scenes for example pedestrians, cars, the ego-vehicle, traffic signs, lane markings, building, and trees.

- *actions*: where acts with a semantic meaning that are being done and can last several frames, are described, for example *isWalking.*

- *events*: where instants in time with a semantic load are reported, they can trigger other *events* or *actions*, for example *startsWalking.*

- *contexts*: are additional described information that are not inside *events* or *actions* because have no specific timing or location, for example weather conditions, properties of the scene (e.g. *Urban* or *Highway*) or general condition about the locations, such as the country name.

Elements have similar properties from the point of view of the attributes, types and hierarchies. Attributes of **elements** are:

- *uid*: it is a unique identifier for a specific element, it can be a simple unsigned integer or a Universal Unique Identifier (UUID) of 32 hexadecimal characters.

- *name*: it is an identifier, simply understandable by humans and it is not unique.
- *type*: this attribute corresponds to the class of the elements identified, for example a *Car* type.

Additional items can be added to the elements, such as:

- *ontology_uid*: a reference to the ontology used for the *type* of the element.
- Element data, for example *object_data*: this element contains static information referring to the object.
- Element data pointers, for example, *object_data_pointers*: are pointers to element data at frames that also reduce the nesting of data.
- *frame_intervals*: it lists the frame intervals where the element exists through an array.

Another main key is the *frames* where all dynamic (temporal) information of the annotation are specified at frame level and each frame is indexed with an integer number. While in the *frame_intervals* key the array of frame intervals for which the whole json file data contains information, it is composed of a starting and ending frame number as a closed interval. Information can be included with the structure previously defined for elements. Frame properties could be:

- *timestamp*: it is the time instant the frame correspond to and can be a relative or an absolut time reference.
- *streams*: includes information for different streams of data collect with a multi-sensor setup, in fact inside this field it is possible to specify, for example, intrinsic calibration parameters and frequency and then it is possible to correlate the stream and the labeled element.

- *transforms*: inside this key the changes of position and orientation with respect to coordinate systems can be managed, it will be better explained below the document.

It is possible to synchronize the frames under the *master frame indexes* that group data from different stream by their timestamp and relate them to a specific instant or interval instead of classifying them under the stream specific frame index. Multiple streams of data could be collected with more than one sensor and with different frequency, in this case it is needed or a downsampling and the master frame indexing follows the slower one or it can follow the faster one. The annotations can be enriched with numerical properties of objects, for example the position, the size or other physical magnitudes, for this reason it is important to take into account the different coordinate systems the scenes could be referring to. In ASAM OpenLABEL the scenes are considered as Euclidean spaces and right-handed Cartesian coordinate systems that could be two or three dimensional. If any geometry is initially expressed with respect to a specific coordinate system, it can be changed thanks to transformation that are available inside the format. These coordinate systems are declared with a friendly name, used as an index and should be in the form of parent-child links to establish their hierarchy. Coordinate systems also have a *type*, they are:

- *scene_cs*: that is the static coordinate system.
- *local_cs*: indicates the coordinate system of a rigid body such as the vehicle carrying the sensors.
- *sensor_cs*: the coordinate system attached to the sensor.
- *custom_cs*: coordinate systems defined by the user.

Coordinate systems can have a *parent* which is the coordinate system they relate to and the list of the *children* coordinate system that refer to it, but also a *pose_wrt_parent* that contains the default or static pose of the specific children with respect to its parent. The *pose_wrt_parent* can be defined three ways:

- 4x4 homogeneous matrix;
- quaternion and translation vector;
- vector of Euler angles with sequence code, and translation vector;

Transforms between coordinate systems can also be defined for each frame with the key *transform* inside the *frame_properties* one and with the following properties: *src* and *dst*, these are the name of the source and destination coordinate systems, listed in the *coordinate_systems* field and another last key *transform_src_to_dst* containing the actual transform expressed in algebraic form. In ASAM OpenLABEL can be used geometric and non-geometric (generic) data types that allow labels and tags to represent any type of information. In this project 3D bounding boxes data type have been adopted in order to annotate the LiDAR point clouds. A 3D bounding box is a cuboid in three dimensional Euclidean space, defined by position, rotation, and size. Position and size are expressed as three elements vectors, while rotation can be defined in two alternative forms, using four elements vector quaternion notation or three element vector Euler notation (to be applied in ZYX order equivalent to yaw-pitch-roll order). The *cuboid* object can have the following keys: *attributes*, *coordinate_system* with respect to which it is expressed, *name* that is an index inside the corresponding object data pointers and *val* that contains an array of values that identify the position, rotation and dimensions of the bounding box. In the case that such rotation is expressed through quaternion notation, the array will have the following structure (x, y, z, qa, qb, qc, qd, sx, sy, and sz) better described in table 2.1 while if the cuboid is defined thanks to the Euler notation it is (x, y, z, rx, ry, rz, sx, sy, and sz) explained in table 2.2.

Attribute	**unit**	**Description**
x	m	Specifies the x-coordinate of the 3D position of the center of the cuboid.
y	m	Specifies the y-coordinate of the 3D position of the center of the cuboid.
z	m	Specifies the z-coordinate of the 3D position of the center of the cuboid.
qa		Specify the quaternion in non-unit form (x, y, z, and w) as in the SciPy convention.
qb		Specify the quaternion in non-unit form (x, y, z, and w) as in the SciPy convention.
qc		Specify the quaternion in non-unit form (x, y, z, and w) as in the SciPy convention.
qd		Specify the quaternion in non-unit form (x, y, z, and w) as in the SciPy convention.
sx	m	Specifies the x-dimension of the cuboid or the x-coordinate.
sy	m	Specifies the y-dimension of the cuboid or the y-coordinate.
sz	m	Specifies the z-dimension of the cuboid or the z-coordinate.

Table 2.1: Available attributes of a 3D bounding box (cuboid) using quaternion. The quaternions conform to the SciPy convention (Source [20]).

Attribute	**unit**	**Description**
x	m	Specifies the x-coordinate of the 3D position of the center of the cuboid.
y	m	Specifies the y-coordinate of the 3D position of the center of the cuboid.
z	m	Specifies the z-coordinate of the 3D position of the center of the cuboid.
rz	rad	Specify Euler angles, rz = yaw.
ry	rad	Specify Euler angles, ry = pitch.
rx		Specify Euler angles, rx = roll.
sx	m	Specify the quaternion in non-unit form (x, y, z, and w) as in the SciPy convention.
sx	m	Specifies the x-dimension of the cuboid or the x-coordinate.
sy	m	Specifies the y-dimension of the cuboid or the y-coordinate.
sz	m	Specifies the z-dimension of the cuboid or the z-coordinate.

Table 2.2: Available attributes of a 3D bounding box (cuboid) using Euler angles (Source [20]).

