h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

LiDAR Sensor Model Calibration and Validation

Master's Thesis conducted at
Valeo Schalter und Sensoren GmbH, for the degree of

Master of Science (M. Sc.)

at Darmstadt University of Applied Sciences
Department of Electrical Engineering and Information Technology

submitted by

Anoop Kengutte Nagaraj
Matr.Nr. 1112308

1st Academic Supervisor	:	Prof. Dr. Thomas Schumann
2nd Academic Supervisor	:	Prof. Dr. Markus Haid
Industrial Supervisor	:	Dr. Ivan Stepanov

Declaration

I hereby declare that this thesis is a presentation of my original research work and that no other sources were used other than what is cited.

I furthermore declare that wherever contributions of others are involved, this contribution is indicated, clearly acknowledged and due reference is given to the author and source.

I also certify that all content without reference or citation contained in this report is original work.

I acknowledge that any misappropriation of the previous declarations can be considered a case of academic fraud.

Darmstadt, May 26, 2025

Anoop Kengutte Nagaraj

Confidentiality

This Master's thesis report contains confidential data of Valeo Schalter und Sensoren GmbH. It may only be made available to university supervisors and authorized members of the board of examiners. Any publication or duplication of this report – even in part – is prohibited. Inspection of this work by third parties requires the express permission of the author and Valeo Schalter und Sensoren GmbH.

Abstract

As the automotive industry moves toward fully autonomous driving, achieving high accuracy and reliability in perception with respect to the surrounding of the system becomes critical. Modern autonomous vehicles rely on a combination of sensors including cameras, radar, and Light Detection and Ranging (LiDAR) to understand and interact with their surroundings. Among these, LiDAR plays a key role due to its ability to provide precise 3D information about the environment. However, to achieve accurate environmental understanding, simulated LiDAR data must closely match real-world sensor behavior. This presents a significant challenge due to the presence of various sensor artefacts which are often not accurately replicated in basic simulations. Effective calibration and validation techniques are therefore essential for bridging the gap between real and simulated LiDAR outputs.

This thesis focuses on improving the realism and fidelity of LiDAR sensor simulation through software-based calibration methods. A comprehensive literature survey outlines the fundamentals of LiDAR technology, its role in Advanced Driver-Assistance Systems (ADAS), and key signal features. The core contribution lies in the implementation of histogram-based evaluation metrics and artefact-specific calibration algorithms, supported by a Graphical User Interface (GUI) for efficient analysis and automation of workflows. The results show improved alignment between real and simulated point clouds, enhancing the utility of LiDAR simulation tools in automotive development. These findings contribute to the advancement of sensor simulation for high-level autonomous driving and support scalable, cost-effective validation strategies.

Acknowledgements

I would like to express my sincere gratitude to my academic supervisor, **Prof. Dr. Thomas Schumann**, for his constant support, valuable feedback, and expert guidance throughout the duration of this thesis. His mentorship played a critical role in the successful completion of this work.

I am also grateful to my second academic supervisor, **Prof. Dr. Markus Haid**, for his valuable input and for serving as a member of my evaluation committee. Furthermore, I would like to thank **Hochschule Darmstadt** for providing excellent academic resources and a stimulating environment that greatly contributed to my research.

I am profoundly grateful to my industrial supervisor, **Dr. Ivan Stepanov**, at **Valeo Schalter und Sensoren GmbH**, whose technical expertise, insightful guidance, and ongoing support were invaluable throughout the practical phases of this project.

I would also like to thank **Dr. Richard Rapp** for giving me the opportunity to work on this project and for his trust and support during my time at **Valeo**.

I extend my appreciation to my colleagues and the entire team at **Valeo** for providing a collaborative and motivating environment. Your support and feedback contributed significantly to this research.

Last but not least, I would like to thank my family and friends for their unwavering support, patience, and encouragement throughout my academic journey.

Anoop Kengutte Nagaraj
Hochschule Darmstadt

Contents

Acknowledgements v

List of Abbreviations viii

1 Introduction 1
- 1.1 Company Overview . . . 2
- 1.2 Motivation . . . 3
- 1.3 Problem Statement . . . 4
- 1.4 Objectives . . . 4
- 1.5 Thesis outline . . . 5

2 Literature Review 7
- 2.1 Advanced Driver Assistance System . . . 7
  - 2.1.1 Stages of Autonomous Driving . . . 8
- 2.2 LiDAR : Light Detection and Ranging . . . 9
  - 2.2.1 The Necessity of LiDAR in Autonomous Driving . . . 9
  - 2.2.2 Evolution of Automotive LiDAR . . . 10
  - 2.2.3 Different generations of LiDAR at Valeo Schalter und Sensoren GmbH . . . 11
- 2.3 LiDAR Range Equation . . . 14
- 2.4 Single-Photon Avalanche Diode (SPAD) Arrays in LiDAR Sensor Technology . . . 19
  - 2.4.1 Operating Principle . . . 19
  - 2.4.2 SPAD vs. APD . . . 19
  - 2.4.3 Time-Bin Histograms and Pile-Up Effects . . . 20
  - 2.4.4 Challenges from Dead-Time and Cross-Talk . . . 22
  - 2.4.5 Advantages over Traditional Sensors . . . 22
  - 2.4.6 Limitations and Mitigation Techniques . . . 23
- 2.5 Alternative Statistical Techniques for LiDAR Simulation and Calibration . . . 23
- 2.6 Software Requirements . . . 27

3 Concept of Solution 29
- 3.1 Sensor Model Use with CarMaker . . . 29
- 3.2 Interactive Calibration Workflow . . . 29
- 3.3 Simulation of LiDAR Artefacts . . . 31
- 3.4 Calibration Methodology . . . 32
- 3.5 Expected Benefits and Outcomes . . . 32
- 3.6 Importance of Analytical Features in Calibration . . . 32
  - 3.6.1 Peak in LiDAR Signal Processing . . . 33
  - 3.6.2 Width in LiDAR Signal Processing . . . 34

3.6.3 Area in LiDAR Signal Processing . . . 35
3.6.4 XYZ Coordinates in LiDAR Data . . . 36
3.6.5 LiDAR Scanning Mechanism and Point Cloud Formation . . . 36

4 Implementation **38**
4.1 Histogram-Based Metrics for LiDAR Simulation Evaluation . . . 40
4.1.1 Parallel Visualization of Real and Simulated LiDAR Data for Comparison . . . 43
4.1.2 Comparative Scenes for Scenario-Level Analysis . . . 46
4.2 Modeling Noise in LiDAR Data for Realistic Simulation . . . 49
4.2.1 Sources of Noise in LiDAR Data . . . 50
4.2.2 Simulating Noise in LiDAR Data: Need and Baseline Estimation 51
4.2.3 Data Collection and Preprocessing . . . 52
4.2.4 Noise Characterization: Area and Distance Histograms and Detection Probability . . . 56
4.2.5 Spatial Misalignment Between Real and Simulated LiDAR Data . 58
4.3 Blooming . . . 60
4.3.1 Effects of Blooming in LiDAR Applications . . . 63
4.3.2 Types of Blooming in LiDAR . . . 63
4.3.3 Vertical Blooming in Our LiDAR Sensor . . . 66
4.3.4 Implementation Approach . . . 66
4.3.5 GUI Enhancement for Multi-File Processing . . . 69
4.3.6 Calibration of Blooming Effect Using Lookup Tables (LookUp Table (LUT)s) . . . 70
4.4 Ghost Points . . . 76
4.4.1 Factors Causing Ghost Particles . . . 76
4.4.2 Effects of Ghost Particles in LiDAR Applications . . . 77
4.4.3 Implementation Approach . . . 77
4.4.4 Selecting Multiple Distances . . . 78
4.4.5 Preprocessing LiDAR Data: Isolating Traffic Sign and Ghost Particles . . . 79
4.4.6 Graphical User Interface for Ghost Point Analysis . . . 80
4.4.7 Future Development: Integration of Ghost Traffic Sign Simulation 82
4.4.8 Ghost Point Extraction Algorithm . . . 86
4.5 Automation in Calibration Workflow . . . 88
4.5.1 Co-Efficient Management and Automation for Blooming Calibration 89
4.5.2 Automating Header Processing for Efficient Data Handling . . . 91

5 Conclusion **94**
5.1 Outlook . . . 95

Bibliography **97**

List of Abbreviations

LiDAR Light Detection and Ranging

ADAS Advanced Driver-Assistance Systems

OEM Original Equipment Manufacturers

EV Electric Vehicles

Radar Radio Detection and Ranging

UGV Unmanned Ground Vehicle

UAV Unmanned Aerial Vehicle

ADCU Autonomous Driving Control Unit

ECU Electronic Control Unit

DARPA Defense Advanced Research Projects Agency

VLP Velodyne LiDAR Puck

ISO International Organization for Standardization

SOTIF Safety Of The Intended Functionality

FMCW Frequency-Modulated Continuous Wave

V2X Vehicle-to-Everything

AI Artificial Intelligence

SoC System on Chip

Q-Q Quantile-Quantile

KDE Kernel Density Estimation

CDF Cumulative Distribution Function

FOV Field of View

SPAD Single-Photon Avalanche Diode

APD Avalanche Photodiode

ToF Time-of-Flight

TCSPC Time-Correlated Single Photon Counting

CCD Charge-Coupled Device

CMOS Complementary Metal-Oxide-Semiconductor

CMOS Complementary Metal-Oxide-Semiconductor

fps Frames Per Second

FLIM Fluorescence Lifetime Imaging Microscopy

PET Positron Emission Tomography

TDC Time-to-Digital Converter

GUI Graphical User Interface

IPG International Performance Group

IDE Integrated Development Environment

PCD Point Cloud Data

KPI Key Performance Index

LUT LookUp Table

JSON JavaScript Object Notation

SLAM Simultaneous Localization and Mapping

ROI Region Of Interest

DBSCAN Density Based Spatial Clustering Algorithm

ACC Adaptive Cruise Control

LKA Lane Keeping Assistance

AEB Autonomous Emergency Braking

BMS Battery Management System

KPI Key Performance Indicators

List of Figures

2.1 Types of Sensors in an Autonomous Vehicle [22] 8
2.2 Velodyne's ULTRA Puck™ (Velodyne LiDAR Puck (VLP)-32C) [17] . . . 10
2.3 SCALA Gen-1 [29] 11
2.4 SCALA Gen-2 [29] 12
2.5 SCALA Gen-3 [29] 13
2.6 Performance comparison between SCALA Gen 1-2-3 [43] 14
2.7 Real scene and corresponding point clouds from SCALA Gen-1, Gen-2, and Gen-3 [29] 14
2.8 Power budget equation in LiDAR systems [18] 15
2.9 Illustration of factors influencing received power in LiDAR systems. [18] 17
2.10 Current-voltage relationship of a p-n diode. [41] 20
2.11 Time bin histogram 21
2.12 Pile up effect 21
2.13 Dead time 22

