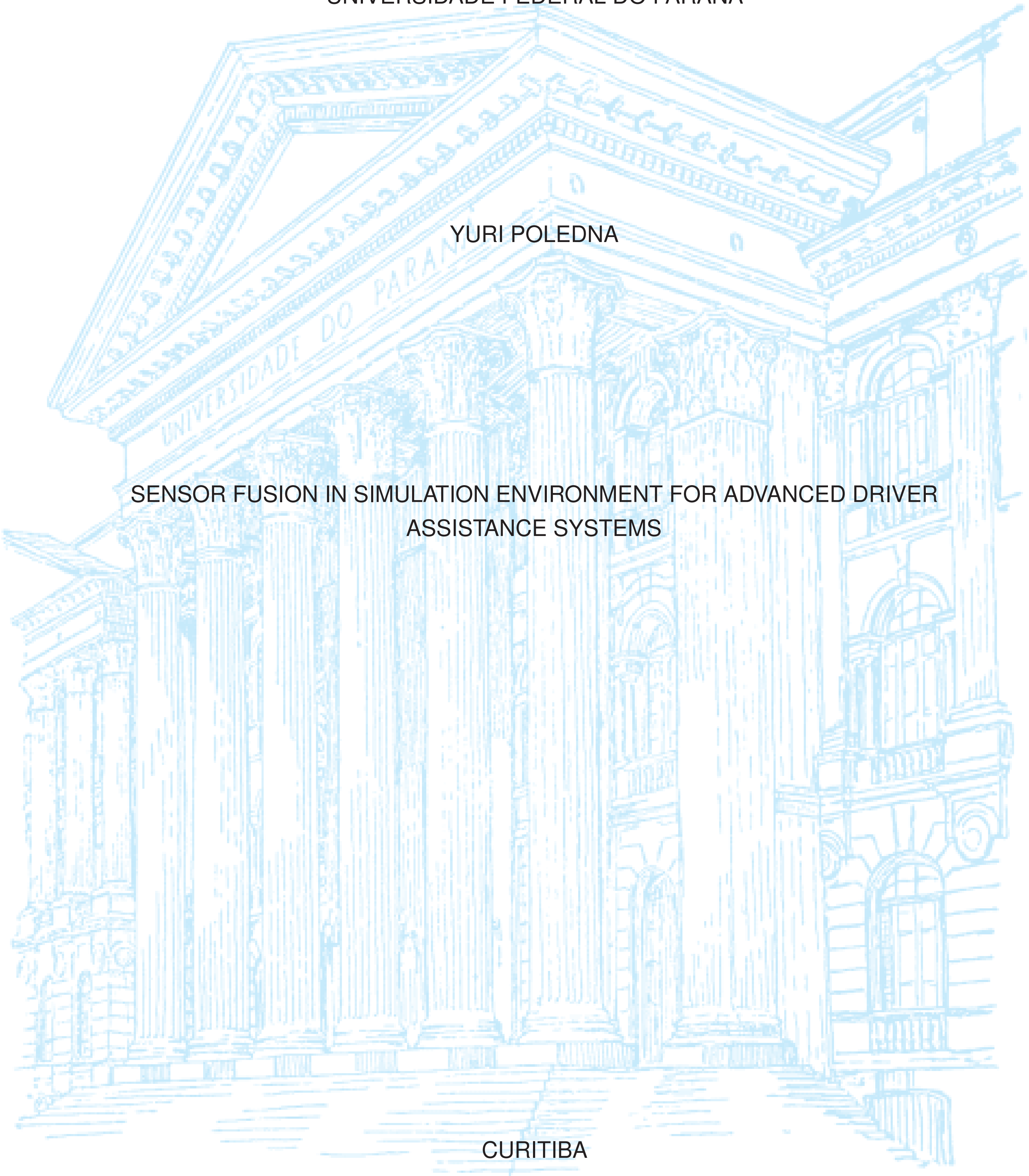
UNIVERSIDADE FEDERAL DO PARANÁ

YURI POLEDNA

SENSOR FUSION IN SIMULATION ENVIRONMENT FOR ADVANCED DRIVER ASSISTANCE SYSTEMS

CURITIBA

2022



YURI POLEDNA

SENSOR FUSION IN SIMULATION ENVIRONMENT FOR ADVANCED DRIVER ASSISTANCE SYSTEMS

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
Co-Orientador: Prof. Dr.-Ing. Werner Huber

CURITIBA

2022

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

Poledna, Yuri
Sensor fusion in simulation environment for advanced driver assistance system / Yuri Poledna. – Curitiba, 2022.
1 recurso on-line : PDF.

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

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

1. Câmera. 2. Radar. 3 Detectores. I. Ribeiro, Eduardo Parente. II. Huber, Werner. III. Universidade Federal do Paraná. Programa de pós-Graduação em Engenharia Elétrica. IV. Título.

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

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 **YURI POLEDNA** intitulada: **SENSOR FUSION IN SIMULATION ENVIRONMENT FOR ADVANCED DRIVER ASSISTANCE SYSTEMS EVALUATION**, 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, 29 de Agosto de 2022.

Assinatura Eletrônica
29/08/2022 17:03:57.0
EDUARDO PARENTE RIBEIRO
Presidente da Banca Examinadora

Assinatura Eletrônica
29/08/2022 14:07:49.0
ANDREI CAMPONOGARA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
29/08/2022 18:32:26.0
GIDEON VILLAR LEANDRO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
29/08/2022 13:57:58.0
EDUARDO TODT
Avaliador Externo (55002387)

For those who are lost in the midst of the jungle of knowledge and will leave no stone unturned to reach the future.

THANKS

Initially I thank Silvia Rossana Caballero Poledna, Edgar Poledna e Christiane Rossana Caballero Alburquerque, which have been the support backbone for me during this thesis period. I also thank the Professor Eduardo Parente Ribeiro, my advisor for the opportunity of working with this topic, and whom has been advising me through many years now without failure, I look forward to collaborate even further in the future. I would like to thank Fabio Luis Reway for the endless support he has provided, during my research, not only in the research, but personally as well, Fabio is one of those people that made everything possible and I am grateful for his support on all fronts. Professor Dr.-Ing. Werner Huber is the Head of the laboratory and without him we would not be able to have the connections we need to make everything possible, also he is the person with an always excited face, and it is clear that loves working with his job, it has been an honor to work by his side, and I am looking forward to our future collaboration. Also in the same line I thank Maikol Funk Drechsler who has provided and helped in more times than necessary and offered to help in various moments, his help has proven without comparison when making tests and discussing matrix transformations.

ApheK Gusso de Brito and Rafael Luiz Bruginski Stonoga were so incredibly special that they deserve an entire comment of their presence in my life. Mr. Aphek, has always been present even in the darkest hour in the moment, a small ray of sunshine to enlighten, and possibly the best programmer I've had the possibility to meet. Mr. Stonoga finally has calmed down in life and came back down to Earth after his amazing flight to metaphorical low earth orbit, I am happy that his back in one piece. To the stars we aim, because even if we fail, we reach the Moon. Tania Lucero Cordutti, introduced me to even newer things I was not aware of, she is always fun and with the wildest stories to tell. Luciane Tomio Favero, shared a lot of moments due to the insanity it was to get to europe, but nevertheless a trip. Hellem Pedroso has magically appeared in my life one day and showed me not only how to live but why. Her magical power of showing things that you don't see is quite impressive, a new youtuber, podcaster, and a lot more, i hope nothing but the best to you. To the friends Juliana Cristina Labatut Perreira, Eduardo Zibetti dos Passos, Diego Garzaro, Tatiana Holub, Julien Victor Prestes da Costa, I would like to truthfully appreciate the support when needed and the laughs when needed as well. Each one of you were uniquely important to me during this time, and I hope to unite everyone back.

Valentina de Melo Cezar de Araújo a superpowered magical firefly, never seen someone so firery about living life, eternally magical, I just hope we can see each other again, and our paths connect once more, much love to you. In the midst of the endless

chasm, that is life, you were always an intese light of inmeasurable power and trust. Never have I ever felt what being with you made me feel.

Samuel Souza Alcantra Queiroz, between moments of joy and anger, this hard headed man has always been there to share a barbecue, or food for thoughts, or an deep conversation of religious thoughts. Still finding where he stands in life, i hope you find the path no matter how long it may take. Enrico Nardi, present even if you need to move furniture around, this man is nothing short of a future GOAT. Murilo Belinasso, oh boy the drama, eventually you will see that life is simpler than we are lead to believe, as Dory would say "Just keep swimming". Never change. Veronica Bezdrighin, this blue hair is freaking amazing person who was been amazing, lover of cakes, and Yasonandan Vadlamudi, I'm mostly proud of having you as friends, I am eternally happy that you care about me, many hugs to you both. Isabella Wosniack, never have I ever seen a girl that fought so much alone, just be aware please, that you do not have to fight alone, alhough hard it is to trust people, it is only by opening your heart up to failure and pain that we find happiness, there is always someone to help, you just need to ask.

Ronja Marquadt, from the bottom of my heart truthfully thank you for having me, you gave me something you do not even know what it is, I hope for the best in your new endeavours as a Technische Designrin. The path you take is as important as the destination, but the length of the path always changes, it was never a race, being open for anything that life as it is throws at you, now or ever, is more important than making a simple decision, life is not about making the right choice constantly, but doing the best you can with what you were given right now. If I could suggest you something, would be to try and fix things instead of getting a new one, to fix something is to be aware of what is wrong precisely, and not generally, but for that, deep knowledge and understanding of that part is needed, which in the process helps understand each other flaws.

Johann Cherian Mathew, my buddy, my friend, so many thing to thank you for, not sure where to start, thanks for being such a good friend. When I think about you I think about a person who has a huge energy, even more than me, who is fun and endlessly entertaining to hang out with, who loves long-boarding, even though just (TRAINTRACKS) learnt it. Matthias Patrzek, really thank you for inviting me over multiple times, know that I will forever be grateful for having you as a friend, you are such an awesome person, with such an amazing personality. You, like me, choose the path of electrical engineering, a choice for sure, but a welcomed one (according to a deeply disputed publication by The Tab(2017), Electrical Engineering is the 4th hardest course). I hope and wish great success in your so vivid career wherever you choose to be. Aim for the stars, you may just reach the Moon in the process. Matthias and Johann, I feel like I found brothers in you that I never had. Thanks for all the amazing times we have had together, and as always looking forward to the next.

“Be faithful in the small thing
because it is in them that lies your strengths“
(Mother Theresa)

RESUMO

Sistemas avançados de assistência ao condutor (do inglês, *Advanced Driving Assistance Systems* – ADAS) são sistemas que percebem o cenário em que o veículo está imerso, e auxiliam o condutor de duas formas: fornecendo informações mais detalhadas sobre o cenário ou atuando de forma direta na condução do veículo. Nesse contexto, a percepção do ambiente se dá pelos dados obtidos por diferentes sensores, dentre eles: câmeras, radares e lidares. A fim de aumentar a acurácia da percepção, estes dados podem ser combinados com diferentes métodos de fusão. Nesse âmbito existe a fusão de dados obtidos por sensores com diferentes princípios de medição, gerando uma representação do cenário em forma de modelo, o qual pode ser descrito matematicamente, que define a cena onde se encontra o veículo. Uma vez que testar algoritmos de ADAS em ambientes reais controlados é dispendioso, pode-se utilizar simulações. Contudo é necessário saber até que ponto a simulação é válida se comparada com a realidade. Este trabalho propõe estimar a distância entre simulação e realidade que pode ocorrer em testes de algoritmos de fusão de dados para algoritmos de direção automatizada. A fusão pode ser feita em vários métodos e níveis. Para isso dois métodos de fusão são implementados: um alto nível e outro de baixo nível. Testes foram realizados em uma pista de testes do CARISSMA e em laboratório com a ajuda da simulação e estimulação de sensores. Foi observado que a fusão baixo nível obteve melhores escores quando comparada com a fusão alto nível. Além disso, observou-se que a distância entre simulação e realidade foi menor na fusão baixo nível do que na de alto nível. Em baixo nível observou-se uma distância entre simulacao e realidade de -5.26%(negativo nota que o simulado teve resultados melhores que o real) em cenários de dia e 20.59% em cenários com chuva. Enquanto no alto nível mostrou-se nos cenários de dia uma distância de 23.99% e na chuva uma distância de 34.67%. Isto mostra que o cenários simulados ainda requerem algum trabalho para serem mais próximos da realidade. Portanto, outros cenários próximos ao cenário de dia, utilizando a mesma metodologia e utilizando algoritmo de baixo nível podem ter resultado parecidos com aqueles dos reais. Este trabalho conclui que os cenários simulados devem ser melhorados ainda mais, para diminuir a distância entre cenários simulados e reais.

Palavras-chaves: Fusão; simulação; fusão de câmera e RADAR; percepção de ambientes; ADAS; *Digital Twin*; *Hardware-in-The-Loop*; *Simulation-to-Reality Gap*.

ABSTRACT

Advanced Driver Assistance Systems (ADAS) are systems that perceive the scenario in which the vehicle is immersed, and assist the driver in two ways: by providing more detailed information about the scenario or by acting directly in driving the vehicle. In this context, the perception of the environment is given by the data obtained by different sensors, among them: cameras, RADARs, and LiDARs. In order to increase the accuracy of perception, this data can be combined with different fusion methods. In this scope there is the fusion of data obtained by sensors with different measurement principles, generating a representation of the scene in the form of a model, which can be described mathematically, that defines the scene where the vehicle is. As testing ADAS algorithms in real controlled environments is expensive, simulations can be used. Nonetheless, it is necessary to know how valid the simulation is when compared to reality. This study proposes to estimate the gap between simulation and realities that can occur in testing data fusion algorithms for automated driving algorithms. Fusion can be accomplished in various methods and levels. For this purpose, two fusion methods are implemented: a low level fusion and a high level fusion. Tests were performed on a CARISSMA test track and in the laboratory with assistance of simulation and sensor stimulation. It was observed that the lower level fusion obtained better scores when compared to the high level fusion. Also, it was observed that the distance between simulation and reality was smaller in the lower level fusion than that based on the higher level fusion. It was noted that mainly the uncertainties added to the extra input layers in the higher level fusion greatly influenced the result of this fusion.At low level we observed a distance between simulation and reality of -5.26%(negative note that the simulated had better results than the real one) in day scenarios and 20.59% in rain scenarios. While the high level showed a distance of 23.99% in day scenarios and 34.67% in rain scenarios. This shows that the simulated scenarios still require some work to be closer to reality. Therefore, other scenarios close to the day scenario, using the same methodology and using low-level fusion algorithm can have similar result to real ones. This paper concludes that the simulated scenarios should be further improved to decrease simulation-to-reality gap.

Key-words: Fusion; simulation; camera and RADAR fusion; enviromental perception; ADAS; Digital Twin; Hardware-in-The-Loop; Simulation-to-Reality Gap.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ABS	Anti-locking Brake Systems
ADAS	Advanced Driving Assistance Systems
ADMA	Automotive Dynamic Motion Analyzer
AEB	Automatic Emergency Braking
AP	Average Precision
CAN	Controller Area Network
CARISSMA	Center of Automotive Research on Integrated Safety Systems and Measurement Area
CBLA	Car to Bicycle Longitudinal
CCRb	Car to Car Rear braking
CCRs	Car to Car Rear static
CRF-Net	Camera Radar Network
DDI	Direct Data Injection
ECU	Electronic Control Unit
EuroNCAP	European New Car Assessment Programme
GMSL	Gigabit Multimedia Serial Link
GPS	Global Positioning System
HiL	Hardware-In-the-Loop
IMU	Inertial Measurement Unit
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
OTA	Over-The-Air
RADAR	Radio Detection and Ranging
RCS	Radar Cross Section

ROS	Robot Operating System
SMOKE	Single-Stage Monocular 3D Object Detection via Keypoint Detection
UDP	User-Defined Packet
VIB	Video Interface Box
VIL	Vehicle in the Loop
VRTS	Vehicle Radar Test System
mAP	mean Average Precision

TABLE OF CONTENTS

1 INTRODUCTION

For previous vehicle generations, autonomous functions in vehicles used to be comfort features. Currently, these functions are considered safety functions. The most important part of an autonomous vehicle is the driving function. This process is based on data inputs regarding its surroundings, which outputs a list of actions that the vehicle must perform. An autonomous vehicle integrates multiple functions of a car into a unique driving function. Ideally, an autonomous vehicle is such that no human input is needed. However, a fully autonomous vehicle is highly complex, with multiple parts, and multiple possible points of failure. To identify and minimize failures, all components that partake into the driving function must be tested and validated.

Scenario recognition is the main input for a driving function. Safe locomotion requires awareness of vehicle immediate surroundings. In the scope of increasing driver perception, Advanced Driving Assistance Systems (ADAS) are systems which recognize and enhance the environment surrounding the vehicle perception. These output data directly to the driver, human or automated driving function (GALVANI, 2019).