Chapter 3

Tool to generate OpenLABEL annotations

3.1 Overview

The designed solution is entirely shaped as a SaaS (Software as a Service), in fact, the user can interface with the service through a web application that has a custom frontend developed for this specific purpose. The service itself is hosted in a server that gives the computing capacity needed to execute the analysis and the frontend interacts with the backend thanks to a RESTful API collection that allows the client to use any of the developed services. The pipeline is composed by the first step where the users are able to upload their own datasets, collected on the road using a standard LIDAR sensor configuration and files and information are structured in a defined manner. Once the dataset is uploaded, the next input from the user is the choice of the annotation model he/she would like the annotation tool to adopt based on the required performances, at this point the user can make the process start, the algorithm will execute a data preparation in order to have the data ready to be given as input to the annotation model, that is the service that follows the data preparation. The Convolutional Neural Network will analyse the LIDAR point clouds given as an input and will perform the objects detection task based on the parameters given by the chosen model. Labels of the identified objects in the point clouds will be stored in a JSON file with a custom structure and format. When the custom labels are ready, the next service developed for this work of thesis will begin to convert these labels in an ASAM OpenLABEL compliant format. Once the converter has finished to operate, the user will be able to retrieve the ASAM OpenLABEL labels of its own dataset by downloading them from the user interface. Another provided service is the merging of a large number of point clouds in order to obtain a more defined and human understandable scene, these merged point clouds can be downloaded by the user in order to be visualized together with the labels. Also the visualizer has been developed during the realization of this project and it allows the user to interact with 3D scenes and bounding boxes but also the ground truth if available, for example the score of the labels can be changed and the scene can be navigated through.

3.2 Selected and Structured Input Data

The designed service takes as an input a structured dataset, in this part it will be described how the data should be collected, structured and enriched with information on the collection process. It has been taken as a model for the input dataset the one called NuScenes produced by Motional. NuScenes is one of the most consolidated and widely accepted and adopted public multimodal large scale datasets, this is mainly due its availability from its release in March 2019 but also because is one of the largest, most innovative and complete automotive datasets. NuScenes supports six tasks:

- *3D Object Detection* that consists in placing bounding boxes around objects that are grouped in ten categories: barrier, bicycle, bus, car, construction_vehicle, motorcycle, pedestrian, traffic_cone, trailer and truck;
- *3D Object Tracking* is the next step after detection where the aim is to track objects across time and are only taken into account seven categories because static objects such as barrier, traffic_cone and construction_vehicle are excluded;
- *Motion Prediction* is thought for the goal of predicting the trajectories of the ego vehicle with a series of x-y positions, each prediction lasts six seconds and must be sampled at 2 hertz.
- another task is to perform the *LiDAR Segmentation* that consists in predicting the category of every point in a set of 3D LiDAR point clouds. There are 10 foreground classes and 6 background classes for a total of 16 categories.
- the *Panoptic Segmentation and tracking* task aims to predict the semantic categories of every point, and additional instance IDs for things, it also

gives temporal coherence and pixel-level association over time. There are 10 thing classes and 6 stuff classes that means 16 categories.

Motional organizes Challenges where uesers can participate and challenge themselves to achieve the best performances on the NuScenes Tasks. Thanks to the popularity of this dataset, a lot of participants present their projects each time and they always improve the results. All these reason make NuScenes the perfect dataset to be adopted for this project.

3.2.1 Data Collection

From the official paper [21] we can extract some information. NuScenes data has been collected by driving through two cities: Boston (Seaport and South Boston) and Singapore (One North, Holland Village and Queenstown). They are both two cities with a dense traffic and highly challenging driving situations, but it has been taken into account also the diversity of the dataset across locations by spanning from left-hand to right-hand traffic, but also changing between different surrounding environments. The car used to realize the dataset are two electric Renault Zoe Supermini with the same sensor layout, compliant with ISO-8855 showed in 3.1 and composed by the sensor listed and described in 3.1.

Extrinsics and intrinsics of every sensor has been accurately calibrated in order to obtain high quality multi-sensor data. Taking as a reference the ego frame, placed in the midpoint of the rear vehicle axle, it has been possible to express the extrinsics of the sensors. In addition, for what concerns the synchronization, since the exposure of a camera is almost instantaneous, it is triggered when the top LiDAR sweeps across the center of the camera's FOV and the timestamp given to the image is the exposure trigger time while the timestamp of the LiDAR scan is the time when the full rotation of the current LiDAR frame is completed. The cameras' frame rate is reduced

Sensor	**Details**
6x Camera	RGB camera model Basler acA1600-60gc, 12Hz capture frequency, Evetar Lens N118B05518W F1.8 f5.5mm 1/1.8", 1/1.8" CMOS sensor of 1600 $\times$ 900 resolution, Bayer8 format for 1 byte per pixel encoding, 1600x900 ROI is cropped from the original resolution to reduce processing and transmission bandwidth, auto exposure with a maximum of 20 ms, JPEG compressed images. Front and side cameras have a 70° FOV with an offset of 55°, rear camera has a FOV 110°
1x Lidar	Spinning, model Velodyne HDL32E, 32 beams with 1080 ($\pm$10) points per ring, 20Hz capture frequency, 32 channels, 360° horizontal FOV, $-30°$ to 10° vertical FOV, $\leq 70m$ range, uniform azimuth angles 80m-100m range, $\pm 2cm$ accuracy, $\sim 1.39M$ points per second.
5x Radar	model Continental ARS 408-21, $\leq$250m range, 77GHz, FMCW, 13Hz capture frequency, $\pm$0.1km/h vel. accuracy.
GPS & IMU	GPS, IMU, AHRS. 0.2° heading, 0.1° roll/pitch, 20mm RTK positioning, 1000Hz update rate.

Table 3.1: NuScenes sensor data.

to 12Hz in order to keep the compute, bandwidth and storage requirement of the perception system low, but this means that not all LiDAR scans correspond to a camera frame because of LiDAR's higher frequency of 20Hz.

3.2.2 Data Format

All information about data such as calibration, maps, vehicle, coordinate and so on are stored inside a relational database with a specific schema defined by Motional. The database is composed of JSON tables where each object is identified by its unique primary key *token*.