3.1 Sensor Model Use with CarMaker 29
3.2 Interactive Calibration Workflow 30
3.3 Conceptual flowchart 31

4.1 Calibration tool GUI 39
4.2 Real vs Simulated Point cloud data of a car 39
4.3 Dataset selection interface 41
4.4 Offset alignment 41
4.5 Optimal offset interface 42
4.6 Histogram comparison GUI 42
4.7 Saved histograms of four distance values 43
4.8 Side view of merged Point Cloud Data 44
4.9 Top view of merged Point Cloud Data 44
4.10 Point Cloud Data of a Real scene 46
4.11 CarMaker setup of the Real scene 47
4.12 CarMaker + Sensor model output of the Real scene 48
4.13 Simulated output of the Real scene 49
4.14 Laboratory Setup 52
4.15 Point Cloud Data of Lab 53
4.16 Point Cloud Data of Panels only 53
4.17 Point Cloud Data of Trimmed Panels 54
4.18 Point Cloud Data of Simulated Panels 55
4.19 GUI of Noise Characteristics panel 56
4.20 Histogram comparison of Area and Distance 57
4.21 Detection Probability of Real and Simulated points 58

4.22 Comparison of Real and Simulated panels at 50m, 55m, and 60m . . . 59
4.23 Traffic Sign [26] . . . 61
4.24 Point Cloud Data of the Traffic Sign . . . 61
4.25 Point Cloud Data of the Traffic Sign with Blooming . . . 62
4.26 Vertical and Horizontal Blooming . . . 64
4.27 Point Cloud of Real-World with Blooming effects . . . 67
4.28 Point Cloud of Traffic sign with Blooming effects . . . 69
4.29 Dataset of Traffic Sign for 3 Distances . . . 70
4.30 Look-Up-Table to calibrate Vertical Blooming . . . 70
4.31 Look-Up-Table to calibrate Horizontal Blooming . . . 71
4.32 Helper tool for Calibrating Parameters . . . 73
4.33 Histogram of Vertical Blooming at 157m . . . 75
4.34 Histogram of Horizontal Blooming at 20m . . . 75
4.35 Point cloud of Traffic sign and Ghost points with surroundings . . . 76
4.36 Snipped point cloud of retained Traffic sign and Ghost points . . . 80
4.37 Panel Ghost Calibration . . . 82
4.38 Area histogram of Actual and Ghost points across 4 datasets . . . 83
4.39 Peak histogram of Actual and Ghost points across 4 datasets . . . 84
4.40 Point count of Actual and Ghost objects across 4 datasets . . . 85
4.41 Simulated actual traffic sign . . . 85
4.42 Complete Scene with Traffic Sign and Ghost Points . . . 87
4.43 Isolated Traffic Sign and Ghost Points After Filtering . . . 88
4.44 JavaScript Object Notation (JSON) file of Vertical Blooming . . . 90
4.45 Reload and Update JSON buttons in GUI . . . 91
4.46 Header Type 1 . . . 92
4.47 Header Type 2 . . . 93
4.48 Code Snippet to process data based on Header format . . . 93

1 Introduction

The automotive industry is undergoing a rapid transformation with the advancement of autonomous driving technology. Modern vehicles are increasingly integrating intelligent systems that enhance safety, efficiency, and driving experience. Features such as Adaptive Cruise Control (ACC), Lane Keeping Assistance (LKA), and Autonomous Emergency Braking (AEB) have already become common. However, the ultimate goal is full autonomy, where vehicles can navigate complex environments without human intervention. This evolution is driven by advancements in sensor technology, Artificial Intelligence (AI), and vehicle-to-infrastructure communication [27].

A crucial aspect of autonomous driving is the perception system, which relies on multiple sensors to interpret the surrounding environment. Cameras provide high-resolution visual information and good object classification but sometimes make mistakes in distance estimation and struggle in low-light or adverse weather conditions. Radio Detection and Ranging (Radar) excel in detecting distance and velocity of objects and work through obstructions like fog and rain, but they lack precise shape and texture recognition. LiDAR bridges these gaps by providing high-resolution 3D spatial data, enabling accurate depth perception and object detection. By combining data from these sensors, vehicles achieve a more comprehensive understanding of their surroundings [42].

Traditionally, automotive systems relied on Cameras and Radars for ADAS. While these technologies significantly improved vehicle safety, they had limitations in fully autonomous applications. The need for precise, real-time environmental mapping has led to the adoption of LiDAR as a key sensor for self-driving technology. Modern autonomous vehicles, such as those operated by Waymo, Cruise, and Motional, rely heavily on LiDAR for accurate perception and navigation in complex urban environments. In the consumer vehicle segment, Valeo's SCALA 1 and SCALA 2 LiDARs are integrated into production cars like the Audi A8, Mercedes-Benz S-Class, and Honda Legend, enabling Level 3 autonomous driving features. These examples highlight the growing importance of LiDAR in both commercial autonomous fleets and ADAS in passenger vehicles.

For autonomous vehicles to operate safely and reliably, LiDAR based perception must be rigorously tested and optimized. This is where simulation plays a crucial role. By simulating real-world scenarios, edge cases, and artifacts, developers can evaluate and refine sensor performance even before the physical sensor is developed. LiDAR system simulation is useful during all stages of sensor development. At the beginning of sensor development, simulations can provide guidance on target Key Performance Indicators (KPI), like resolution and range. During sensor development, simulation can provide synthetic data for algorithm development when no real data is available yet.

After the sensor hardware is finalized, simulation can support algorithm validation by providing synthetic data for scenarios where no real data is available. This approach accelerates sensor development, reduces testing costs, and ensures robustness in diverse driving conditions. Moreover, by fine-tuning simulated LiDAR data to align with real-world behavior, developers can enhance the accuracy and effectiveness of perception algorithms, contributing to the advancement of fully Autonomous Driving Systems [14].

While sensor technologies like LiDAR play a pivotal role in autonomous driving, the accuracy and reliability of these sensors are heavily dependent on the underlying sensor models used in simulations. These models must be rigorously tested and validated to ensure they reflect real-world sensor behavior under various conditions. If the model fails to accurately replicate phenomena such as noise, blooming artifacts, or ghost points, the simulation results may not correspond to actual sensor performance, leading to faulty algorithm development. This is particularly crucial as even small discrepancies can significantly impact the behavior of autonomous systems, especially in edge cases. Therefore, the validation and calibration of the sensor model itself is essential for achieving robust and reliable simulations. This thesis focuses on creating a tool for calibrating and validating LiDAR sensor models, aiming to improve the alignment of simulated and real-world data and thereby ensure the reliability of autonomous vehicle systems.

1.1 Company Overview

Valeo Schalter und Sensoren GmbH, is a global automotive supplier headquartered in Paris, recognized for its innovative solutions and its commitment to the development of cutting-edge technologies in the automotive industry. Founded in 1923, Valeo has evolved into a key player in the automotive sector, particularly in the fields of electrification, autonomous driving, and mobility solutions. The company operates in 33 countries and employs around 113,000 people globally[4], serving a wide range of Original Equipment Manufacturers (OEM) and Tier-One suppliers[36].

Valeo is a global automotive supplier that focuses on delivering innovative technologies to enhance vehicle intelligence, energy efficiency, and safety. Following its recent organizational restructuring, Valeo now operates under three core divisions: Brain, Power, and Light, each reflecting the company's strategic focus areas in the evolving automotive industry:

1. **Brain Division:** This division encompasses all systems related to vehicle intelligence and automation. It includes Advanced ADAS, such as Radar, Cameras, and LiDAR sensors, which enable features like AEB, LKA, and autonomous parking[32]. A significant innovation under this division is Valeo's LiDAR technology. The Valeo SCALA, one of the most advanced LiDAR systems on the market, provides high-resolution 3D environmental mapping with a range of up to 200 meters, playing a crucial role in autonomous driving capabilities[35].

2. **Power Division:** This division focuses on the electrification and energy efficiency of vehicles. It includes electric and hybrid Powertrain Systems, offering components such as electric motors, inverters, and Battery Management System (BMS). These solutions support the global transition to electric mobility by improving performance and reducing emissions[34]. The Power division also integrates Thermal Systems for managing the thermal comfort of passengers and the thermal performance of electric drivetrains, ensuring optimal energy usage and reliability[34].

3. **Light Division:** Dedicated to visibility and vehicle signaling technologies, this division includes Valeo's expertise in Lighting and Wiper Systems. It offers advanced headlamp solutions, interior lighting, and dynamic signal lights that enhance safety and aesthetic appeal. These technologies are key contributors to both passive and active safety in various driving conditions[33].

4. **Sustainability and Innovation:** Across all three divisions, Valeo remains committed to sustainability by reducing CO_2 emissions and promoting eco-friendly innovation. The company advances material science, manufacturing processes, and system integration to support a more sustainable automotive future[31].

Valeo continues to innovate within the automotive industry, with a particular focus on autonomous driving technologies, electric mobility, and safety systems. The company's investment in LiDAR technology marks a significant step forward in the journey toward fully autonomous vehicles.

1.2 Motivation

In recent years, there's been remarkable progress in the field of autonomous driving, with various sensors, including Cameras, Radars, and LiDARs enabling vehicles to perceive and interpret their surroundings. While cameras provide rich visual detail and radars are effective in measuring speed and distance, both have limitations in challenging conditions such as poor lighting, glare, or cluttered environments. LiDAR, with its ability to generate accurate 3D maps of the environment regardless of lighting, has become essential for enhancing the reliability and precision of autonomous perception systems.

To fully leverage LiDAR's capabilities, it's crucial to rigorously test and calibrate these sensors ideally in a wide range of real-world scenarios. But testing in the real world is often costly, time-consuming, and difficult to scale especially when it comes to rare or complex driving situations. This is where simulation becomes an invaluable tool. By creating realistic virtual environments, developers can test and fine-tune their systems early in the process, before physical prototypes are even available.

However, one of the limitations of current simulation tools is that they often fail to capture artefacts that real sensors produce like blooming, ghost points, and sensor noise.

These differences can lead to inaccuracies when the same algorithms are later deployed on real vehicles.

Improving the accuracy of LiDAR simulation and calibration can help bridge this gap. With more reliable simulation tools, it becomes possible to test and improve perception algorithms in a safer, faster, and more cost-effective way. Ultimately, this helps push autonomous driving closer to real-world deployment, with fewer surprises and better performance across diverse scenarios.

1.3 Problem Statement

Despite its growing use in autonomous driving systems, current LiDAR simulation technologies often fail to accurately replicate the behavior of real-world sensors when modeling phenomena such as sensor noise, blooming effects, and ghost points. These gaps result in noticeable discrepancies between simulated and real LiDAR outputs, which can significantly impact the reliability of simulation-based validation processes for perception systems.

The challenge is further compounded by the fact that sensor development and perception algorithms development are often carried out simultaneously. This creates a strong dependency on synthetic LiDAR data during early development that may be inaccurate if the simulation environment fails to mimic key sensor artifacts observed in real-world data.