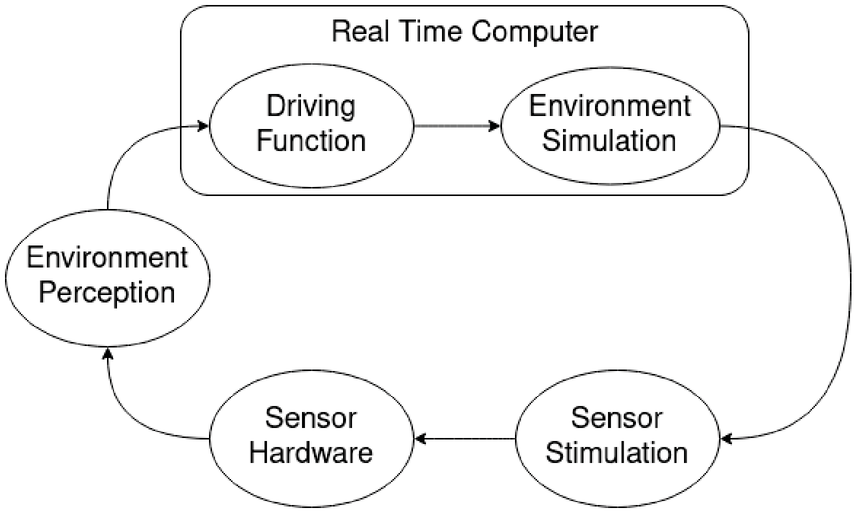
SAE International (2014) formally describes levels of automation and assistance of ADAS. The first is no automation at all, where only warnings are considered, and the last is full automation. At Level 0, the human behind the wheel performs all driving tasks, but ADAS helps the driver to be more aware of his surroundings. Examples of this level include warning systems, such as lane departure warning or blind spot detection, which only warn the driver. Level 1, the vehicle is still driven by the human, however there is some assistance included. For instance, Anti-locking Brake Systems (ABS) is a level 1 system which performs a specific task, in this case to avoid the wheel blockage. Level 2, the vehicle is able to perform combined functions, such as steering and acceleration, however the driver must continue remain fully engaged. Autonomous parking is a great example of this level since it involves vehicle surroundings awareness to know when the spot to park is big enough to fit the current vehicle, as well as when the parking spot is set to steer the steering wheel while the driver accelerates. Level 3 is when the driver is still needed but not required to be in constant surroundings awareness, nevertheless, he must be ready to take control in case the system fails or find no solution to the scenario at hand. Level 4 is when the vehicle performs all driving functions given specific conditions. This means that the driver is not required most of the time. The last level is 5 in which the vehicle is capable of driving itself in any condition and may even lack the driver car interface, such as pedals.

According to Autopromotec Observatory (2021), by 2030, 54% of the cars in Europe will be equipped with ADAS. This is an fourfold increase from the observed participation in 2019 (14%) and higher than the 2025 forecasts (34%). The prospected increase in ADAS participation in the consumer market requires that all systems are thoroughly tested and evaluated before inserting them into production.

Tests are a way to evaluate and validate proposed algorithms. These can be executed in real world scenarios. Real world tests can be controlled or not controlled environments. When using controlled environments, the advantages are that these tests are reproducible and the location of all actors involved is known. When testing in uncontrolled scenarios, the test does not have the same level of repeatability, requires significant levels of annotations due to the fact that the actor location is not known, and these tests introduce more unknown variables to the test, which can increase uncertainty. Controlled real world environment tests are expensive and require bureaucracy and safety controls. In order to decrease costs of making real tests in real world test facilities, scenarios can be simulated. Simulated scenarios lower the human hours required to test the same environment and allow ethically or physically impossible scenarios.

Simulation is the method to test simulated scenarios in laboratory. Ideally, a simulation with results equals or near equal to those seen in the real world with reduced cost per test run and with lowered safety risks should be possible. There is a need to measure how much simulations are distant from their real counter parts. Reway et al. (2022) proposes to use Hardware-In-the-Loop (HiL) to close the simulation-to-reality gap.

FIGURE 1 – HiL ENVIRONMENT



SOURCE: Author (2022)

FIGURE 1 showcases the HiL environment. Initially, there is the simulation on a real time computer, which simulates the environment for the next instant of testing given the initial vehicle position. With this data, it stimulates the sensor stimulators, which stimulate the sensor with simulation data. The sensors output are the input for the environment perception algorithms, which output the vehicle surroundings to the driving function. This will act with the inputs given by the environment perception algorithms and inform the real time computer of its next step. Finally, the environment simulation acknowledges the new vehicle position and closes the loop by computing the next simulation instant.

For a driving function, many sensors can be used, such as camera, Radio Detection and Ranging (RADAR), and Light Detection and Ranging (LiDAR). Deciding factors in the sensor choice are multiple, such as environmental robustness and sensor principle, i.e., a sensor can be active or passive. An active sensor is such that emits some sort of action that the environment reacts to and the sensor reads this reaction, whilst a passive sensor only observes the environment without taking action.

Different sensors have different stimulation methods. Among them, it stands out the Over-The-Air (OTA), and the Direct Data Injection (DDI) . OTA is a method in which there is an air gap between the stimulator and the sensor (hence the name). The actual physical sensor is involved in this method. All outputs from the sensor are the same as if it was on a real world test scenario. Logically, the sensor cannot output data with more resolution or more data points than the stimulator can provide. Therefore a bottleneck to this method is your stimulator. In contrast, DDI is a method in which the physical sensor is not involved in the stimulation. The stimulator has the sensor model stored and then emulates the sensor timings and data outputs, given an input from the simulation. The OTA approach is cheaper than the DDI, but requires more attention towards alignment and stimulator specification. One example of OTA approach for camera can be seen in Reway et al. (2018) and for RADAR OTA simulator can be seen in Rafieinia e Haghighi (2020).

Sensor data must be concentrated in a hardware that has the environment algorithms in execution. This hardware has an algorithm that outputs data to the driving function. This hardware must have all necessary physical connection, and drivers available for the scenario at hand. Each sensor has its own environmental perception, based on not only the environment, but also the sensor relative location to the vehicle. Therefore this software must contain a way to fuse data from multiple sensors with different principles into an unique environmental perception that can be used by the driving function. This process is called fusion. To fuse data from multiple sensors is a technique that increases the system overall performance and robustness. The increase in robustness comes from sensor strength in different fields. In other words, a sensor may not have the best performance with rain, while another may not have a noticeable difference with weather. Fusing sensor data is a crucial step towards surrounding perception (AEBERHARD, 2017).

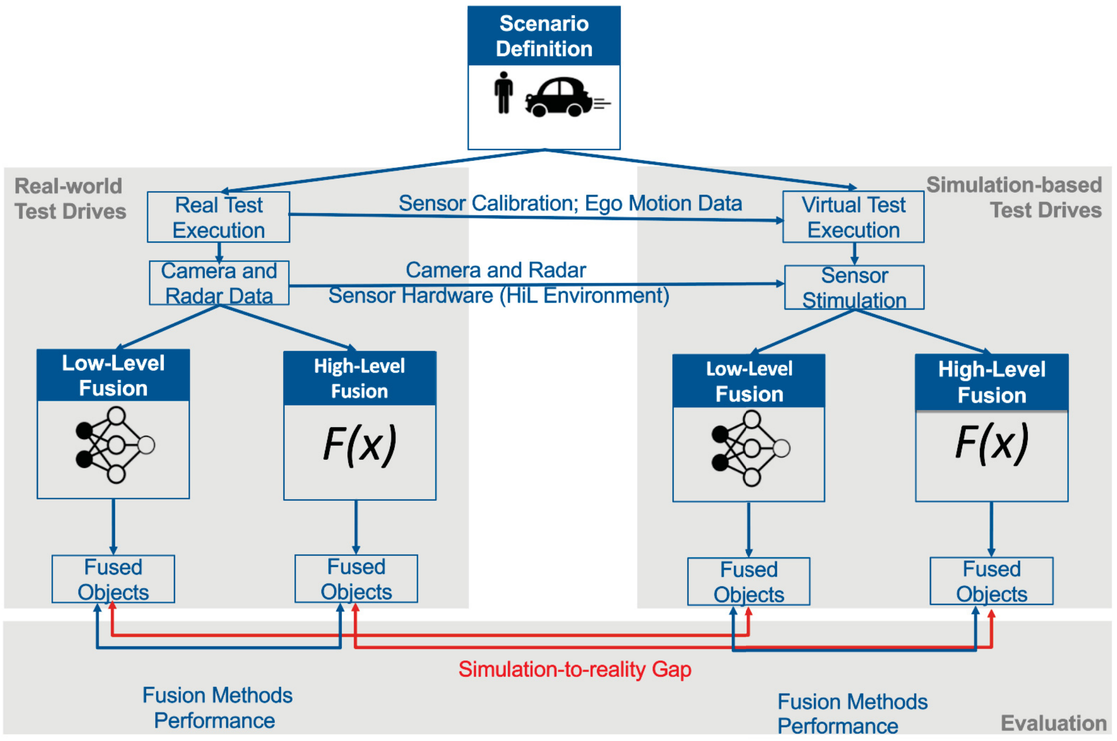
Simulation is an important tool for testing algorithms used in ADAS functions. It is imperative that the simulation-to-reality gap is measured and that the gap is minimal. The German government directive 86/22 reads:

> ”The fulfillment of requirements can also be checked by means of suitable simulation. The simulation tools must be validated. The simulation validation tools must be carried out by means of comparison with a representative selection of real tests; there must be no significant difference between characteristic values from simulation and real tests. The sensor performance 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.” Bundesministeriums für Digitales und Verkehr (2022, p. 40)(freely translated from german)

Therefore, with the main goal of measuring the simulation-to-reality gap, this work proposes to measure the simulation-to-reality gap from the fused data from two sensors, with different principles, camera and RADAR. The real data is captured from scenarios executed in a controlled environment. The simulation is carried out in a HiL environment using digital twins from real data.

This work is a near full-stack development of an autonomous vehicle, covering the extrinsic calibration of sensors on a car, data acquisition, and data analysis, missing only the driving function.

FIGURE 2 – RESEARCH METHODOLOGY



SOURCE: Author (2022)

FIGURE 2 demonstrates the proposed methodology of this work. Initially the scenarios are determined, then executed in real world and using the scenario digital twin, the same test if performed on a simulation environment. With both real and virtual scenarios, data will be collected raw and then processed using high and low level fusions. Both fusion methods output a fused object list, these will be compared to determine the fusion performance. Then comparing real to virtual fusion will determine the simulation-to-reality gap.

1.1 OBJECTIVES

The general objective of this work is to measure the simulation-to-reality gap using camera and RADAR fusion with real world and synthetic data generated from a digital twin.

The specific objectives of this work are:

1. Measure/calibrate camera and radar location extrinsically.
2. Perform tests on real life scenarios and with the real world data create the digital twin environment.
3. Capture the data from real world and digital twin scenarios using the same hardware and HiL .
4. Determine how the results of camera and RADAR fusion deviate for real and virtual environments.
5. Determine which sensor fusion approach delivers the best performance for environment perception.

1.2 WORK STRUCTURE

This work is separated in 4 chapters. Initially, there is a literature review where the concepts related to this work, such as sensors, sensor perception, and fusion, are more thoroughly developed. Followed by the methodology on data acquisition, algorithms utilized, tests performed, and evaluation methods. Then there is a chapter of results analysis, in which used metrics results are used. Finally, comes the conclusion, where qualitative results comparisons, as well as listing future works are made.

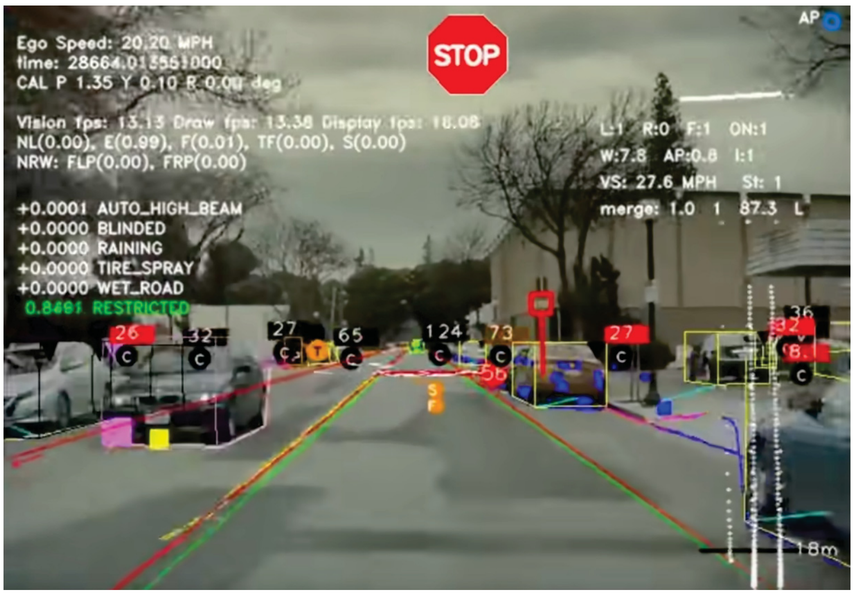
2 LITERATURE REVIEW

This chapter is aimed to the literature review of important concepts related to this work. Firstly, in Section 2.1, the scenario perception is described, with points on how to perceive data using camera and RADAR. Then, in Section 2.2, the calibration and its importance to the fusion is explained. Next, fusion and how to fuse data is discussed in Section 2.3. Finally, in Section 2.4, virtual tests is described.

2.1 PERCEPTION

A scene is what surrounds the vehicle. The vehicle, in which the sensors are installed, is known as ego. In a scene there are actors, which are known as objects. One object has a position relative to the ego, as well as a rotation and a classification.

FIGURE 3 – TESLA AUTO PILOT DEBUG MODE



```
Ego Speed: 20.20 MPH
time: 28664.013551000
CAL P 1.35 Y 0.10 R 0.00 deg

Vision fps: 13.13 Draw fps: 13.38 Display fps: 18.08
NL(0.00), E(0.99), F(0.01), TF(0.00), S(0.00)
NRW: FLP(0.00), FRP(0.00)

+0.0001 AUTO_HIGH_BEAM
+0.0000 BLINDED
+0.0000 RAINING
+0.0000 TIRE_SPRAY
+0.0000 WET_ROAD
 0.8481 RESTRICTED
```

```
L:1   R:0   F:1   ON:1
W:7.8   AP:0.8   I:1
VS: 27.6 MPH    St: 1
merge: 1.0  1  87.3  L
```

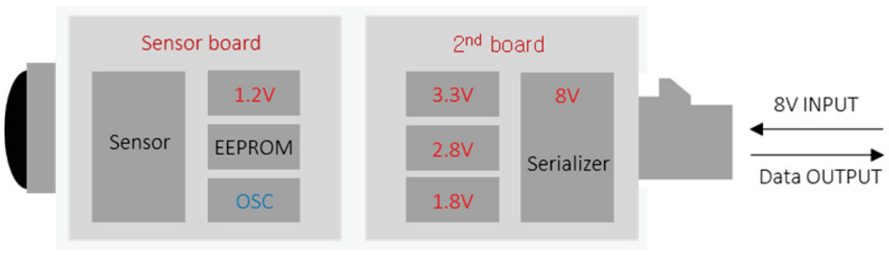
SOURCE: Carscoops (2020)

As a perception example, FIGURE 3 depicts the debug mode from the ”*autopilot*” mode from a Tesla car. In this figure, there are multiple objects, such as cars, lanes, and plaques. Each object has its own position, rotation, and classification in reference to the ego. In Subsection 2.1.1 and Subsection 2.1.2 is discussed the camera and RADAR sensors, respectively. As the camera data is raw, it is necessary the use of a detection algorithm in order to have an object list from camera. This algorithm is described in Subsection 2.1.3. As important as object detection, is to be sure that they are the same object from the previous frames, for this application, a tracking algorithm can be used, this is described in Subsection 2.1.4.

2.1.1 Camera

A digital camera is a unidirectional visual sensor, composed of lens which condense the data received into a light-to-digital converter. This components can be seen in the below figure.

FIGURE 4 – CAMERA BLOCK DIAGRAM



SOURCE: SEKONIX Co., Ltd. Corporate Headquarters (2022)

FIGURE 4 shows the blocks of a camera internals. In the figure, the most left component is the lens. Lenses have a field of view in which light can enter, therefore also must a camera have a field of view. The lens limit the widest the camera can visualize data. The light-to-digital converter is a sensor that converts a light stimulus into a binary data. This binary data must be serialized to be sent to its destine using a communication protocol which both producer and receiver must share to be able to receive the correct message in the figure, the Serializer. The converter has a resolution, which is how many individual light streams can be captured given the physical constrains. Notice that the resolution is measured in pixels and a pixel is the smallest image unit with an attributed color. Color can be represented using a vector with the amount of each color primary colors. Primary colors are red, green, and blue (RGB), but other combinations can be used, such as CMYK (cyan, magenta, yellow, black – also known as key).

Both lens and light-to-digital converter are not perfect. Consequently, these imperfections are measured using an intrinsic matrix, defined as

$$K_{3\times3} = \begin{bmatrix} F/p_x & 0 & 0 \\ s & F/p_y & 0 \\ c_x & c_y & 1 \end{bmatrix}, \tag{2.1}$$

where p_x, p_y is pixels size in world units, F is the focal length in world units, s is the skew coefficient, and c_x, c_y is the optical center in pixels. World units are usually meters or millimeters, but these must to be consistent in all calculations.

