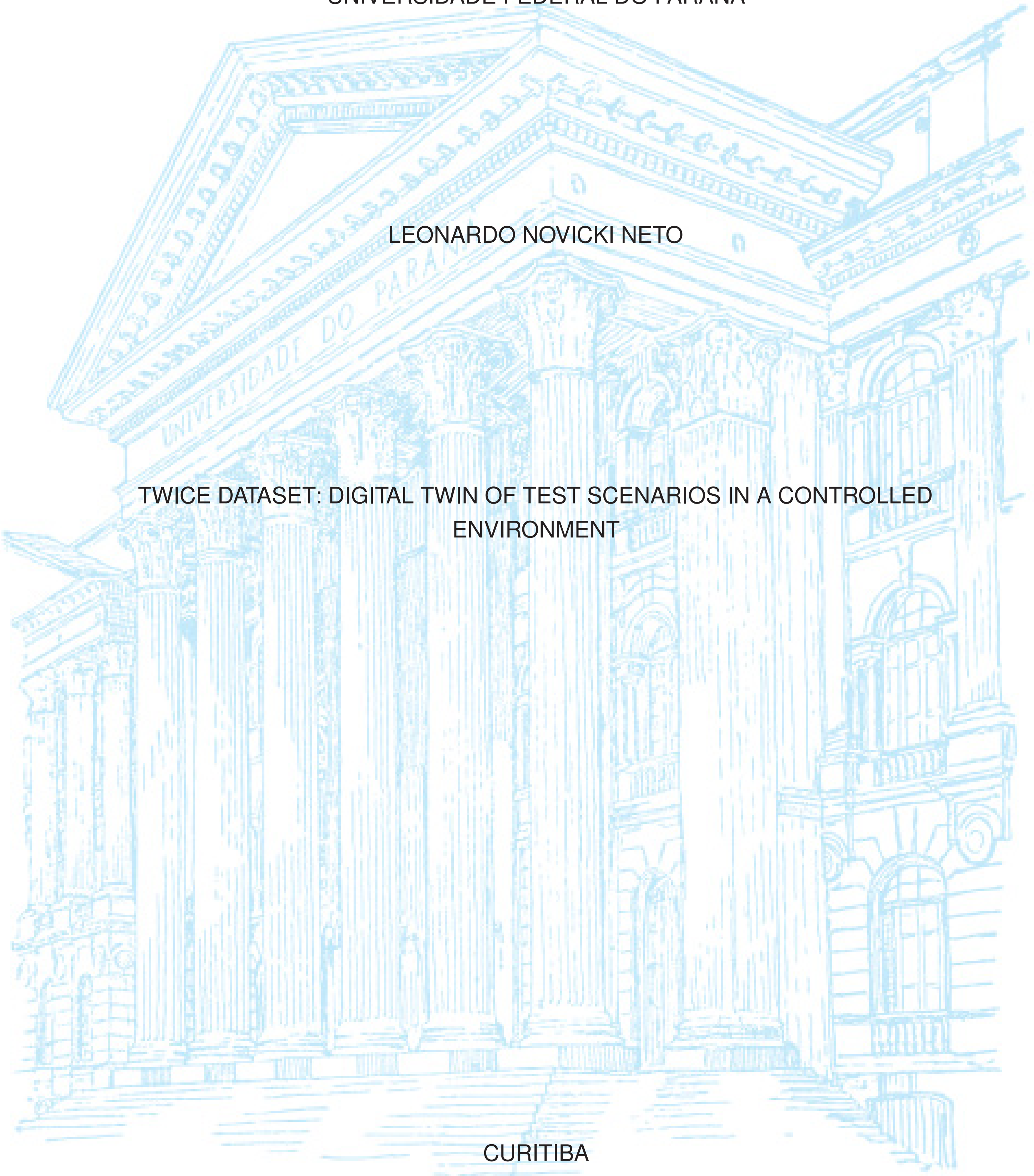


UNIVERSIDADE FEDERAL DO PARANÁ

LEONARDO NOVICKI NETO

TWICE DATASET: DIGITAL TWIN OF TEST SCENARIOS IN A CONTROLLED ENVIRONMENT

CURITIBA

2023

LEONARDO NOVICKI NETO

TWICE DATASET: DIGITAL TWIN OF TEST SCENARIOS IN A CONTROLLED ENVIRONMENT

Dissertação apresentada ao Programa de Pós Graduação em Engenharia Elétrica, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Eduardo Parente Ribeiro
Coorientador: Prof. Dr.-Ing. Werner Huber

CURITIBA

2023

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Novicki Neto, Leonardo
Twice dataset: digital twin of test scenarios in a controlled environment / Leonardo Novicki Neto. – Curitiba, 2023.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica.

Orientador: Eduardo Parente Ribeiro
Coorientador: Werner Huber

1. Radar. 2. Direção de automóveis. 3. Indústria automobilística. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Engenharia Elétrica. III. Ribeiro, Eduardo Parente. IV. Huber, Werner. V. Título.

Bibliotecário: Leticia Priscila Azevedo de Sousa CRB-9/2029

MINISTÉRIO DA EDUCAÇÃO
SETOR DE TECNOLOGIA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENGENHARIA ELÉTRICA - 40001016043P4

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **LEONARDO NOVICKI NETO** intitulada: **TWICE Dataset: Digital Twin of test scenarios in a controlled environment**, sob orientação do Prof. Dr. EDUARDO PARENTE RIBEIRO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 24 de Novembro de 2023.

Assinatura Eletrônica
28/11/2023 13:15:04.0
EDUARDO PARENTE RIBEIRO
Presidente da Banca Examinadora

Assinatura Eletrônica
28/11/2023 22:44:25.0
MARCO ANTONIO SIMÕES TEIXEIRA
Avaliador Externo (PONTIFICA UNIVERSIDADE CATÓLICA DO PARANA)

Assinatura Eletrônica
29/11/2023 09:10:45.0
ANDRÉ AUGUSTO MARIANO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Av. Cel. Francisco H. dos Santos, 210, Bairro Jardim das Américas, Bloco PK/PL - DELT, Setor Tecnologia, Campus Centro Politécnico - Curitiba - Paraná - Brasil
CEP 81530-000 - Tel: (41) 3361-3622 - E-mail: ppgee@ufpr.br
Documento assinado eletronicamente de acordo com o disposto na legislação federal Decreto 8539 de 08 de outubro de 2015.
Gerado e autenticado pelo SIGA-UFPR, com a seguinte identificação única: 330536
Para autenticar este documento/assinatura, acesse https://siga.ufpr.br/siga/visitante/autenticacaoassinaturas.jsp e insira o codigo 330536

"The good life is one inspired by love and guided by knowledge."
-Bertrand Russell

AGRADECIMENTOS

I would like to begin by expressing my gratitude to my parents, Sergio Novicki and Sandra Maria Penkal Novicki, who provided unwavering support and love throughout my master's thesis. My heartfelt thanks extend to my siblings, Sergio Novicki Junior and Francelize Novicki, for their encouragement and support. And my grandparents: Iracy and Pedro Penkal. To my advisor, Professor Eduardo Parente Ribeiro, for accepting me as his student and for providing me with the invaluable opportunity and support I needed to complete my master's thesis. A special thanks go to Fabio Luis Reway. I cannot adequately express my gratitude in words for the incredible personal and professional support during my time in Germany. I would also like to extend my appreciation to Professor Dr.Ing Werner Huber for welcoming me into his team and allowing me to work alongside his exceptional group. Yuri Poledna and Maikol Funk Drechsler, who not only assisted me but also made the work environment enjoyable and fun.

Quoting one of my favorite movies, 'It's a Wonderful Life', which says, 'Dear George, remember, no man is a failure who has friends'. I'd like to express my appreciation to my friends from my physics undergraduate days, Matheus Scherer, Marcelo Ferreira, and Nicholas Melenek. Before I left for Germany, every occasion became an opportunity to gather and celebrate. I had numerous farewell parties, dinners, and even trips. I've never felt more loved, and for that, I want to thank my friends: Rodrigo Tamari, Leonardo Ajuz, Camila Grechoniak, Giulia Egashira, Gustavo Kulisky, Henrique Taguti, Paulo Galvão, Álisson Carvalho, Pedro Carias, Manuella Martins, Alis Frentzel, and João Paulo Krug. I'd also like to extend my gratitude to my friends from Sion: Aline Zablonsky, Andre Moreira, Caroline Goulart, and Maria Clara Ariki. A special thank you to my friend Luisa Moratelli for the unforgettable trips in Europe during my stay in Germany.

I would also like to express my gratitude for the financial support received from: the AWARE Center and its project 'enGlobe – Engineers go global', which is funded by the German Academic Exchange Service (DAAD) and the Federal Ministry of Science and Research (BMBF). And the 'Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil' (CAPES).

RESUMO

Para possibilitar o desenvolvimento de sistemas de percepção para direção automatizada, a indústria automotiva utiliza cenas pré-gravadas com dados de sensores, como por exemplo: câmera, radar e LiDAR. Este conjunto de dados (*dataset*) permite a realização de um processo iterativo no desenvolvimento de algoritmos, a fim de reduzir quantidade de testes reais. Primeiramente, neste projeto é realizada uma revisão de diferentes *datasets* disseminados pela comunidade científica e utilizados pela indústria automotiva. Em seguida, é proposta a criação de um novo *dataset* chamado TWICE, um acrônimo em inglês para "*digital twin of test scenarios in a controlled environment*". Este *dataset* contém dados de sensores reais (câmera, radar e LiDAR) coletados de duas formas diferentes: (1) com a execução de testes em campo em uma pista de testes, e (2) em laboratório, com uma bancada de testes, na qual os sensores são estimulados com ajuda de simulação. Para a criação do *dataset* proposto, os dados coletados devem ser estruturados de maneira que possibilitem seu processamento de forma automatizada. A principal característica é o fato de que os dados coletados em laboratório se referem a um gêmeo digital (*digital twin*) dos testes executados em campo. Ou seja, para cada teste real existe um teste equivalente criado na simulação. Porém, ambos foram executados com o mesmo hardware e software para a percepção do ambiente. A fim de estimar a diferença entre os testes realizados em campo e em laboratório, é proposto um método para estimar a diferença (*gap*) entre realidade e simulação com base no TWICE *dataset*. Isso contribui para que o desenvolvimento de algoritmos de direção automatizada possa ser, em um primeiro momento, auxiliado por testes realizados em laboratório antes da execução de extensivos testes em campo, que demandam grande esforços e custos.

Palavras-chaves: dataset; autonomous driving; digital twin; adverse weather; radar; LiDAR;

ABSTRACT

To enable the development of perception systems for automated driving, the automotive industry uses pre-recorded scenes with data from sensors such as camera, radar and LiDAR. This allows an iterative process in the development of algorithms, in order to reduce the effort in the execution of tests. First, in this project a review of different datasets disseminated by the scientific community and used by the automotive industry is performed. Then, it is proposed the creation of a new dataset called TWICE, an acronym for "digital twin of test scenarios in a controlled environment". This dataset contains data from real sensors (camera, LiDAR and radar) collected in two different ways: (1) by performing field tests on a test track, and (2) in the laboratory, with a test bench, on which the sensors are stimulated with the help of simulation. To create the proposed dataset, the collected data must be structured in such a way that it can be processed in an automated manner. The main characteristic of the TWICE dataset is the fact that the data collected in the laboratory refer to the digital twin of the tests performed in the field. That is, for each real test there is an equivalent test created in the simulation. However, both tests were executed with the same hardware and software for environment perception. In order to estimate the difference between the tests performed in the field and in the laboratory, a method is proposed to estimate the gap between reality and simulation based on the TWICE dataset. This contributes to the fact that the development of automated driving algorithms can be, in a first moment, aided by tests performed in the laboratory before the execution of extensive field tests, which demand great efforts and costs.

Key-words: dataset; autonomous driving; digital twin; adverse weather; radar; LiDAR;

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ADAS	Advanced driver assistance system
ADC	analog-to-digital converter
ADMA-G	Automotive Dynamic Motion Analyzer-GNSS/Inertial System
ADS	Automate Driving Systems
AEB	Automatic Emergency Braking
AI	artificial intelligence
ASAM	Association for Standardization of Automation and Measuring Systems
CAN	Controller Area Network
CCD	Charge-Coupled Device
CCRb	Car to Car Rear braking
CCRs	Car to Car Rear static
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	convolutional neural network
DDI	Direct Data Injection
EU	European Union
EURONCAP	European New Car Assessment Programme
FOG	fiber optic gyroscopes
GA	Genetic Algorithm
GMSL	Gigabit Multimedia Serial Link
GNSS	global navigation satellite system
GPS	Global Positioning System
GPU	graphics processing units
HiL	Hardware-in-the-Loop

INS	inertial navigation systems
LiDAR	Light Detection and Ranging
M&S	Modelling and Simulation
ODDs	operational design domains
OSI	Open Simulation Interface
OTA	Over-The-Air
RAM	Random Access Memory
RCS	Radar Cross Section
RMS	Root Mean Square
ROS	Robot Operating System
RTK	real-time kinematic positioning
SAE	Society of Automotive Engineers
SP80	Spectra Position 80
VIB	Video Interface Box
VIL	Vehicle-in-the-Loop
VRTS	Vehicle Radar Test System
YOLO	You Only Look Once
YOLOP	You Only Look Once for Panoptic Perception

CONTENTS

1 INTRODUCTION

Autonomous vehicles are among the technologies that will have the greatest impact on our society in the future, with the potential to prevent fatal accidents (L.; G., 2015). In 2016, there were 1.35 million deaths due to car accidents (WAYMO LLC, 2021). It is estimated that 94% of vehicle accidents are caused by human error and could be avoided by automated driving systems (SINGH, 2015). Driver assistance systems are already a reality in the current automotive market and help prevent accidents. For example, the Automatic Emergency Braking (AEB) system alerts the driver of a possible collision. Advanced systems can even take emergency maneuvers if the driver does not intervene after the alert. These systems represent only the first step towards a future in which cars will be autonomous. To make this automation possible, the maturity of environmental perception technologies with sensors such as (cameras, radars, and Light Detection and Ranging (LiDAR)) and artificial intelligence algorithms or other statistical methods is necessary. These technologies must be able to detect and classify objects and obstacles in real time during vehicle operation under different adverse conditions.

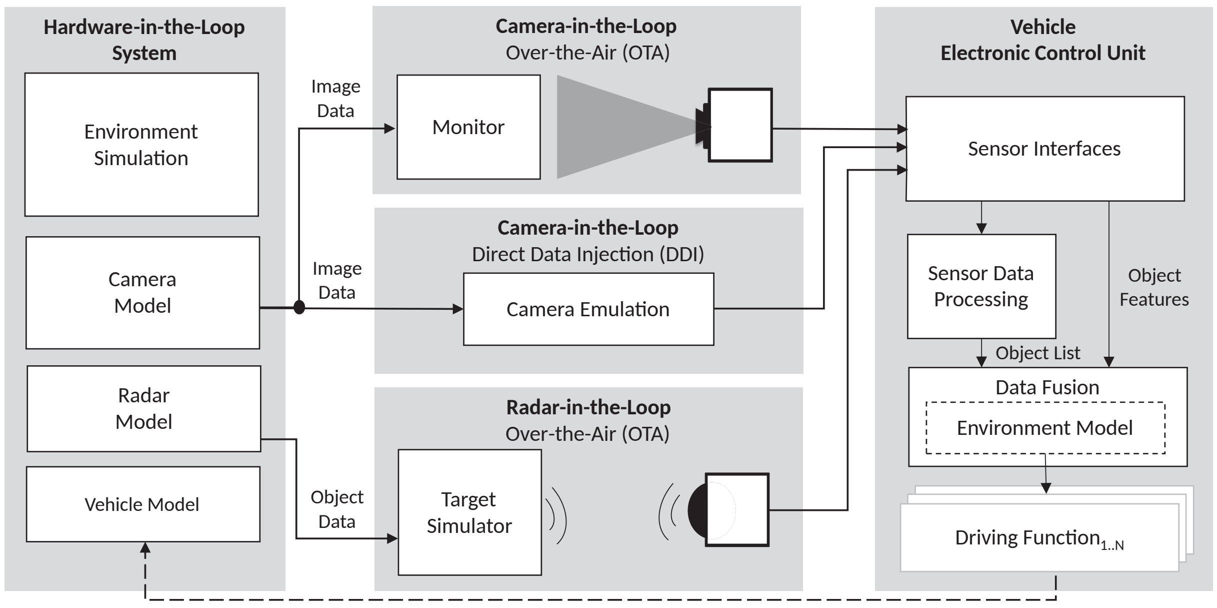
In order to assist the development of robust perception systems, different datasets consisting of pre-recorded scenarios are created. Currently, in the automotive industry, these data are used for neural network training and automated system testing.

The Society of Automotive Engineers (SAE) defined five levels of driving automation in (SAE, 2021). Level zero stands for no automation at all. Advanced driver assistance system (ADAS) such as adaptive cruise control, anti-lock braking systems and stability control start with level one. Level two is partial automation to which advanced assistance systems such as emergency braking or collision avoidance, are integrated. Level three is conditional automation; the driver could focus on tasks other than driving during normal operation, however, the driver has to quickly respond to an emergency alert from the vehicle and be ready to take over. In addition, level three Automate Driving Systems (ADS) operate only in limited operational design domains (ODDs) such as highways. Human attention is not needed in any degree at level four and five. However, level four can only operate in limited ODDs where special infrastructure or detailed maps exist. In the case of departure from these areas, the vehicle must stop the trip by automatically parking itself. The fully automated system, level five, can operate in any road network and any weather condition. No production vehicle is capable of level four or level five driving automation yet (YURTSEVER et al., 2020). Level four and above driving automation in urban road networks is an open and challenging problem. The environmental variables, from weather conditions to surrounding human behavior, are highly non deterministic and difficult to predict. Furthermore, system failures lead to accidents: Tesla's Autopilot failed to recognize a white truck and collided with it, killing the driver.(CNN, 2016)

Testing is a crucial means to assess and validate proposed algorithms, and these tests can be conducted in real-world settings. Real-world tests can be carried out in controlled or uncontrolled environments. Conducting tests in uncontrolled scenarios require extensive annotation efforts due to unknown actor locations, and introduces additional unknown variables that can increase uncertainty. Controlled environments offer advantages such as test reproducibility and knowledge of the precise location of all actors involved. Controlled or not tests in real-world environments are costly and entails bureaucratic and safety protocols. The use of synthetic data is the alternative that emerges in the face of these challenges.

There are multiple simulation software options available that can simulate entire environments, providing the ability to control various factors and replicate situations that would be impractical or impossible to achieve in real-world tests. These simulations can encompass scenarios such as high speeds, traffic infractions, and car crashes. These software platforms are capable of simulating the behavior of all the sensors typically found in autonomous vehicles. However, it is possible to utilize the same sensors utilized in the real-world tests within a Hardware-in-the-Loop (HiL) simulation to generate synthetic data. Figure 1 shows an example of a HiL system. The first block on the left is where the environment and vehicle simulation occurs. Then, it feeds the sensor stimulators with data (center block). Finally, the sensors are stimulated, and their outputs are received by the car Electronic Control Unit (ECU) (right block).

Figure 1 – *HiL system*



Source: (NETO et al., 2023).

There are two methods to stimulate the sensors. The first method is known as Direct Data Injection (DDI), in which the stimulation does not use the physical sensor. The sensor model is stored within the stimulation device, which subsequently mimics the sensor timings and data outputs based on input received from the simulation. The other method is Over-The-Air (OTA), where the setup uses the real sensor and a stimulation device. For the camera, a monitor is used where the simulation image is displayed to be captured by the camera. For the radar, a stimulation device is used to manipulate the electromagnetic waves generated by the radar to simulate an object. However, a constraint in this method is the resolution of the stimulation device.

Simulation plays a significant role in the development of the ADS. Nevertheless, in order for the utilization of this data to be deemed valid, it must adhere to certain requirements outlined in European Union (EU) regulations:

> The credibility can be achieved by investigating and assessing five properties of Modelling and Simulation (M&S) :
>
> (a) capability – what can the M&S do, and what the risks are associated with it;
>
> (b) accuracy – how well does M&S reproduce the target data;
>
> (c) correctness – how sound & robust are M&S data and algorithms;
>
> (d) usability – what training and experience is needed.
>
> (e) fit for purpose – how suitable is the M&S for the ODD and ADS assessment.
>
> (COMMISSION IMPLEMENTING REGULATION (EU) 2022/1426)(L 221/50)

There is also the German government directive 86/22:

> The fulfillment of requirements can also be checked by suitable simulation. In this case, the simulation tools must be validated. The validation of the simulation tools must be carried out by means of comparison with a representative selection of real tests. There must not be any significant difference between characteristic values from simulation and road test. The performance of the sensor technology in terms of detection and classification of objects as a function of different distances and environmental conditions shall be determined for the simulation in real tests. Each simulation series shall be supplemented by real tests if deemed necessary by the technical service. (Translated from German)(2022, p.40)

As became clear, it is fundamental to measure the simulation-to-reality gap in order to validate the use of synthetic data in the development of ADS. The creation of a dataset that contains scenarios recorded in real-world and then reproduced in the laboratory environment generating synthetic data can help other researchers to develop methods to aid in this effort.

Semantic segmentation is a computer vision technique that involves dividing an image into meaningful and distinct regions, assigning each pixel a specific class label. In the context of autonomous driving, semantic segmentation plays a vital role in scene understanding and perception. By segmenting the surrounding environment into different categories such as roads, pedestrians, vehicles, and obstacles, autonomous vehicles can accurately perceive and interpret their surroundings. This enables them to make informed decisions and take appropriate actions, such as path planning, object detection, and collision avoidance.