As a result, reliance on uncalibrated simulated data can lead to misleading test outcomes, delayed integration timelines, and reduced reliability in algorithm performance. Without methods to identify and correct the differences between simulated and real data, the effectiveness of simulation as a development and validation tool remains limited, particularly when dealing with edge cases or safety-critical scenarios.

1.4 Objectives

The primary objective of this thesis is to enhance the accuracy and realism of simulated LiDAR sensor data for autonomous driving applications. This involves analyzing and replicating key real-world sensor behaviors such as blooming effects, ghost points, and point distribution characteristics in the simulation environment. By aligning simulated data more closely with real LiDAR outputs, the goal is to support early-stage function development, sensor validation, and algorithm testing. The ultimate aim is to accelerate development cycles and improve the reliability of autonomous perception systems.

To achieve this goal, the following key objectives are defined:

1. **Development of a Calibration and Validation Tool for Simulated LiDAR Data**
   Develop a tool to visualise and compare simulated LiDAR outputs against real-

world measurements. The tool will focus on identifying discrepancies in point cloud characteristics and support the calibration of unknown parameters in the sensor simulation.

2. **Support for Sensor System Simulation Development**
   Provide meaningful feedback to the sensor simulation team by analyzing deviations in point cloud features such as distance, blooming, noise, and ghost points. The tool helps developers fine-tune the simulation by clearly showing where and how it differs from real-world sensor data, making it easier to improve accuracy.

3. **Validation of Simulated Noise Characteristics**
   Evaluate the simulated point cloud data for realistic noise behavior and compare it with measured LiDAR data. This helps validate whether the simulated environment reflects actual sensor imperfections.

4. **Quick Sanity Testing in Complex Scenes**
   Enable quick testing of the sensor model in complex scenarios like city streets, reflective surfaces, or layered objects to quickly spot major issues before a detailed evaluation.

5. **Streamlining the Calibration Workflow**
   Improve the usability and efficiency of the calibration process by introducing automation and support features that simplify parameter handling and streamline the overall calibration process, making it faster and user-friendly.

By achieving these objectives, this thesis aims to contribute to the advancement of LiDAR sensor simulation, making it a more efficient and reliable tool for autonomous vehicle development.

1.5 Thesis outline

The content of this thesis is divided into five chapters, as illustrated below, which also include this chapter and a conclusion in the end.

Chapter 2 provides a literature review of key topics related to LiDAR sensor simulation and calibration by introducing ADAS and the different levels of autonomous driving. The chapter then explains the basics of LiDAR technology, its importance in autonomous vehicles, and the evolution of LiDAR systems, including those developed at Valeo. It also covers the LiDAR range equation, SPAD sensor technology, and common challenges in LiDAR systems. Finally, it introduces some statistical techniques used for LiDAR data analysis and lists the software tools required for simulation and calibration.

Chapter 3 explains the overall concept behind the solution. It describes both open-loop and closed-loop approaches used in the calibration process. The chapter also covers how LiDAR artefacts are simulated, the calibration steps followed, and the expected outcomes. Finally, it highlights key analytical features such as peak, width, area, and

XYZ coordinates that are important in understanding and improving LiDAR signal processing.

Chapter 4 encompasses the intricate technical aspects of the implementation carried out, involving software-based simulation and calibration techniques. It offers in-depth insights into the evaluation of LiDAR data using histogram-based metrics, the simulation of artefacts such as noise, blooming, and ghost points, and the development of algorithms for their calibration. The chapter also details enhancements to the GUI and outlines automation strategies that streamline the calibration workflow.

Chapter 5 summarizes the key outcomes of the thesis and highlights the contributions of the proposed software-based calibration methods to the field of LiDAR sensor simulation. It also discusses limitations and suggests possible directions for future improvements and extensions of the current work.

Appendices and References provide supplementary definitions and explanations of technical terms used throughout the thesis. The references section includes citations to relevant academic literature, technical documentation, and other sources that support the research conducted.

2 Literature Review

2.1 Advanced Driver Assistance System

The term ”Advanced Driver Assistance System” (ADAS) refers to a group of technologies that help drivers operate cars safely. The human-machine interaction used by ADAS improves vehicle and road safety. ADAS employs automated technology, such as cameras, Radars, LiDARs and other sensors, to recognize surrounding obstacles or driving errors and take appropriate action. Different levels of autonomous driving are possible with ADAS[40].

Since human error causes the majority of traffic accidents, ADAS is designed to automate, adapt, and improve car technology for safety and better driving. ADAS has been shown to lower traffic deaths by reducing human error. By providing technologies that notify the driver to issues, making corrections, and taking control of the vehicle if necessary, safety features are intended to prevent collisions and crashes. Examples of adaptive features include automated lighting, adaptive cruise control, collision avoidance assistance, traffic alerts, warnings of potential obstacles, assistance with lane departure and lane centering, and other features are examples of adaptive features[23]. The concept of an autonomous vehicle has existed long before its commercial implementation. Since allowing automobiles to drive independently benefits humans, attempts to do so have been made for many years. The United States Department of Defense started sponsoring research on autonomous vehicles for military use in the 1970s, including the creation of Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV). The first attempts to create autonomous vehicles date back to the 1970s[2]. LiDAR technology played a crucial role in the success of the DARPA Grand Challenge, enabling vehicles to accurately perceive their surroundings and navigate complex environments without human input.

At present, almost all automobile manufacturers and suppliers are developing their own models, control units, or sensors for autonomous vehicles, for example, the Autonomous Driving Control Unit (ADCU), LiDAR, and Radar sensors for autonomous driving. Modern automobiles are no longer basic mechanical machines. They have instead evolved into complicated systems with several electronics and sensors. It is a mobile computer. There are numerous Electronic Control Unit (ECU)s included in the system, each of which accomplishes a distinct role.

The ADCU, which controls all the unique functions associated with autonomous driving, serves as the master ECU in these. This includes using sophisticated computer vision algorithms to gather data from the LiDAR, camera, and Radar and reach a conclusion. The Radar system is used to measure the vehicle’s approach to any object in

the vehicle's surroundings, which increases driving safety and the LiDAR sensor scans the vehicle's surroundings with a laser.

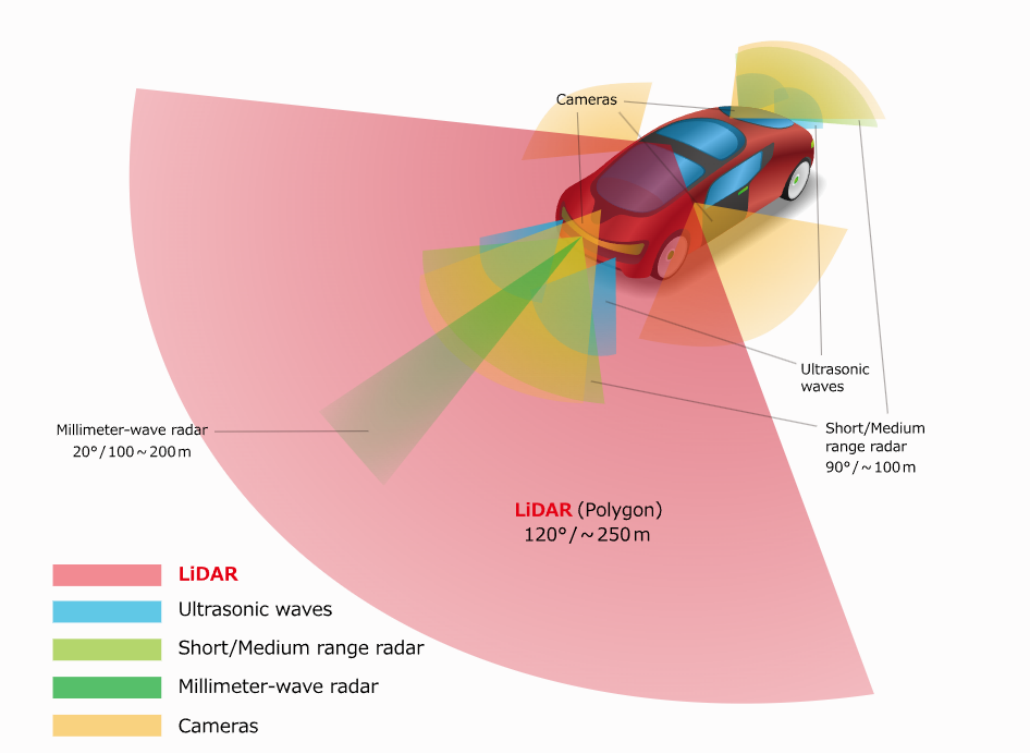

Figure 2.1: Types of Sensors in an Autonomous Vehicle [22]

2.1.1 Stages of Autonomous Driving

Level 0: No Automation - At this level, the driver is the only one who can monitor and operate the car. No automated systems exist to help with driving chores.

Level 1: Driver Assistance - Basic ADAS functions like adaptive cruise control and lane-keeping assistance are included in Level 1 ADAS. These technologies can help with particular tasks, but the driver must still give them their complete attention.

Level 2: Partial Automation - At this stage, vehicles are capable of simultaneously steering and accelerating and decelerating. The driver must, though, be attentive and prepared to take control if necessary. Level 2 automation is exemplified by Tesla's Autopilot.

Level 3: Conditional Automation - Vehicles at Level 3 are capable of doing the majority of driving responsibilities in specified situations, including on the highway. The driver has the option to stop actively driving, but they must be ready to take over if the system requests.

Level 4: High Automation - Without the assistance of the driver, Level 4 vehicles are capable of fully autonomous driving inside specific operational domains or geofenced

locations. However, in rare circumstances, they might still need a human driver.

Level 5: Full Automation - The greatest level of autonomous driving is Level5. At this level, vehicles are capable of operating without any human input in any circumstances. There is no need for a steering wheel or pedals[7].

2.2 LiDAR : Light Detection and Ranging

LiDAR is a remote sensing technology that uses laser pulses to measure distances and generate precise 3D representations of the surrounding environment. By emitting laser beams and calculating the time it takes for the light to return after hitting an object, LiDAR sensors create detailed point clouds that capture shape, depth, and spatial information. This capability makes LiDAR a critical component in perception systems, particularly in applications requiring accurate environmental mapping, such as autonomous driving.

2.2.1 The Necessity of LiDAR in Autonomous Driving

While Cameras and Radar provide critical perception capabilities, they each possess inherent limitations that hinder the reliability of autonomous systems. Cameras, despite offering high-resolution imagery well suitable for object classification, struggle with depth estimation and environmental variations. Radar, although robust in detecting object motion, lacks sufficient spatial resolution to accurately reconstruct detailed scenes and classify objects.

LiDAR bridges these gaps by delivering high-precision depth perception and 3D mapping capabilities. The necessity of LiDAR becomes particularly evident in high-resolution mapping and real-time obstacle detection. Advanced LiDAR systems, with their ability to create dense point clouds, enable autonomous vehicles to navigate urban environments with a high degree of accuracy.

Moreover, the fusion of LiDAR with Cameras and Radars enhances sensor redundancy, mitigating the weaknesses of individual technologies and improving overall system robustness. In applications requiring fine-grained object detection, such as pedestrian identification and lane boundary recognition, LiDAR significantly improves perception accuracy.