A monocular camera is a camera which has only one sensor, in opposition of a stereo camera which has two sensors side-by-side. For a monocular camera, the approximated location of a point can be determined using

$$w \begin{bmatrix} x_{N\times1} & y_{N\times1} & 1_{N\times1} \end{bmatrix}_{N\times3} = \begin{bmatrix} X_{N\times1} & Y_{N\times1} & Z_{N\times1} & 1_{N\times1} \end{bmatrix}_{N\times4} RT_{4\times3}K_{3\times3}, \tag{2.2}$$

where w is a scale factor, N is the amount of points that are being converted from a real world to image perspective or from image to real world perspective, $\begin{bmatrix} x_{N\times1} & y_{N\times1} & 1_{N\times1} \end{bmatrix}_{N\times3}$ are the image points coordinates in pixels, in which $x_{N\times1}$ and $y_{N\times1}$ are column vectors with the pixel coordinates, in respectively, x and y, where the axis origin is the image top left corner. $\begin{bmatrix} X_{N\times1} & Y_{N\times1} & Z_{N\times1} & 1_{N\times1} \end{bmatrix}_{N\times4}$ are the points in world coordinates, where $X_{N\times1}$, $Y_{N\times1}$, and $Z_{N\times1}$ are column vectors with, respectively, the x,y and, z coordinates in the 3D plane and the $1_{N\times1}$ is a column vector with only ones in the same size as the X, the xyz axis is such that z is orthogonal to the xy plane, mathematically $x * y$,

$$RT_{4\times3} = \begin{bmatrix} R_{3\times3} \\ t_{1\times3} \end{bmatrix}_{4\times3}, \tag{2.3}$$

where $R_{3\times3}$ is the camera rotation matrix and $t_{1\times3}$ is the camera translation vector both in world units. The matrix RT converts a point in a 3D world coordinate to a 3D camera coordinate, then with the $K_{3\times3}$ this 3D point is rasterized from 3D to 2D. With some algebraic computation a pixel coordinate can be transformed into a point in world coordinates, or from a point in world coordinates the pixel representation (HEIKKILA; SILVEN, 1997; ZHANG, 2000).

A single monocular camera distances measurements after 40 m becomes increasingly harder, due to one pixel of error representing a big movement in further distances. Camera are also prone to any light block, as any visual sensor, which will be unable to detect what is behind the said block.

2.1.2 RADAR

RADAR is an active sensor which uses electromagnetic waves to quantify and qualify objects. The main measurement made by RADAR is distance, which is calculated by

$$R = \frac{cT_R}{2}, \tag{2.4}$$

where c is the light speed and T_R is the time from wave emission to wave reception. Object velocity is measured by the Doppler effect. Due to the RADAR having more than one antenna, the angle in which the wavefront arrives back at the RADAR informs the detection angle. The final measurement is the Radar Cross Section (RCS) which is given by

$$\sigma = \frac{\text{Reflected power/Unit of Solid Angle}}{\text{Power Density Incident}/4\pi} = \lim_{R\to\infty} 4\pi R^2 \left| \frac{E_r}{E_i} \right|^2, \tag{2.5}$$

where σ is the RCS, R is the distance to the object, E_r is the reflected electromagnetic field and E_i is the incidence electromagnetic field (SKOLNIK, 1980).

RCS is not necessarily bound to the targeted object geometric size. In fact, due to the target physical composition object, its reflection may be bigger or smaller.

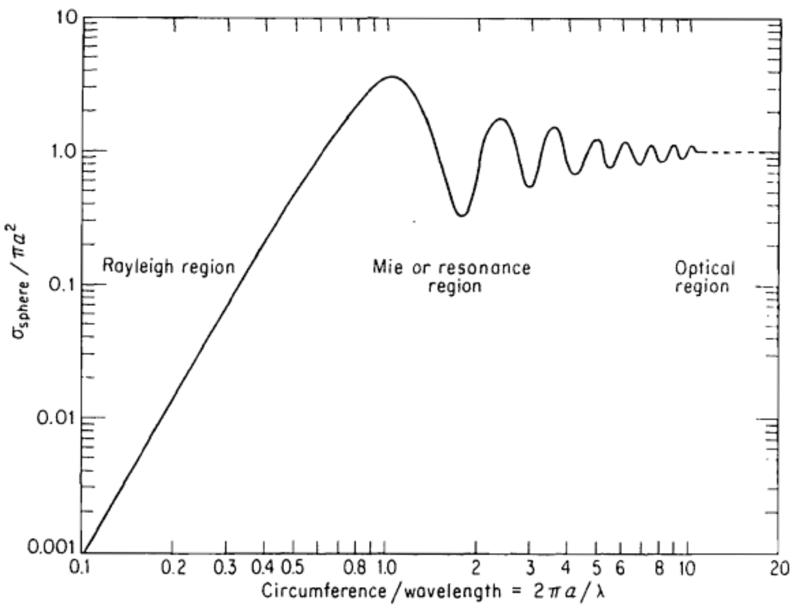
Whilst in normal circumstances a bigger target usually means a greater RCS, in specific circumstances due to scattering the RCS of a bigger target may be lower than a smaller target (WOLFF, 2022).

However RADAR is not a replacement to visual sensors. One example of this is Rayleigh region seen in FIGURE 5. The Rayleigh region is a region where the water droplets are invisible to RADAR. This region is defined via the approximation of a water droplet to a sphere with radius a and wave length λ, and is given by

$$T_{esf} = \frac{2\pi a}{\lambda}.$$

The region $T_{esf} \ll 1$ defines the Rayleigh region, where the water droplets (such as rain) are invisible to the RADAR, $T_{esf} \gg 1$ is the region where $\sigma \approx \pi a^2$. In between these regions, there is the resonance region, also known as Mie region (SKOLNIK, 1980). Such regions can be observed in FIGURE 5, which shows the RCS of a sphere with radius a and wavelength λ. Therefore with higher frequencies RADAR sensors, it perceives more rain than those with lower frequencies.

FIGURE 5 – RAYLEIGH REGION



SOURCE: Skolnik (1980)

The RADAR can be monostatic or bistatic. A monostatic RADAR is when the antenna that receive the signal is in the same location as those which transmit the signal. On the other hand, a bistatic RADAR is when the transmit antenna is separated from the receive antenna.

RADAR is an active sensor that emits a signal that interacts with the target and returns to the RADAR. This returned wave is the measurement in its purest form. An example of this is the Radarlog from INRAS (JOBY AVIATION INC., 2021). It is a

RADAR which dumps all the electromagnetic wave information received on the antennas to a file. Notice that there are uses for RADAR that dump all information, such as to test and develop algorithms, but in a automotive perspective, RADAR usually clusterize or even creates an object list from the received electromagnetic waves. For instance, the Continental ARS408 provides either a list of clusters or an object list. The list of clusters is a perceived objects into points clusterization, while the object list is a higher level of abstraction where the RADAR outputs data in the form of objects with size and dimension. Both cluster and objects are tracked through time and possess covariance, position, rotation, and classification (CONTINENTAL ENGINEERING SERVICES, 2022). The next layer of detection must define what is an object and its properties when using a cluster list.

2.1.3 Detection algorithms

While the RADAR can already output data in object list form, the camera outputs data in its purest form, a raw stream of data. Higher levels of fusion require an input of object lists. For that reason, a detection algorithm for monocular cameras can be considered. With the purpose of having a same baseline to compare algorithms, the KITTI datase was created by Geiger et al. (2012). KITTI is the pioneer dataset and has multiple competitions to instigate the growth of perception algorithms. Multiple scenes in different scenarios were catalogued and labelled with diferent difficulties.

TABLE 1 – KITTI DATASET TARGETS DIFFICULTIES

	Easy	Moderate	Hard
Minimum Bounding Box Height (px)	40	25	25
Maximum Occlusion Level	Fully Visible	Partly Occluded	Difficult to see
Maximum Truncation (%)	15	30	50

SOURCE: Geiger et al. (2019)

TABLE 1 shows the different difficulties in the KITTI dataset, where each dataset difficulty has different parameters, such as bounding box height projected in the image, how truncated a bounding box can be, and, a more subjective, how occluded the box is. KITTI has many sensors in its dataset, but not a RADAR. One example of monocular camera detector algorithm is Single-Stage Monocular 3D Object Detection via Keypoint Detection (SMOKE) proposed by Liu et al. (2020), which is a real time detection algorithm for monocular camera. SMOKE was trained using the KITTI dataset, and achieved the scores listed in TABLE 2. This table shows that the SMOKE has scores near 100% in detection and orientation, however, the 3D position shows the weakness of using only a monocular camera to detect images. Small differences in the projection matrices causes significant errors in the 3D projection.

TABLE 2 – SMOKE PERFORMANCE ON KITTI DATASET

Benchmark	Easy	Moderate	Hard
Car (Detection)	93.21 %	87.51 %	77.66 %
Car (Orientation)	92.94 %	87.02 %	77.12 %
Car (3D Detection)	14.03 %	9.76 %	7.84 %
Car (Bird's Eye View)	20.83 %	14.49 %	12.75

SOURCE: Geiger et al. (2019)

Currently, the highest scoring single monocular camera algorithm with the KITTI dataset is the Pseudo-Stereo-3D proposed by Chen et al. (2022). This algorithm emulates a stereo camera by utilizing a disparity map from the monocular camera. Then using both the monocular camera and the generated second camera the algorithm feeds its neural network with these data. Its scores can be seen in TABLE 3. This has achieved also near 100% in detection and orientation. The improvement in 3D position is significant, but limited to the monocular projection camera matrices.

TABLE 3 – PSEUDO-STEREO-3D PERFORMANCE ON KITTI DATASET

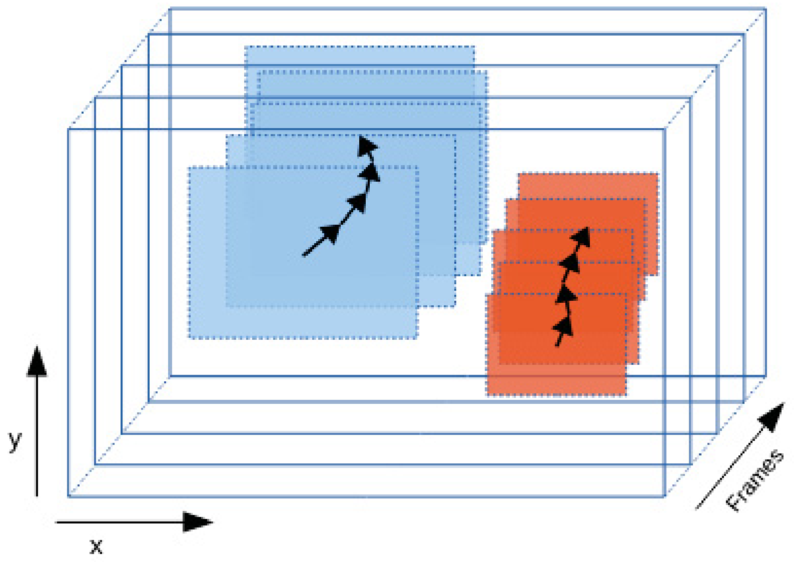
Benchmark	Easy	Moderate	Hard
Car (Detection)	95.75 %	90.27 %	82.32 %
Car (Orientation)	95.60 %	89.78 %	81.68 %
Car (3D Detection)	23.74 %	17.74 %	15.14 %
Car (Bird's Eye View)	32.64 %	23.76 %	20.64 %

SOURCE: Geiger et al. (2019)

2.1.4 Tracking

Given multiple objects through multiple frames in time, a tracking algorithm must be able to maintain the same identification to correct object through referred frames while the object exists as illustrated in FIGURE 6. This figure shows two distinct object one red and one blue. The tracking algorithm must be able to identify that the object with identification blue in frame one is the same object with identification blue in frame two with a different position. Acording to Bochinski et al. (2017), a good tracking algorithm is able to fill in the gaps between false positives and missing detection. Note that tracking is important to data fusion due to the need to fuse the same object from different frames. A Kalman filter can be used as a tracking algorithm. The advantage of using a Kalman filter is the object covariance calculation and object next step prediction. One example of tracking algorithm is motpy, developed by Muron et al. (2021).

FIGURE 6 – TRACKING OBJECT THROUGH FRAMES



SOURCE: Bochinski et al. (2017)

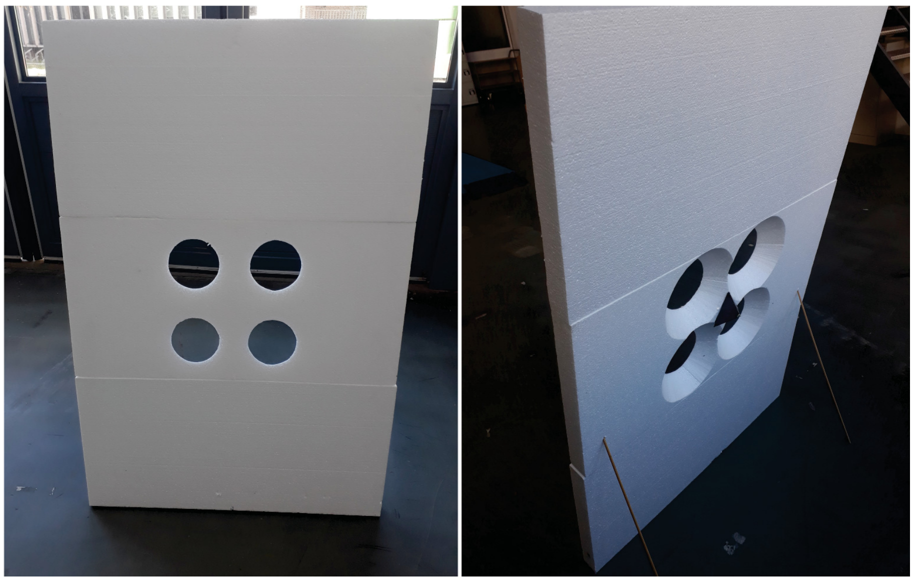
2.2 SENSOR CALIBRATION

As aforementioned, the camera has a lens and a lens-to-digital converter which are not ideal. Due these imperfections is imperative to make the camera image calibration. This calibration is known as intrinsic calibration, which makes possible to project world points on the image, as seen in Subsection 2.1.1. In this regard, OpenCV[1]– developed by Bradski (2000) – describes a tool that uses a squares pattern to calculate the intrinsic matrix using various methods. Square sizes must be measured and the amount of squares must be counted and inform to the calibration software.

Since the sensor calibration is an important step towards comparing sensor data each sensor measures are relative to its own relative reference frame. Then to proper compare and evaluate different sensors, sensor position calibration in the vehicle frame must be performed. This calibration is called extrinsic calibration. Which allows sensor data to be overlapped. An example of sensor calibration algorithm is the Multi Sensor Calibration proposed by Domhof et al. (2019). This algorithm calculates the sensors position relative to a reference sensor, using OpenCV's solvePnP algorithm and a genetic algorithm responsible for finding the most likely result to the proposed sensor setup.

[1] OpenCV is a programming library with function aimed mainly at computer vision.

FIGURE 7 – MULTI SENSOR CALIBRATION TOOL



SOURCE: Domhof et al. (2019)

FIGURE 7 shows the used calibration board in Multi Sensor Calibration to calibrate RADAR, Camera, and Stereo Camera. The target is a Styrofoam calibration tool with a corner reflector placed in the pattern middle. The circles in the target are for camera and LiDAR detection. So that LiDAR does not detect the target edge, the circle has a cone shape.

NVIDIA Driveworks, developed and maintened by NVIDIA Corporation (2022), also has a toolkit for extrinsic calculation of camera position in a vehicle using camera images. This tool uses printed april targets[2] of different sizes, four to be stuck to the car wheels, one to be placed in front of the camera, and two to be placed at each side of the car. Then another camera takes car pictures with the targets. While the calibration camera, takes a picture from the front target. With these data and the measures of printed targets and vehicle sizes, such as wheel diameter and car length, the program can be executed. This extrinsice calibration provides the developer with a calibrated camera position and rotation in relationship to the back axle.

[2] April targets are a special kind of target which the boxes have orientational points and shapes that facilitate the camera detection.

2.3 FUSION

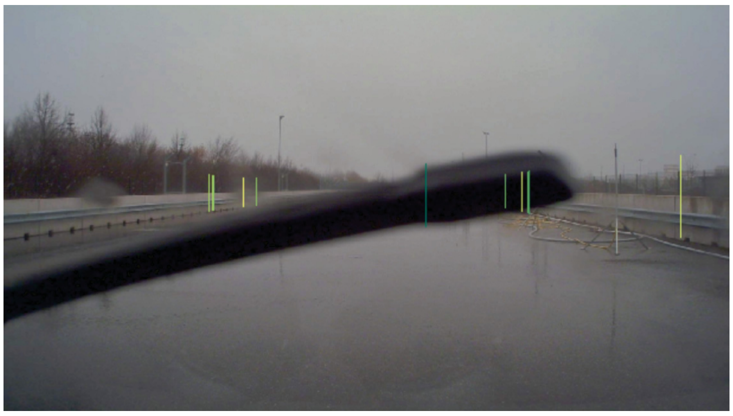
Given a system with multiple sensors in the same environment, fusion consists of combining data from different sensors, which may have quite different measurements, into one combined environment description. According to Aeberhard (2017), each sensor has strong and weak points, by fusing sensor data not only the output is more robust, but also the performance is increased.

Aeberhard (2017) also describes that Low-Level fusion Architecture is applied to fuse raw data which comes directly from the sensors with a central tracking algorithm, while in the High-Level fusion Architecture, each sensor tracks and filters the data before sending to the fusion module. The main advantage of Low-Level fusion when comparing with other fusion levels is that the accuracy is the highest, due to no assumptions being made in sensor level and the data between sensors are independent.