This work aims to create a dataset for the development of autonomous driving systems, which will be made available for use by the research community. Additionally, it demonstrates the application of this dataset in developing a method to measure the gap between simulation and reality. The algorithm performs semantic segmentation of the scenario, separating the free-space area (drivable area) and the lane lines on the road. Then it is possible to compare the performance of the algorithm between real-world and simulation environment.

1.1 OBJECTIVES

The general objective of this project is to organize the recorded data into a labeled dataset with a well-defined structure and a format that is agnostic:

- Based on the dataset, develop a method to estimate the simulation-to-reality gap for testing automated driving systems.

The specific objectives of the project are:

- Make the dataset public and provide examples of its utilization.
- Develop tools to automate data structuring within the ROS framework to the Open Simulation Interface (OSI) format.
- Perform annotations based on ground truth of the scenes and develop a tool to standardize the annotation format (labeling).
- Develop a method to estimate the semantic segmentation gap between simulated and real data.

1.2 WORK STRUCTURE

This work is divided into five chapters. It begins with an introduction to the concepts relevant to this study, followed by a literature review of available datasets for autonomous driving and related concepts like sensors and algorithms. The subsequent chapter covers the methodology, including data acquisition, utilized algorithms, conducted tests, data structuring, annotation, processing, and evaluation methods. Following that, there is a dedicated chapter for the results. Finally, the conclusion encompasses qualitative comparisons of results, along with a discussion of future work.

2 LITERATURE REVIEW

2.1 AVAILABLE DATASETS

The KITTI dataset, developed by (GEIGER et al., 2012), was one of the first publicly available datasets that provided data for the development of object detection and sensor fusion algorithms. First released in 2012, KITTI consists of a dataset recorded with a car equipped with two high-resolution stereo cameras that record in color and grayscale, a LiDAR, and the Global Positioning System (GPS) . The car captured data in urban, rural, and highway environments while driving through the streets of Karlsruhe, Germany. The dataset contains scenes with up to 15 cars and 30 pedestrians visible per image.

Another widely used dataset in the academic community, which offers a large amount of data and sensors, is the nuScenes dataset (CAESAR et al., 2020). Published in 2019, nuScenes was recorded in Singapore and Boston using cars equipped with GPS, Inertial Measurement Unit (IMU), six high-resolution cameras, five radars, and one LiDAR, covering a 360° field of view. The dataset consists of 1000 scenes, each lasting 20 seconds, and is fully annotated with 3D bounding boxes to identify objects and their locations in the scene. Recorded in scenarios with heavy traffic, including various environments such as vegetation, buildings, vehicles, and road signs, this dataset contains 100 times more images than the pioneering KITTI dataset.

The Canadian Adverse Driving Conditions Dataset (PITROPOV et al., 2020) is specifically designed to address adverse driving conditions, offering a diverse range of challenging scenarios. These scenarios encompass various weather conditions, road surfaces, and lighting conditions, ensuring a realistic representation of the complexities encountered during adverse driving situations. From heavy rain and fog to snow-covered roads and low-light environments. This dataset captures the intricacies and uncertainties present in adverse driving conditions with eight cameras, LiDAR and GPS/IMU.

However, collecting data in a real-world environment is extremely expensive and labor-intensive. An alternative approach is the generation of synthetic sensor data. Published in 2020, the All-In-One Drive (AIODrive) dataset (WENG et al., 2021) is a fully synthetic dataset created using the CARLA simulator [1](DOSOVITSKIY et al., 2017). The AIODrive dataset includes atypical scenarios such as high-speed driving, traffic rule violations, and accidents. The simulated sensors in the dataset include cameras, LiDAR, Radar, IMU, GPS, and SPAD-LiDAR.

[1] CARLA is an open-source software designed to generate simulations for autonomous driving system development.

Table 1 – ALGORITHM PERFORMANCE.

Method	*Training Data*	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointRCNN	AIODrive	65,32	46,21	39,38	24,57	19,04	18,32	40,93	30,41	26,68
	KITTI	85,02	75,16	68,14	46,53	38,76	33,96	73,40	56,73	51,87
	KITTI+AIODrive (10k frames)	87,24	76,83	70,563	46,97	40,78	36,03	74,19	59,31	52,93
	KITTI+AIODrive all frames	88,10	77,03	72,41	51,03	42,18	37,26	78,01	60,14	52,89

Source: (WENG et al., 2021)
Caption: 3D detection results on the KITTI dataset when training is augmented with AIODrive data

The authors demonstrated that training an algorithm with a combination of real-world data (KITTI) and synthetic data (AIODrive) achieves better object detection performance compared to training the same algorithm solely with real-world data. This finding confirms the validity and importance of using synthetic data, as observed in Table 1.

Moreover, there are other datasets available to the academic community with different characteristics and purposes. Table 2 provides a comparison of the sensors present in the ego vehicle of each dataset, among other features.

Table 2 – COMPARISON BETWEEN SOME AVAILABLE DATASETS

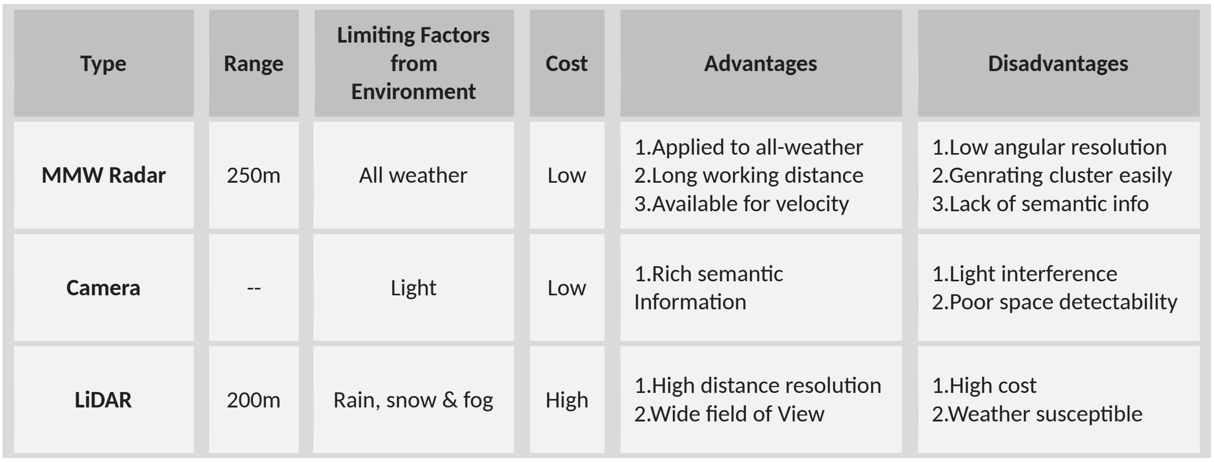
dataset	Citys	Hours	Sequences	Annotation	Stereo	Depth	LiDAR	Radar	SPAD-LiDAR	IMU/GPS	360°
KITTI	1	1,5	22	15k	✓	✓	✓			✓	
Cityscape	27	2,5	0	5k	✓					✓	
Mapillary Vistas	30	-	-	25k							
ApolloScape	4	-	-	140k	✓		✓			✓	
SYNTHIA	1	2,2	4	200k		✓					✓
H3D	4	0,8	160	27k			✓			✓	
SemanticKITTI	1	1,2	22	43k			✓				
DrivingStereo	-	5	42	180k	✓	✓	✓			✓	
Argoverse	2	0,6	113	22k	✓		✓			✓	✓
EuroCity	31	0,4	-	47k							
CADC	1	0,6	75	7k			✓			✓	
Audi	3	0,3	3	12k	✓	✓	✓			✓	✓
nuScenes	2	5,5	1k	40k			✓	✓		✓	✓
A*3D	1	55	-	39k	✓		✓				
Waymo Open	3	6,4	1150	230k			✓				
AIODrive	8	2,8	100	100k	✓	✓	✓	✓	✓	✓	✓

Source: (WENG et al., 2021)
Caption: Available datasets and their atributes

2.2 SENSORS

Generally, there are two types of sensors: passive and active sensors. Passive sensors receive the natural emission from an environment, such as natural light or infrared rays. The camera is the most commonly used passive sensor in a car that captures a 2D array of colored points through a lens. Active sensors send signals and receive the reflection from surfaces and objects. LiDAR and Radar are the most used active sensors in a car. This class of sensors is more expensive than passive sensors,

Figure 2 – Comparison between sensors



Type	Range	Limiting Factors from Environment	Cost	Advantages	Disadvantages
MMW Radar	250m	All weather	Low	1.Applied to all-weather 2.Long working distance 3.Available for velocity	1.Low angular resolution 2.Genrating cluster easily 3.Lack of semantic info
Camera	--	Light	Low	1.Rich semantic Information	1.Light interference 2.Poor space detectability
LiDAR	200m	Rain, snow & fog	High	1.High distance resolution 2.Wide field of View	1.High cost 2.Weather susceptible

Source: (ZHOU et al., 2020).
Caption: Comparison between sensors.

but they can measure the depth of objects by sending and receiving reflected signals. In Figure 2 there is a comparison between sensors present in an autonomous vehicle. Each one has advantages and disadvantages and situations in which factors from the environment limits the operation of the sensor.

2.2.1 Camera

A digital camera is a device that captures and stores digital images. It uses a combination of optics, electronics, and software to convert light into a digital representation of the scene. The camera lens gathers and focuses incoming light from the scene onto the image sensor. It controls factors such as focus, depth of field, and aperture. The image sensor, typically a Charge-Coupled Device (CCD) or a Complementary Metal-Oxide-Semiconductor (CMOS) sensor, is composed of millions of tiny light-sensitive pixels. Each one converts incoming light into an electrical charge. The individual pixel's photodiode accumulates an electrical charge proportional to the intensity of light, then it is converted into an electrical signal. An analog-to-digital converter (ADC) assigns a numerical value to each pixel's charge level, representing the color and intensity of light.

The camera lens has some inherent distortions. A way to describe the internal properties of a camera that affect the imaging process is the intrinsic matrix:

$$K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where fx and fy represent the focal lengths of the camera in the x and y directions, respectively. The values cx and cy denote the principal point, which is the optical center of the camera.

To relate the 3D world coordinates to 2D image coordinates, the intrinsic matrix is combined with the extrinsic matrix (describing the camera's position and orientation in the world) and the projection matrix. The projection matrix maps 3D points in the world coordinate system to their corresponding 2D points in the camera image. The equation for projecting 3D points to 2D image coordinates is as follows:

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}^{\mathsf{T}} = K \cdot \begin{bmatrix} R & t \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^{\mathsf{T}} \tag{2.2}$$

where s represents the scaling factor, u and v are the pixel coordinates, K is the intrinsic matrix, R (rotation) and t (translation) represents the extrinsic matrix, and XYZ are the 3D point coordinates (ZHANG, 2000).

2.2.2 Radar

The radar system consists of a transmitter that emits radio waves, typically in the mm wave frequency range for vehicle applications. The radar system has an antenna that transmits the radio waves into the surrounding environment. This antenna focuses and directs the emitted waves in a particular direction, forming a beam. When the emitted waves encounter objects in the environment, they get reflected back toward the radar system. The radar's antenna receives these reflected waves. As they return to the radar system, they undergo a frequency shift known as the Doppler effect. This shift occurs due to the relative motion between the radar system and the detected objects and it is given by:

$$D_f = f_r - f_t \tag{2.3}$$

where D_f is the Doppler frequency shift, f_t is the frequency transmitted and f_r is the frequency received. It allows the radar to measure the velocity of the objects using the following equation:

$$v = \frac{c \cdot D_f}{2 \cdot f_c} \tag{2.4}$$

where c is the speed of the light and f_c is the operating frequency of the radar. The received waves are processed by the radar's electronic circuitry. It analyzes the changes in frequency, amplitude, and phase of the reflected waves to extract information about the detected objects.

The Radar Cross Section (RCS) is a measure of the reflectivity of an object or target to radar signals. It quantifies how much electromagnetic energy is scattered back to the radar system from the target. A larger RCS indicates a stronger reflection, making the target more detectable by the radar. The RCS of an object depends on various factors, including its size, shape, material composition, and orientation relative to the radar system. It is typically measured in square meters (m²). Mathematically, it is written as:

$$\sigma = \lim_{R \to \infty} 4\pi R^2 \frac{|E_s|^2}{|E_i|^2} \tag{2.5}$$

where R is the distance between radar and target, E_s is the scattered field strength at radar and E_i is the incident field strength at target (YEN, 2005).

2.2.3 LiDAR

A LiDAR sensor is composed of two main components: a transmitter, which consists of a laser that emits light with wavelengths between 250 to 1600 nanometers (nm) - some wavelengths work better in specific environments and targets than others; and a receiver, which is responsible for gathering, analyze, and compute the reflected signal. Regarding the receiver, it generally includes: (a) a telescope to collect the photons; (b) an optical analyzer that can filter specific wavelengths or polarization states from the received light and convert the optical signal into an electrical quantity; and (c) a data acquisition module, which is responsible for calculating the elapsed pulse time and storing the information (RORIZ et al., 2022). The output of LiDAR is a set of 3D points called Point Cloud (PCL).

The two wavelengths for automotive LiDAR currently in use are 905 nm, and 1550 nm, making possible spatial resolution on the order of 0.1 degrees, which allows for extremely high-resolution 3D representation of objects around the vehicle. Rotor-based mechanical LiDAR sensors are the most mature imaging technique being used by the automotive industry. Widely used in driverless systems, they can provide a 360° horizontal FoV, with different vertical FoV ranges. LiDAR challenges that are currently hampering the development are sensor calibration, mutual interference, data compression, point cloud denoising, ground segmentation, and object segmentation and detection methods (RORIZ et al., 2022).

2.2.4 GNSS systems

The global navigation satellite system (GNSS) is an international system of multiple constellations of satellites, including systems such as GPS (United States), GLONASS (Russia), BeiDou (China), Galileo (European Union), and other constellations and positioning systems. GNSS operates in the L-Band (1 to 2 GHz) which can pass through clouds and rain, with a minimum impact on the transmitted signal in terms of path attenuation. GNSS sensors include one or more antennas, reconfigurable GNSS receivers, processors and memory. GNSS is often in combination with real-time kinematic positioning (RTK) systems using ground base-stations to transmit correction data. Odometry and inertial navigation systems (INS) use dead reckoning to compute position, velocity and orientation without using external references. INS combines motion sensors (accelerometers), rotation sensors (gyroscopes), and also magnetic field sensors (magnetometers). For the advanced INS, fiber optic gyroscopes (FOG) are used: with no moving parts, and two laser beams propagating in opposite directions through very long fiber optic spools, the phase difference between the two beams is compared and it is proportional to the rate of rotation. The combination of the above, such as GNSS with INS (GNSS+INS) and other sensors, with algorithms such as Kalman Filter and motion models, is a common approach to improve positioning accuracy and reduce drift. For example, the Spatial FOG Dual GNSS/INS of Advanced Navigation has 8 mm horizontal position accuracy and about 0.005° roll/pitch accuracy (ZHANG et al., 2021).

2.3 SENSORS CALIBRATION

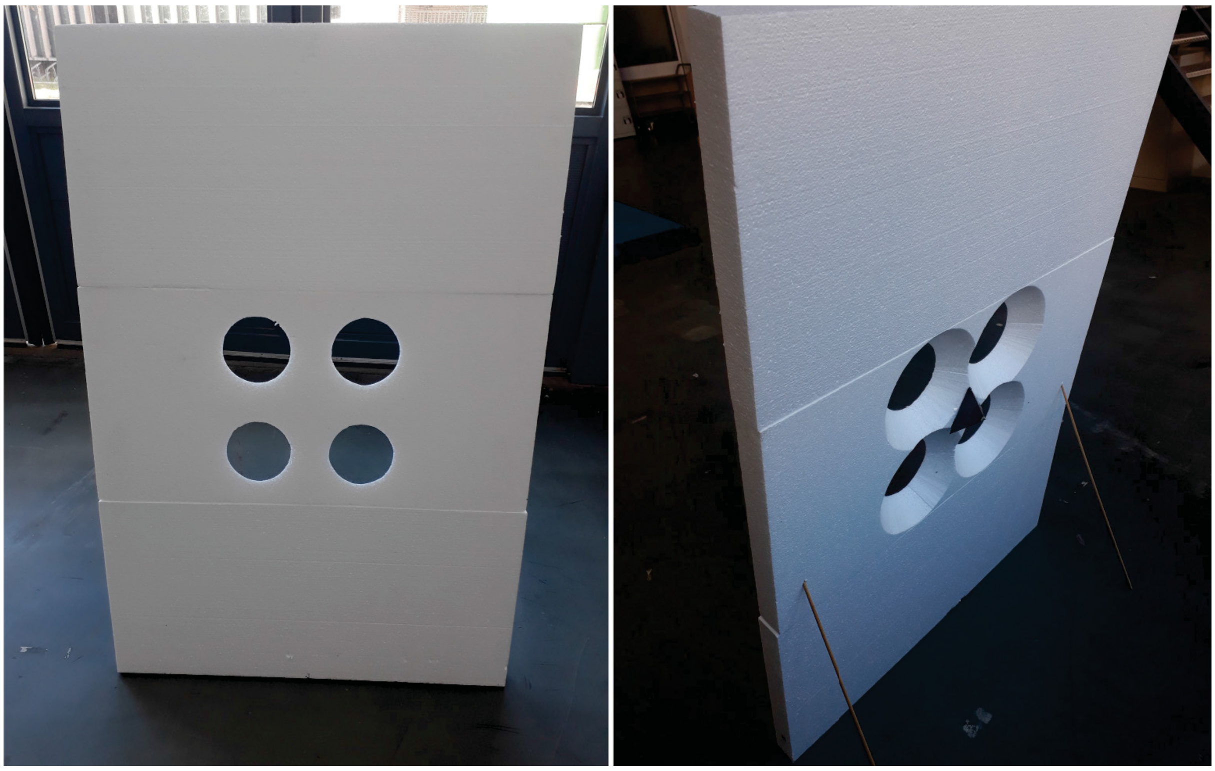
Camera parameters include intrinsics, extrinsics, and distortion coefficients. To estimate the camera parameters, one must have 3-D world points and their corresponding 2-D image points. These correspondences can be obtained using multiple images of a calibration pattern, such as a checkerboard; the sizes of the squares must be known. With these correspondences, the camera parameters can be solved for. The pinhole calibration algorithm is based on the model proposed by (BOUGUET, 2022). The model includes, the pinhole camera model (ZHANG, 2000) and lens distortion (HEIKKILA; SILVEN, 1997). The pinhole camera model does not account for lens distortion because an ideal pinhole camera does not have a lens. To accurately represent a real camera, the full camera model used by the algorithm includes the radial and tangential lens distortion.

To know exactly where which sensor is located in relation to the vehicle. It must be performed an extrinsic calibration. An example of extrinsic calibration is proposed by (DOMHOF et al., 2019). In his article he propose a multi sensor calibration, the algorithm calculates the sensors position relative to each other using a common Styrofoam target (Figure 3) with a corner reflector placed in the middle pattern. Note that the circles have slopes, this was done to enforce that LiDAR detections and camera detections are detected at the front the plate.

However it is necessary to know the position of one of the sensors in relation to vehicle. It can be done using the camera position calculation toolkit for vehicle extrinsic calibration, available in NVIDIA Driveworks (NVIDIA, 2023). This toolkit incorporates a approach utilizing printed April targets [2] with varying sizes. Specifically, four targets

[2] It consists of a black square with a unique pattern of white squares or lines in a grid-like arrangement, designed to be easily detectable and identifiable by computer vision algorithms.

Figure 3 – Styrofoam target.



Source: (DOMHOF et al., 2019).

are affixed to the vehicle wheels, one is positioned in front of the camera, and two are placed on each side of the car. A separate camera captures images of the vehicle with the targets, while the calibration camera captures an image of the front target. By leveraging this data along with measurements of the printed targets and vehicle dimensions (including wheel diameter and car length), the program can be executed. The outcome of this extrinsic calibration is camera position and rotation relative to the rear axle.

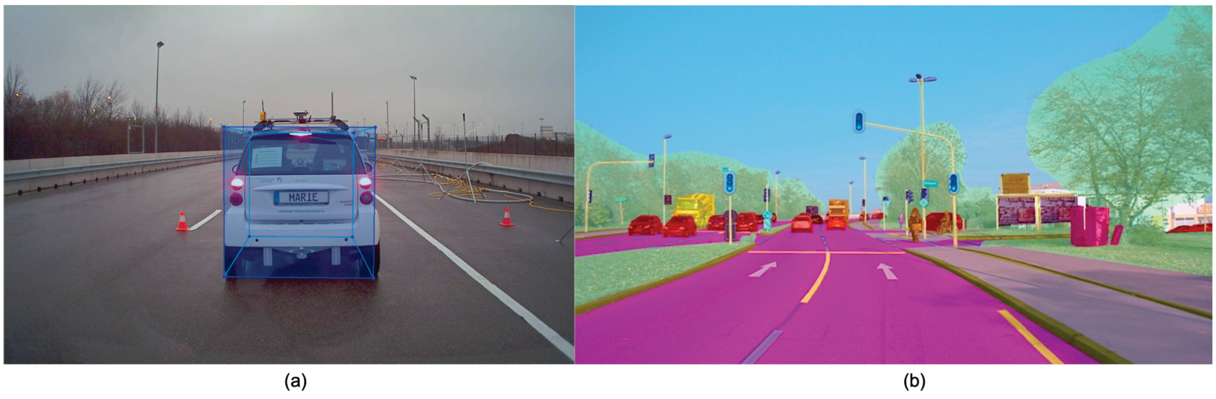
2.4 PERCEPTION

2.4.1 Data annotation

Data annotation is a crucial step in machine learning and artificial intelligence (AI) development. It involves labeling and categorizing datasets to train algorithms to recognize patterns and make predictions. However, the process of data annotation can be tedious and time-consuming, and errors can lead to inaccurate results. Various annotation techniques are employed to extract information from camera data, enabling autonomous vehicles to understand and interpret their surroundings accurately. The annotation process can be performed manually or automated, depending on the specific requirements and available technology.