Despite the challenges associated with cost and adverse weather sensitivity, ongoing advancements in LiDAR technology are addressing these limitations. Innovations in solid-state LiDAR and cost-efficient manufacturing are making LiDAR more accessible for mass-market deployment.

Consequently, LiDAR remains a crucial component in the sensor fusion framework of autonomous vehicles, providing the necessary depth perception and spatial awareness required for safe and reliable operation.

2.2.2 Evolution of Automotive LiDAR

- **Early Development (1970s-1990s):** LiDAR originated in the 1970s, primarily for applications outside the automotive domain, such as meteorology and topographic mapping. Initial experiments suggested its potential in vehicle safety, particularly for collision avoidance. However, LiDAR systems during this period were bulky, expensive, and lacked the technical maturity for integration into vehicles[38].

- **First Automotive Applications (1990s–2000s):** The first significant automotive use of LiDAR emerged during the Defense Advanced Research Projects Agency (DARPA) Grand Challenges in the early 2000s, which demonstrated its feasibility in autonomous driving. Velodyne's early 3D LiDAR systems as in Figure 2.2 played a pivotal role in this evolution by providing detailed environmental mapping capabilities. These systems, however, were limited by their high cost and mechanical complexity, restricting adoption to research and prototyping[17].

Figure 2.2: Velodyne's ULTRA Puck™ (VLP-32C) [17]

- **Miniaturization and Cost Reduction (2010s):** During the 2010s, advances in solid-state LiDAR significantly reduced size, weight, and cost while improving performance. Solid-state LiDAR eliminated moving parts in some models, increasing reliability and scalability for mass production. Companies such as Velodyne and Luminar drove innovation, making LiDAR more accessible for ADAS and autonomous vehicles [3]. This period also saw a growing focus on integrating LiDAR with other sensors like Cameras and Radar to enable robust sensor fusion systems for autonomous navigation[15].

- **Integration with Vehicle Architectures (2010s-Present):** LiDAR systems became integral to modern vehicle architectures, particularly in Level 3 and higher autonomous driving systems. Machine learning and AI advancements enhanced the interpretation of LiDAR data for tasks such as object detection and motion prediction. Safety standards, such as International Organization for Standardiza-

tion (ISO)-21448, Safety Of The Intended Functionality (SOTIF), provided guidelines to ensure reliable functionality in complex driving scenarios[15].

- **Next-Generation LiDAR (2020s and Beyond):** Recent innovations focus on photonic integrated circuits (PICs), Frequency-Modulated Continuous Wave (FMCW) and LiDAR, which promise higher resolution, greater range, and better resistance to environmental interference. LiDAR is now being explored for use in Vehicle-to-Everything (V2X) communication, enabling real-time data exchange between vehicles and infrastructure. As costs continue to decline, LiDAR is poised to play a crucial role in both fully autonomous and connected vehicle ecosystems[8].

2.2.3 Different generations of LiDAR at Valeo Schalter und Sensoren GmbH

Gen 1 (SCALA™ Gen 1) - First Step Towards 3D Perception

- **Performance Focus:** The Gen 1 LiDAR, launched in 2017, the SCALA™ Gen-1 marked the first introduction of 3D scanning technology in series-produced vehicles. It was designed to provide basic environmental awareness to assist with essential ADAS functions[30].
- **Key Features:** It was capable of detecting objects, pedestrians, and other vehicles, but with more limited resolution and range compared to later generations.
- **Applications:** Used for basic functions like adaptive cruise control, lane-keeping assist, and emergency braking.
- **Technical Specifications:** The SCALA™ Gen 1 had a horizontal field of view (FoV) of 145° and a vertical FoV of 3.2°. Its angular resolution was 0.25° horizontally and 0.8° vertically[10].
- **Major Limitation:** The range and precision were not sufficient for more complex autonomous driving tasks, such as full self-driving in difficult or dynamic environments.

Figure 2.3: SCALA Gen-1 [29]

Gen 2 (SCALA™ Gen 2) - Improved Perception for Advanced Functions

- **Performance Focus:** The Gen-2 LiDAR, released in 2021, the SCALA™ Gen 2 was a substantial upgrade in terms of precision, range, and vertical Field of View (FOV). This generation aimed to provide more detailed and accurate 3D models of the vehicle's surroundings[11].
- **Key Features:** With better angular resolution, it allowed for finer object detection, identifying smaller objects and offering more detailed data about the environment.
- **Applications:** It enabled more advanced ADAS features, such as traffic jam pilot systems, which required higher levels of situational awareness. It was also designed to support more advanced features required for Level 3 autonomous driving like partial self-driving in controlled conditions.
- **Technical Specifications:** The SCALA™ Gen 2 offered a horizontal FoV of 133° and a vertical FoV of 10°. Its horizontal angular resolution was 0.125° within ±15° and 0.25° in the outer FoV, while the vertical angular resolution was 0.6°[16].
- **Major Improvement:** Increased range and vertical FOV allowed for more precise detection in more complex driving environments, such as dense city streets or highways with many obstacles.

Figure 2.4: SCALA Gen-2 [29]

Gen 3 (SCALA™ Gen 3) - Advanced Perception for Autonomous Driving

The SCALA™ Gen-3 LiDAR marks a significant leap in LiDAR technology, designed to support the most demanding requirements of Level 3 and Level 4 autonomous driving. This third-generation sensor offers exceptional performance in terms of range, resolution, and robustness in a variety of real-world driving conditions, such as adverse weather (rain, fog) and low-light environments. Capable of detecting small objects, it can reliably identify a tire on an unlit black asphalt road from over 150 meters ahead, an object that other sensors like cameras and Radars may miss. This ability plays a critical role in ensuring unparalleled safety and reliability for autonomous vehicles.

In addition to its hardware advancements, the SCALA™ Gen-3 is equipped with a number of software modules that include perception and AI-based algorithms. These

algorithms enhance the sensor's ability to not only detect objects but also to classify and track them. This comprehensive data is then fused with other sensor inputs to create a detailed, real-time 3D map of the environment surrounding the vehicle, a vital element for autonomous navigation[24].

Key software features include:

- **Blockage Detection:** Ensures continuous and reliable operation by detecting and compensating for any obstructions that might hinder LiDAR's line of sight.
- **Rain and Spray Detection:** Enhances perception by recognizing environmental interference such as rain or road spray, thereby improving robustness under adverse weather conditions.
- **Online Calibration and Misalignment Detection:** Allows for automatic calibration of the LiDAR system in real-time, maintaining accuracy over time and compensating for misalignment that may occur due to road conditions or vehicle vibrations.
- **Technical Specifications:** The SCALA™ Gen 3 provides a horizontal FoV of 120° and a vertical FoV of 26°, with both horizontal and vertical angular resolutions reaching 0.05°, enabling high-definition 3D perception[25].

Figure 2.5: SCALA Gen-3 [29]

This integration of AI-driven perception capabilities with advanced hardware enables full 3D mapping and a comprehensive understanding of the vehicle's surroundings. The data generated is critical for tasks like obstacle avoidance, detecting lane markings, and understanding dynamic traffic scenarios. Moreover, these software modules are optimized to run seamlessly on major System on Chip (SoC) platforms and can be easily embedded into dedicated ECUs or domain controllers within the vehicle's architecture.

With 12.5 million points per second, SCALA™ Gen-3 provides higher resolution, longer detection range, and more precise environmental modeling than any previous generation. These capabilities are essential for the safe and effective deployment of autonomous driving systems, making it a key enabler of self-driving vehicles that can navigate complex, unpredictable environments. The performance evolution of SCALA generations can be seen in Figure 2.6, while Figure 2.7 shows the corresponding point

cloud outputs in a real driving scenario.

PARAMETER	SCALA GEN. 1	SCALA GEN. 2	SCALA GEN. 3 *
Technology	Rotating Mirror (750rpm)	Rotating Mirror	
Range	150 m	150 m x3	+200m to 450m
Scan Angle Horizontal FoV / Angular Resolution Points number	145° / 0.25° 580 points	133° / 0.25° (+/- 15° / 0.125°) 532 + 120 = 652 points	
Scan Angle Vertical FoV / Angular Resolution Multi-Layer number	3.2° / 0.8° 4 layers x4	10° / 0.6° 16 layers	
Data Update Rate / Scanning Frequency	25 fps / 25Hz	25 fps / 25Hz	
Total points per scan per second	580 points x 4 layers x 25 fps 58 000 points cloud x4.5	652 points x 16 layers x 25 fps 260 800 points cloud x17	4.5 millions
Multi-echo	3 measurements per shot	3 measurements per shot	
Accuracy	< 10 cm		
Distance resolution	4 cm	< 10 cm	50 times better than Gen. 2
Distance error	10 cm		
Dimensions Volume	133 x 101 x 70 mm 940 cm3	150 x 93 x 68 mm 948 cm3	
Weight	625 g -10%	565 g	
Car Manufacturers / Models	Audi A8 (2018)	Honda Legend (2021) Mercedes S-Class (2021)	Stellantis (2024)
Autonomous Level Certification	Level-2	Level-3	Level-3

Figure 2.6: Performance comparison between SCALA Gen 1-2-3 [43]

Figure 2.7: Real scene and corresponding point clouds from SCALA Gen-1, Gen-2, and Gen-3 [29]

2.3 LiDAR Range Equation

The received power in a LiDAR system is a crucial factor in determining the accuracy and reliability of distance measurements. It is influenced by multiple factors, including the transmitted power, target reflectivity, atmospheric conditions, and the efficiency of

system components. The power budget equation shown in Figure 2.8 provides a quantitative relationship between these parameters, typically including an inverse-square law dependency on distance ($1/R^2$). This equation not only explains the physical principles of power attenuation but also serves as a foundation for sensor modeling and calibration.

In our sensor model, this behavior is explicitly implemented. For example, the return power decreases with increasing target distance, which we validate using the "Infinite surfaces" test scenario. Additionally, the model accounts for varying target reflectivity by allowing the assignment of different material properties. Our calibration tool incorporates this capability, enabling users to simulate and analyze the effect of material-dependent reflectivity on the received signal. This integration ensures that both the simulation and calibration environments accurately reflect real-world LiDAR sensor behavior.