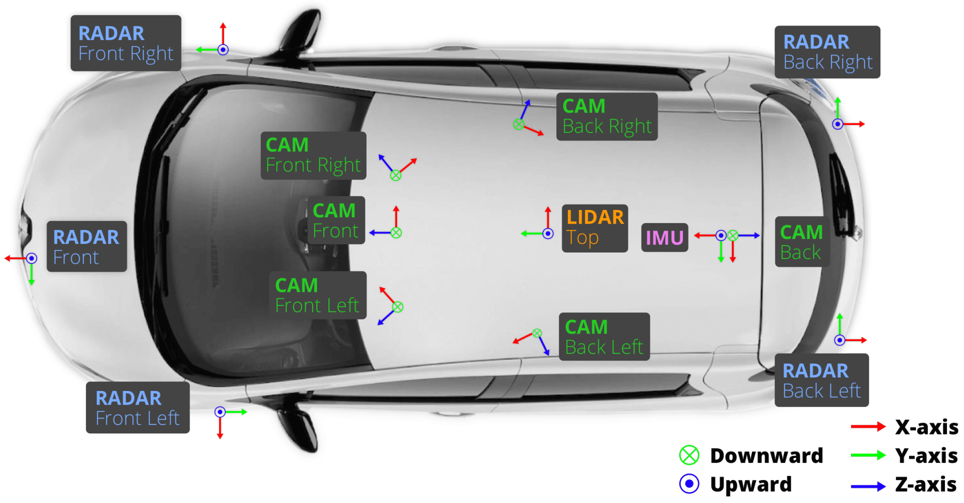

Figure 3.1: Car sensor setup (Source [[22]]).

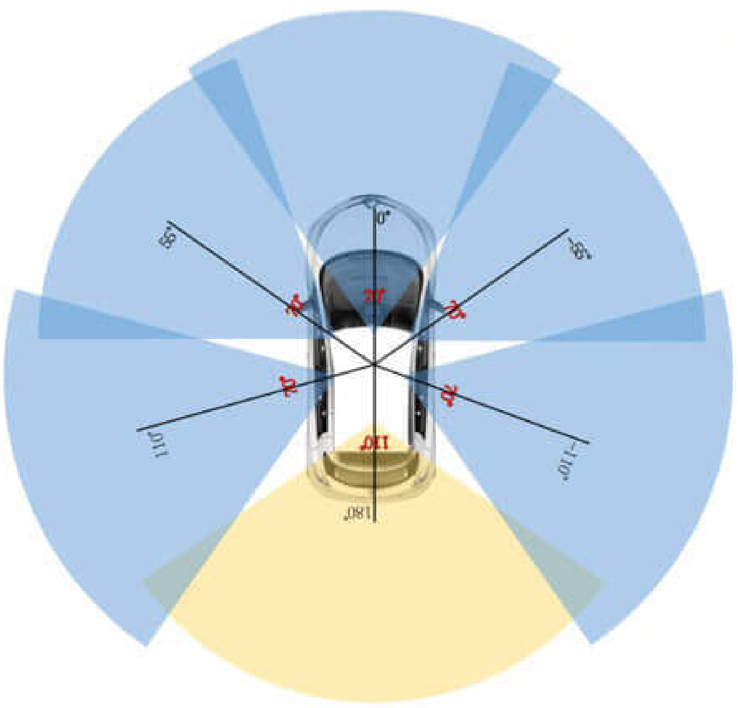

Figure 3.2: Camera orientation and overlap (Source [[22]]).

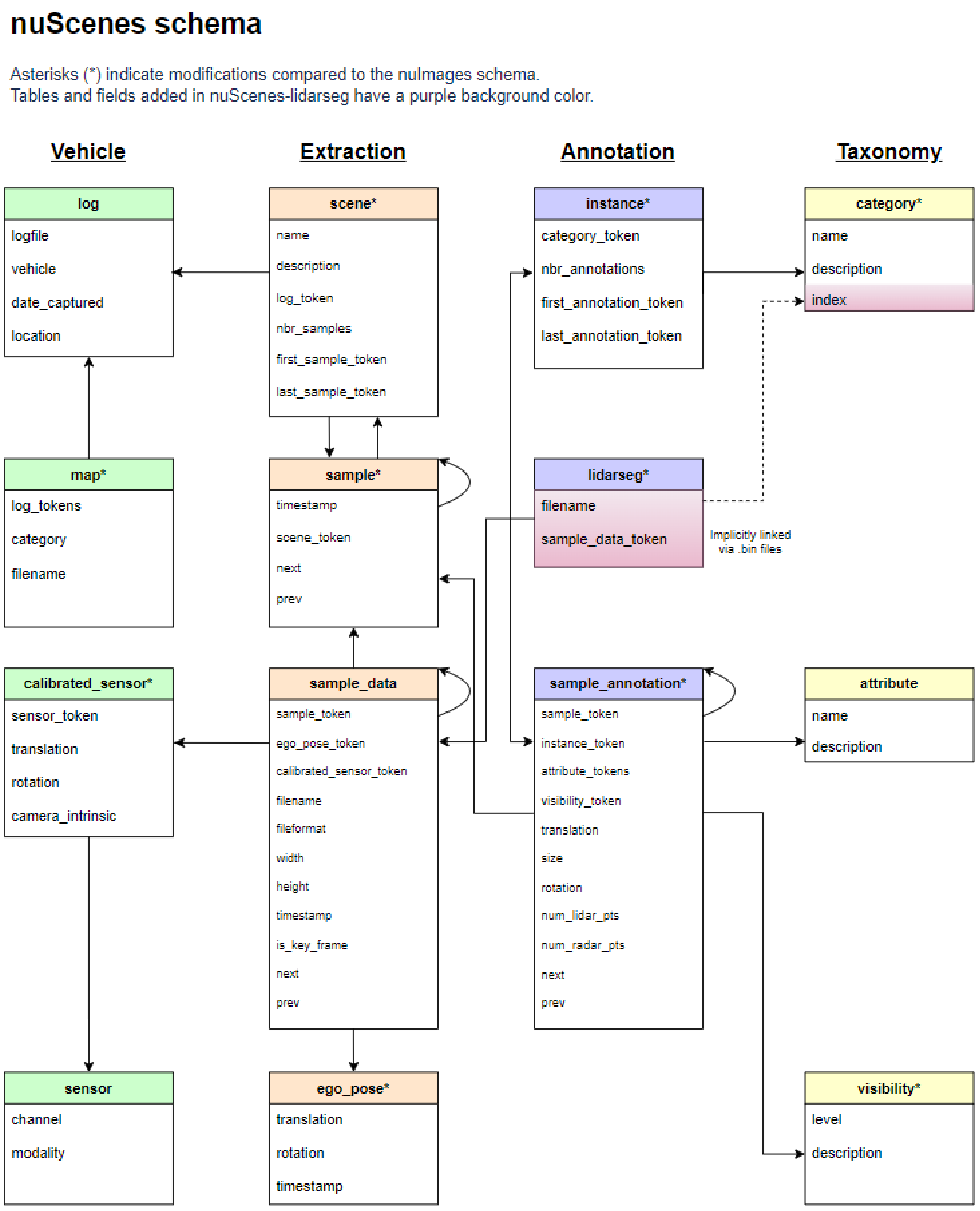

Figure 3.3: NuScenes Relational Database Schema (Source [[22]]).