2.3.1 Fusion Methods

Neural Networks can fuse sensor data. For instance, the CRF-Net addressed in Nobis et al. (2019a) uses RADAR data to enhance data from camera. The algorithm overlaps the RADAR data with the camera data, by making the RADAR points rasterization into the image. This new image with the RADAR points overlapped in the image is the algorithm input. Comparetively, RADAR data has a lower frame rate than a camera. Therefore CRF-Net has a BlackIn system. This system turns off the camera detection for a couple instants so that the neural network does not ignore the RADAR data. The fusion of RADAR data with camera provides significant fusion data reliability due to the camera not generating objects when occluded. For example, in FIGURE 8, the wiper is in the target way, however, the RADAR has no such obstruction, when projecting the RADAR points onto the image the target can be observed, which would not been visible using only camera.

FIGURE 8 – CAMERA AND RADAR FUSION MOTIVATION



SOURCE: Author(2022)

Other fusion algorithms include, but not limited to, Nabati e Qi (2021) which proposed Center Fusion, a neural network which associates RADAR clusters with 3D pillars using a Frustum method to create a characteristic map that enhances the image in a middle level fusion.

While Yadav et al. (2020) shows a similar work to CRF-Net using a RANet and BIRANet in the feature level fusion. With a different approach, Bai et al. (2021) fuses camera and RADAR by making a surroundings track using an elliptic discriminant and mapping this track to a finite set of correlated data in the image plane with a Gaussian mixture.

With a constrained budget, Lim et al. (2021) proposes that fusion of camera and RADAR is less useful than fusing RADAR and LiDAR due to the quick dropout of radar data in the first layers of Neural networks in benefit of images.

Finally, Aeberhard (2017) proposes a statistical High to Hybrid Level Fusion. In which there are three level of data processing: sensor, fusion, and application level each level outputs an object list which is an input for the next level. Each object has a state vector, a dimension vector, a feature vector, and a classification vector. A state vector is composed of position, rotation, velocity, and acceleration parameters. A feature vector informs which features are closest to the ego. A classification vector informs the type the object belongs to. Firstly, this algorithm aligns the inputs spatially and temporally. With the input objects aligned, each input object can be associated to a global object, and then the input object is fused with the global object.

2.4 VIRTUAL TESTS

Simulators are used to perform virtual tests. Simulators are programs that enable to test multiple scenarios in a laboratory. CarMaker is a simulator from IPG Automotive (2021) that has support to real time and is able to simulate RADAR, Camera, LiDAR, and many other sensors. This simulator can simulate the real world distortion of camera such as noise, or the rain disturbances in RADAR and LiDAR. CarMaker is written in C and has an extensive support of this programming language. A scenario editor is also available and allows the user to draw its own scenario, in which the simulation will occur. IPGMovie and MovieNX, are the simulation visualization tools, which accept multiple objects with different 3D formats and allows to add driving functions to the ego vehicle.

Carla is an open source simulator based on the Unreal Engine 4, developed by Dosovitskiy et al. (2017). Python programming is supported and it is able to simulate RADAR, camera, and other sensors. Nonetheless, Carla lacks real time capabilities. Real time is important to make simulations with HiL due to the sensors being real world objects. Carla also lacks RCS support on RADAR objects.

3 METHODOLOGY

In order for there be a complete comprehension of how this work has been executed, this chapter will discuss how the test were made. For this a discussion on how sensors generated data, which sensors where used, and how they were calibrated is made in Section 3.1. Then the comprehension of how the sensor data is acquired, where does it goes after generation, and who is receiving it is explained in Section 3.2. After that, how the sensor data receiver collects the data, using which protocols, platforms, how the data is stored, and how the fusion algorithm receive the data from the sensor receiver is done in Section 3.3. Given fused data, the next step is to evaluate the data, and validate the fusion data, which are done in Section 3.4. Section 3.5 describes which tests were done in real world. At this point there should be completely understood how the system to evaluate the difference between real and simulated scenarios works. Therefore, the predetermined tests in a real world facility with an real world car and data collection can be executed. With this data the scenario virtualization, and its simulation in laboratory, are discussed on Section 3.6.

3.1 DATA GENERATION

Data generation is the process in which sensors perceive the environment. The scope of this work aims only to fuse data from camera and RADAR, LiDAR was utilized for calibration purposes.

3.1.1 Camera

The utilized camera was the Sekonix SF3325-100. This camera has resolution of 1920 px (pixel) horizontally and 1232 px vertically. With frame rate of 30 frames per second. For transmission the camera utilizes the Gigabit Multimedia Serial Link (GMSL) protocol, which provides a connection with gigabit transmission speed via one single cable (INTEGRATED MAXIM, 2022). The used camera can be seen in FIGURE 9.

FIGURE 9 – CAMERA SEKONIX

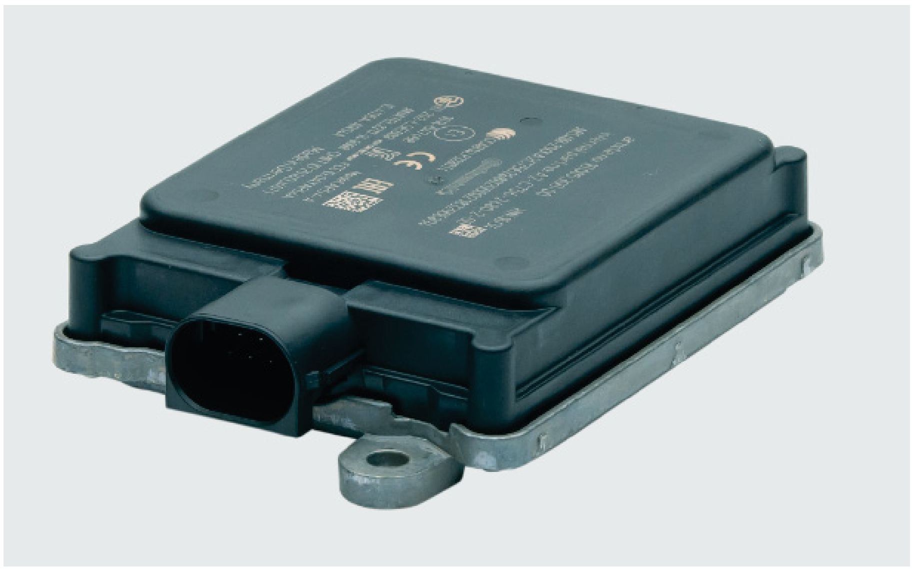
SOURCE: SEKONIX Co., Ltd. Corporate Headquarters (2022)

The Robot Operating System (ROS) camera calibration tool, was utilised for intrinsic calibration. The calibration tool is available and maintained by Stanford Artificial Intelligence Laboratory (2018).

3.1.2 RADAR

The utilized RADAR was the Continental ARS408, an automotive RADAR. According to Continental Engineering Services (2022) the frequency range is between 76 and 77 GHz and operating 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 to $+200$ km/h (where negative velocity is coming towards the RADAR and positive velocity is going away from the target). The ARS408 can be seen in the FIGURE 10. Connection is made with the use of a standard connector which makes connection with an Electronic Control Unit (ECU) using the protocol Controller Area Network (CAN). This RADAR has two operating modes, object list and cluster list.

FIGURE 10 – RADAR ARS408



SOURCE: Continental Engineering Services (2022)

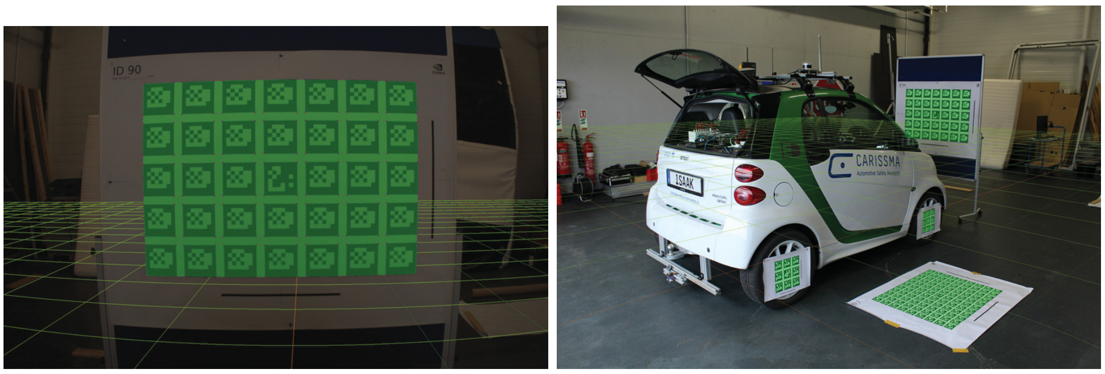
3.1.3 LiDAR

The utilized LiDAR was the Ouster OS1 128 channels. Data is transmitted utilizing a User-Defined Packet (UDP) Ethernet packet. This LiDAR range is 100 m in sunlight and minimum range of 0.3 m with accuracy of 3 cm. Also, it is configurable horizontal resolution of 512, 1024, or 2048, and vertical resolution of 128 channels (OUSTER, INC., 2021). This sensor was only utilized to make the extrinsic calibration.

3.1.4 Extrinsic sensor calibration

The intrinsic calibration calculates the camera distortion. The extrinsic calibration is used to define the sensors location in world coordinates. Firstly, the NVIDIA driveworks plataform was used (available at NVIDIA Corporation (2022)), which provides a toolkit to locate the camera position in relation to the car back axle. Then with these measurements, and considering the camera as the reference sensor, the relative location from any other calibrate sensor to the reference sensor can be measured using Domhof et al. (2019). With the relative position of one sensor to the reference sensor, the point projection determination from the frame of one sensor to the frame of the reference sensor can be done.

FIGURE 11 – NVIDIA EXTRINSICS CALIBRATION



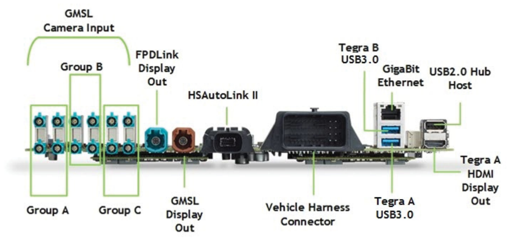
SOURCE: Author(2022)

The reference sensor position was defined as the camera, ergo the NVIDIA extrinsic calibration was used, which is shown in FIGURE 11. In this figure there is a plane drawn out in light green, this is the car plane. The car plane, as defined by International Organization for Standardization (2011), is zero in the car back-axle center. On the figure there are the targets on the car, ground, and on front of the car, with these and the car specifications, Driveworks is able to measure the camera position relative to the back-axle. Due to the RADAR bidimensionality camera projection points, the extrinsic location of the sensors could multiple planes. It was observed that the extrinsic calibration produced better results when using a LiDAR, because it outputs points in three dimensions.

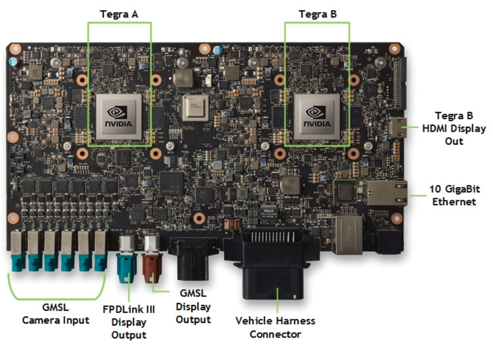
3.2 DATA ACQUISITION

The Nvidia Drive PX 2 AutoChauffeur (P2379) is a development platform for autonomous vehicles. This platform has two cores known as Tegra A and Tegra B. Note that, they come in duplicity to increase security in case of failures or to distribute the processing load. The frontal view from the DrivePX2 can be seen in FIGURE 12a, in which various inputs and outputs interfaces are shown. FIGURE 12b shows the top view, where the separation of the two Tegra cores is highlighted.

FIGURE 12 – DRIVEPX 2



(a) Front View



(b) Top View

SOURCE: NVIDIA Corporation (2018)

Available inputs and outputs for the "Nvidia Drive PX 2" are 3 arrays of 4 GMSL cameras, 1 gigabit Ethernet adapter, one 10 gigabit ethernet adaptor, and 12 CAN interfaces. The Nvidia Drive PX 2 also counts with the Driveworks platform which is a toolkit that provides multiple tools to help the developer create an autonomous vehicle (NVIDIA CORPORATION, 2022).

3.3 COMPONENT IMPLEMENTATION AND INTEGRATION

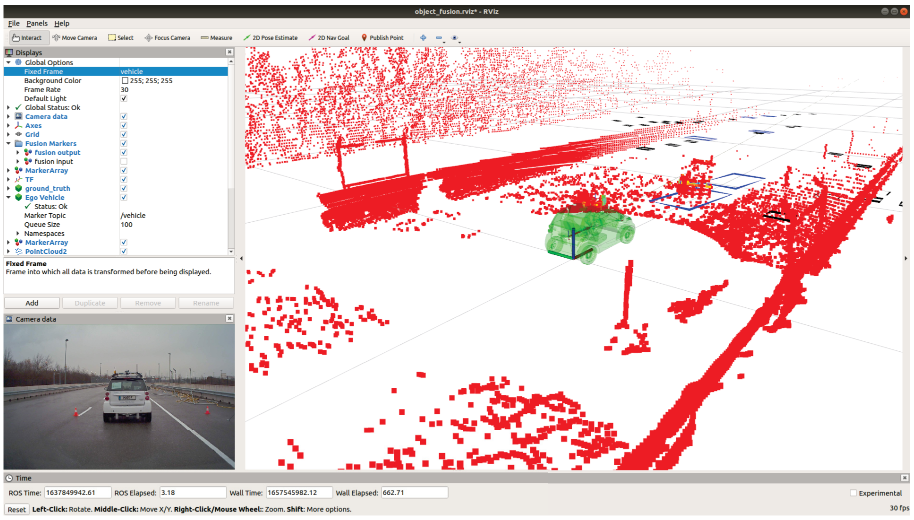
The hardware described in previous sections must have all data that is incoming to the ECU saved utilizing some software. In this work, the utilized software was ROS.

ROS is an open source set of tools to aid the development of robot applications (STANFORD ARTIFICIAL INTELLIGENCE LABORATORY, 2018), which was utilized as for having a great amount of libraries for various sensors and applications that are ROS-based such as Domhof et al. (2019). ROS was also used for the single timing frame for all frames.

ROS has a structure based on topic and nodes which makes the data more accessible. Nodes are computation process that perform tasks and can communicate with each-other using topics. Topics are how they communicate. The topics have an anonymous communication, just as a radio, a publisher topic is not aware of whom is listening to its broadcast, whilst the subscriber node is publisher aware. Due to this, it is possible to have multiple publishers and subscribers to the same topic. Also ROS permits the use of launch files which execute multiple ROS nodes at once and to set parameters which are the ROS equivalent of variables, which can be global or local. ROS topics can be user defined, increasing the amount of data communicated between topics (STANFORD ARTIFICIAL INTELLIGENCE LABORATORY, 2018).

In order to visualize the data acquired, ROS offers RViz, which is a powerful tool to visualize any ROS type message that can be visualized. In the scope of this work, the RViz was taken into account to visualize fusion inputs, outputs, and debugging. The RADAR in RViz is a great example due to the data complexity. The comprehension of RADAR outputs of cluster or object list is simpler with a visualizer. And with the LiDAR data projected on the visualization tool, its is possible to overlap the RADAR to LiDAR.

FIGURE 13 – RVIZ VISUALIZATION TOOL



```
Global Options
  Fixed Frame          vehicle
  Background Color     255; 255; 255
  Frame Rate           30
  Default Light        ✓
Global Status: Ok
Camera data            ✓
Axes                   ✓
Grid                   ✓
Fusion Markers         ✓
  fusion output        ✓
  fusion input
MarkerArray            ✓
TF                     ✓
ground_truth           ✓
Ego Vehicle            ✓
  Status: Ok
  Marker Topic         /vehicle
  Queue Size           100
  Namespaces
MarkerArray            ✓
PointCloud2            ✓

Fixed Frame
Frame into which all data is transformed before being displayed.

ROS Time: 1637849942.61   ROS Elapsed: 3.18   Wall Time: 1657545982.12   Wall Elapsed: 662.71
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
```

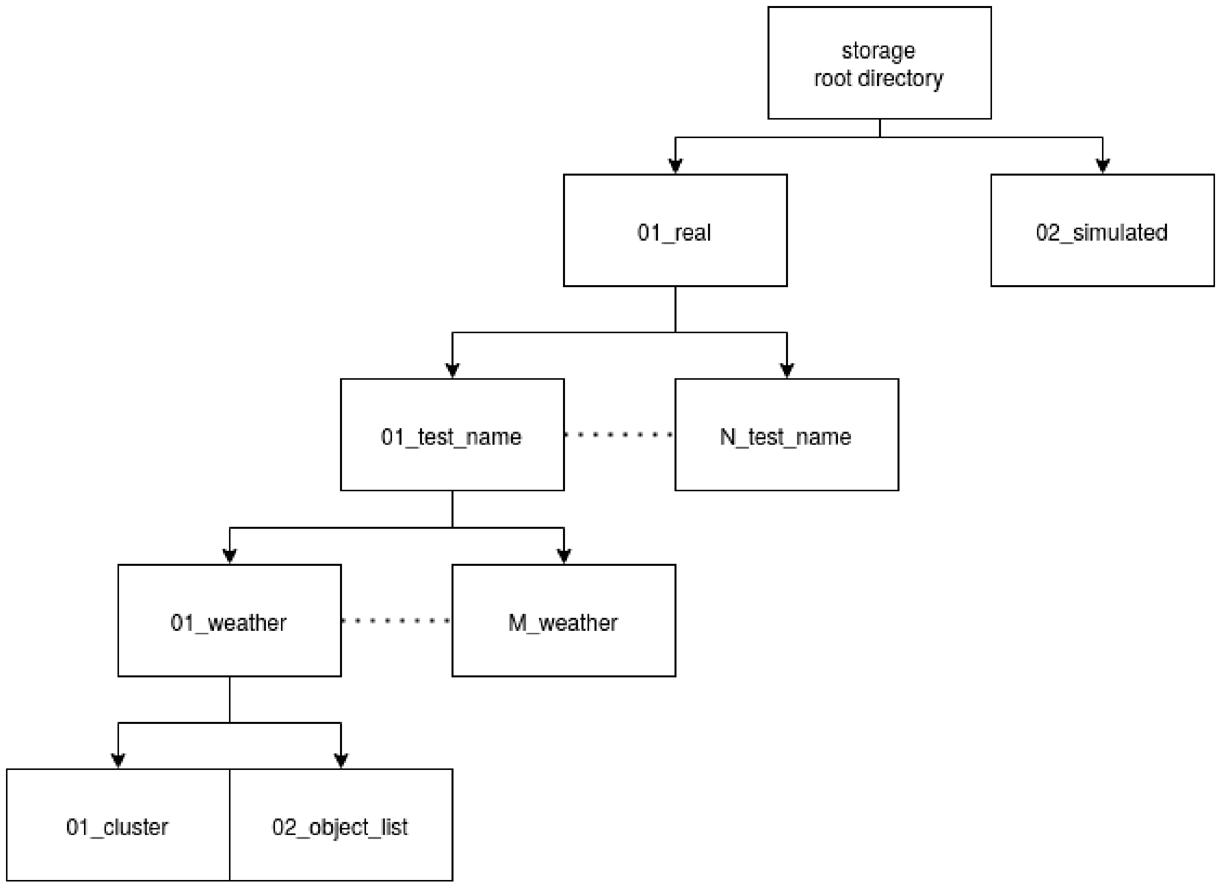
SOURCE: Author(2022)

FIGURE 13 shows the RViz interface, in which is presented the data collected from one tests. Observe that there are two columns in figure, the column to the left shows the list of topics on top and on bottom there is the camera raw image and on the right the XYZ plane is shown, where the sensor measurements and the ego vehicle are. The red dots on the XYZ plane represent the LiDAR points on the world space, and the boxes of blue and black color are the RADAR object list detection.