Power Budget Equation: The received power P_S in a LiDAR system can be expressed as:

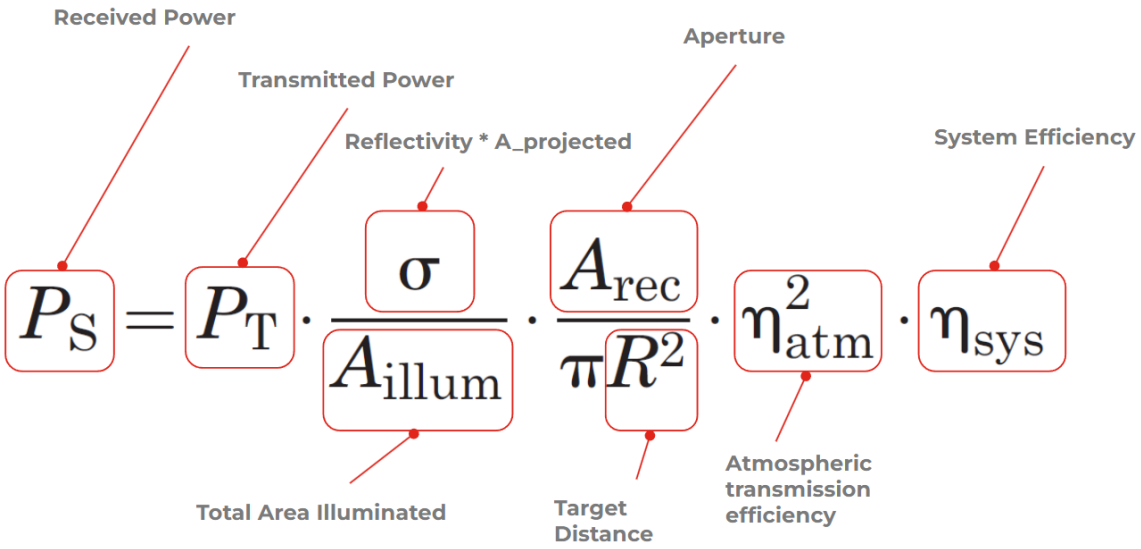

Figure 2.8: Power budget equation in LiDAR systems [18]

Where:

- P_S = Received power
- P_T = Transmitted power
- σ = Reflectivity & Projected area
- A_{illum} = Total area illuminated
- A_{rec} = Receiver aperture area
- R = Target distance
- η_{atm} = Atmospheric transmission efficiency

- η_{sys} = System efficiency

Each term in this equation plays a critical role in determining the performance of a LiDAR system. Below, we analyze the significance of these factors in detail.

Explanation of Terms

- **Transmitted Power (P_T)**: This represents the initial power of the laser pulse emitted by the LiDAR system. Higher transmitted power increases the likelihood of detecting distant or low-reflectivity objects. However, increasing P_T may lead to higher power consumption and potential eye safety concerns[18].
- **Reflectivity and Projected Area (σ)**: Reflectivity is a measure of how much of the transmitted laser pulse is reflected by the target surface. It depends on the material, color, and roughness of the object. The projected area accounts for the effective portion of the target that interacts with the laser beam[18].
- **Total Area Illuminated (A_{illum})**: This term represents the footprint of the laser beam when it reaches the target. A smaller illuminated area increases the energy density of the laser pulse, leading to a stronger return signal. However, a larger illumination area may be required in scanning applications to cover a wider FOV[18].
- **Receiver Aperture Area (A_{rec})**: The receiver collects the returning laser signal. A larger aperture allows for better signal capture, improving detection at greater distances. The optical design of the receiver plays a crucial role in minimizing noise and maximizing sensitivity[18].
- **Target Distance (R)**: The power of the received signal is inversely proportional to the square of the distance. This follows from the inverse-square law, which states that the intensity of an emitted signal decreases as the square of the distance from the source. As a result, detecting objects at long distances requires careful optimization of system parameters[18].
- **Atmospheric Transmission Efficiency (η_{atm})**: The atmosphere absorbs and scatters light, reducing the strength of the returning signal. Factors such as fog, rain, dust, and humidity impact η_{atm}. The squared term accounts for the fact that the laser pulse must travel to the target and back through the atmosphere[18].
- **System Efficiency (η_{sys})**: System efficiency encompasses various losses in the LiDAR system, including optical losses in lenses, mirrors, and detectors, as well as electronic losses in signal processing components. High system efficiency ensures better utilization of received power[18].

Practical Implications

Understanding and optimizing the power budget equation is essential in designing LiDAR systems for autonomous driving, robotics, and remote sensing. Some key considerations include:

- **Minimizing noise**: Wavelength is chosen so that it coincides with a minimum in the spectrum of the solar radiation making it possible to use a band pass filter and block sunlight while letting the laser light pass into the sensor.
- **Optimizing Receiver Aperture**: A well-designed optical system with a large aperture enhances detection at long distances.
- **Managing Atmospheric Effects**: Adaptive signal processing techniques can compensate for atmospheric disturbances.
- **Balancing Power Consumption and Performance**: Systems should be designed to operate within safety regulations while maintaining reliable detection.

The power budget equation provides a fundamental framework for analyzing the efficiency and performance of LiDAR systems. By understanding the role of each parameter, engineers can make informed design choices to optimize LiDAR functionality for various applications[18].

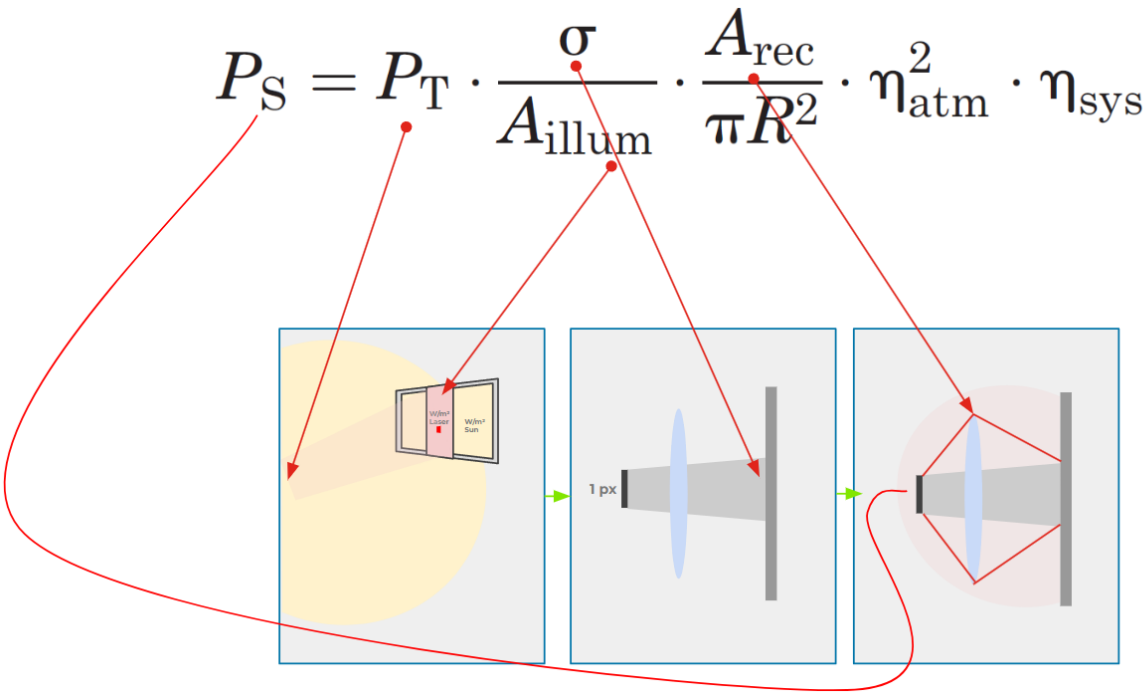

Figure 2.9: Illustration of factors influencing received power in LiDAR systems. [18]

The diagram in Figure 2.9 provides a visual breakdown of the equation's components and their real-world implications.

Target Illumination and Reflectivity

The first segment on the left in Figure 2.9 represents how the laser pulse illuminates a target. The target reflects a fraction of the incident laser power depending on its material properties and surface characteristics. The reflectivity (σ) determines how much energy is scattered back towards the sensor. The equation term $\frac{\sigma}{A_{illum}}$ describes this process.

- The left image in the Figure 2.9 illustrates a laser beam striking a surface, where both the laser intensity and ambient sunlight intensity are considered.
- The reflectivity is influenced by surface properties such as roughness, angle, and composition.
- A highly reflective surface returns more power to the LiDAR sensor compared to a low-reflectivity surface.

Receiver Aperture and Distance Effects

- The middle section of Figure 2.9 shows the receiver aperture and the effect of distance on received power.
- The term $\frac{A_{\text{rec}}}{\pi R^2}$ in the equation accounts for the inverse-square law governing power distribution over distance.
- As the target distance (R) increases, the received power decreases proportionally to $1/R^2$.
- The middle image illustrates how the returning signal is captured by the receiver, with each pixel receiving a fraction of the reflected signal.

Atmospheric and System Efficiency

- The right section of Figure 2.9 represents the impact of aperture, atmospheric and system losses.
- The atmospheric transmission efficiency (η_{atm}) accounts for absorption and scattering effects in the medium (e.g., fog, dust, or rain) that reduce signal strength.
- The system efficiency (η_{sys}) encompasses losses due to optical components, detector sensitivity, and electronic noise.
- The right image in the diagram depicts how optical components and system limitations influence the final received signal.

Key Takeaways

- The power received by the sensor depends on the interplay of target reflectivity, illuminated area, distance, and optical efficiency.
- The equation follows an inverse-square relationship, meaning that increasing the distance significantly reduces the received power.
- System and atmospheric losses must be considered when designing LiDAR sensors for real-world applications.
- The visual representation (Figure 2.9) helps in understanding how different components contribute to the received signal.

Understanding the power budget equation is crucial for optimizing LiDAR performance in various applications, including autonomous driving and remote sensing. The provided equation and diagram shown in Figure 2.9 clarify the physical principles governing LiDAR signal reception and highlight factors affecting the power efficiency of the system. Factors like target distance, surface reflectivity, and system efficiency are implemented in the sensor model and play an important role in the calibration and validation of the sensor model.

2.4 SPAD Arrays in LiDAR Sensor Technology

SPAD arrays are highly sensitive photodetectors capable of detecting, counting, and timing the arrival of individual photons. This exceptional sensitivity makes them a core component in modern LiDAR systems, particularly where high-resolution depth sensing is required under challenging conditions such as long-range detection or low reflectivity surfaces. Their relevance has increased significantly with the push toward higher levels of autonomy in vehicles, where precise 3D mapping is vital.

2.4.1 Operating Principle

SPADs operate in what is known as Geiger mode, where they are reverse biased beyond their breakdown voltage. In this state, a single incident photon is sufficient to trigger an avalanche breakdown, resulting in a large current pulse that is detectable by downstream circuitry. This current pulse signifies the detection of a single photon.

After each detection event, the SPAD must be ”quenched” to stop the avalanche and then reset before it can detect another photon. This leads to a dead time, during which the detector is inactive. This factor must be managed in high-photon-flux environments to avoid saturation or pile-up effects[3].

In LiDAR applications, SPADs are often arranged in linear or two-dimensional arrays, allowing parallel detection. In our sensor setup, the array consists of a vertical line of SPADs, where only a vertical laser line is projected and detected. Each point in the point cloud is not derived from a single SPAD but from a group of SPADs (e.g., 3×9 in the SMART variant, 3×21 in SLIM). These grouped responses are then combined over multiple laser pulses (e.g., 5 shots), leading to a maximum peak amplitude (e.g., 135 for SMART when all SPADs are activated across all pulses).