One widely used type of annotation is 2D bounding box, which involves labeling

Figure 4 – Data annotation.



Caption: (a) 3D bounding box (b) Semantic segmentation.
Source: (a) (NETO et al., 2023) (b) (GEYER et al., 2020).

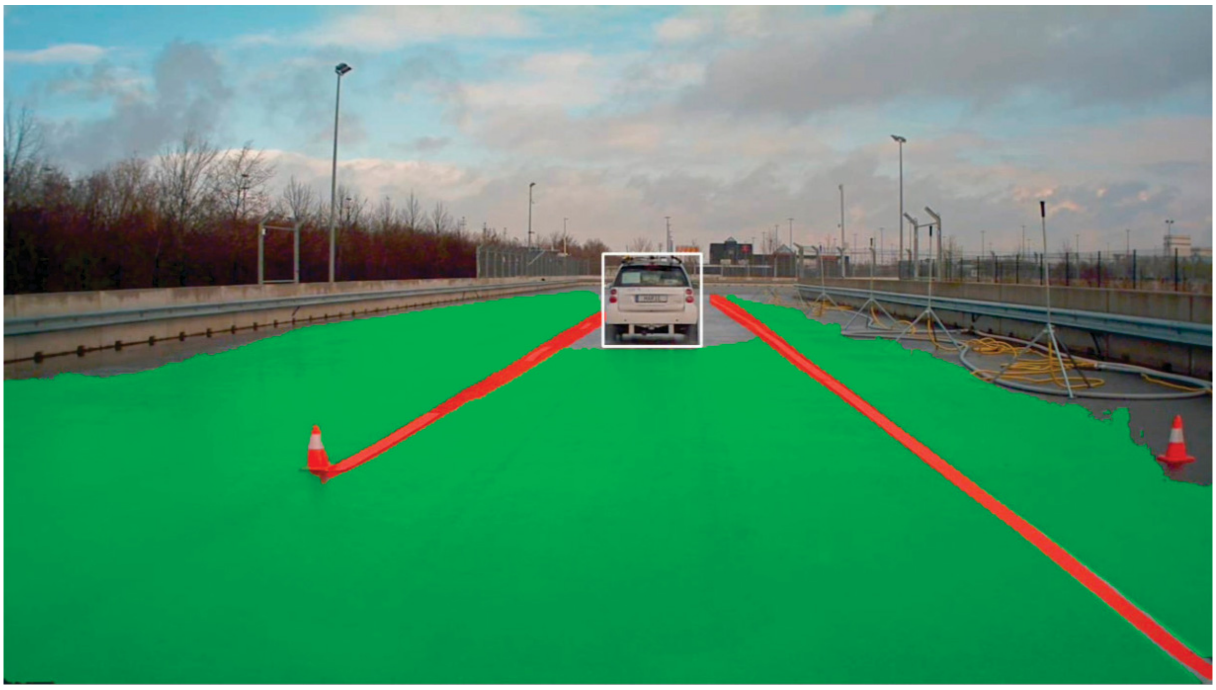
objects of interest, such as pedestrians, vehicles, and traffic signs, with rectangular bounding boxes. The annotation of cuboids, or 3D bounding boxes, goes beyond the 2D plane by capturing the depth and rotation of objects in the scene. Lastly, semantic segmentation annotation assigns pixel-level labels to different regions in an image, enabling the system to understand the scene's layout and identify individual elements as shown in the Figure 4.

2.4.2 Algorithms

The You Only Look Once (YOLO) algorithm is a popular and influential object detection algorithm in the field of computer vision. Before its release, the traditional object detection approaches required multiple passes over an image, YOLO takes a different approach by directly predicting bounding boxes and class probabilities in a single pass. It divides the input image into a grid and applies a convolutional neural network (CNN) to simultaneously predict bounding box coordinates and object class probabilities within each grid cell. This approach makes the algorithm fast and efficient, allowing it to process images in real-time. YOLO also introduces anchor boxes, which are pre-defined shapes of various aspect ratios, to handle objects of different sizes and shapes. By combining these predicted bounding boxes and class probabilities, effectively detects and localizes multiple objects within an image. YOLO has come a long way since its initial release in (REDMON et al., 2015) and has culminated in multiple versions with further improvements per iteration. The algorithm has continued to evolve ever since its initial release. Both YOLOv2 and YOLOv3 were written by Joseph Redmon. After YOLOv3, there came new authors who anchored their own goals in every other YOLO release. The latest version is YOLOv8.

The You Only Look Once for Panoptic Perception (YOLOP) (WU et al., 2022) is an extension of the original YOLO algorithm. Adding the ability to jointly handle three crucial tasks in autonomous driving: object detection, drivable area segmentation and

Figure 5 – YOLOP output.



Source: Author (NETO et al., 2023).
Caption: YOLOP detection: drivable area, lane line and object detection.

lane detection. It was designed to save computational costs, reduce inference time as well as improve the performance of each task. The Figure 5 shows the output of the algorithm. The drivable area (free-space) in green is the region where the car can drive, the red region is the lane line and the white bounding box is the detected object.

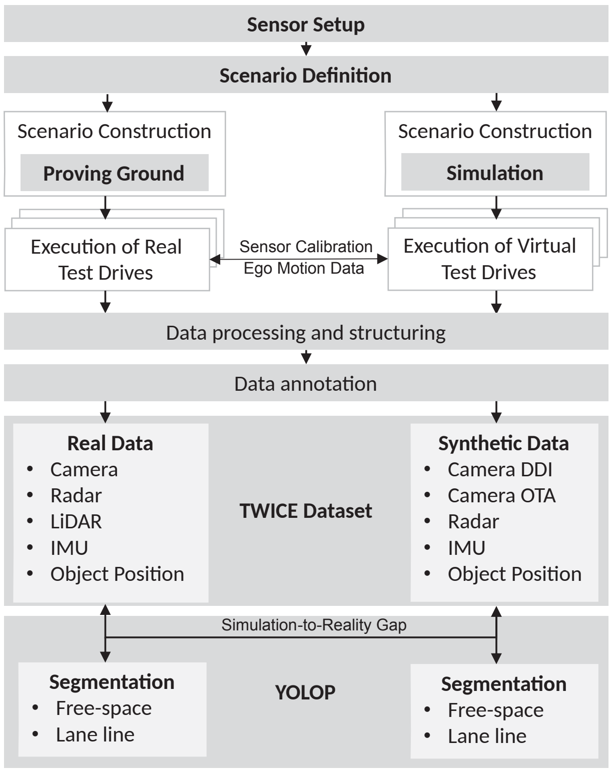
2.5 VIRTUAL TESTS

Carla is an open-source simulator for autonomous driving research that has been developed from the ground up to support development, training, and validation of autonomous driving systems (DOSOVITSKIY et al., 2017). It is based on the Unreal Engine 4 and supports Python programming. The simulation platform supports camera, LiDAR, radar and other sensors. However, Carla lacks: real-time capabilities (which are important for simulations with HiL due to the sensors being real-world objects) and RCS support on radar objects.

CarMaker is a simulator developed by IPG Automotive that can simulate a wide range of sensors in real time. It is capable of simulating real-world distortions such as noise in cameras or rain disturbances in radar and LiDAR. Written in C, CarMaker offers extensive support for this programming language. It also features a scenario editor that enables users to create their own custom scenarios for simulations. The simulation visualization tools, IPGMovie and MovieNX, support multiple 3D formats and allow the addition of driving functions to the ego vehicle.

3 METHODOLOGY

Figure 6 – *Work diagram*



Source: Author.

In Figure 6, the methodology of this work is presented. Firstly, the ego-vehicle[1] was equipped with the necessary sensor setup. Next, specific scenarios were defined and recorded during real-world test runs. The same sensor calibration and positioning of the ego-vehicle and objects were replicated to create corresponding test runs in the simulation environment. Up to this point, the tasks were carried out by Professor Werner's team at CARISSMA. Afterwards, it became my responsibility to process and organize all collected data into a structured format and to annotate the camera frames, thereby completing the dataset creation process. To estimate the gap between simulation and reality, the dataset was utilized in conjunction with an algorithm called YOLOP, which performs semantic segmentation.

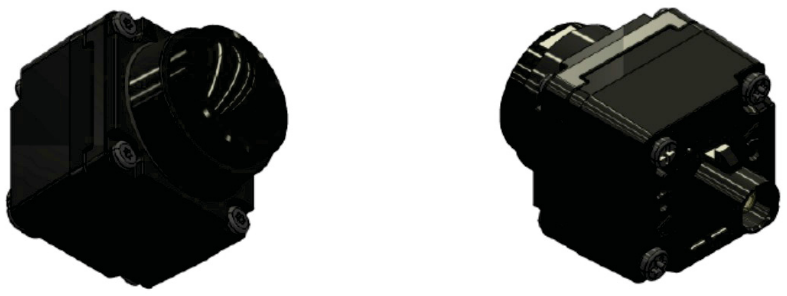
3.1 DATA ACQUISITION

In this section, a description will be provided for all the sensors and components utilized in the data acquisition in the real-world environment.

[1] Vehicle equipped with the sensors.

3.1.1 Camera

Figure 7 – Sekonix SF3325-100 Camera.



Source: (SEKOLAB, 2023).

The utilized hardware was the Sekonix SF3325-100: 1920x1232 resolution (2.3M Pixel), $30\,\mathrm{Hz}$ capture frequency, H:60°;V:$36,6$°. For transmission the camera utilizes the Gigabit Multimedia Serial Link (GMSL) protocol, which provides gigabit transmission speed via one single cable. (SEKOLAB, 2023)

The intrinsic camera calibration was performed using the Matlab camera calibrator (BOUGUET, 2022).

3.1.2 Radar

Figure 8 – Continental ARS408 radar.

Source: (CONTINENTAL, 2023).

The utilized radar was the Continental ARS408 (Figure 8), an automotive radar. It has two operating modes: object list and cluster list. Operates in the frequency range between 76 and 77 GHz and distance from 0 to 250 m with near range distance resolution of 0.39 m and in far range 1.79 m. This radar measures a velocity range from –400 km/h (coming towards the radar) to $+200$ km/h (going away from the radar). Connection is made with the use of a standard connector using the protocol Controller Area Network (CAN).

3.1.3 LiDAR

The utilized LiDAR was the Ouster OS1-128 (Figure 9). The range is 100 m in sunlight and minimum range of 0.3 m with accuracy of 3 cm. It is configurable horizontal

Figure 9 – LiDAR Ouster OS1-128.

Source: (OUSTER, 2023).

resolution of 512, 1024, or 2048, and vertical resolution of 128 channels. The capture frequency is $10\,\mathrm{Hz}$. Data is transmitted utilizing a User-Defined Packet (UDP) Ethernet packet.

3.1.4 GNSS/INS

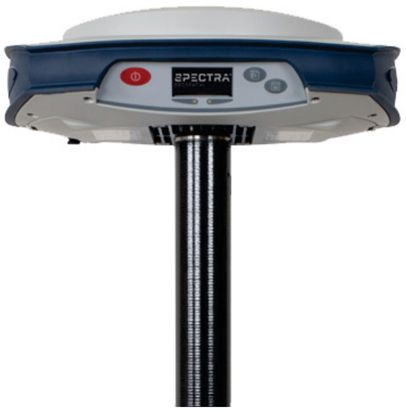
Figure 10 – ADMA G-Pro GeneSys.

Source: (GENESYS, 2023).

In order to collect the car and ego-vehicle position in the test track. It was used the Automotive Dynamic Motion Analyzer-GNSS/Inertial System (ADMA-G) (Figure 10). The system allows for constant measurement of acceleration, speed and position of moving vehicles in all three-dimensional axes. The FOG allows a precision of 1 cm in position. Pitch, roll and yaw angles can be continuously and precisely measured with 0.005° resolution. Connection is made using Ethernet.

The Spectra Position 80 (SP80) (Figure 11) is a GNSS receiver that was used to collect the position of the objects in the scene with a millimeter accuracy (SPECTRA, 2023). It has WiFi, 3.5G cellular and hot-swappable batteries.

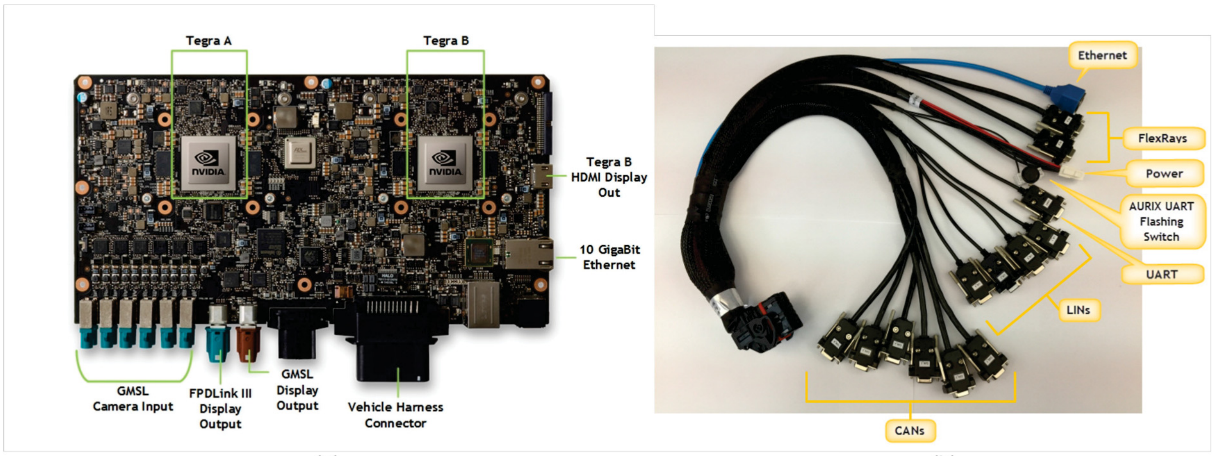
Figure 11 – SP80 GNSS receiver.



Source: (SPECTRA, 2023).

3.1.5 Eletronic Control Unit

Figure 12 – Nvidia Drive PX 2.



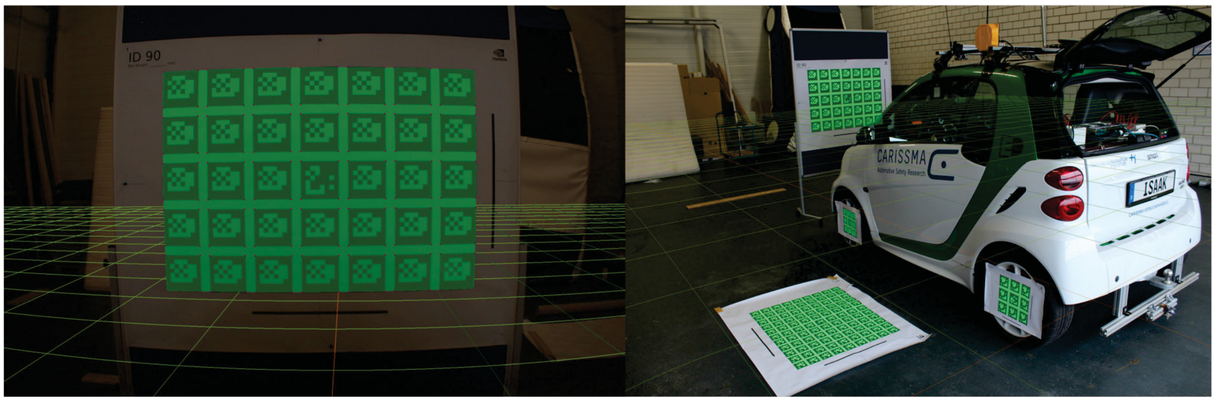
(a) (b)

Source: (NVIDIA, 2018).

The Nvidia Drive PX 2 AutoChauffeur (P2379) is an automotive platform designed to enable autonomous driving capabilities in vehicles. It is a hardware and software system developed by Nvidia, a leading technology company specializing in graphics processing units (GPU) and AI.

The Figure 12 shows Nvidia Drive PX 2 has gigabit Ethernet adapter, one 10 gigabit Ethernet adapter, 3 arrays of 4 GMSL cameras, 1 and 12 CAN interfaces. As show in Figure 12 (a) it has two cores known as Tegra A and Tegra B, they come in pair to distribute the processing load and increase security in case of failures. Also included, is the NVIDIA Driveworks platform that provides multiple tools to help the developer create an autonomous vehicle (NVIDIA, 2023).

Figure 13 – Extrinsic calibration.



Source: CARISSMA (2021).

3.1.6 Extrinsic Calibration

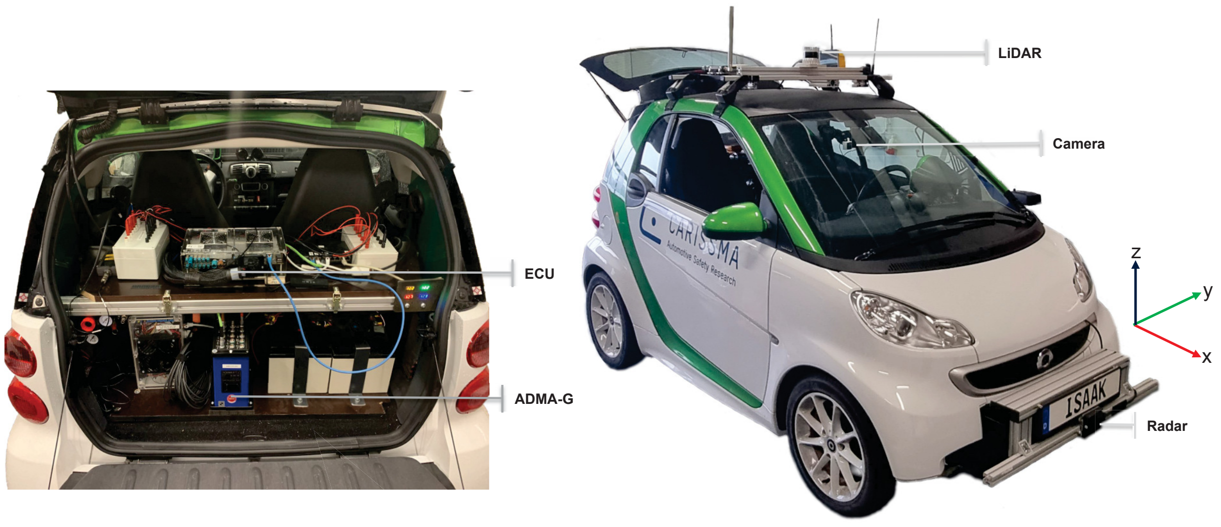
The extrinsic calibration is performed to determine the location of the sensors relative to the ego-vehicle. The car position is defined as zero at the center of the back axle (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). The ADMA-G sensor was mounted in the trunk of the car (Figure 14, and its location relative to the back axle was measured using lasers. To determine the relative location of the other sensors, the calibration proposed by (DOMHOF et al., 2019) was performed. This enabled the determination of the location of the radar and LiDAR relative to the camera. Finally, to obtain the positions of all sensors relative to the ego-vehicle, the calibration available in NVIDIA Driveworks was performed. As shown in Figure 13, targets were placed on the car wheels, on the ground beside the car, and in front of the ego-vehicle to be detected by the ego camera. Using the image shown in Figure 13, captured by the ego camera, and additional pictures captured by an external camera, the calibration tool returned the camera's position relative to the ego back axle.

3.2 HARDWARE IMPLEMENTATION

Figure 14 displays all the hardware described in the previous section, which is installed on the ego-vehicle used in the tests. In order to collect data from all the sensors, it is necessary to run software on the Electronic Control Unit (ECU).

The chosen software for this work was the Robot Operating System (ROS). It provides libraries and tools to assist software developers in creating robotic applications. Offers device drivers, libraries, visualizers, message transmission, package management, and other tools. ROS is licensed under an open-source license (STANFORD ARTIFICIAL INTELLIGENCE LABORATORY ET AL., 2018). Each sensor in ROS has a corresponding node. Each node is an executable that publishes and subscribes to certain topics. For example, Node X can publish to the topic "/topic_1" and Node Y can subscribe to the same topic "/topic_1" and read all the data being published by

Figure 14 – Ego-vehicle

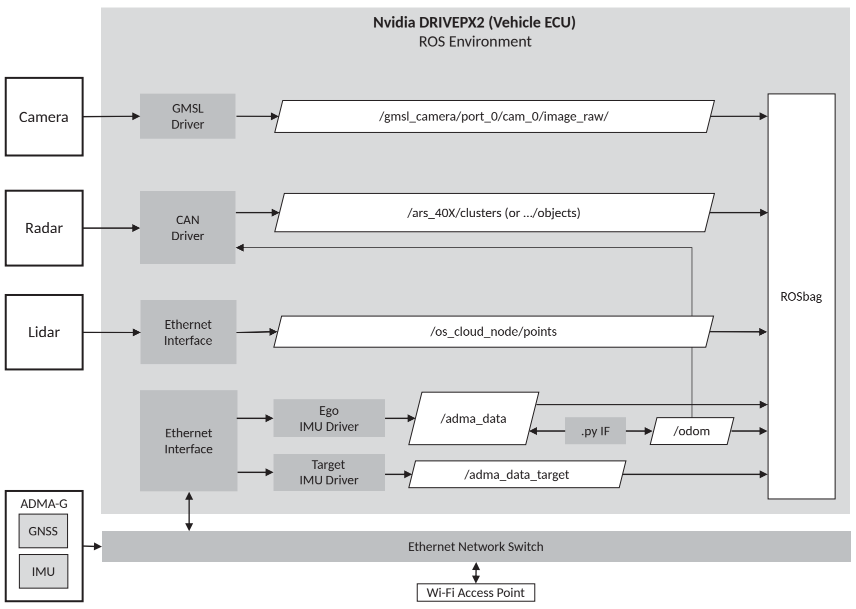


Source: (NETO et al., 2023).