For the purposes of this project, below will be described the parts of the database where the information about the **vehicle** and the data **extraction** are stored i.e. the database status before the annotation process. The table, containing vehicle's information, useful for submitting the custom dataset to the annotation service are:

- *sensor*: it groups all the sensor types with this object:

```
{
    "token": <str>, unique record identifier.
    "channel": <str>, Sensor channel name.
    "modality": <str>, (camera, lidar, radar)
}

```

- *calibrated_sensor*: inside this table are listed the parameters resulted from the calibration process of a sensor between lidar, radar and camera, with respect to the ego vehicle body frame (camera images should be undistorted and rectified) with objects like the next one:

```
{
   "token": <str>, unique record identifier.
   "sensor_token": <str>, identifier pointing to
    the sensor type.
   "translation": <float> [3], coordinate system
    origin in meters: x, y, z.
   "rotation": <float> [4], coordinate system
  orientation as quaternion: w, x, y, z.
  "camera_intrinsic": <float> [3, 3], intrinsic
   camera calibration matrix, it is empty for
  sensors that are not cameras.
}

```

- *map*: also map data can be added to the database, stored as binary semantic masks from a top-down view and they are identified with a specific object, as follows:

```
{
  "token": <str>, unique record identifier.
  "log_tokens": <str> [n], identifiers pointing
   to the logs.
  "category": <str>, map category (
  semantic_prior for drivable surface and
  sidewalk).
  "filename": <str>, relative path to the file
  with the map mask.
}

```

- *log*: contains information about the log that originated the related data:

```
{
  "token": <str>, unique record identifier.
  "logfile": <str>, log file name.
  "vehicle": <str>, vehicle name.
```

```
   "date_captured": <str>, date (YYYY-MM-DD).
   "location": <str>, area where log was
  captured.
}

```

While tables related to the data extraction that must be included in the submitted database are:

- *ego_pose*: it is the ego vehicle pose at a particular timestamp, it is expressed with respect to the global coordinate system of the map. The z-component of the translation key is always equal to 0 since the localization is expressed on a 2D x-y plane overlying the map.

```
{
   "token": <str>, unique record identifier.
   "translation": <float> [3], coordinate system
    origin in meters: x, y, z=0.
   "rotation": <float> [4], coordinate system
  orientation as quaternion: w, x, y, z.
   "timestamp": <int>, unix time stamp.
}

```

- *sample_data*: are the data about a sample returned by cameras, lidar or radar, if the related sample is a key frame the timestamps are very close, while if it is not a key frame the timestamps are the closest:

```
{
   "token": <str>, unique record identifier.
   "sample_token": <str>, identifier pointing to
    the sample to which this sample_data is
  associated.
   "ego_pose_token": <str>, identifier pointing
  to the ego_pose.
   "calibrated_sensor_token": <str>, identifier
  pointing to the calibrated_sensor.
   "filename": <str>, relative path to data-blob
    on disk.
   "fileformat": <str>, data file format.
   "width": <int>, for images it is the width in
    pixels.
   "height": <int>, for images it is the height
  in pixels.
   "timestamp": <int>, Unix time stamp.
   "is_key_frame": <bool>, True if sample_data
  is part of key_frame, else False.
   "next": <str>, identifier pointing to the
  sample data from the same sensor that follows
  this in time. Empty if end of scene.
   "prev": <str>, identifier pointing to the
  sample data from the same sensor that precedes
    this in time. Empty if start of scene.
}

```

- *sample*: the sample is a keyframe that will be annotated, often result of a downsampling to match the sample of all sensors and reduce the amount of data to be annotated without a significative loss of information. For NuScenes samples have a 2Hz frequency with approximately the same timestamp as part of a single LIDAR sweep.

```
{
   "token":<str>, unique record identifier.
   "timestamp":<int>, unix time stamp.
   "scene_token":<str>, identifier pointing to
  the scene.
   "next":<str>, identifier pointing to the
  sample that follows this in time. Empty if end
   of scene.
   "prev":<str>, identifier pointing to the
  sample that precedes this in time. Empty if
  start of scene.
}

```

NuScenes also provides samples grouped in 1000 scenes, i.e. 20s log sequence of consecutive frames extracted from a log, where object identities are preserved for the whole duration of the scene and not across scenes. Scenes are not mandatory for the submission to the developed service but can be useful for future development. The *scene* object is structured as follows

```
{
   "token": <str>, unique record identifier.
   "name":  <str>, short string identifier.
```

```
    "description": <str>, longger description of the
    scene.
    "log_token": <str>, identifier pointing to log
    from where the data was extracted.
    "nbr_samples": <int>, number of samples in this
    scene.
    "first_sample_token": <str>, identifier pointing
    to the first sample in scene.
    "last_sample_token": <str>, points to the last
    sample in scene.
}

```

Data must be wrapped inside a unique compressed folder, grouped in different folders as follows:

- **sweeps**: it contains all the collected data file, grouped by sensor in different folders named in relation with the sensor name:
  - *CAM_BACK*
  - *CAM_BACK_LEFT*
  - *CAM_BACK_RIGHT*
  - *CAM_FRONT*
  - *CAM_FRONT_LEFT*
  - *CAM_FRONT_RIGHT*
  - *LIDAR_TOP*
  - *RADAR_BACK_LEFT*
  - *RADAR_BACK_RIGHT*
  - *RADAR_FRONT*

  - *RADAR_FRONT_LEFT*
  - *RADAR_FRONT_RIGHT*

  it is not mandatory for the developed service.

- **samples**: it is the result of the downsampling of the *sweeps* folder and has the same structure. It is mandatory for the developed service, in particular the data files contained in the folder *LIDAR_TOP* are used for the annotation and together with the files in *CAM_FRONT* they are taken as input in order to be visualized together with the annotations.
- **v1.0-mini**: is the folder of the relational database that stores the JSON tables named and structured as previously described.
- **maps**: if available, there are stored the maps and it is not mandatory.

3.3 Annotation tool

The annotation is thought to be fully automated and this goal is achieved making use of **MMDetection3D**, an open source object detection toolbox based on **PyTorch**, the popular machine learning framework based itself on the **Torch library**. OpenMMLab released MMDetection3D for the first time in July 2020 due to the absence of a universal codebase in 3D object detection. The toolbox has been chosen not only because it is an open source project but also because it represents one of the most powerful tool of its kind, in addition it is very well documented and it supports state-of-the-art single-modality/multi-modality 3D object detectors such as MVXNet, VoteNet, PointPillars and so on, in indoor/outdoor datasets, including ScanNet, SUNRGB-D, Waymo, nuScenes, Lyft, and KITTI and many more. It is proved that it trains faster than other codebases such as OpenPCDet, votenet or Det3D. Thanks to MMDetection3D it is possible to implement an inference of the model PointPillars. It is a deep neural network and a fast encoder for object detection from 3D point clouds. The differencies of detecting object from lidar point clouds, instead of using images are that the former is three-dimensional and a sparse representation while the latter is two-dimensional and dense, for this reason the object detection must move away from the classical image convolutional pipelines while taking into account different approaches. In fact, techniques that project a 3D point cloud to a 2D image are fast and have high reference resources but are characterized by a very low accuracy, instead, methods that performs the feature extraction directly using the 3D point clouds have a relatively high accuracy but the inference is slow. PointPillars revolutionized this rule, since, when it was developed, other models were designed to extract the features directly from the 3D point cloud at a maximum of 30Hz inference speed while it can reach 62Hz. The secret of the model high speed is removing the expensive 3D convolutional layer and opting for dense 2D convolutions on pillars. As presented in [23], PointPillars main concept is to see the point cloud as a group of pillar and it can be divided in three major parts as also showed in figure 3.4:

- **Pillar Feature Net**: considering that each point in the cloud is characterized by the coordinates x, y, z and reflectance r, the first part consists in a conversion of the point cloud into a pseudo-image by discretizing the point cloud into an evenly spaced grid in the $x - y$ plane, as a result a set of pillar is created. Then, all the points are enriched with the x_c, y_c, z_c distances to the arithmetic mean of all points in the pillar and with the couple x_p and y_p that are the offset from the pillar x, y center. Each augmented lidar point is now $D = 9$ dimensional. In addition, in order to take into account the sparsity of the point cloud, it is imposed a limit both on the number of non-empty pillars per sample (P) and on the number of points per pillar (N) to create a dense tensor of size (D, P, N). In the case that a single sample or a pillar holds too much data to be represented in this tensor it is randomly sampled, on the other hand if data are not enough a zero padding is applied. At this point a simplified version of the neural network PointNet is used to apply a linear layer to each point i.e. a 1x1 convolution across the tensor that is a very efficient computation, followed by Batch Norm and ReLU to generate a (C, P, N) sized tensor that after a max operation results in a (C, P) sized tensor. After the encoding process, the feature are distributed with the same location of the original pillar in order to originate a pseudo-image of size (C, H, W).

- **Backbone**: the backbone is composed of two sub-networks: one top-down network that generate features at increasingly small spatial resolution and a second network that applies an upsampling and a concatenation of the top-down features. The top-down backbone can be seen as a series of blocks $Block(S, L, F)$, each one running at a different stride S. L is the number of 3x3 2D conv-layers and F the output channels each followed by BatchNorm and a ReLU. In order to ensure the block operates at stride S after the input blob of stride S_{in}, the first convolution inside the layer has stride S/S_{in}, while all next convolutions in the block have stride 1. Before applying again BatchNorm and ReLU, features are upsampled, $Up(S_{in}, S_{out}, F)$ from S_{in} to S_{out} thanks to a transposed 2D convolution with F final features. The result are features that are concatenation of all features produced at different strides.

- **Detection Hand**: in the phase of 3D object detection, the Single Shot Detector (SSD) setup is adopted in order to match the priorboxes to the ground truth with 2D Intersection over Union (IoU). Having a 2D match, the height and elevation become additional regression targets.

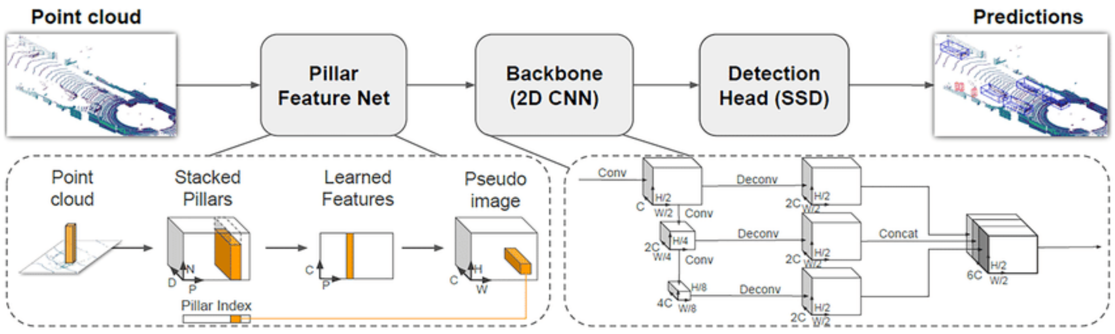

Figure 3.4: PointPillars Network Structure (Source [23]).

3.4 Converter

The output of MMDetection3D is annotations in a format compliant with the one defined by NuScenes for the submission of Object Detection task. For this task NuScenes provides the annotations for the training and validation data sets but not for the test one. In order to get the annotation at inference time it is possible to make use of a maximum of 6 past camera images, 6 past radar sweeps and 10 past lidar sweeps, that result in a window of approximately 0.5s. The annotation resulted from the detection are stored in a database that consists in a single JSON file called results_nusc. It has the following keys:

- the key *meta* has a corresponding value that is an object itself composed by five keys with a boolean value based on whether the data from the specific sensor in the key have been used for the object detection.

```
    {
        "use_camera":   <bool>
        "use_lidar":    <bool>
        "use_radar":    <bool>
        "use_map":      <bool>
        "use_external": <bool>
    }

```

- the value of the other key, *results*, that mirrors the sample_annotation table inside the NuScenes database, is another object with samples tokens as keys, each key correspond to a sample and its value is a list of objects composed as follows:

```
{
    "sample_token": <str>, identifies the sample/keyframe for which objects are detected.
    "translation": <float> [3], estimated bounding box location in meters, a list of x-y-z coordinates of the cuboid center in the global frame.
    "size": <float> [3], estimated bounding box size in meters: width, length, height.
    "rotation": <float> [4], Estimated bounding box orientation as quaternion in the global frame: w, x, y, z.
    "velocity": <float> [2], estimated bounding box velocity in m/s in the global frame: vx, vy.
    "detection_name": <str>, the predicted class for this sample_result, e.g. car, pedestrian.
    "detection_score": <float>, object prediction score between 0 and 1 for the class identified by detection_name.
    "attribute_name": <str>, name of the predicted attribute or empty string for classes without attributes.
}

```

for what concerns the attribute_name, it is ignored for classes without attributes. There are a few cases (0.4%) where attributes are missing also for classes that should have them. We ignore the predicted attributes for these cases.

Detection challenges take into account ten different detection classes, they are:

- *car* with the associated attributes *vehicle.{moving, parked, stopped}* and detection range equal to equal to 50m.
- *bus* with the associated attributes *vehicle.{moving, parked, stopped}* and detection range equal to 50m.
- *bicycle* with the associated attributes *cycle.{with_rider, without_rider}* and detection range equal to 40m.
- *barrier* with a detection range equal to 30m.
- *construction_vehicle* with the associated attributes *vehicle.{moving, parked, stopped}* and detection range equal to 50m.
- *motorcycle* with the associated attributes *cycle.{with_rider, without_rider}* and detection range equal to 40m.
- *pedestrian* with the associated attributes *pedestrian.{moving, standing, sitting_lying_down}* and detection range equal to 40m.
- *traffic_cone* with a detection range equal to 30m.
- *trailer* with the associated attributes *vehicle.{moving, parked, stopped}* and detection range equal to 50m.
- *truck* with the associated attributes *vehicle.{moving, parked, stopped}* and detection range equal to 50m.