2.4.2 SPAD vs. APD

The distinction between SPADs and Avalanche Photodiode (APD)s is fundamental. APDs function below the breakdown voltage, amplifying current in proportion to light intensity without entering self-sustaining conduction. In contrast, SPADs, due to their operation above the breakdown voltage, produce a digital-like pulse even from a single

photon. This results in an extremely high amplification factor but necessitates quenching and introduces non-linear behaviors under high photon flux. These behaviors impact LiDAR signal processing algorithms, particularly in Time-of-Flight (ToF) systems where accurate temporal resolution is critical [41]. The current-voltage characteristics that distinguish these two types of photodetectors are depicted in Figure 2.10.

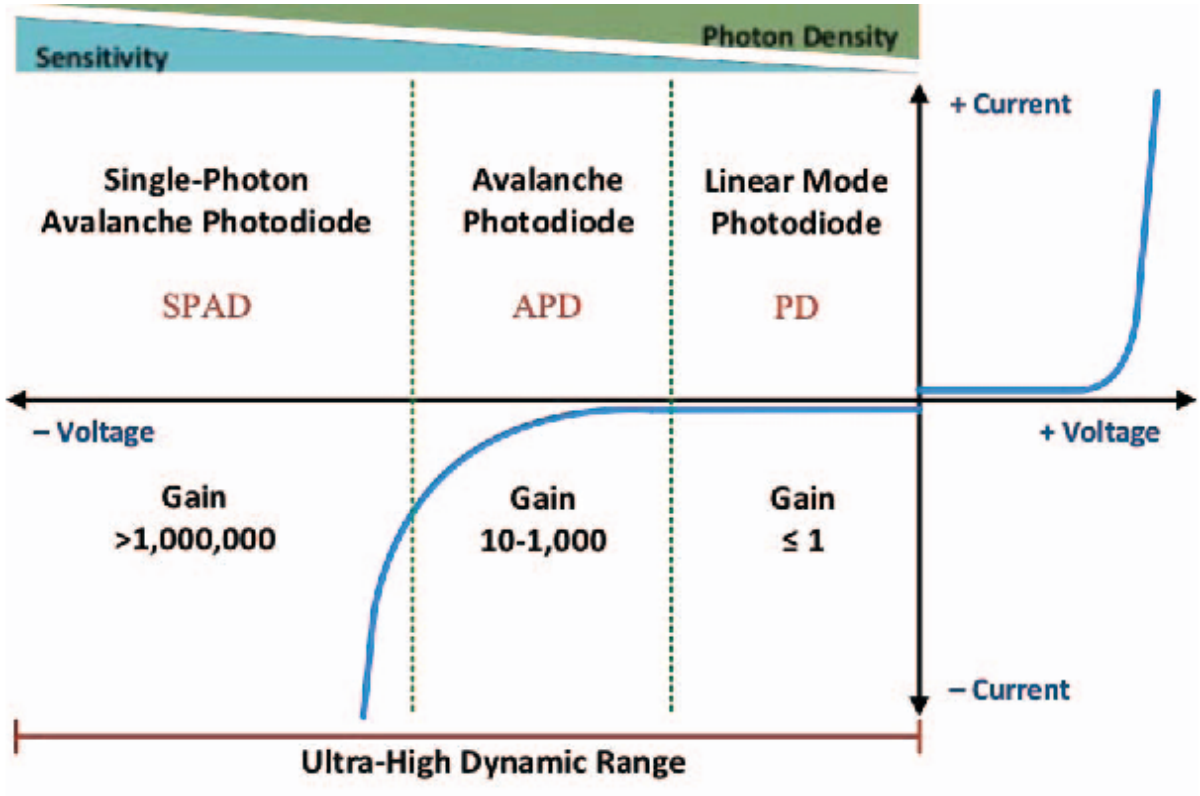

Figure 2.10: Current-voltage relationship of a p-n diode. [41]

2.4.3 Time-Bin Histograms and Pile-Up Effects

One key characteristic of SPAD arrays is their ability to generate time-bin histograms, which reflect the time distribution of detected photons. At low to medium light intensity, the probability of triggering in a given time-bin is approximately proportional to the incoming photon flux. Thus, the histogram shape closely resembles the actual light pulse profile, as illustrated in Figure 2.11.

However, at high intensities, pile-up effects become significant. In such cases, SPAD elements fire early and enter a dead-time phase, leaving later portions of the light pulse undetected. As a result, the observed histogram skews toward earlier bins, shifting the peak and the weighted mean of the distribution closer to the sensor. This distortion, shown in Figure 2.12, can affect distance estimation and must be accounted for in the peak detection algorithm.

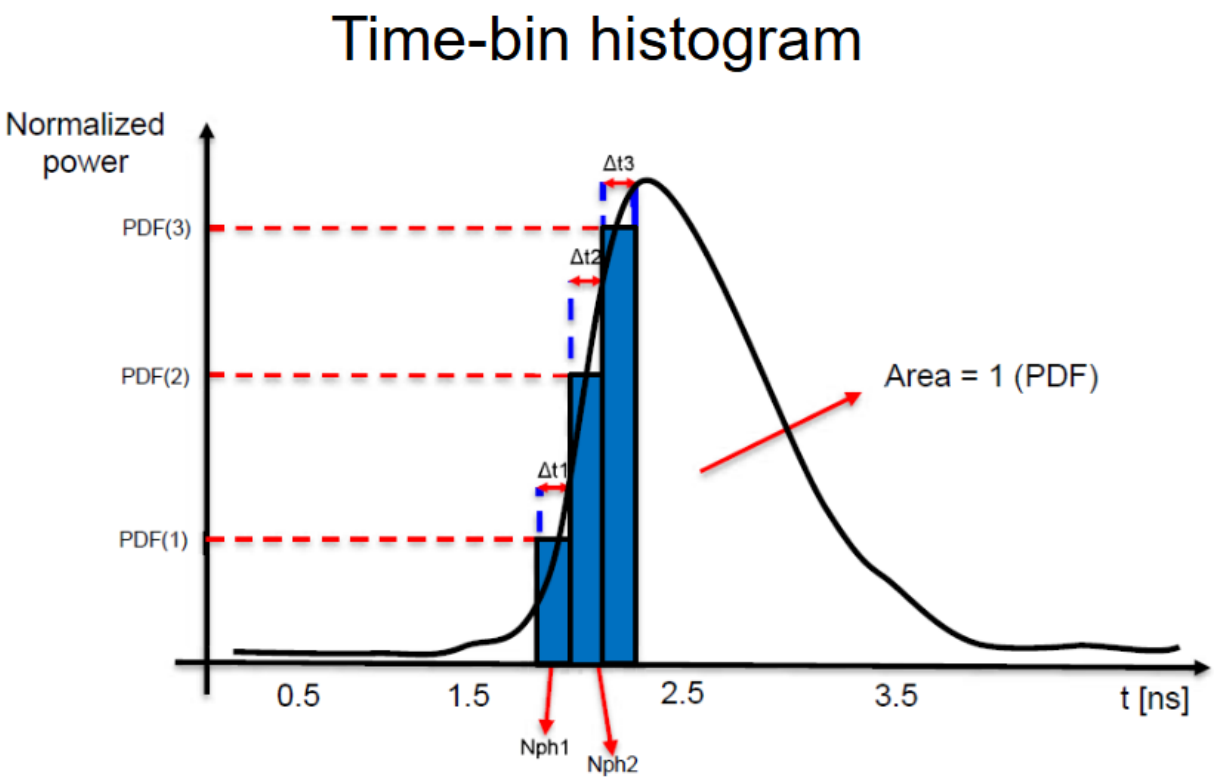

Figure 2.11: Time bin histogram

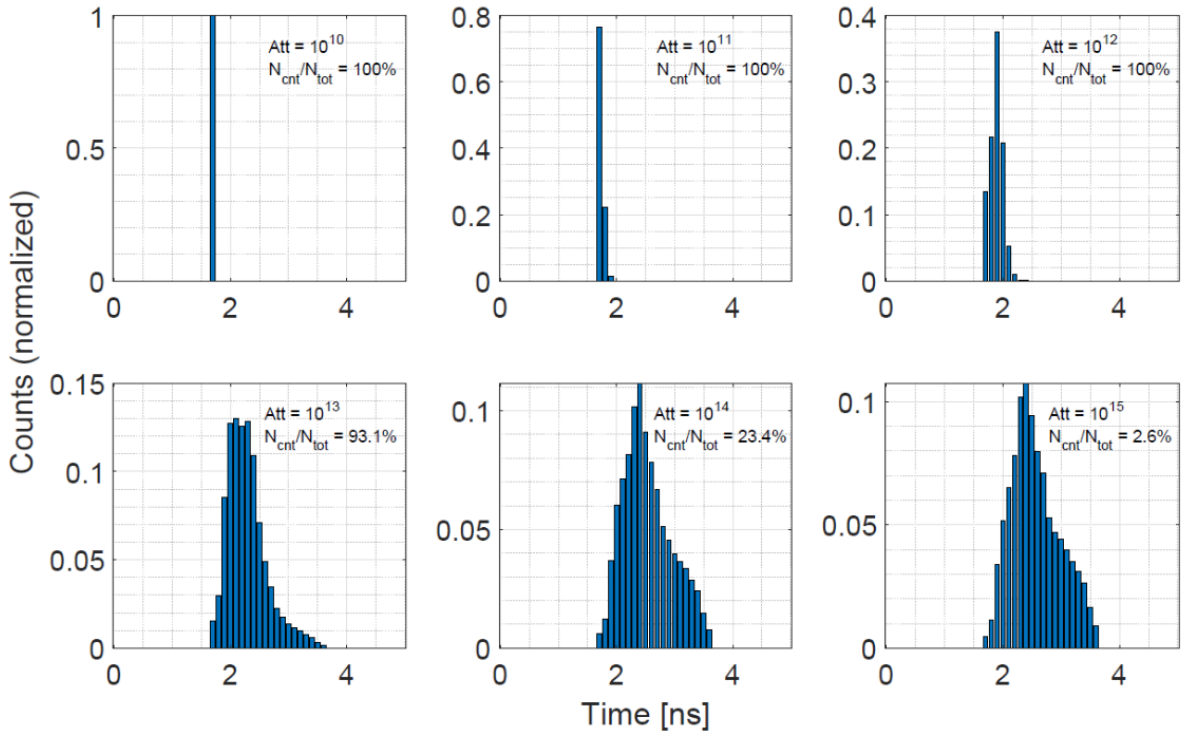

Figure 2.12: Pile up effect

2.4.4 Challenges from Dead-Time and Cross-Talk

SPAD arrays also exhibit additional effects stemming from dead-time variability. The duration of dead-time is inherently stochastic where some SPADs recover faster, while others remain inactive for longer. Moreover, cross-talk between adjacent SPAD pixels can result in false detections, further complicating the signal interpretation. These factors introduce noise into the detection process and require robust statistical modeling and filtering to maintain accuracy in ToF measurements [12]. The dead-time effect is shown in Figure 2.13.