Node X. Therefore, publishing to a topic is the same as sending data to that topic, and subscribing is the same as reading data from that topic. Publishers do not know who reads their messages, but subscribers know who published them. ROS was also used for the single timing frame for all frames. As shown in the diagram of the Figure 15, each sensor node published messages on a different topic. As we can see each sensor has a connection with the ECU, the Wi-Fi access point is to connect the sensor from the object of the scenario. It can be another ADMA-G or the SP80.

These messages are stored in a ROSbag, which is a file that stores all the data published on ROS topics.

Figure 15 – ROS environment



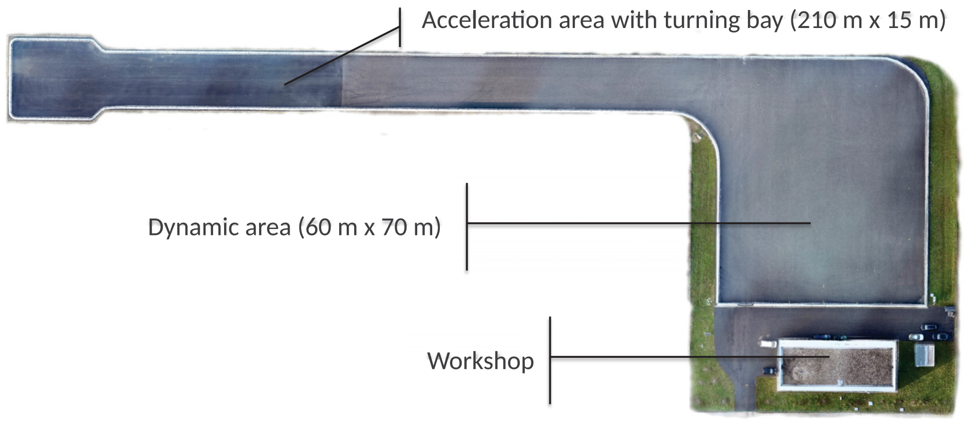
Source: (NETO et al., 2023).

3.3 TEST ENVIRONMENTS

To compare and validate real-world scenarios with simulations, the utilization of a controlled environment is crucial. Such an environment allows for control over significant variables, including scenario participants and even weather conditions (by generating artificial rain). This section will discuss both the real-world and laboratory environments.

3.3.1 Real-World

Figure 16 – The CARISSMA outdoor proving ground.


Acceleration area with turning bay (210 m x 15 m)
Dynamic area (60 m x 70 m)
Workshop

Source: CARISSMA.

The CARISSMA Institute for Automated Driving has an outdoor testing facility, illustrated in Figure 16, that was utilized as a proving ground for conducting our experimental scenarios. This testing facility is situated in Ingolstadt, Germany, and covers an area of 215 m in length and 15 m in width. In addition to this, there is a dynamic testing region measuring 60 m by 70 m. This facilities also provides validated targets from 4A [2] which are important for conducting tests involving collisions. In order to conduct experiments under rainy conditions, an artificial rain generation system was utilized, which allowed for controlled generation of rainfall. This system involved the use of a sprinkler system that was able to generate artificial raindrops.

3.3.2 Laboratory

The simulation software chosen was the IPG CarMaker (IPG, 2023). This software has support for HiL, target virtualization and Vehicle-in-the-Loop (VIL). The support to VIL is crucial once that the software receives the ego data from the real test and replicates all in the simulation. Therefore, even the variation in camera angle due to the acceleration and braking of the ego vehicle was replicated in the synthetic test runs.

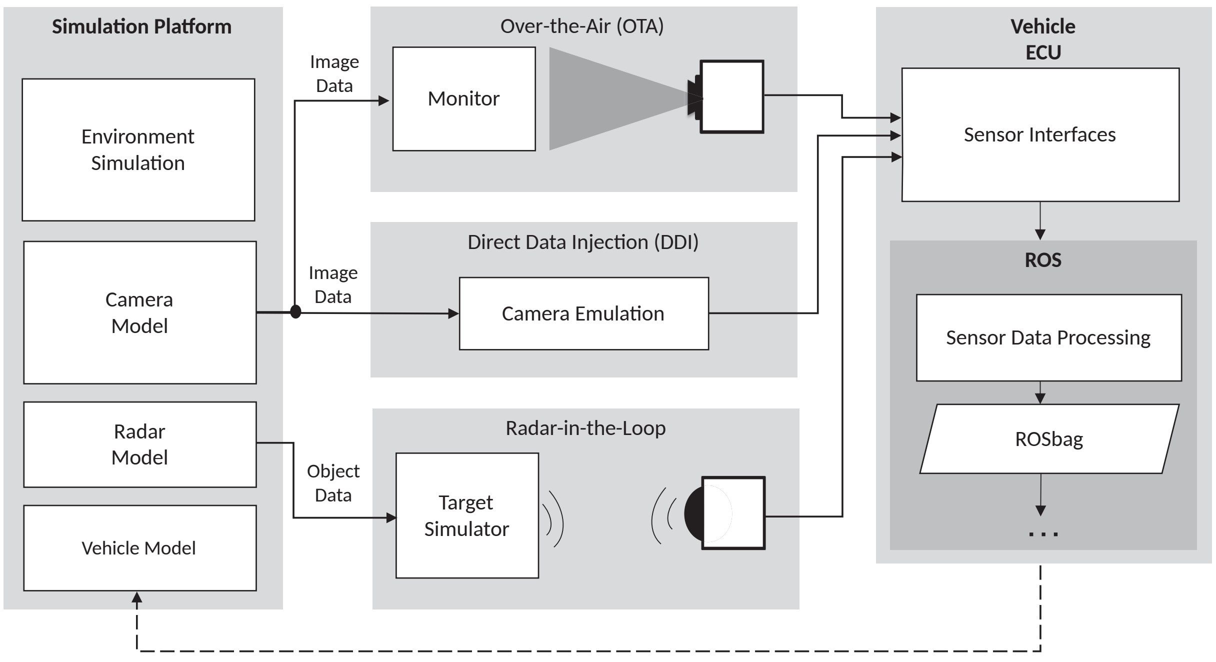
[2] 4activeSystems is a company that specializes in providing advanced solutions and technologies for the automotive industry.

Even the extrinsic calibration performed in the sensors of the real-world ego vehicle was replicated in the virtual tests.

The schema detailing the laboratory bench is presented in Figure 17. As it can be seen the setup is in a Hardware-in-the-Loop environment. However the loop is not completed because there is no driving function. The final diagram flow ends with the ROSbag where all the data is recorded. It was employed two different methods for the stimulation of the camera sensor. The first type involves Direct Data Injection (DDI) from the simulation using the IPG Video Interface Box (VIB). The second type is a camera OTA setup (REWAY et al., 2018) which uses the same camera as the real-world test scenario to record a video of the simulation displayed on a monitor inside a dark box. Both of them contain exactly the same frame image. The radar was mounted inside an anechoic chamber. The radar was installed in one side, and the stimulator in the opposite side. The stimulator used was a Vehicle Radar Test System (VRTS), developed by Konrad GMBH. It would receive data from the simulation, and by manipulating the EM waves from the radar, generates a virtual target with distance, velocity, acceleration and RCS provided. Figure 18 shown all the hardware used in the laboratory bench.

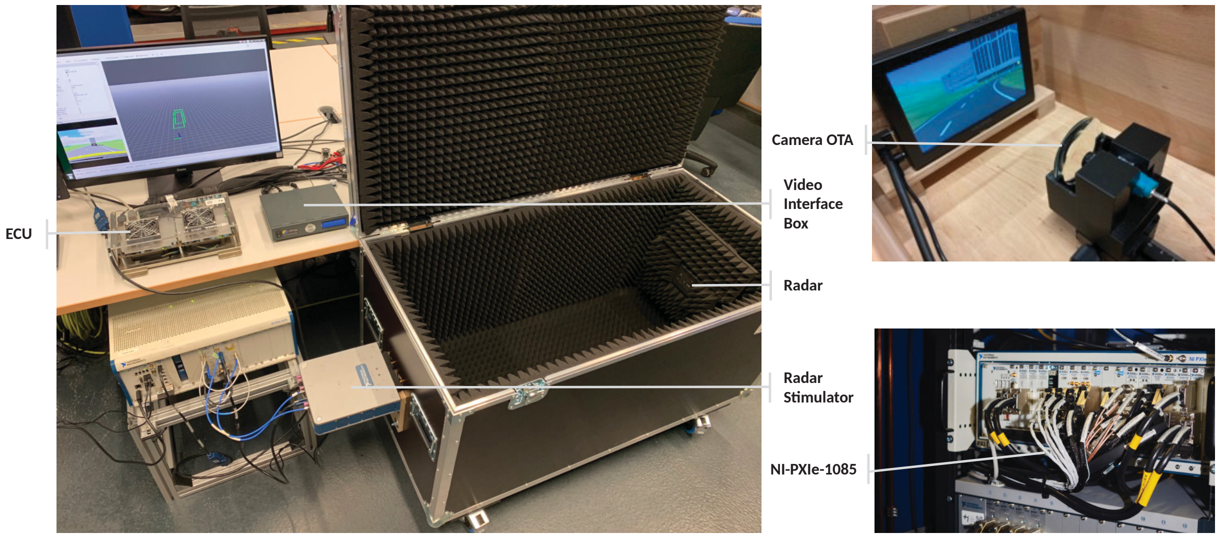
To maintain real-time performance, the NI-PXIe-1085 from National Instruments was utilized. With CarMaker's support, this real-time computer can calculate positions in real time and subsequently output the data to the sensor stimulators. These stimulators (VRTS, VIB and Monitor) are responsible for stimulating the sensors, and the resulting sensor data is recorded by the same ECU used in the real-world tests. Even the diagram

Figure 17 – Data acquisition in the Laboratory: Stimulation of environment sensors (virtual tests, but integrated hardware of camera and radar).



Source: (NETO et al., 2023).

Figure 18 – Laboratory bench.



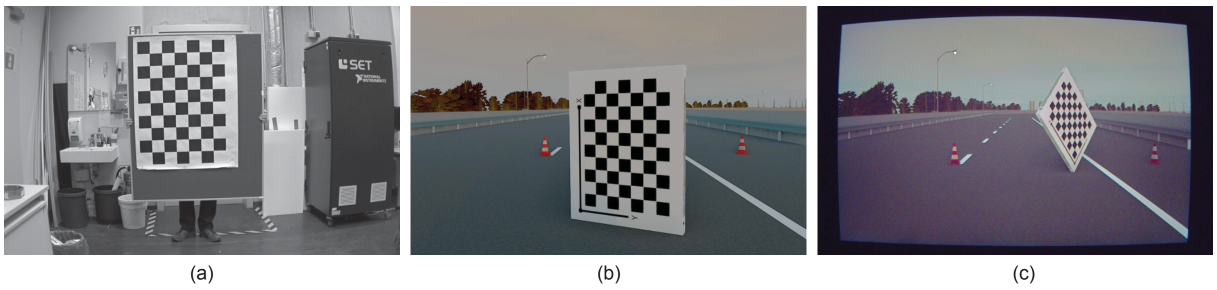
Source: CARISSMA Institute of Automated Drive.

shown in Figure 15 used in the real-world tests remains largely unchanged, with the exception of the LiDAR sensor, which was not replicated in the synthetic tests. And the position data of the objects in the scenario is sourced from the CarMaker simulation rather than the ADMA-G and SP80 sensors.

3.3.3 Intrinsic Camera Calibration

There are three types of cameras used in the test scenarios: the camera used in the real world and the two cameras used in the laboratory setup (VIB and OTA). The calibration process was performed using the Camera Calibrator app in Matlab. Multiple images of a checkerboard with a known size were captured by the cameras from different positions and angles. For the laboratory cameras, a 3D checkerboard with a predetermined size was inserted into the simulation, as depicted in Figure 19 (b) and (c).

Figure 19 – Intrinsic Camera Calibration.



Source: (NETO et al., 2023).

3.4 TEST SCENARIOS

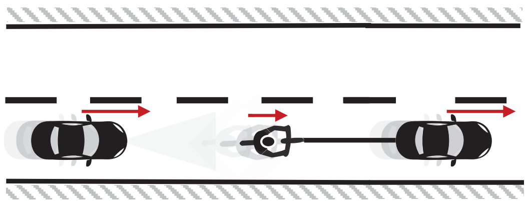
With the test environment established, the next step was to define the recorded test scenarios. A total of eight different test scenarios were recorded, featuring various scenes with cars, pedestrians, cyclists, and trucks. Some of the scenarios were inspired in the cataloged test scenarios of EURONCAP [3]. These scenarios can be divided into static ego scenarios and dynamic ego scenarios, depending on whether the ego vehicle is stationary or in motion.

3.4.1 Dynamic ego scenarios

3.4.1.1 CBLA

Inspired by EURONCAP scenarios, the Car to Bicycle Longitudinal (CBLA) test is an Automatic Emergency Braking (AEB) test where the ego vehicle approaches a bicycle with a velocity ranging from 25 km/h to 60 km/h, while the bicycle moves away from the car at 15 km/h. In this work, the specific test conducted was the CBLA-50, with the ego vehicle traveling at 50 km/h. The bicycle dummy utilized in the test was a validated dummy provided by the manufacturer 4A and was pulled with the assistance of another car equipped with a ADMA-G sensor to infer the position of the dummy.

Figure 20 – Car to Bicycle Longitudinal Adult (CBLA).



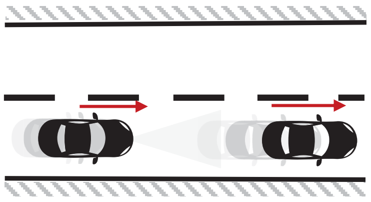
Source: (NETO et al., 2023).

3.4.1.2 CCRb

Also inspired by EURONCAP scenarios, the Car to Car Rear braking (CCRb) is a AEB test in which both vehicles are moving. The ego vehicle travels towards the target at a velocity of 50 km/h. The target car initially moves at the same speed but then decelerates either at a rate of -2 m/s² or -6 m/s² until it comes to a stop. This results in a motion where the target moves away from the ego vehicle until reaching a maximum distance, after which the ego vehicle approaches the target as it comes to a stop. This test allows for the evaluation of sensors under both conditions: a target moving away and a target approaching.

[3] European New Car Assessment Programme (EURONCAP) is an independent organization that conducts safety tests and provides ratings for cars sold in Europe. Its evaluations cover crashworthiness, occupant protection, pedestrian safety, and safety assist systems, aiming to promote the development of safer vehicles and inform consumers about their safety performance.

Figure 21 – Car to Car Rear braking (CCRb).

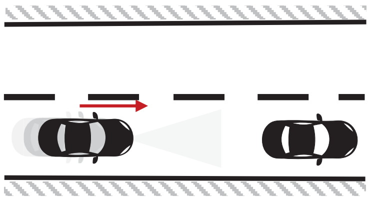


Source: (NETO et al., 2023).

3.4.1.3 CCRs

The last one inspired by EURONCAP scenarios, the Car to Car Rear static (CCRs) is an AEB test scenario where a target vehicle is stationary at a predetermined distance, while the ego vehicle approaches it at a velocity of 50 km/h. In this study, the test was conducted using a standardized dummy provided by 4A, which includes validated RADAR and camera targets.

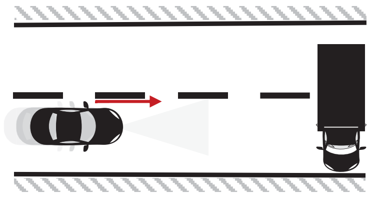
Figure 22 – Car to Car Rear static (CCRs).



Source: (NETO et al., 2023).

3.4.1.4 Truck perpendicular

Figure 23 – Truck perpendicular.



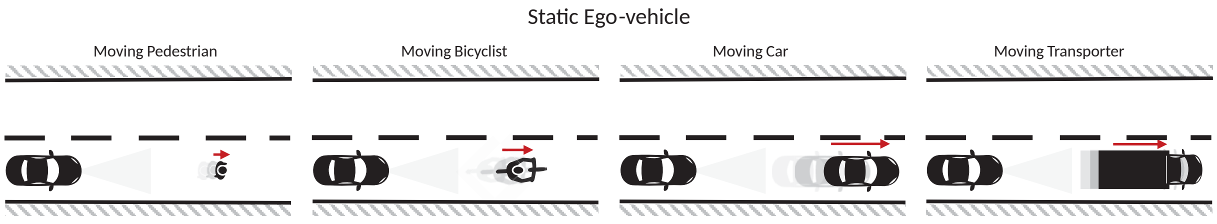
Source: (NETO et al., 2023).

Inspired by the first fatal collision involving a software-controlled car (CNN, 2016), this scenario involves a stationary truck positioned perpendicular to the road, Figure 23. The ego vehicle accelerates towards the truck and comes to a stop just before colliding with the side of the truck.

3.4.2 Static ego scenarios

With the ego vehicle stationary, four different types of objects (Figure 24) move away from the sensor setup. To obtain the position of these moving objects, the bicyclist, pedestrian, and truck were equipped with SP80 sensors, while the car was equipped with an ADMA-G sensor.

Figure 24 – Four different scenarios with the static ego-vehicle.



Source: (NETO et al., 2023).

3.5 DATA FORMAT

The data collected during the tests was stored in ROSbags. However, accessing this data can be impractical. Typically, RVIZ[4] is used as a visualization tool, but it requires running all the ROSbag data (one file which contains data from all the sensors). Therefore, it was necessary to convert the data into a well-structured and agnostic format that can be easily accessed by a broader audience. In the absence of a standardized format for this type of data, the Open Simulation Interface (OSI) serves as a suitable solution.

3.5.1 Google Protocol Buffers

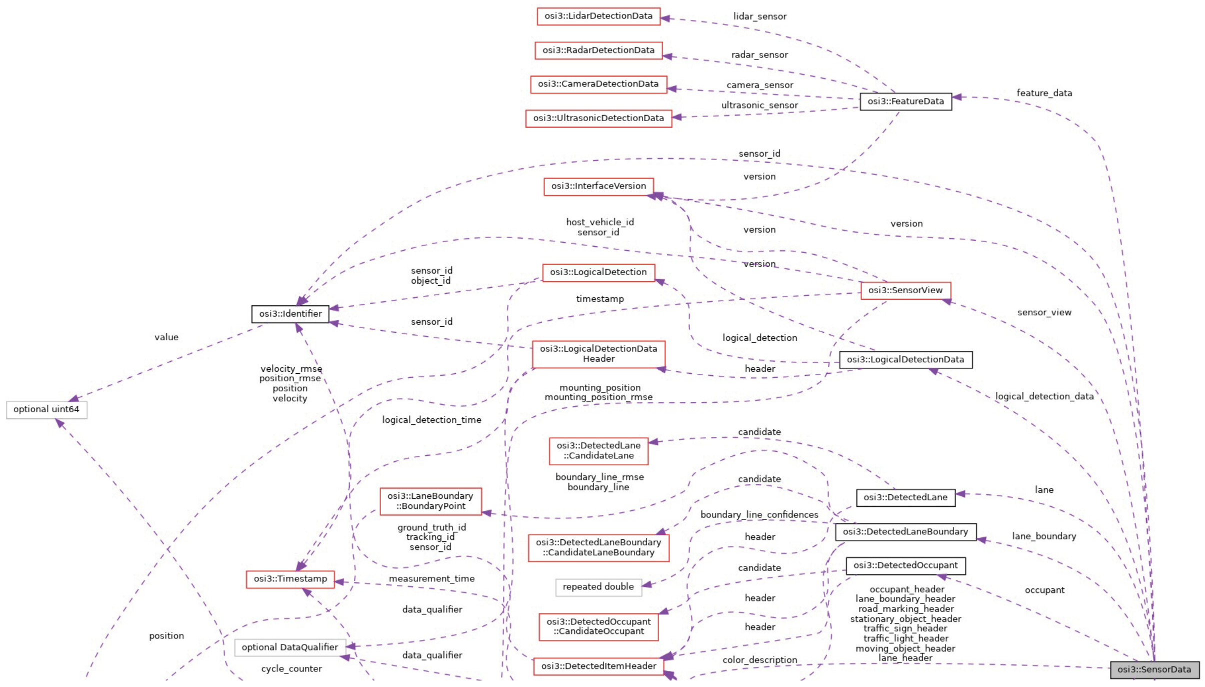
Google Protocol Buffers (protobuf) is a language-agnostic data serialization framework developed by Google. It provides a way to define the structure of structured data in a language-agnostic format and enables efficient serialization and deserialization of data for communication and storage purposes.

Protobuf enables the definition of data structures using a simple language known as Protocol Buffer Language. This definition acts as a contract or schema for the data. Once the structure is defined, the protobuf compiler can be utilized to generate code in multiple programming languages. This code offers convenient APIs for encoding and decoding data in the specified format.

Protobuf offers several advantages over other data interchange formats, such as JSON or XML. It provides a compact binary representation, resulting in smaller message sizes and faster serialization/deserialization. It offers strong type checking, efficient data parsing, and support for optional and repeated fields (GOOGLE, 2023). However, there is a limitation on the message size. By default is 64MB, it can be changed, but the problem is that you must always parse an entire message at once before start using any of the content. This means the entire message must fit into the Random Access Memory (RAM)

[4] RVIZ is a visualization tool in the Robot Operating System (ROS) ecosystem. It allows users to visualize sensor data, robot models, and other relevant information in a 3D environment.