The conversion algorithm is entirely written in Python language making use of the NuScenes devkit, in order to easily navigate the database and the prediction resulted from the detection task, also the *vcd* library developed by the ASAM member organization called Vicomtech. It is a very useful tool to instantiate the OpenLABEL objects and wrap information within them. The converter takes as inputs the database structured as previously explained and it retrieve all the information from the collection process stored in the relational database, such as the ego pose, timestamps, logs, sensor information, location, samples file name and they are grouped in order to be organized following the OpenLABEL guidelines. Moreover, the coordinate systems are added to the OpenLABEL file with their parents, children and pose with respect to parent, the coordinate systems of the maps are added as *scene_cs* with vehicles as children, vehicles reference systems are added as *local_cs* with map as parent and sensors as children, sensors are added as *sensor_cs* with vehicle as parent. In order to make the database as standardized as possible, the pose with respect to the parent is expressed as a 4x4 spatial transformation matrix that represent rotation and translation of the reference system with respect to the parent, because it is a more popular formality. The transformation from the quaternion used by NuScenes to the 4x4 transformation matrix is the following:

$$R = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y+q_zq_w) & 2(q_xq_z-q_yq_w) \\ 2(q_xq_y-q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z+q_xq_w) \\ 2(q_xq_z+q_yq_w) & 2(q_yq_z-q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix}$$

Where $\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$

All the streams such as GPS/IMU, lidar, radar and camera are added, in particular the cameras streams are reported with the pinhole camera intrinsic properties through an intrinsics matrix K defined below, sensors also come with a short description.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where f_x and f_y are the focal lengths expressed in pixels while c_x and c_y are the principal components offset. At this point each frame is identified with ordered integers starting from 0, based on where the frame is located in time and cuboids can be added to the frame they refer to. The name given to each cuboid is composed by the class of the detected object plus the integer n for the $n-th$ object detected for its class in the whole dataset, for example if a car is detected and it is the third until that moment, the name of such object will be "*car3*". Another important information reported in the ASAM OpenLABEL file is the name of the coordinate frame that the cuboid translation and orientation are expressed with respect to. Since each annotation is made with respect to the global reference frame, all the cuboids will now refer to the coordinate system of the a specific map. At the end, the coordinates of the cuboid's center point, its orientation as a quaternion and the width, length and height in meters are taken from the annotation where they belong to different keys and are stacked in a single array, compliant to ASAM OpenLABEL. The final task of the code is to enrich the cuboids with the detection score and the attribute in the *object_data* object and the conversion is completed.

Chapter 4

Results

The architecture of the developed system is composed by multiple micro-services that run in a local machine. Thanks to the Flask web micro-framework, a RESTful set of APIs have been exposed in order to access the micro-services from the front-end through the HyperText Transfer Protocol. The back-end is composed by five requests:

- a POST request used to upload the custom dataset to be annotated.
- a POST request that when called make the annotation of the dataset start.
- a POST request useful to execute the conversion process and generate the ASAM OpenLABEL file.
- a GET request to retrieve the OpenLABEL compliant annotation in a .zip file.
- a GET request to download the merged point clouds.

Each request sends back a response body in JSON format containing four key-value pairs, a boolean one if the process succeed or failed, two for the timestamps when the request has been received and the other when it is sent back, one last key contains the output file of the process encoded in base64 if the method generates one, otherwise it is "*None*".

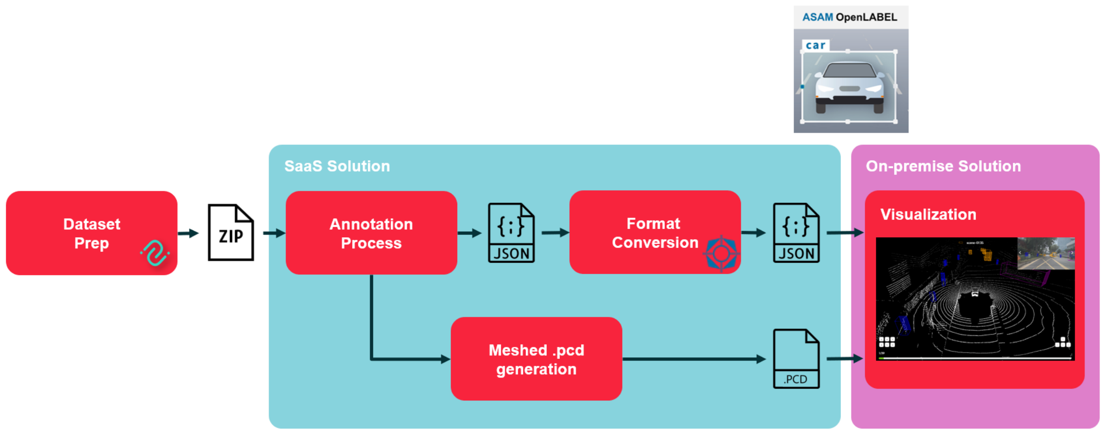

Figure 4.1: Structure of the developed system.

In order to understand the performances of PointPillars, the model *"point pillars_hv_fpn_sbn-all_8xb4-2x_nus-3d.py"* present in the MMDetection3D framework trained by OpenMMLab, it has been evaluated with 8 gpus using the available checkpoint *"hv_pointpillars_fpn_sbn-all_4x8_2x_nus-3d_202 00620_230405-2fa62f3d.pth"* in the MMDetection3D repository. Motional uses some metrics in order to evaluate the models, the metrics are listed and described below. Firstly the well known **mean Average Precision mAP** from the **AP** that in this case is calculated by defining a match considering the 2D bounding boxes center distance d on the ground plane rather than intersection over union (IOU), both in order to decouple detection from object size and orientation and because if object with small footprints (for example pedestrians and bikes) are detected with small translation error the IOU is 0. This approach is used to not penalize vision-only methods which often have large localization error. Four thresholds for center-distance are fixed and they are $\mathbb{D} = \{0.5, 1, 2, 3\}$ meters. For each threshold the Average Precision is calculated by taking as a match the predictions with the ground truth objects that have the smallest center-distance under the specific threshold, moreover, only recalls and precisions greater than 0.1 are considered. The Average precision is obtained by integrating the precision-recall curve varying the confidence score, where the precision p is the number of the correct predictions (true positive TP) over the total predictions (True Positive + False Positive $TP + FP$), and the recall r is the ratio between the true positive an the total number of ground truth (True Positive + False Negative $TP + FN$). At the end AP is averaged over all the match thresholds and the mean is computed across all the classes $\mathbb{C}$.

$$mAP = \frac{1}{|\mathbb{C}||\mathbb{D}|} \sum_{c \in \mathbb{C}} \sum_{d \in \mathbb{D}} AP_{c,d}$$