ROS is a framework that can be programmed using Python, or C/C++ programming language. In this work, Python was mainly used as the programming language. But for data collection and calibration modules that were programmed in C/C++, such as the multi sensor calibration tool were used.

A rosbag, a ROS program, can be used to record the topics of interest to the developer. This is an useful tool because it provides the means to play the same data equally, with the same clock, as of recording time. This permits to make a real time analysis in a second moment and not necessarily in the same instant as the recording or testing.

FIGURE 14 – STORAGE TREE



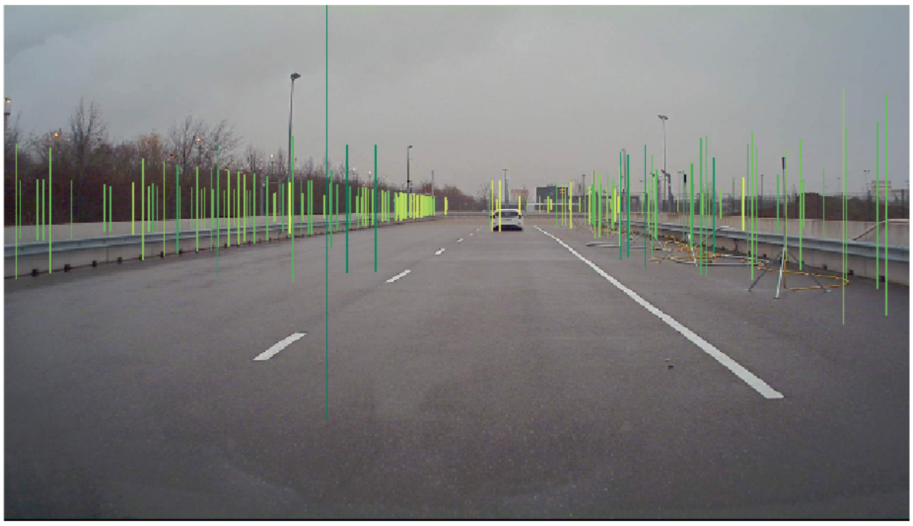
SOURCE: Author(2022)

As important to store data, it is to know which data is related to while test, therefore a storage format consisting of numbered directories in a tree environment was utilized. FIGURE 14 shows the storage tree. Starting at the data set root, there are two separated main branches, 01_real and the $02\_simulated$, as they are equal in structure, only one main branches will be explained. Branch 01_real divides itself into N branches, being N the number of tests made, therefore $N \in \mathbb{N}^*$. The choice to use a preceding 0 before the number is to prevent the eventual tenth test being listed before the first. After the test number follows a underscore and the name test. Then on each test branch, there are M weather conditions, where the name also follows the same test name convention and $M \in \mathbb{N}^*$. Finally, for each weather were made tests with two conditions, one with the RADAR in cluster list mode with the name $01\_cluster$, and other with the RADAR in object list mode with the name $02\_objlist$. Within these last branches were saved the rosbag data files. As per International Organization for Standardization (2008), which specifies at least three test runs for each test scenario, three test runs where executed for each bottom branch.

3.3.1 Fusion Algorithms

The Camera Radar Network (CRF-Net), which was created and maintained by Nobis et al. (2019b), is a neural network which was used to fuse data from camera and RADAR. To fuse RADAR and camera sensors, CRF-Net receives in the input layer an image plus, which is an image with the RADAR points projected to the camera using the camera RADAR transformation matrix provided by the multi sensor calibration and the intrinsics matrix provided by the ROS camera calibration. With the RADAR points projected on the image, lines with 3 meters tall in the location these of these points are estimated, the RADAR data is added as a new layer to the image. In FIGURE 15 there is the RADAR points projection on to the image with 3m tall lines. The CRF-Net is keras framework based, keras maintained by (CHOLLET et al., 2015).

FIGURE 15 – CRF-NET IMAGE PLUS



SOURCE: Author (2022)

The CRF-Net is a low-level to the data of camera since prediction is made directly on raw data from the camera sensor, but for the RADAR the data is not as raw. Instead of CRF-Net receiving data from the electromagnetic waves directly from the RADAR receiver, it receives clusters with some post processing, such as clusterization, classification and tracking. Therefore, is a lower level than object list fusion but not the lowest possible fusion. In this work, the CRFnet fusion will be known as the Low-Level Fusion. Originally, CRF-Net was not made ROS compatible, only accepting not real time data, in this work, the CRF-Net was converted to a ROS node so that predictions can be made at near real-time rates.

The Aeberhard (2017) proposed algorithm was implemented in this work and published at Poledna et al. (2022). The object fusion algorithm is a statistics based fusion. This is a hybrid-level fusion algorithm, where the inputs are object lists. To use this algorithm, each sensor provides object lists. With the RADAR is a simple change of configuration, but with the camera, a module needs to be added as the utilized camera does not have the object list creation. In this work this fusion will be referred to as the High-Level Fusion.

To convert the raw data from the camera into object list, SMOKE Neural Network was used, maintained by Liu et al. (2020). The SMOKE module is pyTorch framework based, pyTorch maintained by Paszke et al. (2019). SMOKE was originally implemented as a full data set algorithm. In this work, SMOKE was converted to a ROS node so that use in near real time updates can be done. However, SMOKE does not track the object, consequently a Multi Object Tracker was implemented, maintained by Muron et al. (2021). With the tracker the previous frame object is tracked to be the same as the current object or a new object. The importance of this is to make possible that the fusion algorithm proposed by Aeberhard (2017) has its inputs in correct identification and the covariances of each object is sustained and not mixed.

3.4 EVALUATION METHOD

An important concept for evaluation is the ground truth, which is the actual object position in the test. In this section, fusion evaluation and the simulation-to-reality gap estimation will be described.

The CRF-Net fusion output is a 2D bounding box, therefore validation occurs with the ground truth 2D projected bounding box. The 2D bounding box validation is done by using the Intersection over Union (IoU)[1] between bounding boxes of ground truth and the detection. The IoU measures the relationship between bounding box coordinates and ground truth coordinates. A high IoU indicates a predicted bounding box that is close of the ground truth. With this value, it is possible, with a threshold of 0.5 for the detection to be a true positive, to create a confusion matrix. The confusion matrix is created by tabulation of True Positives (in which the model accurately predicted the label ground truth), the True Negatives (where the model does not predict the label and does not predict a ground truth), the False Positives (in which there is a prediction, but it is not part of the ground truth), and False Negatives (which the model does not predict a label, but it is part of the ground truth). With the confusion matrix, generated from the IoU, the Average Precision (AP) can be calculated. The precision measures the True Positives, from all positive predictions, and is defined as

$$\textit{Precision} = \frac{TP}{TP+FP}, \tag{3.1}$$

where TP is the number of True Positives, and FP is the number of False Positives. Recall measures the true positives of all predictions, such as

$$\textit{Recall} = \frac{TP}{TP+FN}, \tag{3.2}$$

[1] Also known as the Jaccard Index.

where FN are the number of False Negatives. With Recall, and Precision defined, the AP is defined as

$$AP = \int_0^1 precision(recall)d(rec), \tag{3.3}$$

where $precision(recall)$ is the precision-recall curve, which demonstrates the relationship between precision and recall for different thresholds. The AP is the area under the curve of the precision-recall. With AP of each class, the mean Average Precision (mAP) can be calculated. The mAP is a metric used for object detection evaluation. It is a numeric value in the range $[0, 1]$, which can be defined as

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i, \tag{3.4}$$

where AP_i is the average precision for a class and N is the number of classes. The mAP measures the average precision for each class and then the mean average precision for all classes. In this work, it was utilized the Cartucho et al. (2018) python mAP implementation[2], which is an adaptation of the MATLAB code into Python, of the PASCAL VOC 2012 mAP criterium.

The High-Level Fusion outputs an object list with objects and covariance, then to validate this fusion the ground truth will be projected onto the 2D camera frame.

To determine the best performing fusion algorithm in these proposed scenarios, only real data is used. Each fusion algorithm will have its mAP calculated, utilizing only the 2D bounding box in the frame and the ground-truth. By comparing each method mAP values, the best performing algorithm is determined.

3.4.1 Simulation-to-Reality Gap

The Simulation-to-Reality Gap is a measure that allows to know how good is simulation compared to a real scenario. With a near perfect simulation, when compared with reality, a real scenario may not be needed to test the proposed algorithm. Also, good simulation allows testing impossible scenarios, even if they are physically, or morally impossible to be executed.

The simulation-to-reality gap will be assessed by difference of fusion performance in the same scenario, virtual and real. Ideally,

$$\mathcal{G} = \mathcal{R} - \mathcal{S} = 0, \tag{3.5}$$

where $\mathcal{G}$ is the simulation-to-reality gap, $\mathcal{R}$ is the real scenario performance (in this case mAP), and $\mathcal{S}$ is the simulated scenario performance (in this case mAP). Should $\mathcal{G} \neq 0$ there is a gap between simulation and reality. $\mathcal{G} < 0$ indicates that simulation is

2 Code available at: https://github.com/Cartucho/mAP.

outperforming the real scenario and $\mathcal{G} > 0$ shows that the simulation is under-performing the reality.

3.5 TEST SCENARIOS

With scenario location defined, the car setup made and calibrated, the last thing to be discussed are the scenarios that will be performed by the vehicles to evaluate and validate the algorithms. Multiple scenarios were chosen to be tested, some created by European New Car Assessment Programme (EuroNCAP) (which will be demonstrated in Subsection 3.5.1) and some created without these restrictions, such as static ego, and perpendicular. Static ego tests will be discussed in Subsection 3.5.2 and perpendicular will be discussed in Subsection 3.5.3. But in the scope of this work only selected tests will be evaluated and validated. Each test was made in at least three different conditions, day, night, and rain, with at least 3 test runs each.

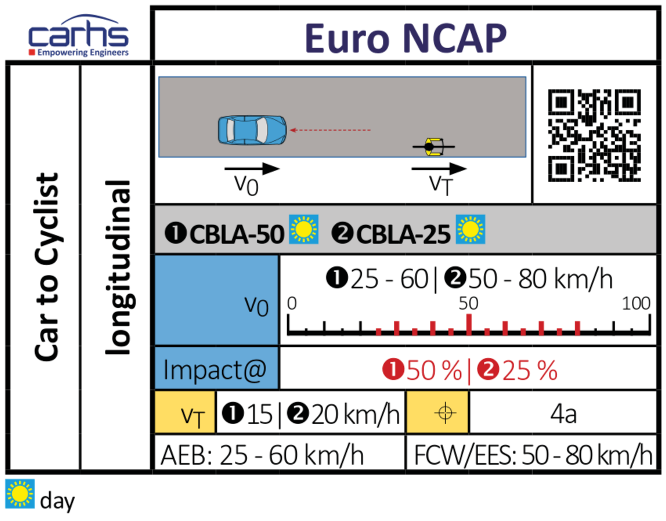
3.5.1 EuroNCAP

EuroNCAP is a voluntary non-profit European program that was formed in 1996, which rates car safety on a rating system from zero to five stars where five stars is the best. To make this rating, the program has created multiple scenarios that are reproducible with multiple vehicles so that the results are comparable. (EURO NCAP, 2022)

3.5.1.1 CBLA - 50

The Car to Bicycle Longitudinal (CBLA) is an Automatic Emergency Braking (AEB) Test in which the ego is going towards a bicycle with velocity between 25 km/h to 60 km/h and the bicycle is going away from the car with velocity of 15 km/h. There are two versions of this test one with the car hitting the bicycle at 50%, and another at 25% the car lateral size. The test information can be seen in FIGURE 16. In this work, the test made was the CBLA-50 with the ego going at 50 km/h. The bicycle dummy used was a validated dummy from manufacturer 4A that was pulled with the assistance of another car. In the scope of this work, this scenario will not be evaluated due to validity of this test concerns, such as the rig used to pull the dummy being seen by the RADAR.

FIGURE 16 – EURO NCAP CBLA TEST

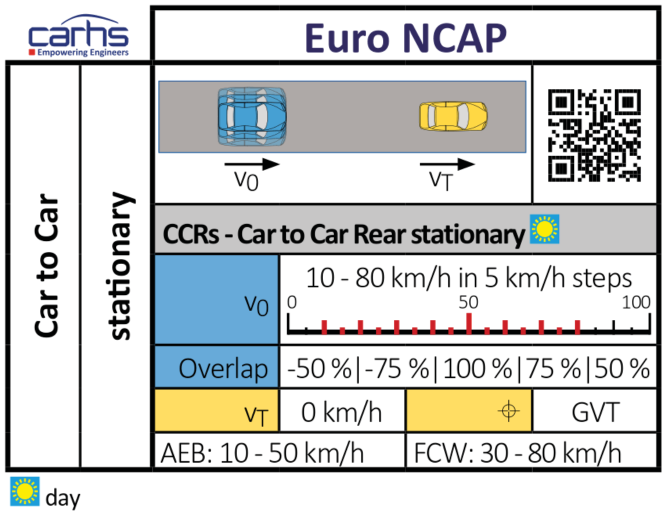


SOURCE: carhs training GmbH (2020a)

3.5.1.2 CCRs

The Car to Car Rear static (CCRs) is an AEB test in which there is a target stopped at a fixed distance while the ego comes at it with velocity of 50 km/h. In this work, the test was performed with a fixed dummy from 4A which has RADAR and camera validated targets. The notation can be seen on FIGURE 17.

FIGURE 17 – EURO NCAP CCRS TEST

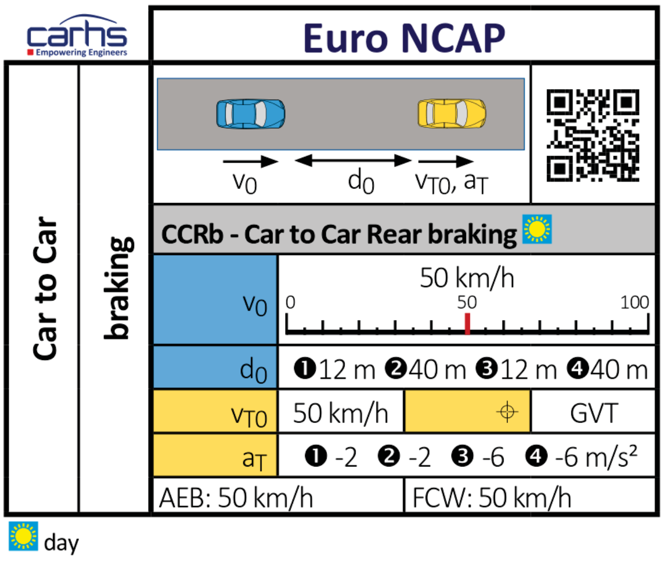


SOURCE: carhs training GmbH (2020b)

3.5.1.3 CCRb

The Car to Car Rear braking (CCRb) is as well an AEB test, but the scenario both vehicles are moving. The ego goes towards with velocity of 50Km/h and the target, yellow car in figure, goes at 50Km/h when it stops with a deceleration of -2ms^{-2} or -6ms^{-2}. This causes an motion where the target goes away from the ego up until an maximum distance and then the ego gets closer to the target when the target stops, with this test is possible to evaluate the sensors under both conditions, a target going away and getting closer. This test was not made to the EuroNCAP conditions, so in this work it will not be evaluated. This test notation can be seen in FIGURE 18

FIGURE 18 – EURO NCAP CCRS TEST


carhs Empowering Engineers

Euro NCAP

Car to Car | braking

v_0 d_0 v_{T0}, a_T

CCRb - Car to Car Rear braking ☀

v_0	50 km/h (0 – 50 – 100)		
d_0	❶12 m ❷40 m ❸12 m ❹40 m		
v_{T0}	50 km/h	⌖	GVT
a_T	❶ -2 ❷ -2 ❸ -6 ❹ -6 m/s²		
AEB: 50 km/h	FCW: 50 km/h		

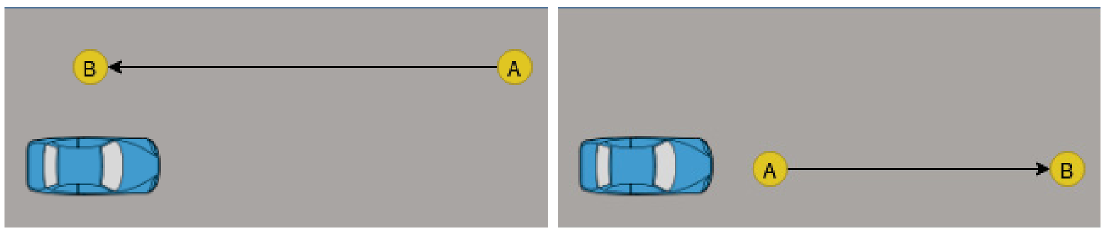
☀ day

SOURCE: carhs training GmbH (2020c)

3.5.2 Static

The static ego goal is to test the sensor setup in static conditions. In this test case, the ego stays static while the target moves away or towards the ego. When the target move towards, the target comes from the opposite lane. This test representation can be seen in FIGURE 19a and FIGURE 19b which show, respectively, the scenario towards and away. The target representation is a point due to this test being made with car, pedestrian, bicycle and truck.