Figure 25 – SensorData OSI diagram.



Source: (TIMO HANKE, NILS HIRSENKORN, CARLO VAN DRIESTEN, 2017).

3.5.2 Open Simulation Interface

The Open Simulation Interface (OSI) is an industry-standard interface developed by the Association for Standardization of Automation and Measuring Systems (ASAM). It serves as a standardized framework and protocol for communication and data exchange between various software components used in simulation and testing domains.

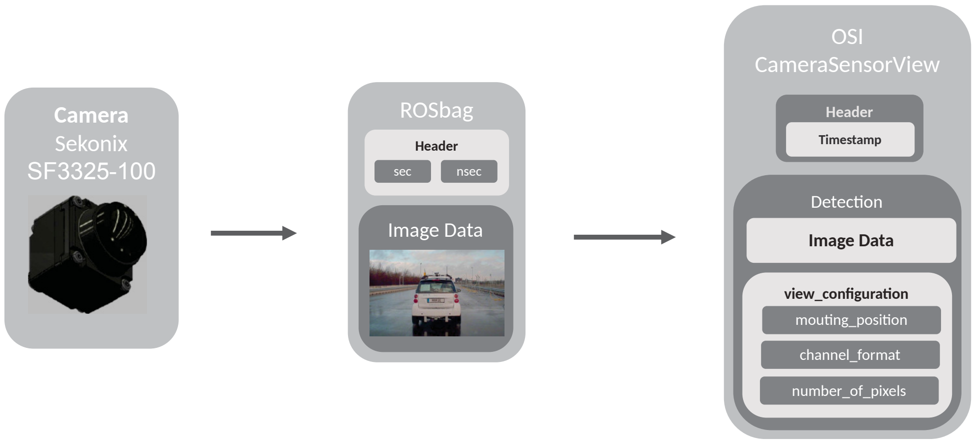
ASAM OSI started as a generic data exchange interface compliant with the ISO 23150[5] (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2021) logic interface for the environmental perception of automated driving functions in virtual scenarios.(TIMO HANKE, NILS HIRSENKORN, CARLO VAN DRIESTEN, 2017). Based on the Google Protocol Buffers. With OSI is possible not only store the data from the ego sensors, but also the measured object position (ground-truth) of the scenario. The OSI framework has a structured format for storing the data collected during testing. This format is organized into classes and subclasses. The top-level class that contains data from all sensors is called `SensorData`, Figure 25 shown part of the structure. The hierarchy of the classes descends from right to left. The words on the dashed lines represent the classes in the Protobuf schema. Within this class, there is a subclass called `FeatureData`, which further contains subclasses for radar data (`RadarDetectionData`) and LiDAR data (`LidarDetectionData`). These subclasses have repeating structures to store, for example, the multiple clusters detected by the radar at a specific moment in

[5] The document specifies the logical interface between in-vehicle environmental perception sensors (for example, radar, lidar, camera, ultrasonic) and the fusion unit which generates a surround model and interprets the scene around the vehicle based on the sensor data.

time.

OSI also defines the units of measurement and the coordinate system used for storing the data and provides well-defined structures, such as `Spherical3d`, which represents the location of clusters relative to the ego vehicle in spherical coordinates: elevation, azimuth, and distance. The timestamp in the OSI structure (`Timestamp`) is stored in seconds and nanoseconds. It begins from 0 and does not have a reference definition, such as the Unix epoch. The timestamp of all sensors is derived from the ROS timestamp. At each moment in time when the sensor publishes data in the ROS node, a timestamp is acquired. Since each sensor has a different data acquisition frequency, the timestamps are unique to each sensor.

Figure 26 – Camera data.



In the Figure 26 is shown how the camera frames was stored in the ROSbag and how it is stored in the OSI format. A sequence of these frames, each with its corresponding timestamp, forms a video.

Figure 27 illustrates how the radar data was stored in the ROSbag and how it is stored in the OSI format. A coordinate conversion was performed, transforming the data from rectangular coordinates in the ROSbag to spherical coordinates in the OSI. Additionally, the covariance was converted to position root mean square. In this case the radar was operating in cluster mode.

The radar also operates in object list mode, therefore the data is slightly different. In the Figure 28 is shown how the data was converted to the OSI format. In this case was generated a list of objects and they were stored in the class `DetectedMovingObject`.

In the OSI format the location of the sensor relative to the ego-vehicle is always stored in the field class `MoutingPosition`. The Figure 29 shown how the data was converted from the ROS format to the OSI. The field `Detection` is a repeated structure where all the points of the LiDAR detection are stored.

Figure 27 – Radar cluster data.

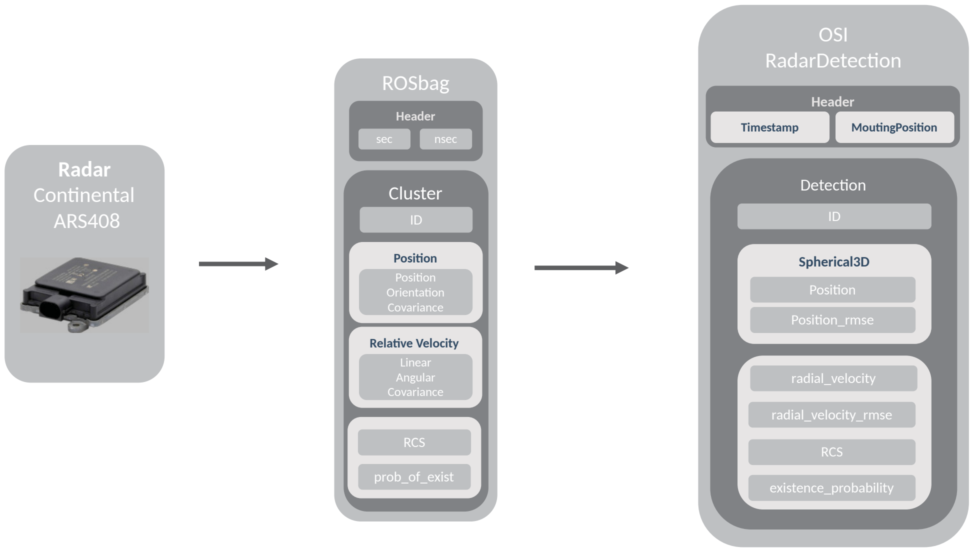


Figure 28 – Radar object list data.

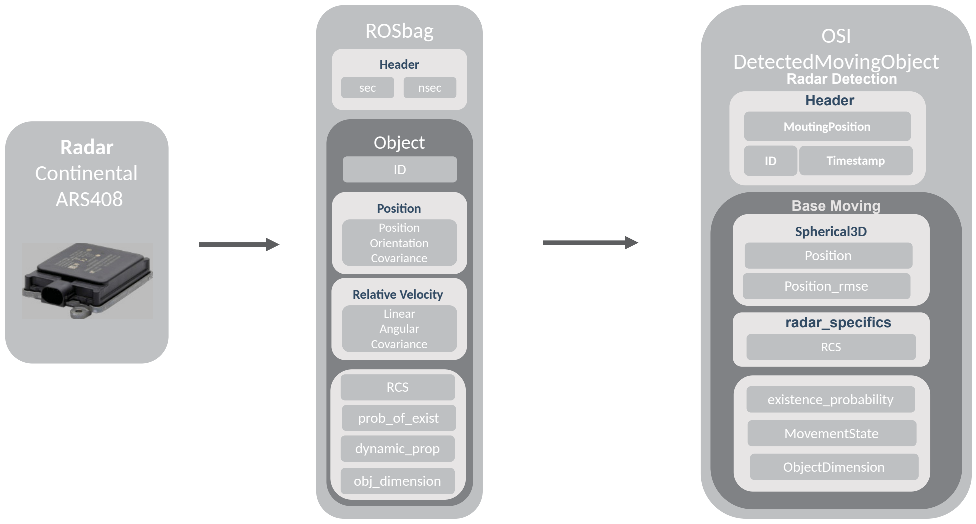
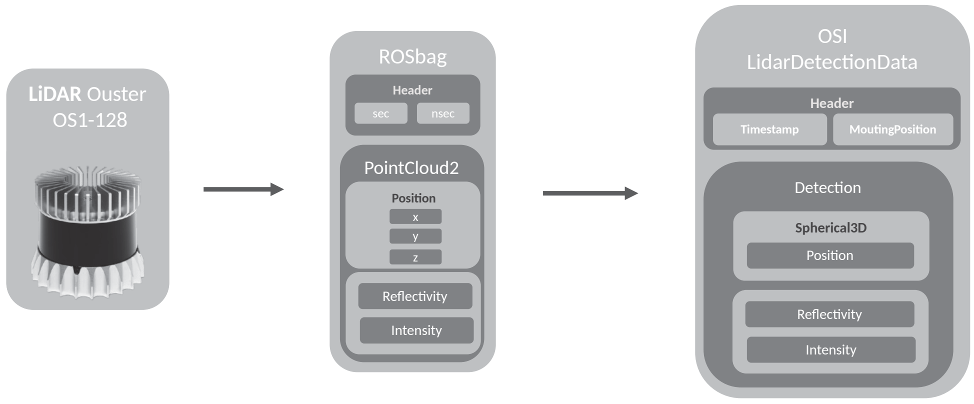
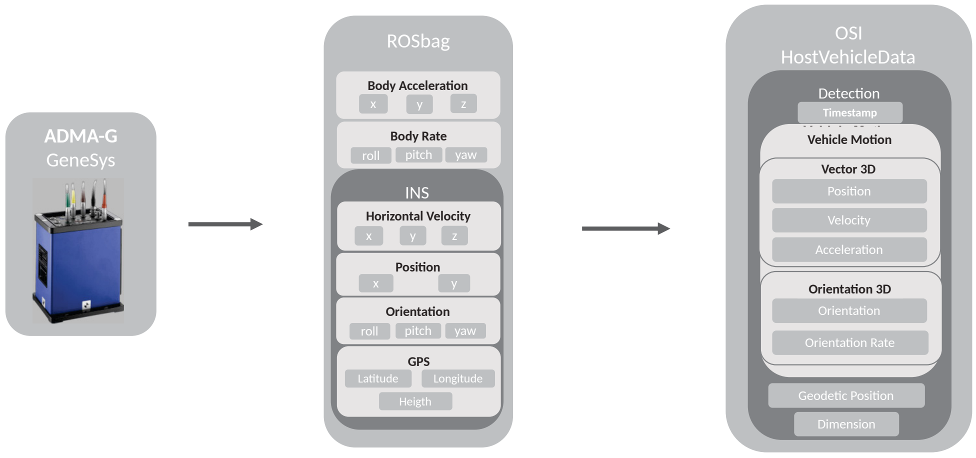


Figure 29 – LiDAR data.



The ADMA-G sensor provide not only the position data, but also acceleration, velocity and orientation angles. In the Figure 30 is possible to see how the data was converted from the ROSbag and how it is structured in the OSI format. The class

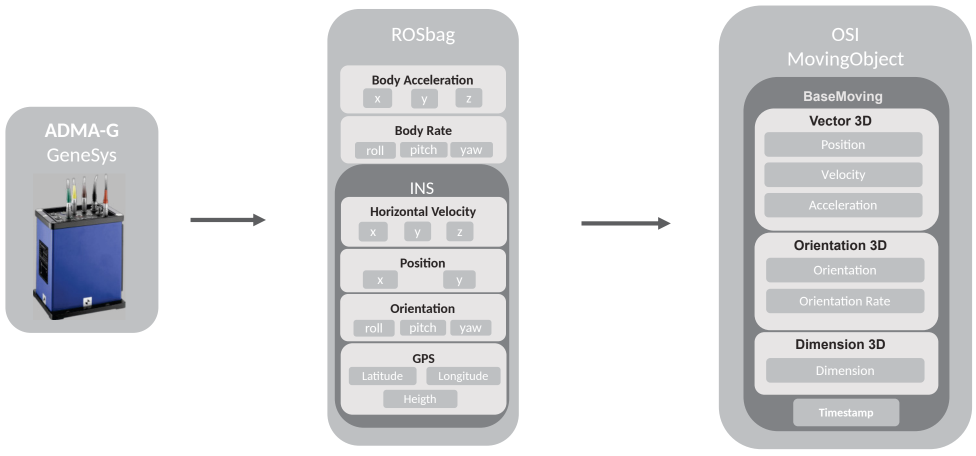
Figure 30 – ADMA-G ego data.



HostVehicleData has the structure to store the data from ego IMU/GNSS system. The OSI data follows the (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011), so the position of the car is set zero in the middle of the back axle of the ego-vehicle.

To store the position of the objects of the test scenarios, the class MovingObject was used. When the object of interest was the car, all the data collected by the ADMA-G was stored as shown in the Figure 31.

Figure 31 – ADMA-G car data.

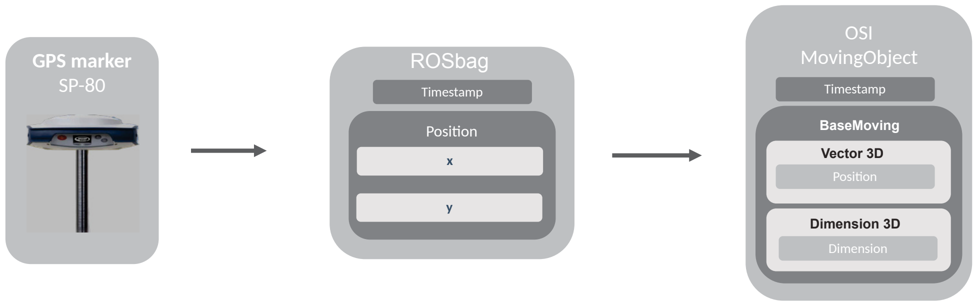


When the object of interest was the pedestrian, bicyclist or the truck. The SP80 was used and therefore was capted only the position data. Figure 32 shown how the data is structure in OSI.

To store the position of the 4A target in the CCRs scenario, was used the StationaryObject class that belong to the GroundTruth OSI class.

The timestamp for all the data is the ROS system timestamp. In other words, the timestamp for all the sensors is the time when the data was collected by the vehicle ECU.

Figure 32 – SP80 data.



Each sensor has an individual `.osi` file assigned to it. In the case of the radar, IMU, and ground-truth of objects, the `SensorDataSeries` is utilized. It is a time series of the `SensorData` and generate a list of detections contained within the protobuf message. However, for the Camera and LiDAR sensors, the protobuf has a default size limit of 64Mb. As a result, each detection is stored in its individual protobuf message and subsequently parsed into a single .osi file using the Python's `struct` package.

3.6 DATA ANNOTATION

Data annotation refers to the process of labeling or tagging data with meaningful and relevant information. In the context of automated driving, data annotation plays a crucial role in understanding and analyzing various aspects of the environment, such as object detection, scene understanding, and behavior prediction.

3.6.1 OpenLABEL

ASAM OpenLABEL is a standardized format and framework also developed by the ASAM. It is specifically designed for the annotation of data in the field of automated driving and related applications.

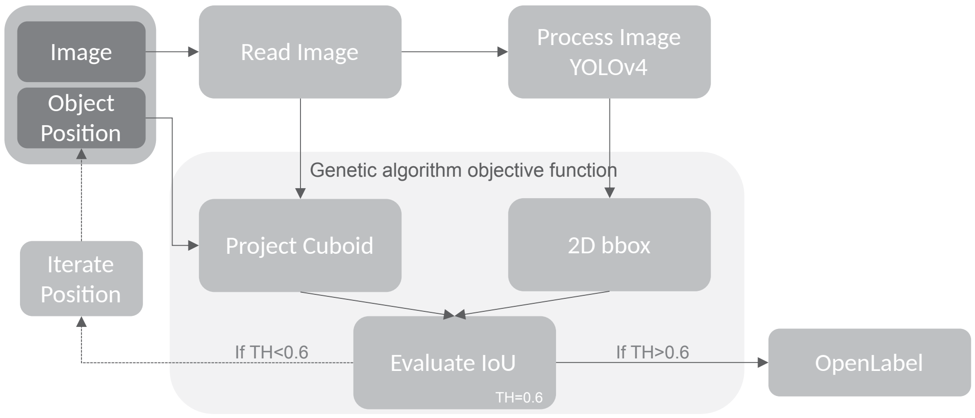
It provides a structured and standardized way to annotate sensor data, images, and other relevant information. It allows for consistent and interoperable annotations, enabling efficient data sharing and collaboration among different stakeholders in the automated driving ecosystem. ASAM OpenLABEL uses the JSON format and can therefore be easily parsed by tools and applications. It specifies which coordinate systems are used as reference for the label. It stores important information such as the camera intrinsic matrix and sensors synchronization.

3.6.2 Genetic Algorithm

A Genetic Algorithm (GA) is an optimization method that can solve both constrained and unconstrained problems. It is inspired by the process of natural selection in biological evolution. The algorithm works by iteratively modifying a population of individual solutions. In each iteration, individuals are randomly selected from the current population and serve as parents to create offspring for the next generation. With each generation, the population evolves towards an optimal solution.

3.6.3 Annotation process

Figure 33 – Labelling schema.

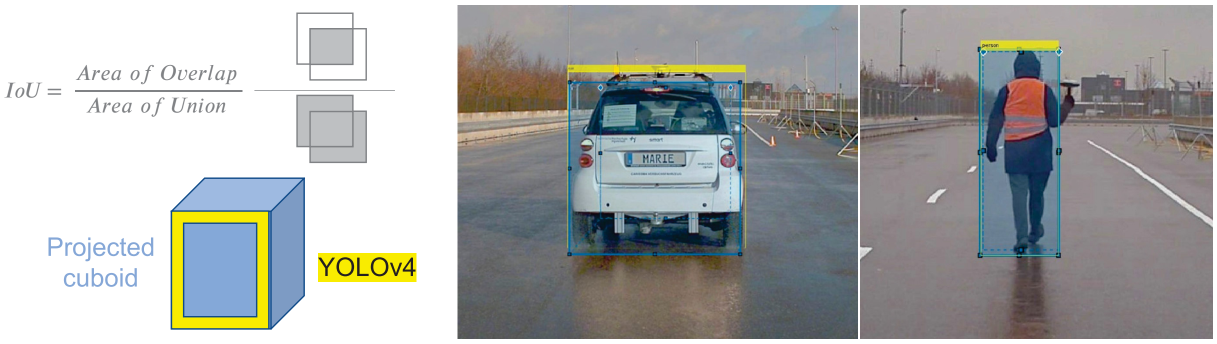


One of the biggest advantages of executing the test runs in a controlled environment is known all the actors involved. Once that the position of the ego and the objects of interest were collected, it is possible to use this information to execute the data annotation.

At a first sight, having the intrinsic camera matrix, the location of the camera in relation to the world, and the position of the object of interest, one could simply project the object dimensions onto the image and get the bounding box. However, there are many uncertainties involved in the process that must be taken into account. Manually supervise this process is time consuming and can lead to more errors. The solution was a semi-supervised method described in Figure 33. The script was implemented using Matlab, which provides convenient packages for development purposes (Automated Driving Toolbox). The initial step involved importing the OSI data into Matlab. Since Matlab currently lacks direct support for Google Protocol Buffers, the data was effortlessly converted to JSON format for integration with Matlab. Subsequently, the algorithm synchronized the timestamps of the sensor data, extracted the ego and object of interest positions, and calculated the relative position of the object with respect to the ego. Leveraging this information, a cuboid was projected onto the camera image using the camera's intrinsic matrix.

To supervise this process, the Matlab object detector utilizing YOLOv4 with a pre-trained network for autonomous driving applications was employed. To enhance efficiency, a subset of frames (1 frame selected every 75 frames) from each test run, capturing different object distances relative to the ego, was selected. These frames were then processed through YOLOv4, generating 2D bounding boxes. These bounding boxes, along with the previously described cuboid projection, were passed to the objective function of a genetic algorithm. The fitness function allowed for slight variations in the object's position, with the output to be optimized being the Intersection over Union (IoU) between the front face of the cuboid and the bounding box for each frame. Figure 34 illustrates the calculation of IoU. The average IoU across the selected frames was computed. If the IoU was unsatisfactory, the object's position (x, y) was adjusted by the genetic algorithm, and the process iterated until a satisfactory IoU result was achieved. The initial population size of the GA was set to 200, and two stopping criteria were employed: a fitness limit (IoU of 0.9 or higher) or a maximum time of 300 seconds.

Figure 34 – IoU between projected cuboid and YOLOv4 detection.



Once achieved a satisfactory result for the IoU the correction on the position is constant over all the frames of the test run. The cuboid is stored into the OpenLABEL format using real-world coordinates. The original positions of the objects are stored in the IMU files of the dataset for each scenario.