In addition, a set of True Positive metrics are defined, they are all based on prediction matched with a ground truth box using $d = 2m$ center distance. Tp errors are:

- **Average Translation Error (ATE)**: it is the Euclidean center distance in 2D expressed as meters.
- **Average Scale Error (ASE)** is the 3D intersection over union (IOU) after aligning orientation and translation (1-IOU).
- **Average Orientation Error (AOE)** is the smallest yaw angle difference between prediction and ground truth, expressed in radians and measured on a full 360° period except for barriers where it is 180°.
- **Average Velocity Error (AVE)** is the absolute velocity error, calculated as the L2 norm of the velocity differences in 2D expressed as $\frac{m}{s}$.
- **Average Attribute Error (AAE)** is defined as (1 - acc) where $acc =$ *attribute classification accuracy.*

For each TP metric the mean (mTP) is computed over all classes:

$$mTP = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} TP_c$$

AVE and AAE are not calculated for cones and barrier because respectively they are stationary and there are no attributes for them. Since cones do not have a well defined orientation neither their AOE can be obtained. NuScenes also define its custom detection score (NDS) in order to take into account the detection performance and the quality in terms of box location, size, orientation, attributes and velocity. It is defined as below:

$$NDS = \frac{1}{10}[5mAP + \sum_{mTP \in \mathbb{TP}} (1 - min(1, mTP))]$$

mTP is bounded between 0 and 1 since mAVE, mAOE and mATE can be greater than 1. As reported in [24] the performances with 8 GPUs are:

Object Calss	**AP**	**ATE**	**ASE**	**AOE**	**AVE**	**AAE**
car	0.503	0.577	0.152	0.111	2.096	0.136
truck	0.22	0.857	0.224	0.220	1.389	0.179
bus	0.294	0.855	0.204	0.190	2.689	0.283
trailer	0.081	1.094	0.243	0.553	0.742	0.167
construction_vehicle	0.058	1.017	0.450	1.019	0.137	0.341
pedestrian	0.392	0.687	0.284	0.694	0.876	0.158
motorcycle	0.317	0.737	0.265	0.580	2.033	0.104
bicycle	0.308	0.704	0.299	0.892	0.683	0.010
traffic_cone	0.555	0.486	0.309	nan	nan	nan
barrier	0.466	0.581	0.269	0.169	nan	nan

Table 4.1: Evaluation results of an inference of PointPillars with MMDetection3D (Source [24]).

The system has been tested on the "mini" version of the NuScenes dataset. Once the annotations are generated in the NuScenes compliant format they are successfully and efficiently created following the ASAM OpenLABEL format. In the table 4.3 are listed the object classes with the total number

mAP:	0.3197
mATE:	0.7595
mASE:	0.2700
mAOE:	0.4918
mAVE:	1.3307
mAAE:	0.1724
NDS:	0.3905
Eval time:	170.8s

Table 4.2: Metrics mean (Source [24]).

of boxes generated over all the samples and converted to ASAM OpenLabel format. These labels can be downloaded and it has been developed a visualizer for such labels in order to understand the correctness of the annotations.

classes	number of boxes
car	2728
truck	2035
trailer	225
bus	126
construction_vehicle:	611
bicycle	678
motorcycle	367
pedestrian	2316
pedestrian	2316
traffic_cone	478
barrier	146

Table 4.3: Number of bounding boxes per class.

The visualizer is developed using the language Python and the library Open3D. Thanks to this application it is possible to navigate through the point clouds, enriched with images from all camera in order to better understand the surroundings, in addition it is possible to show only the generated bounding boxes or if available also the ground truth, at the end the user can filter the bounding boxes to be displayed by their detection score. As it can be seen in figure 4.2, bounding boxes maintain their orientation, location and size during the conversion.

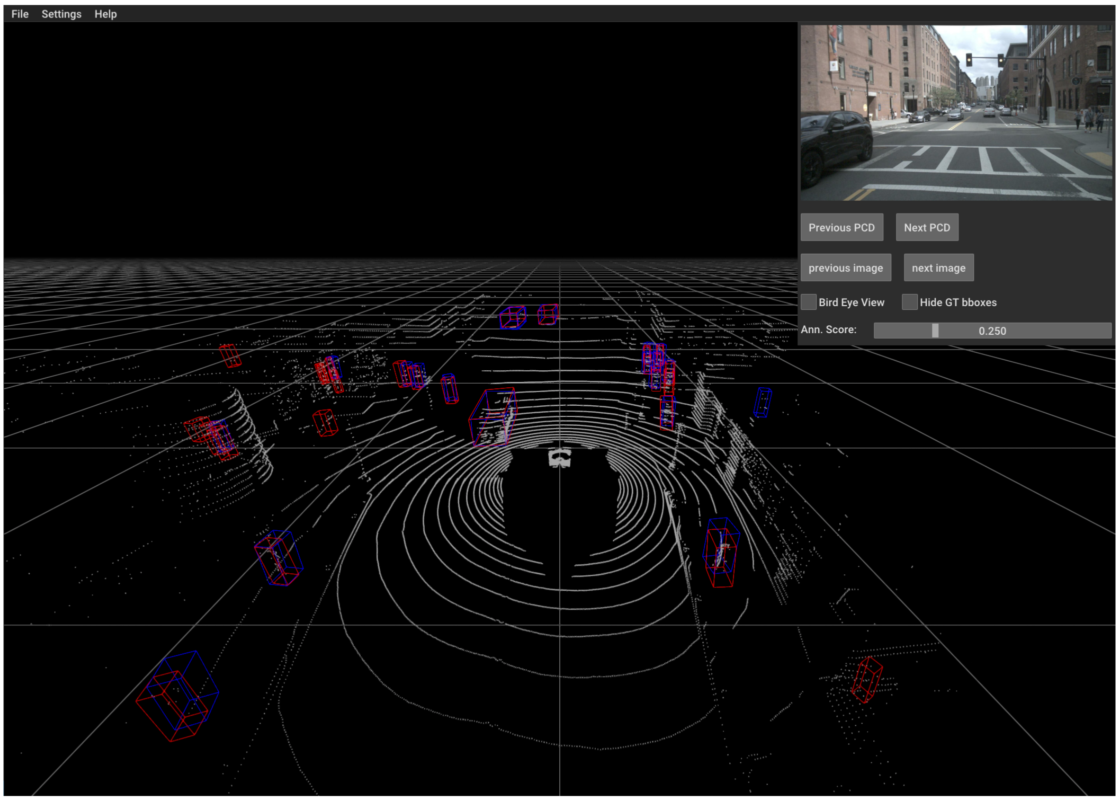

Figure 4.2: Visualizer

In order to obtain a real software as a service, Reply also provided a user interface thanks to a web application developed using HTML, CSS and JavaScript, in order to make the user capable of accessing the offered service in a simple way, making it easier to grow in popularity between organizations and reducing even more the time needed to interface with the system.