FIGURE 19 – STATIC TESTS



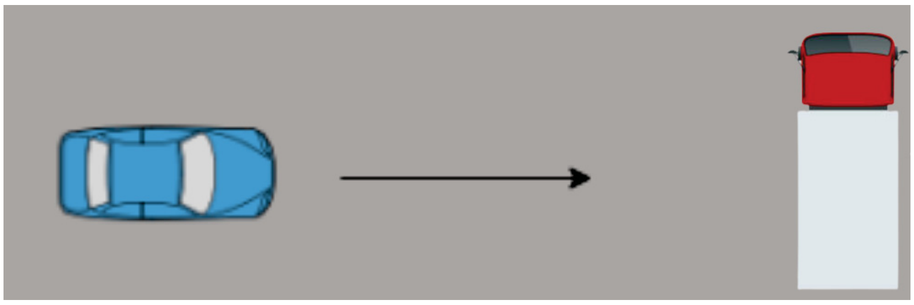
(a) Target Towards Ego (b) Target Away Ego

SOURCE: Author (2022)

3.5.3 Perpendicular

The perpendicular test was made with the reason to simulate the first fatal crash from an software controlled car. With that reason the target utilized was a truck. The ego would then accelerate towards the truck and stop right before its side. With this test, sensor are evaluated when facing a target much bigger than itself, a truck.

FIGURE 20 – DYNAMIC TESTS PERPENDICULAR



SOURCE: Author (2022)

3.6 TEST ENVIRONMENTS

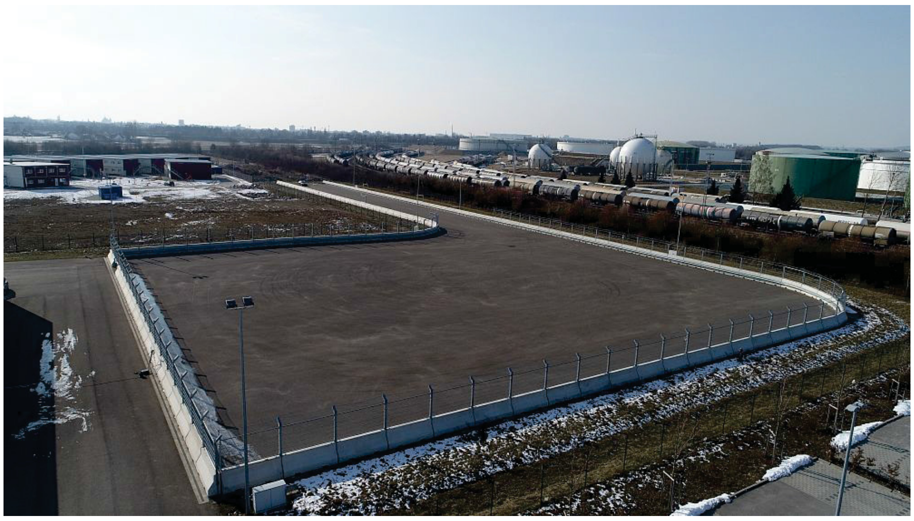
The test environments are a crucial part to this work, a controlled reproducible virtually scenario is a must have for there be a valid comparison between the virtual and real environments. In an uncontrolled environment, such as city street, the ground truths must be annotated by hand and cannot be fully precise, while in a controlled environment all participants position in the test are known. In these section, real scenario and its virtual counterpart are discussed.

3.6.1 Real tests

The controlled facility in the real world was the Center of Automotive Research on Integrated Safety Systems and Measurement Area (CARISSMA) Outdoor Facilities in Ingolstadt(Germany), from now on refereed as "the outdoor". The outdoor is a facility that belongs to the CARISSMA laboratory and provides a safe and reproducible scenario. There are two parts, the acceleration lane and the roundabout. The test here proposed were all made in the acceleration lane.

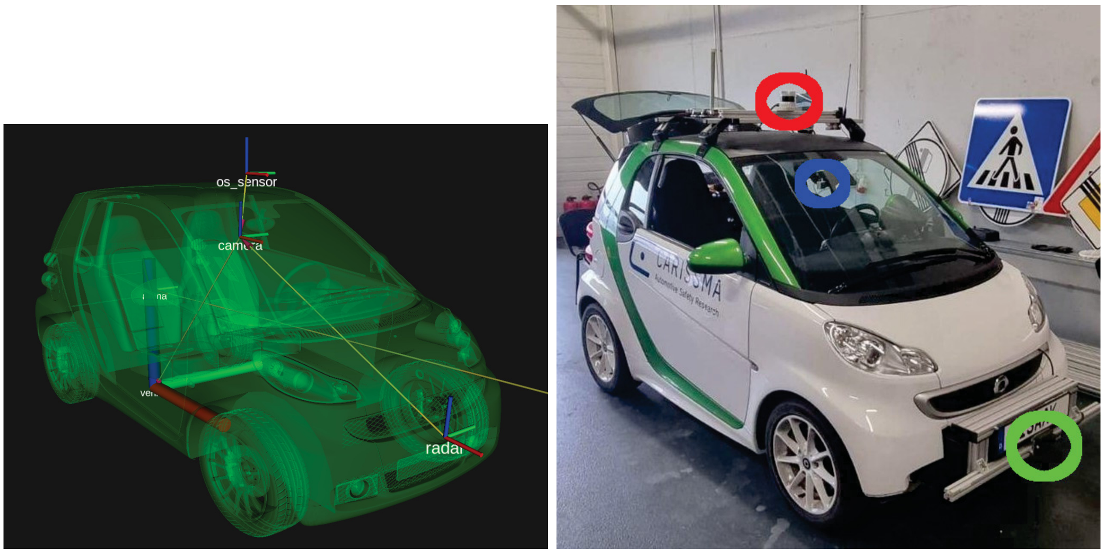
FIGURE 21 shows the outdoor from a side-top perspective. The outdoor provides, for those who use it, precise data from car and target location using Global Positioning System (GPS). A portable GPS is available, for the cases that the target does not support an full , but vehicles for the outdoor, such as, the ones used in this work possess an IMU of high precision, the Automotive Dynamic Motion Analyzer (ADMA) from GENESYS systems, which does provide not only data of location but also rotation, velocity, and acceleration, developed by GeneSys Elektronik GmbH (2022). The outdoor has validated targets from 4A which are important for tests which might incur on collision.

FIGURE 21 – CARISSMA OUTDOOR FACILITIES



SOURCE: CARISSMA(2018)

FIGURE 22 – EGO DIGITAL TWIN



(a) Front View (b) Ego sensors installed

SOURCE: Author(2022)

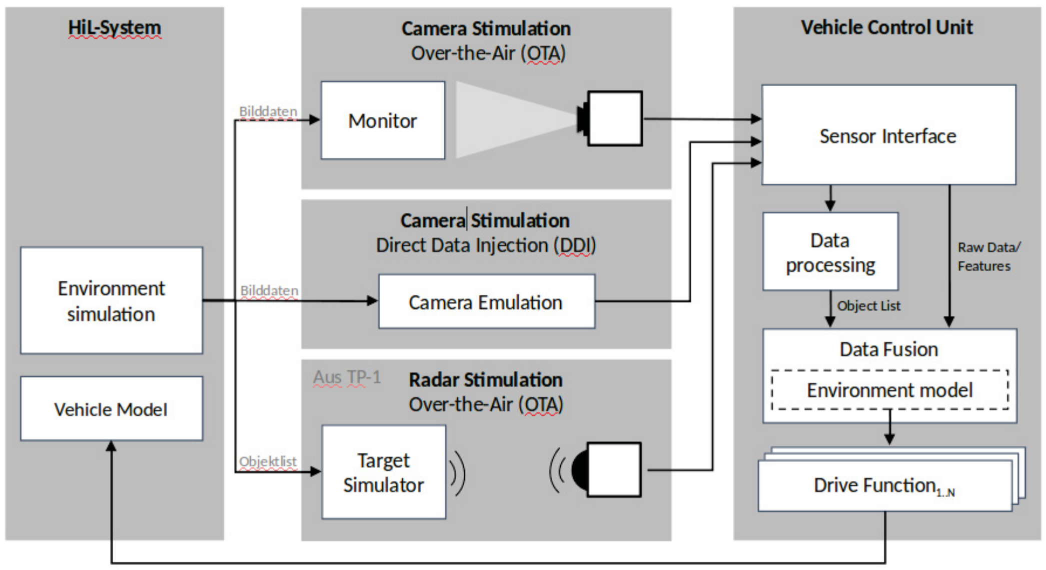
The utilised car to install the sensors, listed in previous section, can be seen in FIGURE 22b. The car is an E-Smart modified to have more power supply points and sensor anchor points. On FIGURE 22b the red circle shows the LiDAR, the green circle the RADAR, and the blue circle the camera.

3.6.2 Virtual tests

After the real tests were made and the data was collected, the virtualization process begun. Digital twin from the outdoor map, the ego with sensor locations, targets locations, velocities from ego and target, and positions were created. The ego digital twin can be seen in FIGURE 22a, in which can be seen the sensor coordinated frames for each frame, the color was chosen to be be transparent green to facilitate the coordinated frames of each sensor visualization, but the car can be colored to the correct colors. The real life ego can be seen in FIGURE 22b.

To simulate the digital twin from the real scenario, the software CarMaker was used, which is developed by IPG Automotive (2021). For this project between the many features, the most important ones are HiL, Vehicle in the Loop (VIL), and target virtualization. VIL support is important due to the ego vehicle must move exactly equal to the real scenario so that the movement is a digital twin as well. VIL overwrites the variables from CarMaker ego so that it will move equally to that of the real life ego. HiL is important due its real time capabilities. Note that with HiL, tests can be made with the exact same sensors in laboratory as if they were in real life. Also, with CarMaker it is possible to add the sensor in the exact location and rotation of the real life sensor, as by calibration, and CarMaker can add the disturbances of weather and model of lens disturbances.

FIGURE 23 – HIL USECASE



SOURCE: Author(2022)

HiL only makes sense when used in real time, but to keep the real time it is not used a personal computer, it is used a real time computer from National Instruments in this case the NI - PXIe-1085. This has a support with CarMaker that allows for the real time computer to calculate the positions in real time and then output the data to the sensor stimulators which will stimulate the sensors and the sensors data will be recorded by the DRIVEPX. FIGURE 23 shows how the HiL can be used in a full stack development of a autonomous vehicle. It is interesting to notice that should the stimulator be changed for a installation in a real car the sensor setup from a hardware and software perspective would be exactly the same, the difference is that the sensor would be stimulated by real world data and not a simulated data. The last box from

FIGURE 23 is not implemented in this work, but is an integral part for a full autonomous vehicle setup drive.

The stimulators are two important parts for this digital twin to properly work. For the camera it was utilized the IPG Video Interface Box (VIB), the VIB is a physical piece of hardware that receives data from simulated camera and serializes and injects it directly into to the data stream. This is a direct approach to simulation of cameras, in which outputted data is in the same protocol as the camera, effectively being a camera from hardware purposes. The other option which was used was OTA transmission, where a camera was installed in front of a seven inch monitor then the data of such camera was recorded. This is a much cheaper option and development status can be seen in Reway et al. (2018). In this work both options were recorded, but for the scope of this work only the OTA camera will be discussed. To stimulate the RADAR the Vehicle Radar Test System (VRTS) was used, which is developed and maintained by Konrad GMBH. The VRTS is a system in which the RADAR is positioned in stimulator front and stimulated over the air. Both the RADAR and the stimulator are inside a box which has radio absorbents to decrease the noise added to the system. When the RADAR emits the electromagnetic wave, the stimulator receives this wave, and given the data receive from the HiL, it manipulates the received wave to match that of one it would be reflected if the target was at the distance, velocity, acceleration, and RCS that HiL has provided.

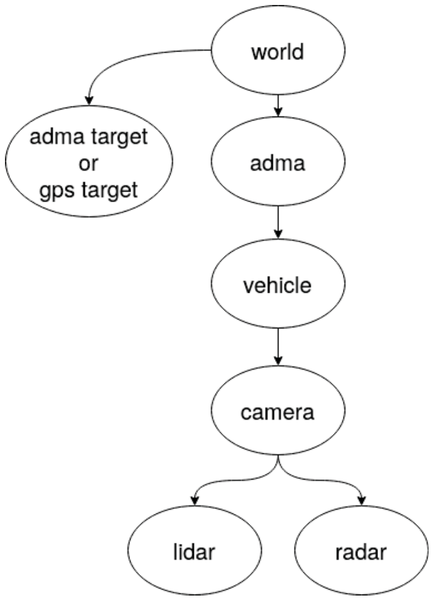
4 RESULTS

With the real world tests executed and the data collected, in this chapter the performance, evaluation, validation of the fusion, results and measure of the difference between real and simulated tests is discussed. Firstly, it is discussed the calibration results and the digital twin on Section 4.1. Then, it is described the High-Level Fusion results in Section 4.2. Finally, the Low-Level fusion results on Section 4.3.

4.1 CALIBRATION AND DIGITAL TWIN

Given that each sensor and the sensor mount has a different position and rotation, each actor in the vehicle has a different coordinated frame. As the sensors are in the vehicle a parent son relationship can be made. FIGURE 24 shows the relationship that was created in this work and that is discussed in this section.

FIGURE 24 – TRANSFORMATION FRAMES STRUCTURE



SOURCE: Author(2022)

The relationship of each transformation frame can be seen in FIGURE 24. For simplicity, the frames were named as the sensor they reference to, in other words, the ADMA frame is the reference frame of the sensor ADMA. Firstly the world frame is created, given this frame a frame ADMA is created. This frame represents the car inertial movement and therefore is not static. The subsequent frames relationships later frames will move together statically in relation with ADMA around the frame world.

Given a static transformation frame at zero known as world, which projection coordinates are known, a son transformation frame ADMA is created. ADMA, mentioned in Subsection 3.6.1, is the car IMU, which provides the car location and rotation relative the world transformation frame and inertial absolute measurements. With these, and the previously known installation of the sensor on the car, the ADMA transformation frame son relationship with the vehicle transformation frame is created. The vehicle transformation frame is static in relationship to ADMA transformation frame, and sits at the middle of the back axle.

As mentioned in Subsection 3.1.4, the camera extrinsic calibration was made using (NVIDIA CORPORATION, 2022), which provided with the camera location relative to the car back axle. The result of extrinsic calibration is shown in table TABLE 4, which shows the rotation in XYZ axis and its translation. With this the camera transformation frame relative to the vehicle can be created, where vehicle is camera's parent.

TABLE 4 – EXTRINSIC CAMERA LOCATION

Roll	Pitch	Yaw	X	Y	Z
0.4669	3.4517	2.8506	1.3199	-0.0716	1.2784

SOURCE: Author (2022)

The multisensor calibration algorithm used, mentioned in Subsection 3.1.4, calculated the transformation frames of the RADAR and LiDAR with referenced to the camera, but unaware of its location. The results of such calibration are shown in TABLE 5.