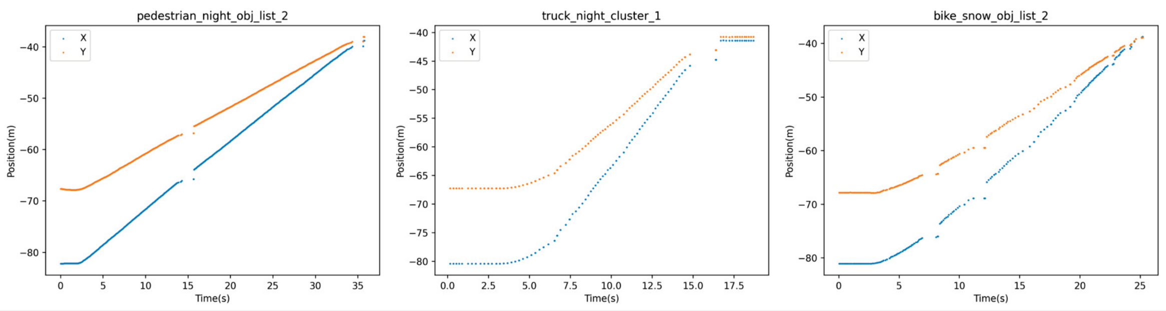
3.6.4 Timestamp synchronization

The OpenLABEL format includes a structure to store sensor data synchronization, considering that each sensor has a different capture frequency. The approach taken was to match the sensor data indexes based on the closest timestamp and store this synchronized data in OpenLABEL format. The synchronization was established using the camera's frame as a reference for the radar, ADMA-G sensor data from the ego vehicle, and the object position. Similarly, the radar data was used as a reference for synchronizing the camera data, ADMA-G sensor data from the ego vehicle, and the object position. Generating two OpenLABEL files, camera and radar.

3.7 DATA PROCESSING

During the data collection, some issues were encountered. The ground-truth position of the objects in the scene was transmitted to the ego ECU via Wi-Fi, as depicted in Figure 15. However, there were instances where the transmission failed, resulting in the non-recording of the position data. This issue was more prevalent when the SP80 sensor was used, but it also occurred with the ADMA-G sensor in scenarios where the car was the object of interest. Figure 35 illustrates the plot of position X and Y over time, demonstrating significant data gaps.

Figure 35 – SP80 raw data.



Source: Author (2023).

The initial task involved identifying the data gaps. The ADMA-G sensor operates at a capture frequency of 100 Hz, and during test runs where data gaps exceeded 150 milliseconds, a warning message was included with the data. For scenarios involving the SP80 sensor, data interpolation was performed to fill these gaps. If the script detected a gap exceeding 500 milliseconds, a linear interpolation model was utilized to fill the gap while maintaining the original data frequency provided by the sensor. A warning message was included with the data, indicating the specific time instance when the data gap occurred.

Another correction made to the objects position data, was relative to the reference point. Except for the ego-vehicle, the reference point for the position is the center of the object. It was necessary to know where the sensors were positioned on the object to correct the position according to the OSI model.

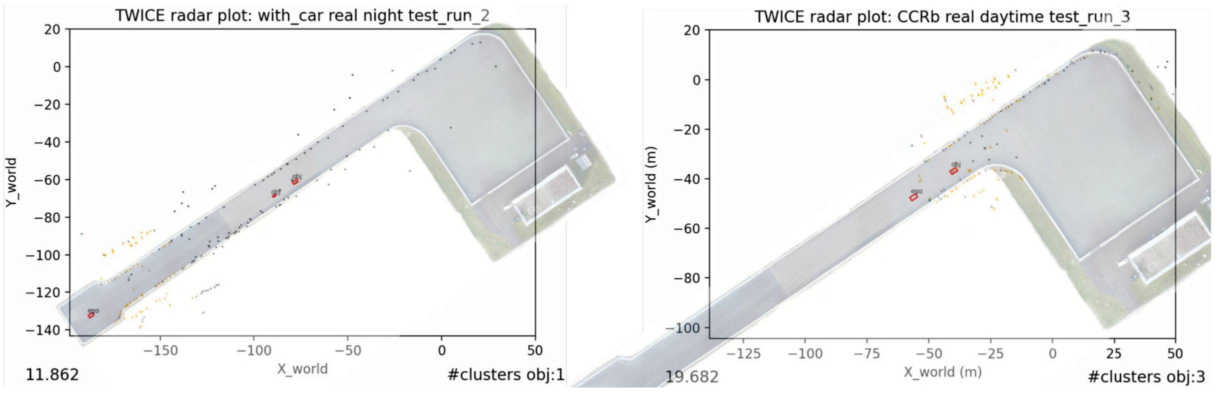
The CBLA test scenario was recorded using a 4A dummy riding a bicycle, which was being pulled by a car equipped with the ADMA-G sensor. Ideally, the dummy would be placed on a platform with a GNSS sensor to accurately determine its position. However, in this case, the position of the dummy was estimated using the position data acquired by the sensor inside the car. Both the car and the dummy (post generated) position are stored in the dataset.

During the test runs conducted with the Truck as the target, its license plate was visible in the captured frames. However, since the data would be made public, it was necessary to blur the license plate to ensure privacy. While typically a license plate detection algorithm trained for this purpose would be used, none of the publicly available algorithms were able to blur the license plates in all the required frames. As a solution, the cuboid generated during the annotation process was utilized to locate the license plate in each image and apply the necessary blurring.

3.8 DATA EVALUATION

To utilize the generated dataset, this study employs the YOLOP algorithm for comparing results between simulation and reality. As illustrated in Figure 5, the algorithm performs lane line segmentation, drivable area (free-space) segmentation, and object detection. However, in order to assess the algorithm's performance, it is crucial to have ground-truth information for these parameters. The data annotation process provides the necessary information for evaluating object detection. For the drivable area and lane line segmentation, it was necessary to create ground-truth data to both real and simulated data.

Figure 36 – Track ground-truth with Radar data.



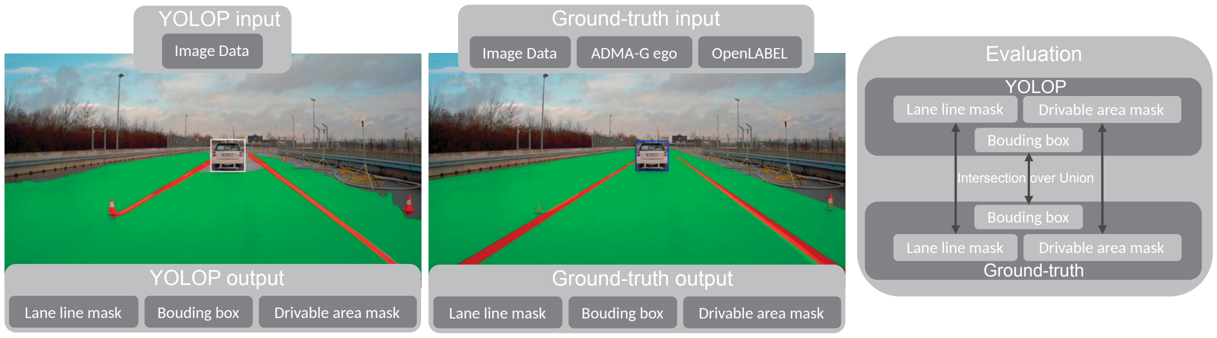
Source: Author (2023).

To generate the ground-truth data for drivable area and lane line segmentation, radar data and the ego vehicle's ADMA-G data were utilized. Since no measurement of the track has been carried out, sensor data alone was sufficient to determine the track's coordinates. In Figure 36, there is an illustration of the process, which involved projecting radar clusters onto the ego-vehicle's position. With this information, the boundaries of the real-world track could be estimated and then this data was projected onto the camera image. Lane line segmentation required estimating the lane's position through multiple test runs, as direct measurement wasn't available during testing. By determining the position of the lane line and the real-world track boundaries, ground-truth data could be generated even while the ego-vehicle was in motion.

The YOLOP algorithm was modified to export the detections in the form of segmentation masks. These masks are represented by boolean matrices with the same dimensions as the image. In the segmentation masks, a pixel value of 1 indicates the presence of the drivable area or the lane line. For each frame, two segmentation masks are generated: one for the drivable area and another for the lane line. The bounding boxes generated, was also exported.

A script was developed to conduct the evaluation process. First, a test run is selected, and the YOLOP algorithm is applied to the camera frames, generating

Figure 37 – IoU between ground-truth and YOLOP.

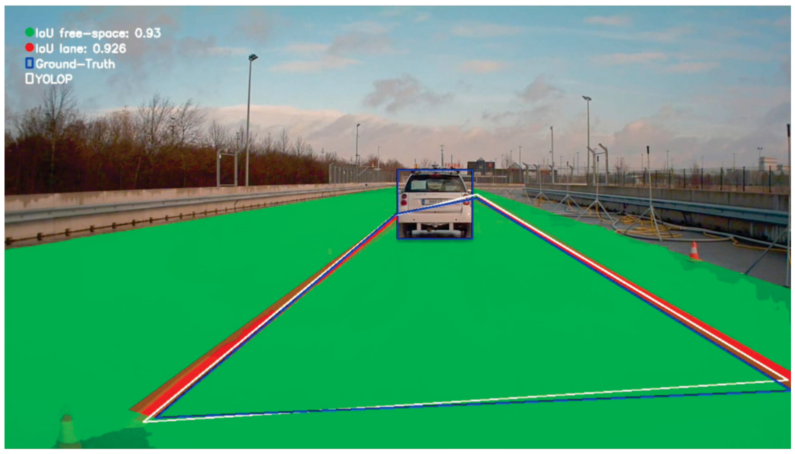


segmentation masks. Next, ground-truth segmentation masks are created in the same format. The third step involves comparing the segmentation masks and object detection results with the cuboids stored in the OpenLABEL files. Figure 38. The metric used to compare the ground-truth with the detection is once again the Intersection over Union (IoU). For the drivable area, it was performed the direct intersection and union of the boolean masks to obtain the IoU. However, for the lane line segmentation, this same calculation did not adequately represent how well the lane line segmentation performed by the YOLOP algorithm. The solution was to use the lane line detections to draw a polygon of the lane and then calculate the IoU of the polygon generated by the ground-truth with the polygon of the YOLOP lane line detection, as shown in

To evaluate the Camera Over-the-Air data, it was necessary to rectify the images to exclude the black borders (Figure 19 (c)) and then be able to project the ground truth. Once rectified, it was necessary to re-calibrate to obtain the new intrinsic matrix. The same images, underwent the same rectifying process, used in the original calibration were utilized.

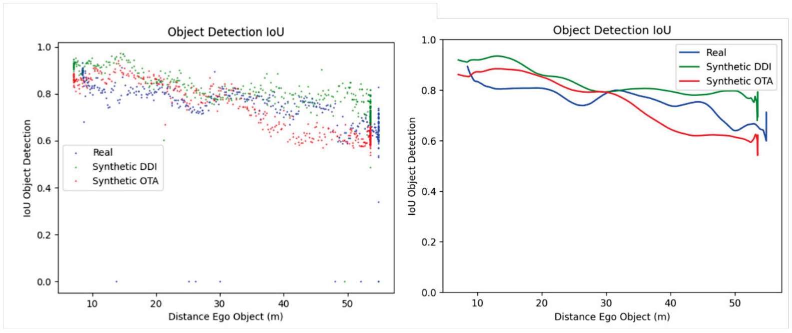
Smoothing. After obtaining all the data, it was observed that there was a need

Figure 38 – Lane polygon YOLOP/Ground-truth.



Source: Author (2023) (NETO et al., 2023).

Figure 39 – Raw Data vs. Smoothed Data



Source: Author (2023).

to apply a smoothing technique to facilitate a better comparison between the real and synthetic data. The choice was made to use LOESS (Locally Weighted Scatterplot Smoothing), a non-parametric regression technique used to model the relationship between two variables in a scatterplot. In Figure 39, it can be observed both the raw data and the results of the smoothing technique.

3.8.1 Simulation-to-reality gap

Since all test runs recorded in the real proving ground have corresponding digital twins, the evaluation method described above allows the comparison of scores between the same test runs, making it possible to estimate the simulation-to-reality gap.

Before delving into the estimation, it is important to define what is the simulation-to-reality gap. This measure serves to determine the degree to which a simulation accurately represents a real-world scenario. With a near-perfect simulation, it is possible to skip testing in the real environment when evaluating the proposed algorithm. Furthermore, a reliable simulation allows the testing of scenarios that may be physically or ethically impossible to execute in reality.

To estimate the simulation-to-reality gap, the IoU for object detection, drivable area, and lane line segmentation was calculated frame by frame:

$$IoU = \frac{\text{YOLOP} \cap \text{Ground-Truth}}{\text{YOLOP} \cup \text{Ground-Truth}} \tag{3.1}$$

By plotting the data from the real test alongside the simulation, it is possible to compare the performance of the simulation.

To quantify this data, a Root Mean Square Error (RMSE) analysis was conducted on the real data in comparison to both synthetic data (DDI and OTA):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{3.2}$$

The calculation involved comparing the IoU values frame by frame for Object Detection, Drivable Area, and Lane Line segmentation within each scenario.

4 RESULTS

With all the data organized and annotated, this chapter will discuss the resulting dataset and its utilization for measuring the simulation-to-reality gap.

4.1 CALIBRATION

As described in Subsection 3.1.6 the extrinsic calibration of the Camera, Radar and LiDAR were performed using two different algorithms.

Figure 40 – Frames transformation.

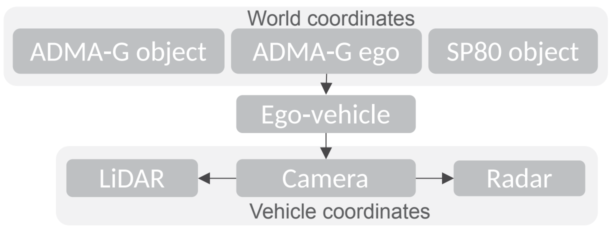


Figure 40 depicts the structure of the transformation frames. Using the ADMA-G sensor, it is possible to determine the world coordinates. The NVIDIA extrinsic calibration tool was employed to ascertain the camera's position relative to the midpoint of the ego-vehicle's back axle. Then the algorithm developed by (DOMHOF et al., 2019) was utilized to derive the relative positions of the Radar and LiDAR with respect to the Camera. Table 3 contains the final results of the extrinsic calibration of the sensors, all of them taking the midpoint of back axle as reference.

Table 3 – EXTRINSIC CALIBRATION RESULTS.

	X	Y	Z	Roll	Pitch	Yaw
Camera	1.3199	-0.0716	1.2784	0.4669	3.4517	2.8506
Radar	2.7774	0.0407	0.5082	0.0174	0.0715	-0.0094
LiDAR	1.2144	0.0506	1.6251	-0.0021	0.0207	-0.0016
ADMA-G	-0.126	0.099	0.737	–	–	–

Source: CARISSMA (2022)
Caption: Extrinsic calibration results. (X, Y and Z in meters. Angles in degrees.)

4.2 DATA STRUCTURE

The data is stored in a structure depicted in Figure 41. There are two types of scenarios: dynamic_ego, where the ego is moving, and static_ego, where the ego is stationary. Each one contains four scenarios.

Figure 41 – Directory structure of the dataset.

```
TWICE
└── Scenarios
    ├── dynamic_ego
    │   ├── CBLA(...)
    │   ├── CCRb
    │   │   ├── real
    │   │   │   ├── rain
    │   │   │   │   ├── cluster
    │   │   │   │   │   ├── test_run_1
    │   │   │   │   │   │   ├── Camera
    │   │   │   │   │   │   │   ├── camera_sv_350_300.osi
    │   │   │   │   │   │   │   └── open_label_camera.json
    │   │   │   │   │   │   ├── IMU_ego
    │   │   │   │   │   │   │   └── ego_sv_350_300.osi
    │   │   │   │   │   │   ├── IMU_obj
    │   │   │   │   │   │   │   └── obj_sv_350_300.osi
    │   │   │   │   │   │   ├── Lidar
    │   │   │   │   │   │   │   └── lidar_sd_350_300.osi
    │   │   │   │   │   │   └── Radar
    │   │   │   │   │   │       ├── radar_sd_350_300.osi
    │   │   │   │   │   │       └── open_label_radar.json
    │   │   │   │   │   ├── test_run_2(...)
    │   │   │   │   │   └── test_run_3(...)
    │   │   │   │   └── object_list(...)
    │   │   │   ├── daytime(...)
    │   │   │   ├── night(...)
    │   │   │   └── snow(...)
    │   │   └── synthetic
    │   │       └── (...)
    │   ├── CCRs(...)
    │   └── truck_perpendicular(...)
    └── static_ego
        ├── bike(...)
        ├── car(...)
        ├── pedestrian(...)
        └── truck(...)
```

The user must first select the desired scenario, then choose between a real or synthetic scenario, followed by the weather condition, radar operation mode, and finally the test run. The directory structure for all scenarios is the same as depicted for the CCRb. Inside the Camera directory, there are the `.osi` files containing the images and the OpenLABEL `.json` files storing the annotation data. The Radar folder also contains two files: one `.osi` file with the radar data and one OpenLABEL `.json` file storing the radar's synchronization with the rest of the sensors. The left sensors (ADMA-G, LiDAR and SP-80) have their respective data stored in `.osi` files as well. For the synthetic data, there are two types of camera data (DDI and OTA), each stored in separate directories. Each camera directory contains its respective `.osi` and OpenLABEL `.json` files.

The dataset contains 221.495 camera frames of which 99.516 are collected in the real proving ground and 121.979 generated in the laboratory. This frames make part of 289 test-runs, that totals more than 2 hours of recording. In the Table 4 the distribution of the test runs according to the scenarios and weather conditions is shown. 135 were recorded in the real proving ground and 154 in the laboratory. This additional synthetic test runs is due to the CBLA tests in laboratory. Were made with and without the car pulling the bicyclist.

Table 4 – NUMBER OF TEST RUNS IN EACH SCENARIO.

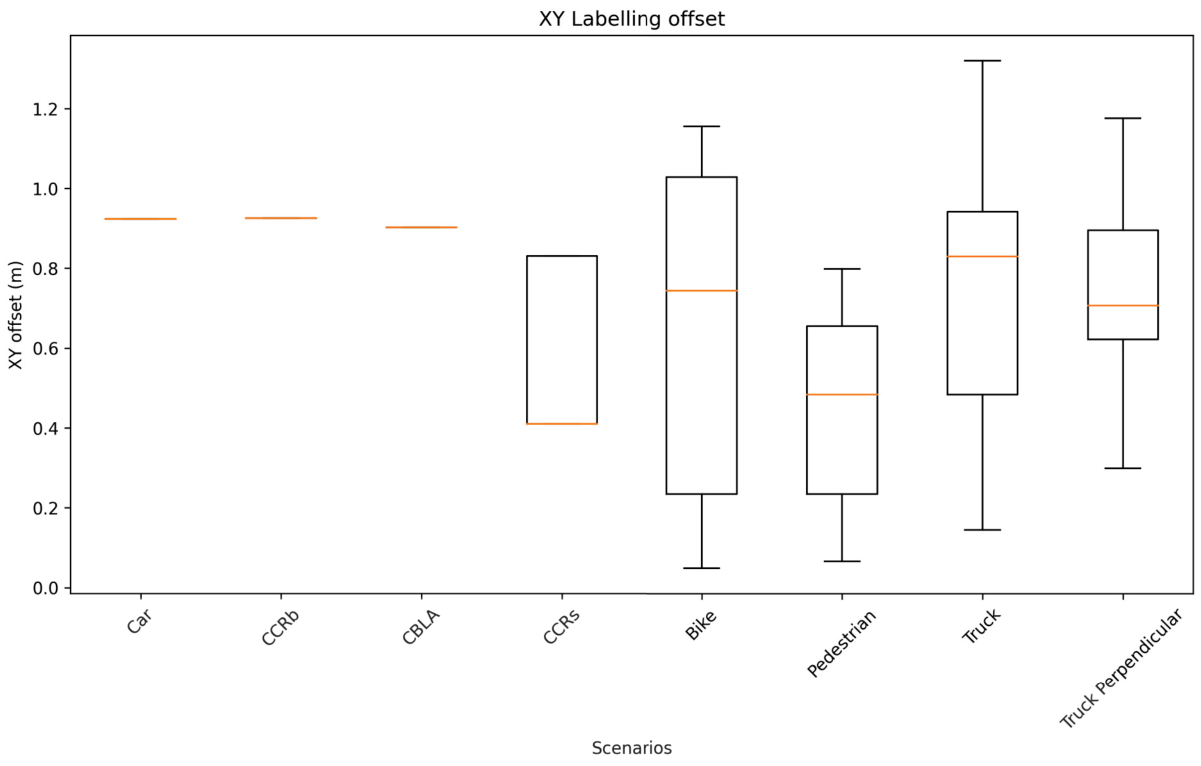
	daytime		night		rain		snow	
Type	real	synthetic	real	synthetic	real	synthetic	real	synthetic
CBLA	7	14	6	12	6	12	-	-
CCRb	6	6	6	6	6	6	-	-
CCRs	5	5	6	6	6	6	-	-
Truck perpendicular	-	-	4	4	-	-	4	4
Bike	6	6	5	5	6	6	4	4
Car	6	6	6	6	6	6	4	4
Pedestrian	6	6	6	6	6	6	1	1
Truck	4	4	3	3	-	-	4	4

Source: Author (2022).

4.3 DATA ANNOTATION

Following the method described in section 3.6, it was possible to perform data annotation. Figure 42 illustrates the Root Mean Square (RMS) of x and y offset applied to the object's position. It demonstrates that objects relying on the SP80 sensor (Bike, Pedestrian and Truck) exhibited greater variation in the position offset across the test runs. This can be attributed to inconsistencies in maintaining the sensor's mounting position throughout the tests. In the case of the bicycle, the sensor was installed on the handlebar, but it did not remain in a fixed position during all the tests. For the pedestrian, the sensor was held in the hands, making it impossible to maintain consistency. Another

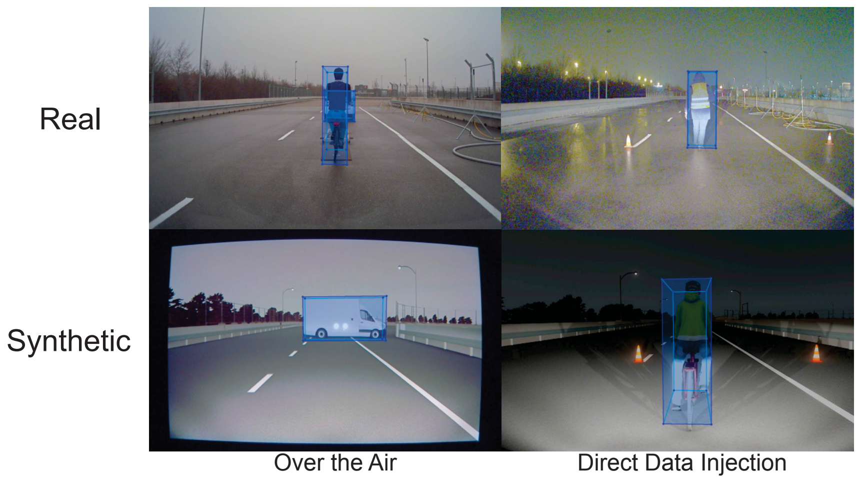
Figure 42 – Object position offset.