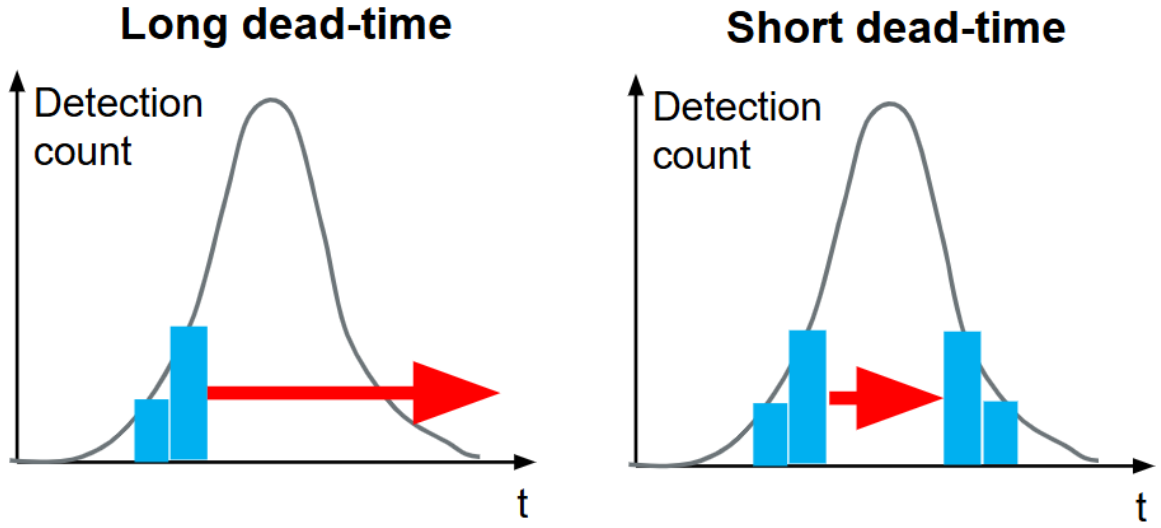

Figure 2.13: Dead time

2.4.5 Advantages over Traditional Sensors

SPAD arrays exhibit several performance advantages when compared to conventional photodiodes, Charge-Coupled Device (CCD)s, or Complementary Metal-Oxide-Semiconductor (CMOS) sensors:

- **Single-photon sensitivity**: SPADs can operate in extremely low-light conditions.
- **Sub-nanosecond temporal resolution**: This enables accurate depth measurements using ToF techniques.
- **High frame rates**: Some SPAD implementations can achieve frame rates up to 700,000 Frames Per Second (fps), supporting ultra-fast dynamic scene capture.
- **CMOS compatibility**: This makes SPADs easier to integrate into compact, scalable, and cost-effective sensor modules.

These features enable SPAD arrays to outperform traditional sensors, which require stronger illumination and suffer from slower response times[6].

2.4.6 Limitations and Mitigation Techniques

Despite their advantages, SPAD arrays are not without limitations:

- **Dead time and saturation**: At high photon rates, the dead time can lead to pile-up errors where only the earliest photons are detected, skewing the depth measurement.
- **Saturation**: At some point, all SPADs are triggered and the amplitude is not sensitive to additional photons.
- **Afterpulsing**: Trapped carriers may trigger false detections.
- **Crosstalk**: Electrical or optical interference between adjacent SPAD elements can affect signal integrity.

To address these challenges, researchers have developed advanced pile-up correction algorithms, time-gating techniques, and multi-event histogramming approaches, which improve data accuracy under realistic conditions.

SPAD arrays represent a technological leap in photodetection and imaging, with wide-ranging applications from automotive LiDAR to quantum communication and biomedical diagnostics. Their integration into LiDAR systems offers crucial improvements in range, resolution, and robustness, helping push the limits of what is achievable in autonomous navigation and sensing. As research continues, innovations in SPAD pixel architecture, Time-to-Digital Converter (TDC) integration, and noise mitigation will further enhance their role in future sensor systems.

2.5 Alternative Statistical Techniques for LiDAR Simulation and Calibration

LiDAR sensor simulation and calibration involve comparing real and simulated data using statistical techniques. Histogram-based methods are commonly used for such analysis; however, alternative algorithms can provide different insights and accuracy. This section explores the standard histogram approach and presents four alternative methods: Quantile-Quantile (Q-Q) Plots, Kernel Density Estimation (Kernel Density Estimation (KDE)), Cumulative Distribution Functions (Cumulative Distribution Function (CDF)), and Violin Plots. These techniques are discussed in the context of their applicability, advantages, and limitations in LiDAR sensor simulation and calibration. In this thesis, only histogram-based methods are implemented, while the alternative approaches are included to suggest possible directions for future enhancement of the calibrator.

1. **Histogram-Based Analysis**
   Histograms are widely used in LiDAR data analysis to compare distributions of metrics such as Peak, Area, Width and Distance. They group data into fixed bins

and count occurrences, making them easy to interpret and apply.

Applications in LiDAR Calibration:

- Comparing distributions of real and simulated data.
- Analyzing distance, peak, area and width measurements.
- Identifying calibration mismatches across scenarios or object classes.
- Supporting statistical validation of sensor models.

Advantages:

- **Simple and Intuitive**: Easy to interpret, making it suitable for quick visual assessments.
- **Effective for Large Datasets**: Efficiently summarizes large point clouds with minimal computational effort.
- **Flexible Binning**: Allows adjustment of granularity based on analysis needs.
- **Overlay Visualization**: Facilitates side-by-side comparison of multiple datasets.

Limitations:

- **Bin Width Sensitivity**: Poor bin selection may hide patterns or exaggerate noise.
- **Loss of Detail**: Individual data points are grouped, reducing granularity.
- **Limited Smoothness**: Results in a stepwise distribution that may not reflect the underlying density.
- **Misleading with Sparse Data**: Inaccurate representation with low sample sizes.

2. **Quantile-Quantile (Q-Q) Plot Analysis**
Q-Q plots are graphical tools that compare the quantiles of two probability distributions. In the context of LiDAR calibration, Q-Q plots allow engineers to visually assess whether the real and simulated data distributions align. A straight 45-degree line indicates a close match between the datasets.

Applications in LiDAR Calibration:

- Identifying whether real and simulated data follow the same statistical distribution.
- Detecting discrepancies and deviations from ideal calibration.
- Assessing tail behavior, particularly useful in identifying anomalies.
- Supporting statistical model validation for distribution fitting.

Advantages

- **No Binning Required**: Provides a smooth, continuous comparison of distributions.
- **Sensitive to Distribution Differences**: Detects mismatches across the entire range, especially in tails.
- **Useful for Model Evaluation**: Helps select the best-fitting distribution for LiDAR measurements.
- **Clear Visual Interpretation**: Ideal for diagnosing systematic deviations.

Limitations

- **Requires Statistical Expertise**: Interpretation may not be intuitive for non-experts.
- **Poor Fit for Multimodal Data**: May fail to represent complex, multi-peaked distributions.
- **Sensitive to Outliers**: Extreme values can distort the plot.
- **Less Effective with Small Datasets**: May produce unreliable results with sparse data.
- **Limited Applicability**: Only suitable for data where each real point can be correlated with the simulated point.

3. **KDE: Kernel Density Estimation**
KDE is a non-parametric technique that estimates the probability density function of a dataset by smoothing individual data points using a kernel function, commonly Gaussian. It provides a continuous curve representing the underlying data distribution.

Applications in LiDAR Calibration:

- Smoothing noisy or sparse LiDAR datasets.
- Comparing continuous distribution trends between real and simulated data.
- Detecting fine-grained differences not visible in histograms.
- Visualizing complex data structures in object surfaces or terrain.

Advantages:

- **Smooth Distribution Representation**: Avoids the step-like nature of histograms.
- **Greater Detail Resolution**: Captures subtle patterns and distribution features.
- **No Binning Needed**: Eliminates artifacts caused by arbitrary bin choices.
- **Customizable Bandwidth**: Enables tuning of smoothness based on data characteristics.

Limitations:

- **Bandwidth Sensitivity**: Improper selection can result in oversmoothing or noise amplification.
- **Computational Complexity**: Resource-intensive for large datasets.
- **Kernel Selection Matters**: Different kernel shapes can lead to varying outputs.
- **Visual Interpretation May Vary**: Requires careful analysis to avoid misinterpretation.

4. **CDF: Cumulative Distribution Function**
The CDF describes the probability that a variable takes on a value less than or equal to a given threshold. It provides a complete view of data distribution, from minimum to maximum values, in a cumulative format.

Applications in LiDAR Calibration:

- Performing statistical hypothesis testing between real and simulated datasets.
- Comparing cumulative behavior of distance, intensity, or area values.
- Computing percentile-based metrics for calibration validation.
- Detecting systematic shifts in data distributions.

Advantages:

- **Complete Representation**: Captures the entire distribution in a single curve.
- **Bin-Free Comparison**: Provides a smoother comparison than histogram-based methods.
- **Useful for Statistical Testing**: Enables use of the Kolmogorov-Smirnov test and other metrics.
- **Percentile Analysis**: Helps define thresholds (e.g., 90th percentile intensity).

Limitations:

- **Poor Multimodal Visibility**: Fails to highlight distribution peaks clearly.
- **Less Intuitive Visuals**: May be harder to interpret compared to histograms or KDE.
- **Small Differences Hard to Spot**: Subtle variations in distribution can be visually compressed.
- **Requires Alignment of Dataset Sizes**: Unequal data lengths may affect accuracy.

5. **Violin Plots**
   Violin plots combine KDE with box plots to present both statistical summaries and data distribution in one visualization. They are effective in comparing distributions between multiple groups or metrics.

   Applications in LiDAR Calibration:

   - Comparing real vs. simulated attributes like intensity, area, and width.
   - Identifying outliers and multimodal distributions in LiDAR returns.
   - Visualizing multiple categories or object types simultaneously.
   - Summarizing distribution and variability in a single compact view.

   Advantages:

   - **Rich Visual Insight**: Merges summary statistics and full distribution.
   - **Highlights Multimodal Trends**: Easily identifies multiple peaks and skewed data.
   - **Outlier Detection**: Clearly shows data points that deviate from the norm.
   - **Comparative Analysis**: Effective for visualizing differences across datasets or scenarios.

   Limitations:

   - **Subjective Interpretation**: Density shapes can be misread without statistical context.
   - **Visual Clutter**: Can become overwhelming when comparing many groups.
   - **Computational Load**: More resource-intensive than basic plots.
   - **Requires Normalization**: Unequal sample sizes can skew visual appearance.

2.6 Software Requirements

In the development of LiDAR sensor simulation and calibration tools, a robust set of software tools was employed to ensure accurate modeling, efficient data processing, and effective visualization. The chosen tools provided capabilities for real-time simulation, GUI development, and point cloud analysis, each playing a critical role in supporting the objectives of this thesis. The software environment was selected to balance computational performance with ease of development and integration into existing automotive toolchains.