Conclusion and Further Development

Conclusion

In conclusion the tool can be used to automatically generate labels for a custom dataset, followed by a manual check that integrates the already present OpenLABEL structure, it extremely improves the annotation process, making it faster and more efficient, in fact, a standardized format results in a minimal learning curve for annotators that means reducing cost and effort. The conversion makes annotation output compliant with a huge number of tools and applications, for example participant companies are Deepen AI, Kognic, Ansys Inc., Five, Peak Solution GmbH, Tata Consultancy Services Pvt. Ltd, understandAI GmbH, Vicomtech and so on. The system developed for this project become very useful when it comes to validate ADAS/Automated Driving Systems through a simulated environment. In fact, it helps in building a realistic ground truth to be reproduced in a virtual environment, ensuring a faithful digital twin of the real world. Generated labels could be also taken as input for a synthetic scenarios generator, dramatically increasing the quantity and the quality of automotive annotated datasets. New and better annotation models can be trained with a huge amount of labeled data, thanks to the partial automation and the standardization of the labels. At the ASAM International Conference 2022, TATA Consultancy Services estimated a 62.4% of reduction in time if the annotation process is supported by AI, automation and standardization. This is an interesting starting point to reach the fully automated annotation and standardization.

Further Development

The project previously described can be seen as a starting point for many further developments. For the first micro-service, the annotation tool, it can evolve towards better models since, day by day, teams of researchers develop different architecture and improve existing models, obtaining always better performances. It can be implemented a model that combines both lidar point clouds and camera images, for example choosing the Bird's-Eye-View (BEV) models that increase the robustness but need the support of powerful hardware. For what concerns the converter micro-service, it can be added the functionality of converting **actions** and **events** into the OpenLABEL file, while maintaining it updated with the additional information introduced by the new adopted annotation models. In order to make the process more reliable and ensure the compatibility with other tools, the next step can be the development of an OpenLABEL format validation software that would helps even with the debugging process. For the user interaction point of view, a visualizer integrated in the web application would be an improvement but also a page where metrics and data are showed, evaluated and described.

Bibliography

[1] Santokh Singh. *Critical reasons for crashes investigated in the national motor vehicle crash causation survey.* Tech. rep. 2015 (cit. on p. 1).

[2] Witold Pawłowski, Krzysztof Goniewicz, David C Schwebel, Jiabin Shen, and Mariusz Goniewicz. «Road traffic injuries in Poland: Magnitude and risk factors». In: *European journal of trauma and emergency surgery* 45 (2019), pp. 815–820 (cit. on p. 1).

[3] Travis J Crayton and Benjamin Mason Meier. «Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy». In: *Journal of Transport & Health* 6 (2017), pp. 245–252 (cit. on p. 1).

[4] SAE On-Road Automated Vehicle Standards Committee et al. «Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles». In: *SAE International: Warrendale, PA, USA* (2021) (cit. on p. 3).

[5] Rami Debouk. «Overview of the Second Edition of ISO 26262: Functional Safety—Road Vehicles». In: *Journal of System Safety* 55.1 (2019), pp. 13–21 (cit. on pp. 11, 13, 14).

[6] Kent Beck et al. «Manifesto for agile software development». In: (2001) (cit. on p. 15).

[7] Lıga Bormane, Jūlija Gržibovska, Solvita Bērziša, and Jānis Grabis. «Impact of requirements elicitation processes on success of information system development projects». In: *Information Technology and Management Science* 19.1 (2016), pp. 57–64 (cit. on p. 17).

[8] Mike McCormick. «Waterfall vs. Agile methodology». In: *MPCS, N/A* 3 (2012) (cit. on pp. 18, 19).

[9] Samar Al-Saqqa, Samer Sawalha, and Hiba AbdelNabi. «Agile software development: Methodologies and trends.» In: *International Journal of Interactive Mobile Technologies* 14.11 (2020) (cit. on p. 20).

[10] Blend Berisha, Endrit Mëziu, and Isak Shabani. «Big data analytics in Cloud computing: an overview». In: *Journal of Cloud Computing* 11.1 (2022), p. 24 (cit. on p. 22).

[11] Erik Brynjolfsson, Lorin M Hitt, and Heekyung Hellen Kim. «Strength in numbers: How does data-driven decisionmaking affect firm performance?» In: *Available at SSRN 1819486* (2011) (cit. on p. 22).

[12] Dániel Fényes, Balázs Németh, and Péter Gáspár. «A novel data-driven modeling and control design method for autonomous vehicles». In: *Energies* 14.2 (2021), p. 517 (cit. on p. 23).

[13] dSPACE. URL: `https://www.dspace.com/en/inc/home/applicationfields/comp/data_driven_development.cfm` (visited on 10/12/2023) (cit. on p. 23).

[14] Alexandra L'heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. «Machine learning with big data: Challenges and approaches». In: *Ieee Access* 5 (2017), pp. 7776–7797 (cit. on p. 24).

[15] Philip Koopman and Michael Wagner. «Challenges in autonomous vehicle testing and validation». In: *SAE International Journal of Transportation Safety* 4.1 (2016), pp. 15–24 (cit. on p. 27).

[16] Nidhi Kalra and Susan M Paddock. «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 (cit. on pp. 27, 28).

[17] Riccardo Donà and Biagio Ciuffo. «Virtual testing of automated driving systems. A survey on validation methods». In: *IEEE Access* 10 (2022), pp. 24349–24367 (cit. on p. 31).

[18] ASAM e.V. URL: `https://www.asam.net/` (visited on 10/12/2023) (cit. on p. 33).

[19] Marius Dupuis and Dr. Klaus Estenfeld. «Standards for simulation and testing: Covering the real and the virtual world». In: Oct 11th - 13th 2022 (cit. on p. 38).

[20] ASAM e.V. URL: `https://www.asam.net/index.php?eID=dumpFile&t=f&f=4566&token=9d976f840af04adee33b9f85aa3c22f2de4968dd#top-1b636ab4-0871-4afc-ac24-758cd64f39fa` (visited on 10/14/2023) (cit. on pp. 48, 49, 51, 56, 57).

[21] Holger Caesar et al. «nuScenes: A multimodal dataset for autonomous driving». In: *arXiv preprint arXiv:1903.11027* (2019) (cit. on p. 61).

[22] nuScenes. URL: `https://www.nuscenes.org/nuscenes` (visited on 11/13/2023) (cit. on pp. 63, 64).

[23] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. «Pointpillars: Fast encoders for object detection from point clouds». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2019, pp. 12697–12705 (cit. on pp. 73, 74).

[24] OpenMMLab. URL: `https://mmdetection3d.readthedocs.io/en/latest/advanced_guides/datasets/nuscenes.html` (visited on 11/27/2023) (cit. on pp. 83, 84).