Source: Author (2023).

factor to consider was the need for data interpolation. In scenarios where the ADMA-G captured the object position (Car, CCRb and CBLA), it was observed greater consistency with no variation in the x and y offset across different test runs. This was due to the fixed positioning of the sensor. Once an optimal offset was determined, it remained constant, eliminating the need for the genetic algorithm to vary it. The same applies to the CCRs scenario, where the target was static. There is only two different positions and can be explained by a displacement of the target during the test runs.

Figure 43 – Projected cuboid onto objects.



Source: Author (2023).

Figure 43 displays the final result of the annotation data process. The left column in the first row is the CBLA scenario, and the right column is the pedestrian scenario, both recorded in the real proving ground. In the second row are scenarios captured in the laboratory, the left column is the truck perpendicular scenario captured by the Camera OTA setup, and the right column is the bicyclist scenario captured using the Camera DDI setup.

Figure 44 – OpenLABEL structure example.

```
{
  "openlabel": {
    "metadata": {
      "schema_version": "1.0.0"},
    "frames": [
      {
        "frame_properties": {
          "timestamp": 0.012592077,
          "Streams": {
            "Camera1": {
              "stream_properties": {
                "sync": {
                  "frame_stream": 0,
                  "timestamp": 0.012592077
                },
                "intrinsics_pinhole": {
                  "camera_matrix_3x4": [...],
                  "distortion_coeffs_1xN": [...],
                  "height_px": 1208,
                  "width_px": 1920
                }}},
            "IMU_ego": {
              "stream_properties": {
                "sync": {
                  "frame_stream": 1,
                  "timestamp": 0.008427381
                }}},
            "IMU_obj": {...},
            "Radar": {
              "stream_properties": {
                "sync": {
                  "frame_stream": 0,
                  "timestamp": 0.051609992
                }}}}},
        "objects": [
          {
            "name": "Car_1",
            "type": "Car",
            "object_data": {
              "cuboid": {
                "name": "cuboid",
                "val": [136.0989,2.2972,0.6804,
                        0,0,0.7900,
                        2.695,1.559,1.565]
              }}},(...)
```

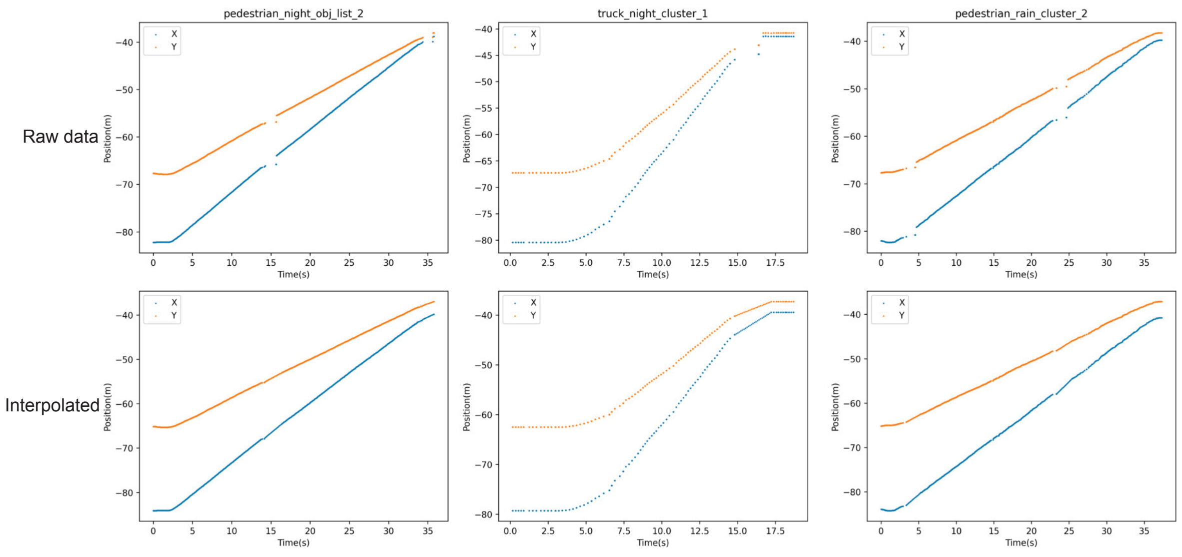
Figure 44 displays an example of the fields used in the OpenLABEL files. There are multiple frames, each containing properties such as timestamp, camera intrinsics, synchronization with other sensors, and the objects present. The cuboids are stored within the object field, and the vector includes the following parameters: X, Y, Z, width, length, height, roll, pitch, and yaw.

The Radar sensor has an OpenLABEL file that contains only the synchronization with the others sensors, without any information about the objects.

4.4 DATA PROCESSING RESULTS

As explained in section 3.7, the data from the SP80 sensor presents some gaps. The first row of Figure 45 shows the raw data, while the second row displays the interpolated data generated to fill these gaps. This interpolated data was used to generate the cuboids presented in section 4.3. Therefore, it was tested and validated.

Figure 45 – Raw vs Interpolated position data.



Source: Author (2023).

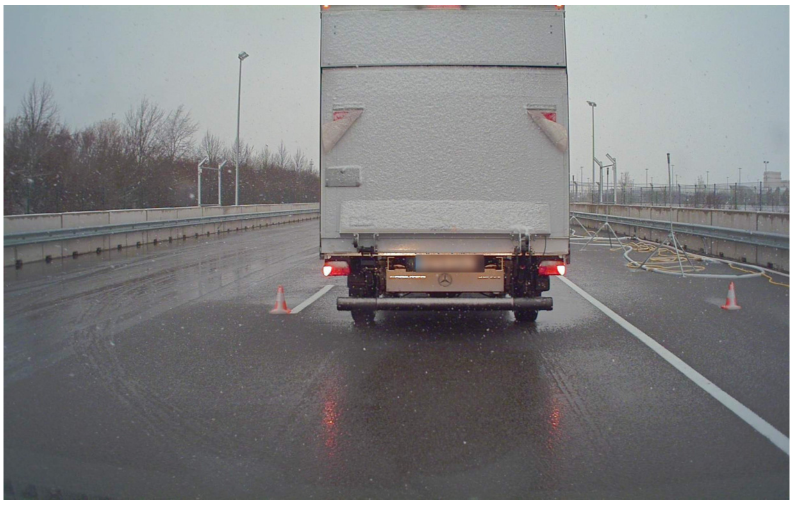
The result of the license plate blur process can be seen in Figure 46. As previously described, the data from the SP80 sensor was utilized to generate the blur effect.

4.5 DATA VISUALIZATION/USABILITY

Once all the data was annotated, structured, and organized in an agnostic format, it was necessary to develop tools to use and visualize the data. A Python notebook was created for this purpose.

File selector. With the structured data in place, it is now possible to select the desired scenario using a script and access all sensor data. The `file_selector.py`

Figure 46 – Blurred license plate.

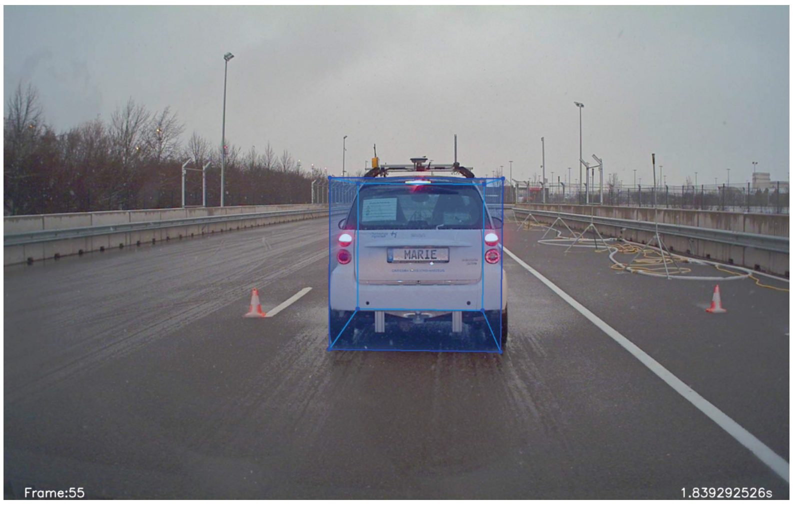


Source: (NETO et al., 2023).

script allows users to choose the desired scenario along with different parameters. If the user selects a combination of parameters that were not recorded, an error will be returned. The script also provides warnings about the scenario, such as the lack of data in the ground-truth sensors and the interpolation performed for the SP80 sensor. Following is the code part of the Python notebook:

```
weather = "snow" # daytime, rain, night, snow
scenario = "truck" #CCRb, CCRs, car, bike, pedestrian,(...)
scenario_type = "real" # real, synthetic
radar_mode = "cluster" # cluster, object_list
test_run = 1 # Usually has 3 test_runs, but it varies from 1 to 4.
camera_path,(...),lidar_path = file_selector(scenario,weather,(...),test_run)
```

Figure 47 – Frame exported using the Python notebook.

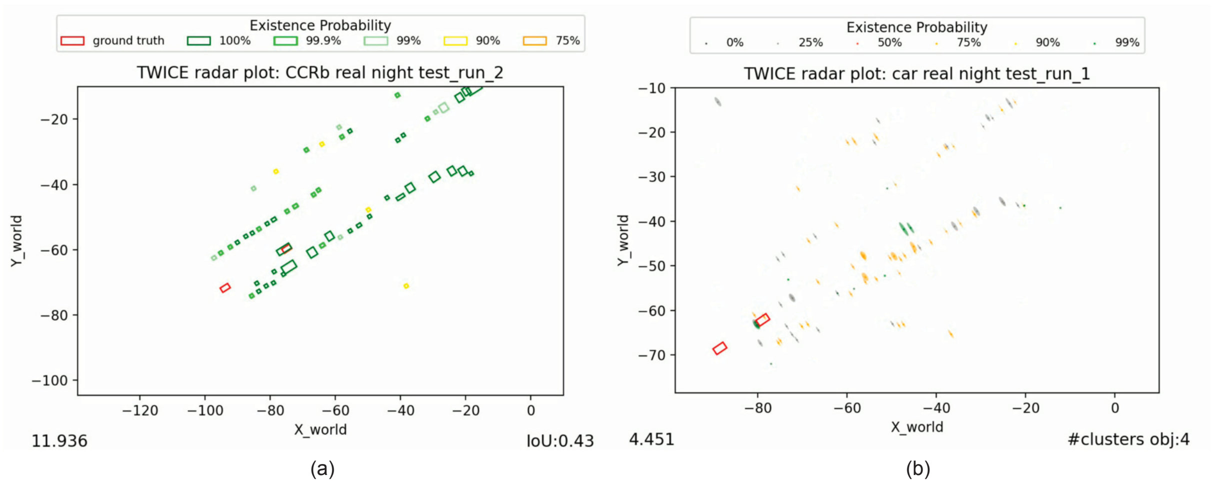


Source: (NETO et al., 2023).

Camera. It is possible to export a selected frame or the entire video. The user can choose whether to include the timestamp and the frame number in the output. Additionally, there is an option to project cuboids onto the objects in the scenario, as displayed in the Figure 47.

Radar. Similar to the camera, it is possible to export a single frame of the radar data or the entire video of the scenario. Alongside the radar data, the position data of the ego and the objects are included. In Figure 48 (a), the data from the radar operating in object list mode is displayed, showing the position, dimensions, and existence probability of the detected objects. By using the ground-truth position of the object of interest, it was possible to calculate the Intersection over Union of the object with the radar detection. This information is displayed in the bottom right of Figure 48 (a) and can be exported as a `.csv` file for further analysis.

Figure 48 – Operation mode: (a) Object List (b) Cluster.



Source: (NETO et al., 2023).

In Figure 48 (b), the radar is shown operating in cluster mode. Similar to the object list mode, it displays the cluster positions, existence probability, and ellipses representing the covariance in the object's position. With the ground-truth information, it is possible to perform the segmentation of the clusters that belong to the object of interest and count the number of the object's clusters in each frame. This data can also be exported to `.csv`.

LiDAR. For the LiDAR data, a script was developed to convert the data from `.osi` to `.ply`, which is a widely known format for point clouds. This script generates an individual `.ply` file for each timestamp. This way, users can easily visualize the LiDAR data.

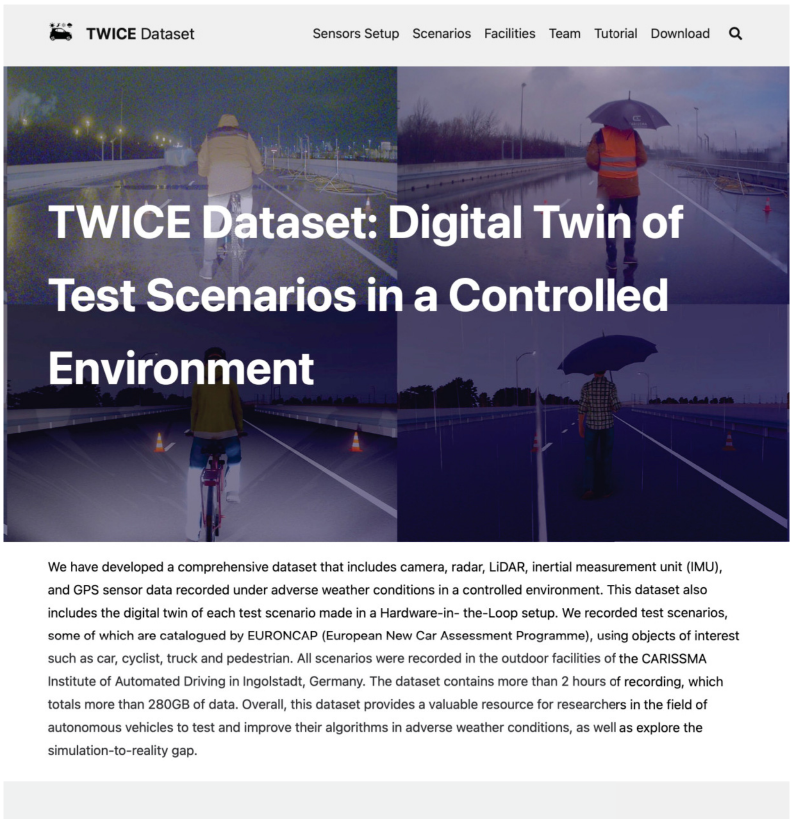
However, it is also possible to access the information of the point cloud without converting the data. The LiDAR data is structured in OSI format, containing the position in spherical coordinates, as well as the information of reflectivity and intensity for each point.

General Data. The Python notebook also demonstrates how to access the OSI data, such as retrieving the position of an object in a selected frame and more. Additionally, it is possible to convert all the data belonging to the `SensorDataSeries` class to the `.json` format.

4.6 WEBSITE

To enhance user accessibility and usage of the dataset, a website has been created, providing essential information and data samples. This website allows users to download the dataset and the necessary scripts for data manipulation. The site can be accessed at this address: `twicedataset.github.io/site`

Figure 49 – TWICE Website.


TWICE Dataset
Sensors Setup
Scenarios
Facilities
Team
Tutorial
Download
TWICE Dataset: Digital Twin of Test Scenarios in a Controlled Environment

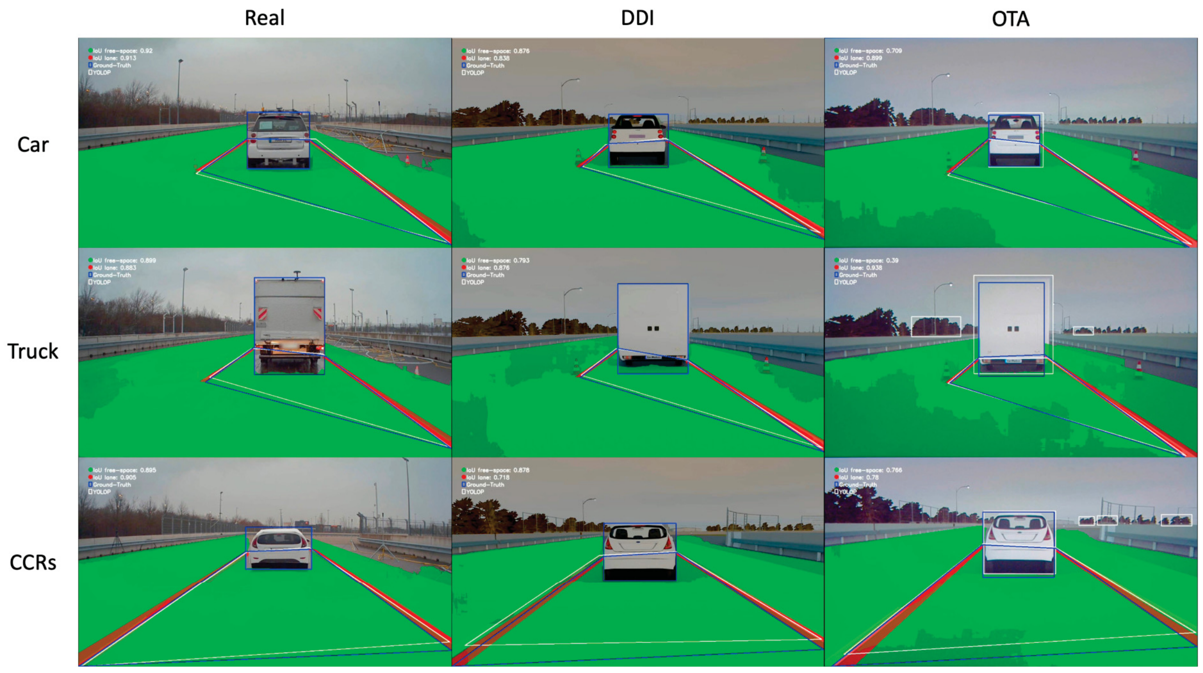
We have developed a comprehensive dataset that includes camera, radar, LiDAR, inertial measurement unit (IMU), and GPS sensor data recorded under adverse weather conditions in a controlled environment. This dataset also includes the digital twin of each test scenario made in a Hardware-in- the-Loop setup. We recorded test scenarios, some of which are catalogued by EURONCAP (European New Car Assessment Programme), using objects of interest such as car, cyclist, truck and pedestrian. All scenarios were recorded in the outdoor facilities of the CARISSMA Institute of Automated Driving in Ingolstadt, Germany. The dataset contains more than 2 hours of recording, which totals more than 280GB of data. Overall, this dataset provides a valuable resource for researchers in the field of autonomous vehicles to test and improve their algorithms in adverse weather conditions, as well as explore the simulation-to-reality gap.

Source: Author (2023).

4.7 YOLOP

Following the methodology described in section 3.8, the measurement of the simulation-to-reality gap was performed under various weather conditions for four scenarios, two of them with dynamic egos: CCRb and CCRs, and two with static egos: Car and Truck. The developed script generates an `.csv` file as output, containing the Intersection over Union (IoU) values between the ground-truth and YOLOP detections for the detected object, drivable area, and lane line segmentation in each frame. Additionally, the script calculates the distance between the ego-vehicle and the object of interest. In Figure 50, it can be observed the YOLOP detections along with the ground-truth projection. The green area represents the drivable space (free space), while the red area depicts the lane line segmentation. The lighter colors correspond to the YOLOP detections.

Figure 50 – YOLOP detections with ground-truth.



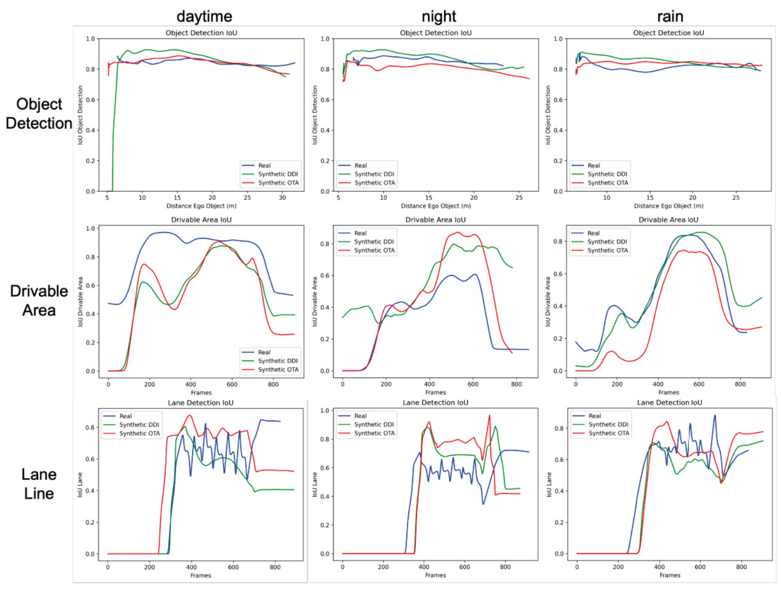
Source: Author (2023).