TABLE 5 – EXTRINSIC SENSOR LOCATION WITH RELATION TO CAMERA FRAME OF REFERENCE

	qx	qy	qz	X	Y	Z
Camera	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
LiDAR	-0.05101	-0.03989	-0.00719	-0.12005	0.13016	0.33903
RADAR	-0.05908	0.01085	0.01288	1.50490	0.03421	-0.68115

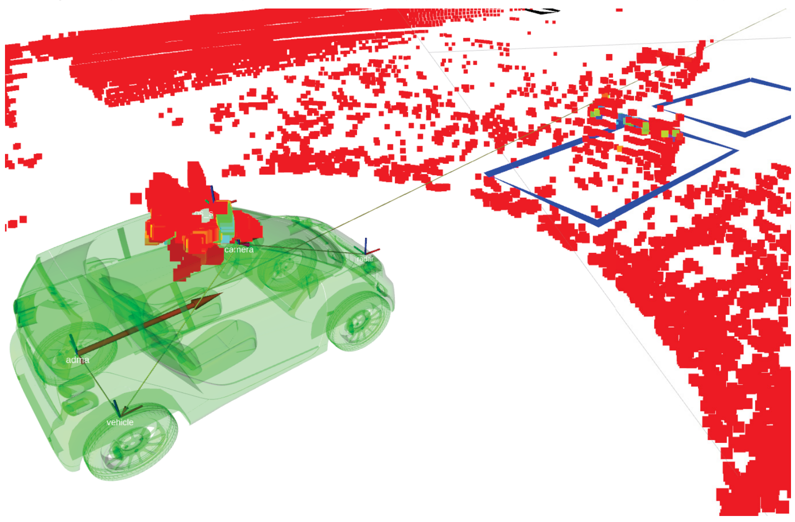
SOURCE: Author (2022)

With the RADAR and LiDAR transformation frames, sons of the camera transformation frame, it is known the relationships between all the transformation frames of the ego vehicle.

The target also has one transformation frame, which its parent is the world transformation frame. The target transformation frame is created on information of the target ADMA of the target GPS antenna data.

As it is known the relationship between camera and vehicle transformation frames, it is possible, using the sensors transformation frames overlap sensor outputs on a tridimensional frame.

FIGURE 25 – OVERLAP OF SENSOR OUTPUT ON TRIDIMENSIONAL FRAME



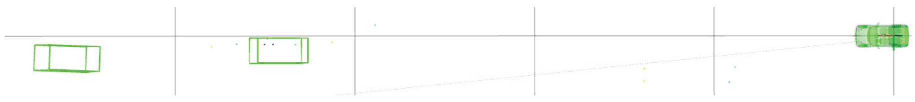
SOURCE: Author(2022)
NOTE: RADAR objects are the blue boxes.

FIGURE 25 shows the RVIZ visualization on the tridimensional grid. In this figure shows the multiple transformation frames and its relationships. The red arrow that point towards the car front are the ADMA frame movement in relation to the world frame. Red points are LiDAR output and boxes are RADAR output. As this is the CCRb AEB test, the dummy is seen in front of the car, represented as a point cloud. LiDAR shows the outline of the dummy, while RADAR shows the object that represent the dummy. Whilst it is possible to project camera to this plane, it was not done, as it is not part of this project scope.

4.1.1 Camera perception

As mentioned on Subsection 3.3.1, the camera perception was made using the SMOKE algorithm. It was observed 20 detection frames per second, where the camera produces 30 frames per second. SMOKE has been observed with the recorded data and it was found similar behavior to that found in the KITTI dataset, where it has a visual good location of the target 2D and orientation, but has difficulty measuring the distance.

FIGURE 26 – SMOKE PREDICTION



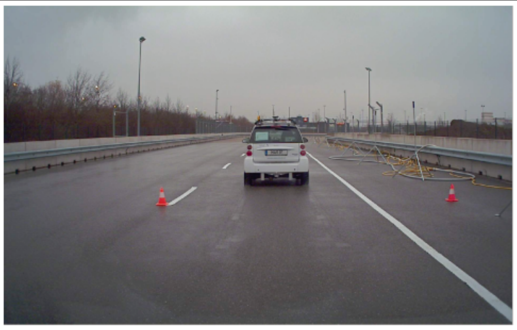
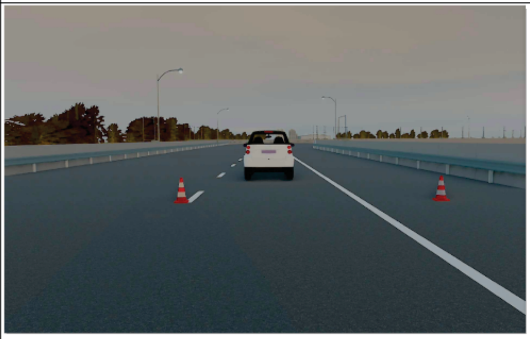
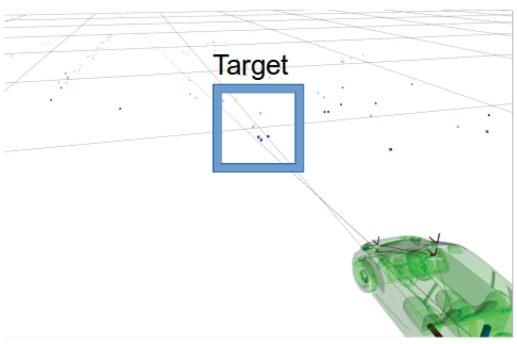
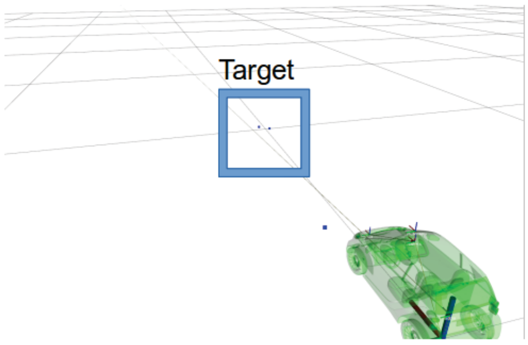
SOURCE: Author (2022)
NOTE: Each grid division is at 5 m apart

FIGURE 26 shows the ego vehicle on the right pointing to the left. The green boxes on the image are SMOKE detection, and points are from RADAR. The furthermost green box to the left is a second SMOKE detection of the same object, a false detection.

4.1.2 Simulation using HIL

In the outdoor, the tests were executed three times each, due to the amount of tests, time, and resources required to do each of them. With the outdoor data obtained, the movement of the ego vehicle and target was observed. Based on this data, simulation profiles were created. This made possible to create the simulation in which the virtual tests were executed.

TABLE 6 – HIGH-LEVEL FUSION INPUT

	Real Test	Simulated Data
Camera Data		
RADAR Data		

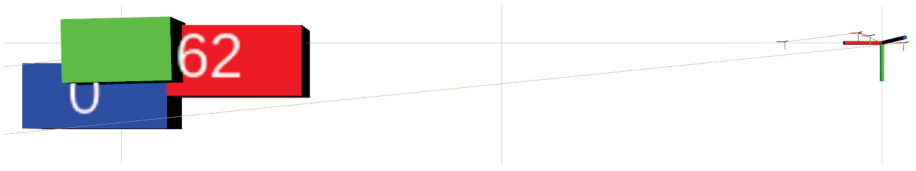
SOURCE: Author (2022)

TABLE 6 shows the comparison between real life test scenarios (first column) to the virtual digital twin (second column), of camera (first row) and RADAR (second row) data. As it is not in the scope of this work, no further analysis on the simulation-to-reality gap of sensors analysis is made at this work. It is important to note that in the simulated RADAR there is only one detection and its shadow. There is only one detection because the utilized stimulator can only stimulate one point. The shadow is inherent to this specific design and its origins were not investigated in this work. The date and time where the original scenario were executed at are important to the camera, then this simulation was executed as if it was being executed in the same time as the real counterpart to maintain the lighting from the sun position. And clouds were added to be a visual match. Notice that the exact amount of clouds were not measured in the outdoor, therefore this was the best effort.

4.2 HIGH-LEVEL FUSION EVALUATION

With calibration data and perception it is possible to fuse the object lists using High-Level Fusion, mentioned in Subsection 3.3.1. To make use of the proposed algorithm, it was utilized python as a programming language in the ROS workspace. The input can be seen in FIGURE 27 and the output in FIGURE 28. Both show the ego vehicle on the right, indicated as the axes XYZ in red, blue, and green. The grid size is of 10 m in any direction.

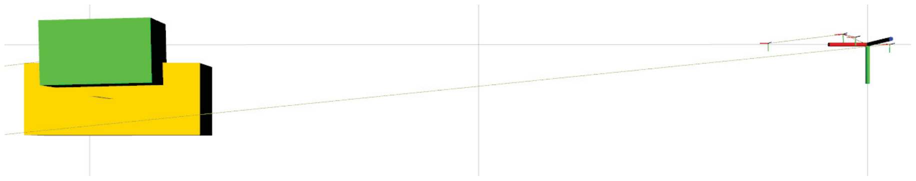
FIGURE 27 – HIGH-LEVEL FUSION INPUT



SOURCE: Author(2022)
NOTE: Each grid division is at 5 m apart

FIGURE 27 shows two boxes, one red and another blue, representing the fusion inputs. The red one is the SMOKE/Camera perception output, which has the identification number of 62. The blue one is the RADAR detection which has identification number of 0, after tracking and RADAR perception. It should be noted that the boxes have different sizes, location, and covariances. The green box is the ground truth for this frame in the tridimensional plane. It is noteworthy that both RADAR and Camera detections are not entirely over the ground truth.

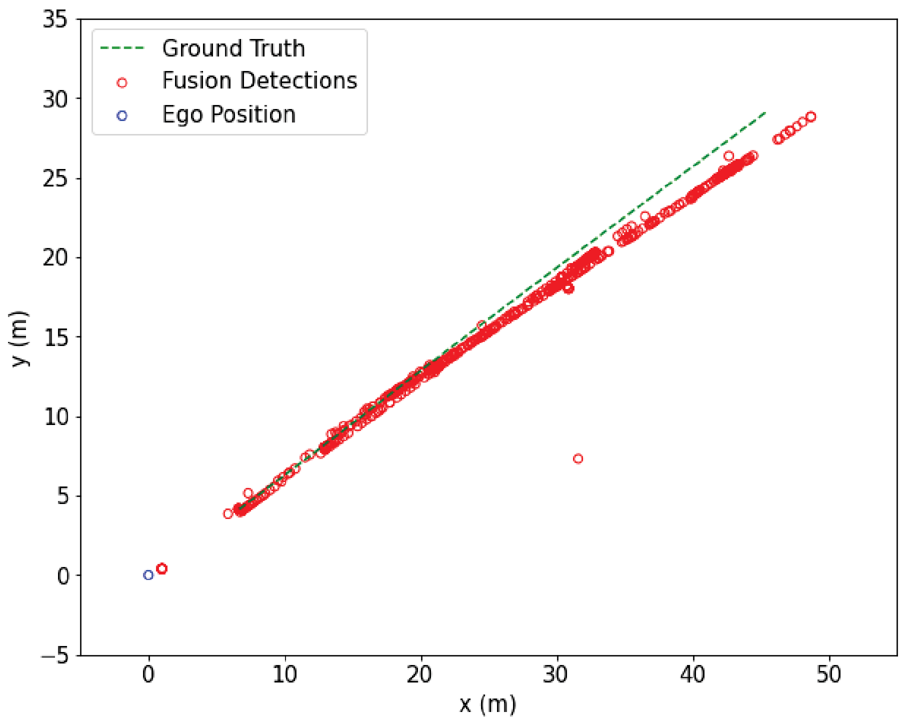
FIGURE 28 – HIGH-LEVEL FUSION OUTPUT



SOURCE: Author (2022)
NOTE: Each grid division is at 5 m apart

FIGURE 28 shows only one box yellow which fuses RADAR and camera boxes. This yellow box is the fusion output. It is important to notice that the algorithm was able to fuse data from different sensors with different identification numbers into a third different identification number, which fuses the states, covariances, classification, and existence of each object into a single object. The green box is the ground truth, showing that the fusion output is not entirely over the ground truth in this specific frame.

FIGURE 29 – HIGH-LEVEL FUSION OUTPUT FOR TESTRUN 1 STATIC EGO, DAYTIME, TARGET CAR



SOURCE: Author(2022)
NOTE: X and Y axis reference are the Ego vehicle position

FIGURE 29 shows the low-level fusion predictions in red and the ground truth in green. Note that there is a slight deviation to the right of the ego vehicle. That is due to errors in lower algorithms compounded. The RADAR object detection adds error, when compared to cluster list detection, and the camera detector also adds errors. This errors are added by the fusion. Also, it is important to acknowledge that a 1 degree error in 50 m accounts for a 0.8726 m lateral displacement at 50 m. It is noteworthy that this algorithm has produced 13 different objects for this specific scenario, which demonstrates that even though the path seems consistent, the algorithm has determined the destruction and construction of multiple objects instead of the only one there was in this specific scene.

TABLE 7 – HIGH-LEVEL FUSION PERFORMANCE ON STATIC EGO SCENARIO, TARGET MOVING AWAY FROM THE EGO, DAYTIME

	REAL SCENARIO	SIMULATED SCENARIO	
Testrun	mean Average Precision	mean Average Precision	$\mathcal{G}$
1	32.69%	4.71%	27.98%
2	29.71%	10.78%	18.93%
3	25.53%	0.48%	25.05%
mean	29.31%	5.32%	23.99%

SOURCE: Author (2022)

TABLE 7 shows the results of the High-Level Fusion on the scenario daytime with the vehicle ego static and the target a car, in which the real scenario achieve a mean of 29.31%, whilst the simulated scenario has achieved 5.32%, computing to a 23.99% of simulation-to-reality gap. This significantly lower outcome of the High-Level Fusion on simulated scenarios shows that the compounded errors on simulation are too significant for this simulation using this algorithm to be a valid counterpart to that of making real tests, even on a scenario which does not involve weather disturbances.

TABLE 8 – HIGH-LEVEL FUSION PERFORMANCE ON STATIC EGO SCENARIO, TARGET MOVING AWAY FROM THE EGO, DAYTIME RAIN

	REAL SCENARIO	SIMULATED SCENARIO	
Testrun	mean Average Precision	mean Average Precision	$\mathcal{G}$
1	33.31%	5.72%	27.59%
2	40.80%	4.97%	35.83%
3	42.53%	1.95%	40.58%
mean	38.88%	4.21%	34.67%

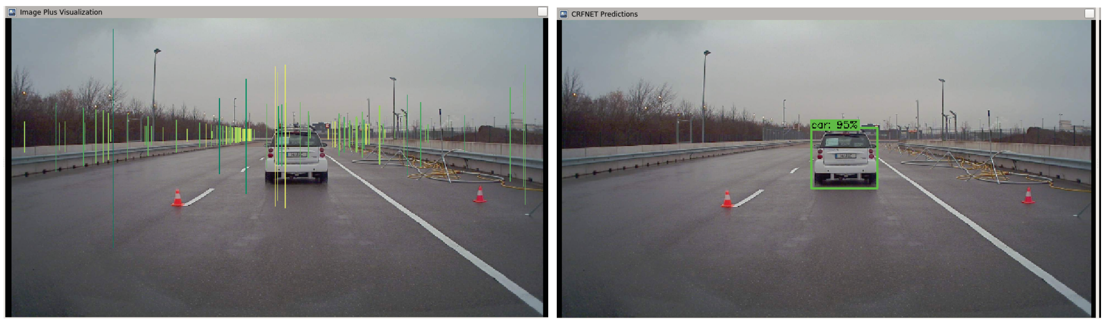
SOURCE: Author (2022)

Similarly, in TABLE 8, it was evaluated the results of the rain scenario of this fusion method, in which the mean average precision for the real scenario was 38.88% and the simulated only 4.21% computing an simulation-to-reality gap of 34.67%. The weather disturbances demonstrated a not as significant in the simulation, but still is low. This comes to show that for this algorithm, the presented simulation is not close enough of reality to make a switch between the real to virtual at this stage in the tested weather conditions.

4.3 LOW-LEVEL FUSION EVALUATION

This section will show the fusion results of a specific real and simulated test runs. It will be shown how the Low-Level Fusion can be seen on RVIz, followed by the presentation of the results using the mAP metric.

FIGURE 30 – LOW-LEVEL FUSION INPUT AND OUTPUT



(a) Low-Level Fusion Input (Image Plus) (b) Low-Level Fusion Output

SOURCE:Author(2022)

FIGURE 30a shows the Low-Level Fusion input in ROS RViz and FIGURE 30b depicts the Low-Level Fusion output. In the output image, it is possible to observe that the Low-Level Fusion predicted that the object is 95% certain to be of the class car. And on the Image Plus it can be seen the projection of the RADAR clusters on to the image, with color coded using the colormap summer from openCV, mapped by RCS.

The testrun that will be evaluated is the static ego, with a car moving away from the ego, in daytime, as shown in FIGURE 19b. The results are condensed in TABLE 9.

TABLE 9 shows that for this fusion method, in real scenarios, it was observed a mAP mean between test runs of 62.01% whilst in simulated scenarios the mean was of 67.30%. Using EQUATION 3.5, the simulation-to-reality gap is determined to be -5.18%. This demonstrates that the performed simulation in this work, with clear weather conditions, are close to the reality outcome, for this specific algorithm. The first testrun outperformed the real test by -17.89%, but testruns 2 and 3, showed closer

TABLE 9 – LOW-LEVEL FUSION PERFORMANCE ON STATIC EGO SCENARIO, TARGET MOVING AWAY FROM THE EGO, DAYTIME

	REAL SCENARIO	SIMULATED SCENARIO	
Testrun	mean Average Precision	mean Average Precision	$\mathcal{G}$
1	62.37%	80.36%	-17.89%
2	62.85%	67.02%	-4.17%
3	60.83%	54.54%	6.29%
mean	62.01%	67.30%	-5.26%

SOURCE: Author (2022)

simulation to reality. As this is a artificial intelligence based algorithm, it is harder to pinpoint exactly why testrun 1 was more distant than the others.

TABLE 10 – LOW-LEVEL FUSION PERFORMANCE ON STATIC EGO SCENARIO, TARGET MOVING AWAY FROM THE EGO, DAYTIME RAIN

	REAL SCENARIO	SIMULATED SCENARIO	
Testrun	mean Average Precision	mean Average Precision	$\mathcal{G}$
1	25.95%	14.91%	11.04%
2	49.11%	29.30%	19.81%
3	58.45%	27.51%	30.94%
mean	44.50%	23.90%	20.59%

SOURCE: Author (2022)

In the same scenario, but in rain conditions is described in TABLE 10, where it was observed a mean average precision on real scenario of 44.50%, and on its digital twin counterpart of 23.90%, resulting in a mean simulation-to-reality gap of 20.59%. This demonstrates that for this scenario on rain, using this algorithm, the rain is not well comprehended by this test method, and could not be as well simulated as the daytime scenario, demonstrating that there has to be a greater effort on to the rain scenario for it to be closer to reality than it is currently. As the simulated version of this test run has under-performed comparatively to its real counterpart, it is a great indicative that the rain behaviour was not fully copied. The rain amount in the real scenario was known, but the size of the water droplets could not be copied to the simulation, causing a effect in which the simulated raindrops are bigger than those of its counterpart.

5 CONCLUSION

In this work, camera and RADAR sensors were installed in a vehicle and then this sensors had its location, regarding the vehicle coordinates, measured. The camera location was measured using a method developed by NVIDIA Corporation (2022), in the car reference frame. With this measurement, the measurement of the RADAR location was possible. Camera was used as a reference sensor to the multisensor calibration algorithm, developed by Domhof et al. (2019). With both RADAR and camera extrinsic measurements, data could be overlapped in a tridimensional plane, and data from the tridimensional plane could be projected onto a bidimensional plane, such as the camera image. Tests were made on a controlled environment, in which all actors position was measured, in order to the creation of a digital twin of the environment, ego vehicle, and target vehicle. While the test execution, data from the sensor was collected using the DRIVEPX, which has inputs for all proposed sensors. For data acquisition, multiple ROS nodes were used, which stored data in multiple folders. With the digital twins, using the same hardware, and with simulation, data from the simulated scenario was recorded. These data was fused using high and low fusion methods, and compared in this work. With these data it was measured the simulation-to-reality gap for both fusion approaches.

The simulation-to-reality gap was measured, using high and low fusion methods. The simulation-to-reality gap was -5.29% for Low-Level Fusion using CRFnet in daytime in the evaluated testrun of static ego with target car. And for High-Level Fusion, the simulation-to-reality gap was 23.99%, in the same scenario. For the rain version of this scenario, Low-Level Fusion achieved a 20.59% simulation-to-reality gap and High-Level Fusion achieved 34.67% of this metric. It is clear that the adverse conditions are not fully well comprehended at the current simulation level, an increase of the simulation-to-reality gap was seen in both algorithms. Which shows a clear need for a better comprehension on adverse weather conditions. This does not indicates that the simulation is not good, but from the perspective of the algorithm, it is clear that the simulation is not the same as the real scenario. Main reasons include, the RADAR stimulator having only one point, for camera the rain boxes must be improved.

It was also observed that the Low-Level Fusion is able to detect an object in the scene. This method achieves a mAP of 93.72% of object detection, without object class acknowledgment. But when classes are included in the measurement its mAP is 62.37%. This shows that this algorithm is able to detect an object, but its efforts on classification, are not ideal at this stage.

In RADAR, there is an algorithm that clusterizes electromagnetic waves and creates a bounding box, which includes error due to its approximation using the shape of the clustered received waves. The camera detector, SMOKE, also includes errors of visual position. This is backed by Aeberhard (2017) which stated that a Lower level fusion is more accurate than a Higher Level fusion, due to the less uncertainties added in the algorithm.

The High-Level Fusion method has seen a decrease of mAP, due to the added uncertainties of the additional algorithm added. But the High-Level Fusion has its advantages, it is a well defined method where all boxes are debuggable, while the Low-Level Fusion is not fully debuggable. At this stage, improvements can be made to the High-Level Fusion where for the Low-Level Fusion it is more difficult to make improvements without creating to many generalizations. It was observed indeed that the Low-Level Fusion outperformed the High-Level Fusion for this test environment, with this HiL, sensor stimulator pairs, and simulation software. It is highly noteworthy that the simulation has under-performed, under this work conditions, to those seen in reality, which is a counter-intuitive conclusion. Usually simulations are better than reality, but that is utilizing software-in-the-loop, with simulations, as the ones here presented, using HiL is an attempt to close the simulation-to-reality gap, towards a more thoroughly tested driving function and autonomous vehicles level 5. But it is important to note that with the setup here presented, scenarios similar to those of daytime with the algorithm of Low Level fusion can be tested in a simulation environment and have close to reality outputs.

During the execution of this work, the comparison in real and virtual test environments has shown that objects with a radar cross-section smaller than 10 dBsm and larger than 1 dBsm (vehicles, pedestrians, bicycles) can be realistically simulated in the laboratory using the test setup. Due to the large reflection power of larger objects, unwanted reflection points occur inside the VRTS chamber.

This work discussed on how to make tests and the importance of calibration, it also has make substantial progress to measure the simulation-to-reality gap in two different algorithms, with digital twins of multiple scenes. The presented dissertation described tests with multiple scenarios, in multiple weather conditions. All these tests were made and a stable and reproducible methodology was determined. The data that was the input for all this work will be made public, as well as, all the code that was used in github. Even though in this work all possible tests were not evaluated, the methodology of this work allows for anyone with interest in this to rework the data, and test this and any other algorithms. As technology evolves, it is expected that the simulation-to-reality gap will be even smaller, with simulations even closer to reality, than those that were here discussed.

Future works spawned from this work are, for example, an approach to increase the accuracy of the High-Level Fusion method can be done, with the inclusion of better camera detection algorithms, and modifying the type of Kalman filter used, may provide better results. With the testruns executed in this work, it is possible to evaluate the efficacy of using the DDI and OTA camera stimulation and compare its results, as well as, can be used a DDI RADAR stimulator, when they become available, and compare the OTA to the DDI approach. The simulator could be changed to an even more realistic camera simulation, which may increase camera detection and accuracy further. Also, although CRFnet makes detections accurately, its classification is not as good as it is its detection, therefore in the future, it would be interesting to retrain CRFnet to increase its classification accuracy, also it would be in good interest to CRFnet to generate detections even without RADAR. Furthermore, it is important to measure when the simulation-to-reality gap becomes so great that simulations become unsuitable as real world replacements.

REFERENCES

AEBERHARD, M. P. **Object-level fusion for surround environment perception in automated driving applications**. [S.l.]: Technische Universität Dortmund, 2017. Cited 7 times on pages 18, 29, 30, 37, 38, 57.

AUTOPROMOTEC OBSERVATORY. **By 2030, 54% of all cars in Europe will be equipped with Adas systems**. [S.l.: s.n.], 2021. Access on: 22 jun 2022. Available at: https://www.autopromotec.com/en/autopromotec-adas-2021/a607. Cited 1 time on page 17.

BAI, J.; LI, S.; HUANG, L.; CHEN, H. **Robust Detection and Tracking Method for Moving Object Based on Radar and Camera Data Fusion**. v. 21. [S.l.: s.n.], 2021. P. 10761–10774. Cited 1 time on page 30.

BOCHINSKI, E.; EISELEIN, V.; SIKORA, T. **High-Speed tracking-by-detection without using image information**. [S.l.: s.n.], 2017. P. 1–6. Cited 1 times on pages 26, 27.

BRADSKI, G. **The OpenCV Library**. [S.l.: s.n.], 2000. Cited 1 time on page 27.

BUNDESMINISTERIUMS FÜR DIGITALES UND VERKEHR. **Bundesministeriums für Digitales und Verkehr 86/22**. [S.l.: s.n.], 2022. Available at: https://www.bundesrat.de/SharedDocs/drucksachen/2022/0001-0100/86-22.pdf?__blob=publicationFile&v=1. Cited 1 time on page 19.

CARHS TRAINING GMBH. **AEB CBL Car to bicycle Longitudinal (Euro NCAP)**. [S.l.: s.n.], 2020. Access on: 25 jul 2021. Available at: bit.ly/3qN1Nvx. Cited 1 time on page 41.

_______. **AEB CBL Car to Stationary Longitudinal (Euro NCAP)**. [S.l.: s.n.], 2020. Access on: 25 jul 2021. Available at: https://bit.ly/3iLzcCA. Cited 1 time on page 41.

_______. **AEB CCRs Car to Car Stationary (Euro NCAP)**. [S.l.: s.n.], 2020. Access on: 25 jul 2021. Available at: bit.ly/3iLzcCA. Cited 1 time on page 42.

CARSCOOPS. **This Is What Tesla's Autopilot Sees On The Road**. [S.l.: s.n.], 2020. Access on 27 mar 2022. Available at: https://www.youtube.com/watch?v=fKXztwtXaGo. Cited 1 time on page 21.

CARTUCHO, J.; VENTURA, R.; VELOSO, M. **Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots**. [S.l.: s.n.], 2018. P. 2336–2341. Cited 1 time on page 39.

CHEN, Y.-N.; DAI, H.; DING, Y. **Pseudo-Stereo for Monocular 3D Object Detection in Autonomous Driving**. [S.l.: s.n.], 2022. Cited 1 time on page 26.

CHOLLET, F. et al. **Keras**. [S.l.: s.n.], 2015. Access on: 25 jul 2022. Available at: https://keras.io. Cited 1 time on page 37.

CONTINENTAL ENGINEERING SERVICES. **ARS 408 Continental Engineering Services**. [S.l.: s.n.], 2022. Available at: https://conti-engineering.com/components/ars-408/. Acesso em: 27 mar. 2022. Cited 2 times on pages 25, 32.

DOMHOF, J.; KOOIJ, J. F.; GAVRILA, D. M. **An Extrinsic Calibration Tool for Radar, Camera and Lidar**. [S.l.: s.n.], 2019. P. 8107–8113. Cited 4 times on pages 27, 28, 33, 34, 56.

DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. **CARLA: An Open Urban Driving Simulator**. [S.l.: s.n.], 2017. P. 1–16. arXiv: 1711.03938. Available at: http://arxiv.org/abs/1711.03938. Cited 1 time on page 30.

EURO NCAP. **Euro NCAP | How to read the stars**. [S.l.: s.n.], 2022. Access on 27 mar 2022. Available at: https://www.euroncap.com/en/about-euro-ncap/. Cited 1 time on page 40.

GALVANI, M. **History and future of driver assistance**. v. 22. [S.l.: s.n.], 2019. P. 11–16. Cited 1 time on page 16.

GEIGER, A.; LENZ, P.; URTASUN, R. **Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite**. [S.l.: s.n.], 2012. Cited 1 time on page 25.

_______. **The KITTI Vision Benchmark Suite**. [S.l.: s.n.], 2019. Access on: 25 jul 2022. Available at: https://bit.ly/3PBpykv. Cited 0 times on pages 25, 26.

GENESYS ELEKTRONIK GMBH. **ADMA-G: Pro+/Eco+/Eco for Automotive/Railway**. [S.l.: s.n.], 2022. Access on 27 mar 2022. Available at: https://genesys-offenburg.de/en/adma-g/. Cited 1 time on page 43.

HEIKKILA, J.; SILVEN, O. **A four-step camera calibration procedure with implicit image correction**. [S.l.: s.n.], 1997. P. 1106–1112. Cited 1 time on page 23.

INTEGRATED MAXIM. **Gigabit Multimedia Serial Link (GMSL)**. [S.l.: s.n.], 2022. Access on: 27 mar 2022. Available at: https://www.maximintegrated.com/en/products/interface/high-speed-signaling/gmsl-serdes.html. Cited 1 time on page 31.

IPG AUTOMOTIVE. **CarMaker**. Access on: 25 jun 2021. 2021. Available at: https://ipg-automotive.com/. Cited 2 times on pages 30, 45.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Intelligent transport systems — Lane change decision aid systems (LCDAS) — Performance requirements and test procedures**. v. 2008. Geneva, CH, mai. 2008. Cited 1 time on page 36.

________. **Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary**. v. 2011. Geneva, CH, 2011. Cited 1 time on page 33.

JOBY AVIATION INC., I. **Radar Cross Section**. [S.l.: s.n.], 2021. Access on: 27 mar 2022. Available at: https://inras.at/en/radarlog/. Cited 1 time on page 24.

LIM, T. Y.; MARKOWITZ, S.; DO, M. N. **RaDICaL: A Synchronized FMCW Radar, Depth, IMU and RGB Camera Data Dataset with Low-Level FMCW Radar Signals**. v. 15. [S.l.: s.n.], 2021. P. 941–953. Cited 1 time on page 30.

LIU, Z.; WU, Z.; TÓTH, R. **SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation**. [S.l.: s.n.], 2020. Cited 2 times on pages 25, 38.

MURON, W. et al. **Multi Object Tracker**. [S.l.: s.n.], 2021. Access on 27 mar 2022. Available at: https://github.com/wmuron/motpy. Cited 2 times on pages 26, 38.

NABATI, R.; QI, H. **CenterFusion: Center-based Radar and Camera Fusion for 3D Object Detection**. [S.l.]: IEEE, 2021. P. 1526–1535. ISBN 978-1-6654-0477-8. arXiv:

2011.04841. Available at: https://ieeexplore.ieee.org/document/9423268/. Cited 1 time on page 30.

NOBIS, F.; GEISSLINGER, M.; WEBER, M.; BETZ, J.; LIENKAMP, M. **A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection**. [S.l.]: IEEE, 2019. P. 1–7. ISBN 978-1-7281-5085-7. arXiv: 2005.07431. Available at: https://ieeexplore.ieee.org/document/8916629/. Cited 1 time on page 29.

_______. **A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection**. [S.l.: s.n.], 2019. P. 1–7. Cited 1 time on page 37.

NVIDIA CORPORATION. **5.x_Linux_DPX_SDK**. [S.l.: s.n.], 2018. Access on: 09 mar 2021. Available at: https://docs.nvidia.com/drive/active/5.0.10.3L/nvvib_docs/index.html. Cited 1 time on page 34.

_______. **NVIDIA DriveWorks | NVIDIA Developer**. [S.l.: s.n.], 2022. Access on: 09 mar 2021. Available at: https://developer.nvidia.com/drive/driveworks. Cited 5 times on pages 28, 33, 34, 48, 56.

OUSTER, INC. **Datasheet rev06-v2p2-os1**. [S.l.: s.n.], 2021. https://data.ouster.io/downloads/datasheets/datasheet-rev06-v2p2-os1.pdf. Access on: 27 mar 2022. Cited 1 time on page 32.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; DESMAISON, A.; KOPF, A.; YANG, E.; DEVITO, Z.; RAISON, M.; TEJANI, A.; CHILAMKURTHY, S.; STEINER, B.; FANG, L.; BAI, J.; CHINTALA, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H.; LAROCHELLE, H.; BEYGELZIMER, A.; D'ALCHÉ-BUC, F.; FOX, E.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 32**. [S.l.]: Curran Associates, Inc., 2019. P. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Cited 1 time on page 38.

POLEDNA, Y.; REWAY, F.; DRECHSLER, F. M.; HUBER, W.; ICKING, C.; PARENTE, E. P. **An Open-Source High-Level Algorithm in ROS for Automated Driving Applications (to appear)**. [S.l.: s.n.], 2022. P. 1–8. Cited 1 time on page 37.

RAFIEINIA, F.; HAGHIGHI, K. **ASGARDI: A Novel Frequency-based Automotive Radar Target Simulator**. [S.l.: s.n.], 2020. P. 1–4. Cited 1 time on page 18.

REWAY, F.; HOFFMANN, A.; WACHTEL, D.; HUBER, W.; KNOLL, A.; RIBEIRO, E. **Test Method for Measuring the Simulation-to-Reality Gap of Camera-based Object Detection Algorithms for Autonomous Driving**. [S.l.: s.n.], 2022. P. 1249–1256. Cited 1 time on page 17.

REWAY, F.; HUBER, W.; RIBEIRO, E. P. **Test Methodology for Vision-Based ADAS Algorithms with an Automotive Camera-in-the-Loop**. [S.l.: s.n.], 2018. P. 1–7. Cited 2 times on pages 18, 46.

SAE INTERNATIONAL. **Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving System**. [S.l.: s.n.], 2014. Cited 1 time on page 16.

SEKONIX CO., LTD. CORPORATE HEADQUARTERS. **SEKONIX|Camera**. [S.l.: s.n.], 2022. Access on 27 mar 2022. Available at: http://sekolab.com/products/camera/. Cited 0 times on pages 22, 31.

SKOLNIK, M. I. **Introduction to Radar Systems**. 2. ed. Singapore: McGraw-Hill Book Co., 1980. ISBN 9780070665729. Cited 2 times on pages 23, 24.

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY. **Robotic Operating System**. [S.l.: s.n.], 23 mai. 2018. Available at: https://www.ros.org. Cited 3 times on pages 32, 34.

WOLFF, C. **Radar Cross Section**. [S.l.: s.n.], 2022. Access on: 27 mar 2022. Available at: https://www.radartutorial.eu/01.basics/. Cited 1 time on page 24.

YADAV, R.; VIERLING, A.; BERNS, K. **Radar+RGB Attentive Fusion for Robust Object Detection in Autonomous Vehicles**. [S.l.: s.n.], 2020. P. 3–7. arXiv: 2008.13642. Available at: http://arxiv.org/abs/2008.13642. Cited 1 time on page 30.

ZHANG, Z. **A flexible new technique for camera calibration**. v. 22. [S.l.: s.n.], 2000. P. 1330–1334. Cited 1 time on page 23.