With this information, it was possible to generate plots for comparing the algorithm's performance in real-world and laboratory environments across different weather conditions. For Object Detection, a plot was created, depicting the distance between the ego-vehicle and the object of interest on the X-axis and the IoU on the Y-axis. For Drivable Area and Lane Line Segmentation, plots were generated, showing the frame numbers on the X-axis and the IoU on the Y-axis. All plots include real-world data as well as data from the two types of cameras used in the laboratory: Direct Data Injection (DDI) and Over-the-Air (OTA).

CCRb. As shown in Figure 51, the curves of the real data and synthetic (DDI and OTA) data follow the same pattern, with the exception of the Drivable Area at night. In this case, the simulation outperformed the real data. In the Lane Line segmentation, an oscillation of the IoU in the real data is noticeable. This is due to the YOLOP having difficulty detecting the dashed lane line.

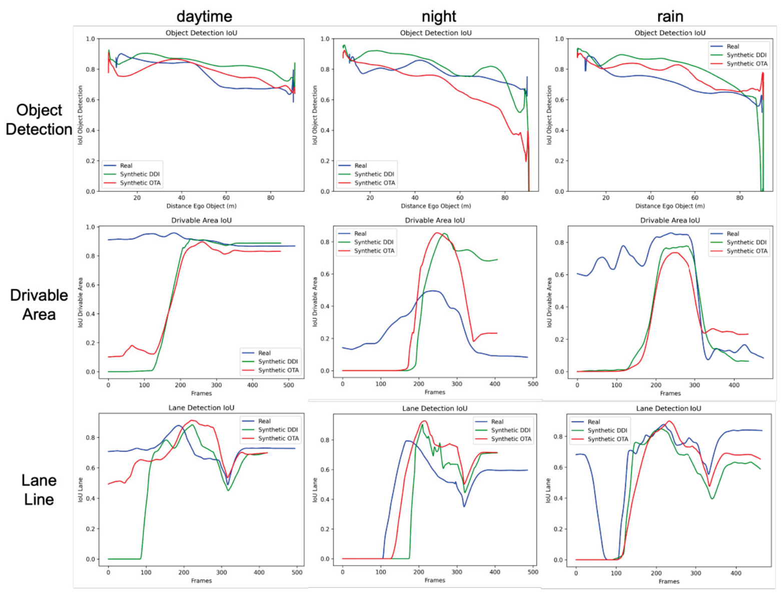
CCRs. Similar to CCRb,Figure 52 demonstrates that the real and synthetic data exhibit the same behavior, with the exception of the drivable area at night, where the simulation outperformed the real data. It is also noticeable that in the initial frames of

Figure 51 – CCRb YOLOP detections.



Source: Author (2023).

Figure 52 – CCRs YOLOP detections.



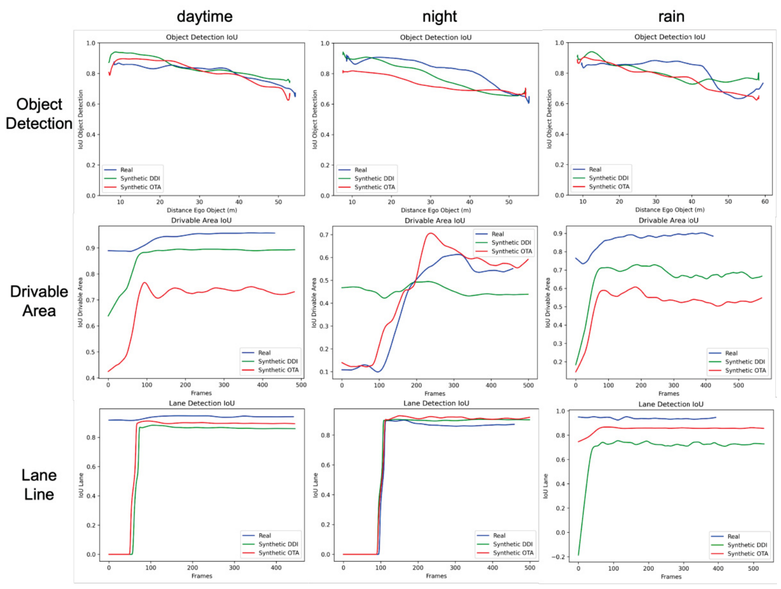
Source: Author (2023).

the simulation, the drivable area was not detected. As the ego-vehicle approaches the area where the lane lines are located, the drivable area starts being detected by the YOLOP algorithm.

Car. As shown in Figure 53 the Object Detection follows the same pattern in both real and synthetic data. The same can be said for the lane line segmentation, with the exception of the initial frames. In the Drivable Area, the real data outperformed the simulation in the daytime and rain scenarios, but exhibited similar behavior at night time.

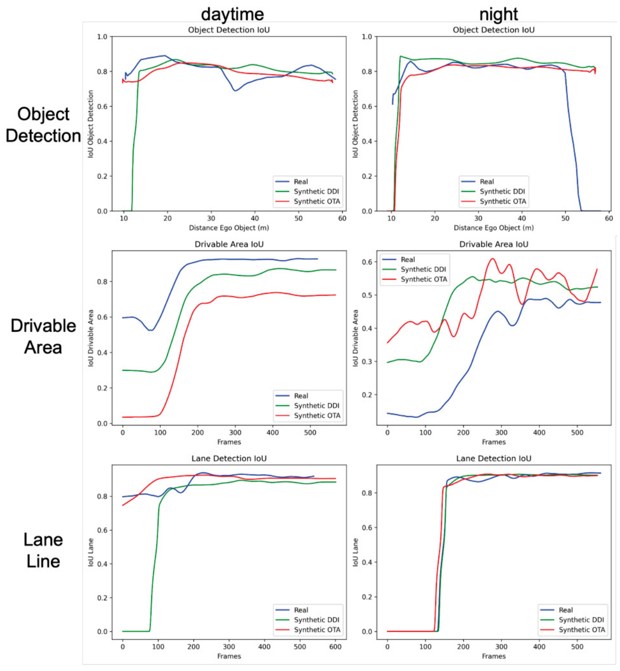
Truck. With slight variations, Figure 54 shows that the real and synthetic data exhibit consistent behavior in both weather conditions. In the initial frames, when the

Figure 53 – Car YOLOP detections.



Source: Author (2023).

Figure 54 – Truck YOLOP detections.

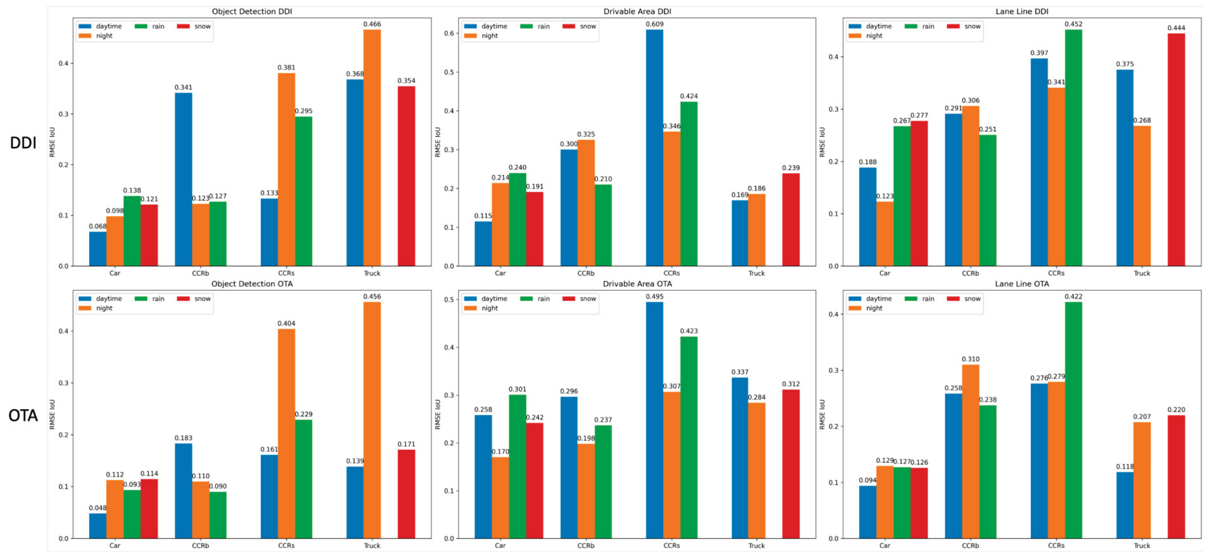


Source: Author (2023).

Truck is too close to the ego-vehicle, the Object Detection algorithm struggles and the lane line is not yet visible.

RMSE. To visualize the results, the mean RMSE values for each scenario and weather condition were computed and presented in the form of a plot shown in Figure 55. The first line is the RMSE calculation between the real scenario and the DDI camera. And the second line is the RMSE (Equation 3.2) between the real scenario and the OTA camera. This calculation was carried out frame by frame for Object Detection

Figure 55 – RMSE plot.



Source: Author (2023).

(first column), Drivable Area (second column) and Lane Line (third column). They were calculated for the scenarios: Car, CCRb, CCRs and Truck. In different weather conditions: daytime (blue), rain (green), night (orange) and snow (red). Examining the results, it becomes evident that the OTA camera displays great consistency when compared to the DDI camera. With the exception of the CCRb, the simulation error compared to real scenarios during the daytime was smaller as expected. This is related to the challenge of simulating adverse weather conditions.

Overall, the simulation performed very well, proving to be valid and consistent for automotive tests. The challenge remains in simulating adverse weather conditions, where the algorithm performed better than the real tests.

5 CONCLUSION

In this study, data obtained from cameras, radar, LiDAR, and IMU was gathered both from real-world proving ground and from laboratory. This data was systematically organized and structured to create an autonomous driving dataset. For each testing scenario conducted on the real-world proving ground, a corresponding digital twin was generated within the laboratory environment. The real-world data was acquired within a controlled environment in Ingolstadt (Germany) under adverse weather conditions. The location of all the actors participating in the test scenarios were determined. Utilizing the ROS framework on the NVIDIA DRIVEPX platform, the data was captured and stored in ROSbags. The same hardware setup, deployed in a Hardware-in-the-Loop (HiL) arrangement, was employed to gather data in the laboratory environment. The collected data underwent processing and conversion into the Open Simulation Interface (OSI) format. The data was annotated using ground-truth information regarding the positional attributes of the objects of interest. This annotation process was performed by a semi-supervised algorithm. The annotated data was then stored in the OpenLABEL format. The dataset contains more than 2 hours of recording, which totals more than 280GB of data and can be downloaded at: `https://twicedataset.github.io/site/`.

Using the TWICE dataset, a method was developed to assess the simulation-to-reality gap. This approach involved employing the YOLOP algorithm, which encompasses Object Detection, Drivable Area, and Lane Line segmentation. Given that the TWICE dataset includes a digital twin for each real test run conducted, it became feasible to compare algorithm performance between real and simulated data. In section 4.7, the figures illustrate a comparison of the algorithm's performance under three distinct configurations: real-world camera input and two simulated inputs, namely Direct Data Injection (DDI) and Over-the-Air (OTA) camera data. Overall, it can be inferred that the simulation closely emulates the patterns observed in real-world scenarios. However, during adverse weather conditions, the simulation tends to deviate more significantly. During nighttime scenarios, the simulation occasionally outperforms the real-world performance. This discrepancy indicates that the simulation did not entirely capture the real-world scenario with complete fidelity. Finally, a Root Mean Square Error (RMSE) analysis was conducted on the real data in comparison to both synthetic data (DDI and OTA). The calculation involved comparing the IoU values frame by frame for Object Detection, Drivable Area, and Lane Line segmentation.

The TWICE dataset holds the potential to serve as a valuable resource for the academic community, aiding them in development of their own methodologies to assess the simulation-to-reality gap. This, in turn, contributes to the validation of synthetic usage within the industry, leading to resource and time savings. Up to the current date, there exists no other public available dataset for autonomous driving that encompasses both real and synthetic data within a controlled environment, under adverse weather conditions. This dataset was the first publicly available dataset developed at the CARISSMA institute, leaving a know-how in the areas of data structuring, processing, formatting, and annotation. Additionally, it is the first public dataset that uses the Open Simulation Interface Format (OSI) and OpenLABEL annotation format, both maintained by the Association for Standardization of Automation and Measuring Systems (ASAM).

Future work can involve the development of a new dataset with a more sophisticated simulation and sensors, using the same data structure developed in this work. It can also include simulating LiDAR data, which was not feasible in this dataset due to the high computational cost. Furthermore, it is possible to utilize the dataset with other algorithms, such as YOLOP, to estimate the Simulation-to-Reality gap.

BIBLIOGRAPHY

BOUGUET, J.-Y. Camera Calibration Toolbox for Matlab. CaltechDATA, May 2022. DOI: 10.22002/D1.20164. Cit. on pp. 25, 30.

CAESAR, H.; BANKITI, V.; LANG, A. H.; VORA, S.; LIONG, V. E.; XU, Q.; KRISHNAN, A.; PAN, Y.; BALDAN, G.; BEIJBOM, O. nuScenes: A multimodal dataset for autonomous driving. In: CVPR. [S.l.: s.n.], 2020. Cit. on p. 20.

CNN. **Who's responsible when an autonomous car crashes?** [S.l.: s.n.], 2016. https://money.cnn.com/2016/07/07/technology/tesla-liability-risk/index.html. Accessed June 20, 2023. Cit. on pp. 16, 39.

CONTINENTAL. **Continental Radar ARS 408-21**. [S.l.: s.n.], 2023. https://conti-engineering.com/wp-content/uploads/2020/02/ARS-408-21_EN_HS-1.pdf. Accessed June 13, 2023. Cit. on p. 30.

DOMHOF, J.; KOOIJ, J. F.; GAVRILA, D. M. An Extrinsic Calibration Tool for Radar, Camera and Lidar. In: 2019 International Conference on Robotics and Automation (ICRA). [S.l.: s.n.], 2019. P. 8107–8113. DOI: 10.1109/ICRA.2019.8794186. Cit. on pp. 25, 26, 33, 52.

DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. CARLA: An Open Urban Driving Simulator. In: PROCEEDINGS of the 1st Annual Conference on Robot Learning. [S.l.: s.n.], 2017. P. 1–16. Cit. on pp. 20, 28.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: CONFERENCE on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2012. Cit. on p. 20.

GENESYS. **ADMA-G**. [S.l.: s.n.], 2023. https://genesys-offenburg.de/en/adma-g/. Accessed June 13, 2023. Cit. on p. 31.

GEYER, J.; KASSAHUN, Y.; MAHMUDI, M.; RICOU, X.; DURGESH, R.; CHUNG, A. S.; HAUSWALD, L.; PHAM, V. H.; MÜHLEGG, M.; DORN, S.; FERNANDEZ, T.; JÄNICKE, M.; MIRASHI, S.; SAVANI, C.; STURM, M.; VOROBIOV, O.; OELKER, M.; GARREIS, S.; SCHUBERTH, P. A2D2: Audi Autonomous Driving Dataset, 2020. arXiv: 2004.06320 [cs.CV]. Available from: https://www.a2d2.audi. Cit. on p. 27.

GOOGLE. **Protocol Buffers Documentation**. [S.l.: s.n.], 2023. https://protobuf.dev/overview/. Accessed June 21, 2023. Cit. on p. 40.

HEIKKILA, J.; SILVEN, O. A four-step camera calibration procedure with implicit image correction. In: PROCEEDINGS of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 1997. P. 1106–1112. DOI: 10.1109/CVPR.1997.609468. Cit. on p. 25.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO 23150:2021 Road vehicles — Data communication between sensors and data fusion unit for automated driving functions — Logical interface. In. Cit. on p. 41.

________. ISO 8855:2011 Road vehicles — Vehicle dynamics and road-holding ability. In. Cit. on pp. 33, 44.

IPG. **IPG Automotive**. [S.l.: s.n.], 2023. https://ipg-automotive.com/en/. Accessed June 15, 2023. Cit. on p. 35.

L., W.; G., A. Ten technologies which could change our lives. **European Parliamentary Research Service**, 2015. Available from: https://www.europarl.europa.eu/EPRS/EPRS_IDAN_527417_ten_trends_to_change_your_life.pdf. Cit. on p. 15.

NETO, L. N.; REWAY, F.; POLEDNA, Y.; DRECHSLER, M. F.; RIBEIRO, E. P.; HUBER, W.; ICKING, C. **TWICE Dataset: Digital Twin of Test Scenarios in a Controlled Environment**. [S.l.: s.n.], 2023. arXiv: 2310.03895 [cs.RO]. Cit. on pp. 16, 27, 28, 34, 36–40, 50, 58, 59.

NVIDIA. **DRIVE PX 2 AutoChauffeur (P2379)**. [S.l.: s.n.], 2018. https://docs.nvidia.com/drive/active/5.0.10.3L/nvvib_docs/index.html#page/NVIDIA%2520DRIVE%2520Linux%2520SDK%2520Development%2520Guide%2Fboard_hardware_dpx2.html%23wwconnect_header. Accessed June 14, 2023. Cit. on p. 32.

________. **NVIDIA Driveworks**. [S.l.: s.n.], 2023. https://developer.nvidia.com/drive/driveworks. Accessed June 14, 2023. Cit. on pp. 25, 32.

OUSTER. **Ouster OS1 LiDAR**. [S.l.: s.n.], 2023. https://ouster.com/products/scanning-lidar/os1-sensor/. Accessed June 13, 2023. Cit. on p. 31.

PITROPOV, M.; GARCIA, D.; REBELLO, J.; SMART, M.; WANG, C.; CZARNECKI, K.; WASLANDER, S. Canadian Adverse Driving Conditions Dataset, Jan. 2020. DOI: 10.1177/0278364920979368. Cit. on p. 20.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection, June 2015. Available from: http://arxiv.org/abs/1506.02640. Cit. on p. 27.

REWAY, F.; HUBER, W.; RIBEIRO, E. P. Test Methodology for Vision-Based ADAS Algorithms with an Automotive Camera-in-the-Loop. In. ISBN 9781538635438. DOI: 10.1109/ICVES.2018.8519598. Cit. on p. 36.

RORIZ, R.; CABRAL, J.; GOMES, T. Automotive LiDAR Technology: A Survey. **IEEE Transactions on Intelligent Transportation Systems**, Institute of Electrical and Electronics Engineers Inc., v. 23, p. 6282–6297, 7 July 2022. ISSN 15580016. DOI: 10.1109/TITS.2021.3086804. Cit. on p. 24.

SAE. **Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles**. [S.l.], 2021. Cit. on p. 15.

SEKOLAB. **Sekonix Camera**. [S.l.: s.n.], 2023. http://sekolab.com/products/camera/. Accessed June 13, 2023. Cit. on p. 30.

SINGH, S. Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey. **Traffic Safety Facts Crash•Stats. Report No. DOT HS 812 115**, 2015. Available from: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115. Cit. on p. 15.

SPECTRA. **SP80**. [S.l.: s.n.], 2023. https://spectrageospatial.com. Accessed June 13, 2023. Cit. on pp. 31, 32.

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY ET AL. **Robotic Operating System**. [S.l.: s.n.], 23 May 2018. Available from: https://www.ros.org. Cit. on p. 33.

TIMO HANKE, NILS HIRSENKORN, CARLO VAN DRIESTEN. **Open Simulation Interface**. [S.l.]: Technische Universität München, 2017. Available from: https://www.ee.cit.tum.de/hot/forschung/automotive-veroeffentlichungen. Cit. on p. 41.

WAYMO LLC. **Waymo safety report**. 2021. Available from: https://storage.googleapis.com/waymo-uploads/files/documents/safety/2021-12-waymo-safety-report.pdf. Cit. on p. 15.

WENG, X.; MAN, Y.; PARK, J.; YUAN, Y.; CHENG, D.; O'TOOLE, M.; KITANI, K. All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds. **arXiv**, 2021. Cit. on pp. 20, 21.

WU, D.; LIAO, M. W.; ZHANG, W. T.; WANG, X. G.; BAI, X.; CHENG, W. Q.; LIU, W. Y. YOLOP: You Only Look Once for Panoptic Driving Perception. **Machine Intelligence Research**, Chinese Academy of Sciences, v. 19, p. 550–562, 6 Dec. 2022. ISSN 27315398. DOI: 10.1007/s11633-022-1339-y. Cit. on p. 27.

YEN, G. G. 8 - Fault-Tolerant Control. In: CHEN, W.-K. (Ed.). **The Electrical Engineering Handbook**. Burlington: Academic Press, 2005. P. 1085–1105. ISBN 978-0-12-170960-0. DOI: https://doi.org/10.1016/B978-012170960-0/50085-2. Available from: https://www.sciencedirect.com/science/article/pii/B9780121709600500852. Cit. on p. 24.

YURTSEVER, E.; LAMBERT, J.; CARBALLO, A.; TAKEDA, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 58443–58469, 2020. ISSN 21693536. DOI: 10.1109/ACCESS.2020.2983149. Cit. on p. 15.

ZHANG, Y.; CARBALLO, A.; YANG, H.; TAKEDA, K. Perception and Sensing for Autonomous Vehicles Under Adverse Weather Conditions: A Survey, Dec. 2021. DOI: 10.1016/j.isprsjprs.2022.12.021. Cit. on p. 25.

ZHANG, Z. A flexible new technique for camera calibration. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 22, n. 11, p. 1330–1334, 2000. DOI: 10.1109/34.888718. Cit. on pp. 23, 25.

ZHOU, T.; YANG, M.; JIANG, K.; WONG, H.; YANG, D. **Mmw radar-based technologies in autonomous driving: A review**. v. 20. [S.l.]: MDPI AG, Dec. 2020. P. 1–21. DOI: 10.3390/s20247283. Cit. on p. 22.