1. **C++**
   C++ is a high-performance programming language extensively used in system programming, real-time applications, and simulation development due to its object-oriented features, efficient memory management, and rich standard libraries[28]. In this thesis, C++ was used to implement and extend LiDAR sensor simulation tools, enabling fast generation and processing of sensor data. Its compatibility with other automotive simulation environments ensured smooth integration of the sensor model into the larger simulation framework. C++ is used as an interface between the sensor model and IPG Carmaker or Applied Intuition ADP as environment simulations.

2. **C# and Windows Forms**
   C# is a modern programming language developed by Microsoft, widely used for building desktop applications, web services, and game engines. Combined with Windows Forms, a part of the .NET framework it enables the rapid development of interactive GUIs[20]. In this work, C# and Windows Forms were employed to design and implement a GUI-based calibration tool. This application allows for efficient visualization and parameter tuning of LiDAR sensor outputs, enhancing usability for engineers involved in calibration tasks.

3. **CloudCompare**
   CloudCompare is an open-source 3D point cloud and mesh processing software, recognized for its capabilities in analyzing, comparing, and visualizing 3D datasets such as those generated by LiDAR systems[5]. It was extensively used during this thesis for comparing real and simulated point clouds, enabling detailed analysis of calibration quality. Its transformation and filtering features proved essential for aligning datasets and minimizing noise, thereby ensuring accurate and high-fidelity evaluation of the sensor models.

4. **International Performance Group (IPG) CarMaker**
   IPG CarMaker is a widely adopted simulation platform designed for virtual testing of vehicle dynamics and driver assistance systems. It provides a realistic driving environment that supports the integration and testing of various sensors, including LiDAR[1]. In this project, CarMaker was used to simulate diverse driving scenarios, allowing the validation of LiDAR behavior under dynamic conditions without the need for repeated real-world testing. The automation features of CarMaker were instrumental in running numerous test cases efficiently.

5. **Visual Studio**
   Microsoft Visual Studio is a comprehensive Integrated Development Environment (IDE) that supports multiple programming languages and offers advanced debugging, code management, and UI design capabilities[19]. It was utilized as the primary IDE for both C++ and C# development throughout this thesis. Its integrated tools and seamless support for version control facilitated efficient development, debugging, and maintenance of the software components involved in simulation and calibration workflows.

3 Concept of Solution

The goal of LiDAR sensor calibration is to minimize the discrepancy between real and simulated sensor data, ensuring a realistic perception pipeline for autonomous driving systems. Achieving this requires not only precise sensor modeling but also a robust calibration framework that can adapt to varying test conditions and parameter sets. This thesis proposes a structured approach to sensor calibration that emphasizes both accuracy and efficiency.

3.1 Sensor Model Use with CarMaker

In this setup, synthetic driving scenarios are generated using the simulation platform ”CarMaker.” These scenarios include detailed road geometry, traffic participants, environmental elements, and vehicle dynamics. Based on this information, CarMaker produces a virtual representation of the world, which includes a point cloud mimicking what a real LiDAR would observe.

This point cloud is passed as input to the LiDAR sensor model, which simulates the sensor's response to the virtual environment. The simulated LiDAR data, typically in the form of point clouds, is exported for evaluation. Analysis tools such as histograms of Peak, Area, and Width are used to assess sensor performance, range characteristics, or behavior in specific situations.

While this method is effective during early sensor model development and validation, it follows a fixed pipeline and lacks flexibility for rapid iterative refinement. Since each test cycle must be manually repeated for any parameter change, it becomes time-consuming, especially when fine-tuning or comparing calibration outcomes as shown in Figure 3.1.

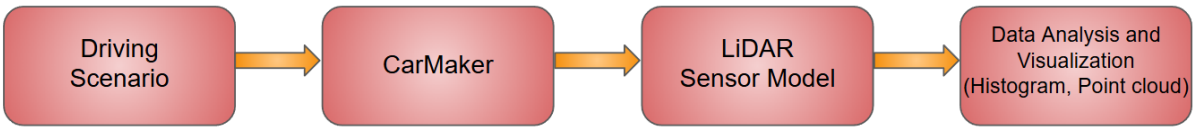

Figure 3.1: Sensor Model Use with CarMaker

3.2 Interactive Calibration Workflow

In the proposed Interactive Calibration Workflow, the calibration process is structured to enable iterative tuning and immediate feedback. The loop starts with the Calibration

Tool, which has features to adjust the parameters related to blooming, noise statistics, ghost particles or even generate entire scenes with aforementioned artefacts. These parameters are passed to the Sensor Model, which uses them to generate a simulated LiDAR point cloud for a given scenario.

The output of the sensor model is then passed to the Visualization and Analysis Module, where it is analyzed using mathematical tools such as histograms or point clouds. This module evaluates how closely the simulated data matches the corresponding real-world data.

Based on this comparison, visual feedback or quantitative error metrics (e.g., histogram deviation, point cloud overlap error) are generated and fed back into the Calibration Tool. This enables immediate parameter adjustment and supports rapid convergence toward a realistic and accurate sensor simulation.

This closed-loop approach significantly reduces iteration time and enhances the ability to fine-tune the LiDAR model under various operating conditions, compared to the more static and manual open-loop process as shown in Figure 3.2.

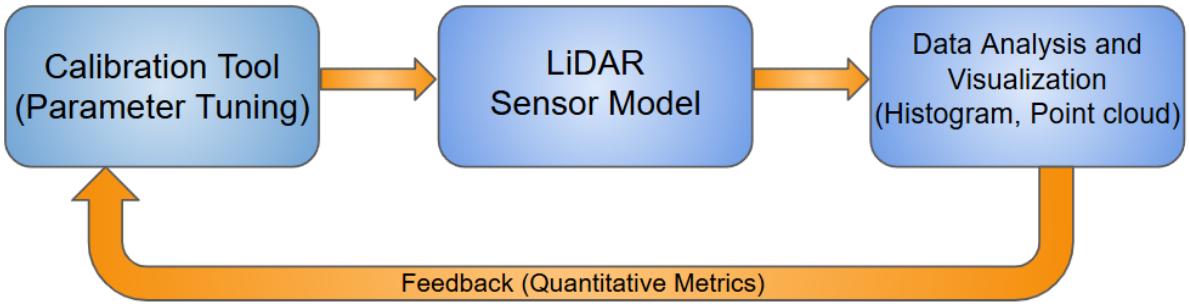

Figure 3.2: Interactive Calibration Workflow

Overview of Conceptual Flowchart

The calibration framework is designed around a feedback-driven loop that enables iterative parameter refinement and real-time analysis of LiDAR simulation output. By decomposing the workflow into clearly defined modular blocks ranging from ideal synthetic data generation, sensor modeling, and visualization, to calibration and configuration the approach supports scalability, reusability, and fine-grained control over the simulation accuracy. The following flowchart outlines the structure and information flow of the proposed calibration process, as shown in Figure 3.3.

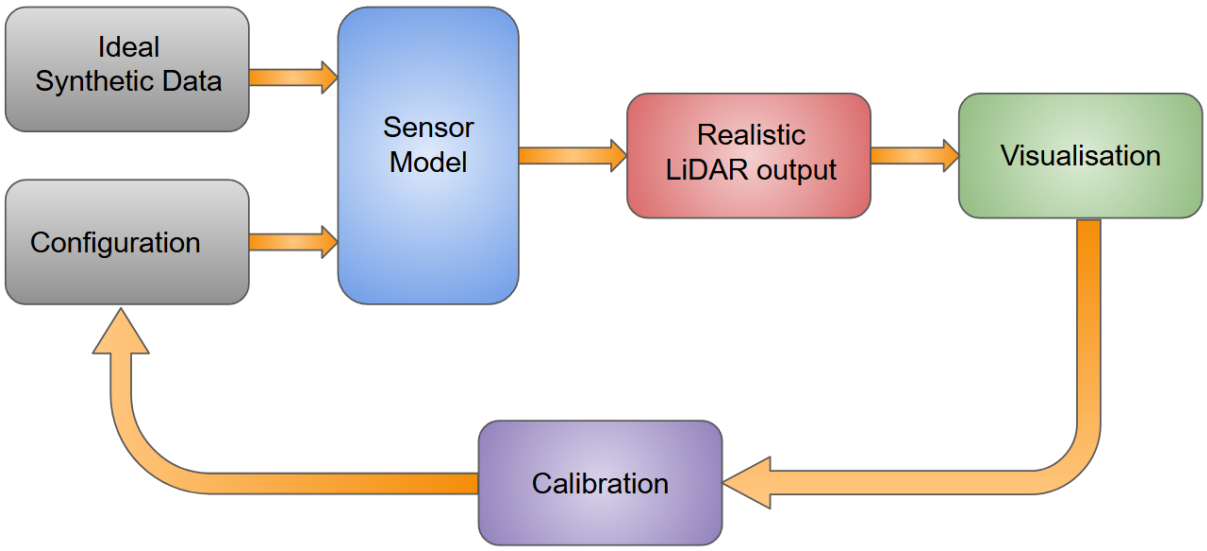

Figure 3.3: Conceptual flowchart

The process begins with the generation of Ideal Synthetic Data, a point cloud or spatial representation based on ground truth conditions. Alongside this, a set of configuration parameters such as blooming, noise statistics, ghost particles are provided to the Sensor Model. This model transforms the synthetic input into a more realistic representation by applying various physical and sensor-specific effects, resulting in the Realistic LiDAR Output.

This output is passed to the Visualization Module, where it is analyzed using tools such as Cloud Compare or histograms to assess its similarity to real-world sensor data. Based on this comparison, the Calibration Module adjusts the configuration parameters and feeds them back into the system, forming a closed feedback loop. The calibration tool iteratively updates parameters to reduce discrepancies, thereby aligning the sensor model more closely with actual LiDAR behavior.

While both ”Sensor Model with CarMaker” and ”Interactive Calibration Workflow” are considered during development, the focus of this thesis lies in the design and enhancement of the latter, which offers higher responsiveness, reduced manual intervention, and improved accuracy.

3.3 Simulation of LiDAR Artefacts

One of the key capabilities of the proposed solution is the ability to validate the simulation of LiDAR-specific artefacts that commonly impact sensor performance. These include blooming effects, ghost objects, sensor noise and other signal anomalies. Accurately modeling these artefacts in the synthetic input data is crucial to ensuring that the simulated sensor output closely resembles what would be observed in real-world scenarios.

By providing a flexible environment to calibrate or validate the simulation of these artefacts under controlled conditions, the tool allows developers to study their impact and iteratively adjust simulation parameters. This helps in identifying and mitigating potential sources of error in the LiDAR signal, improving the robustness and realism of the simulated data. Moreover, the ability to visualize these artefacts through the closed-loop calibration tool supports more targeted and effective sensor tuning.

3.4 Calibration Methodology

The core strength of the closed-loop system lies in its ability to facilitate on-the-fly calibration. Since the input is controlled and generated directly by the calibration tool, any discrepancies between the simulated and real-world data can be immediately observed and addressed. This swiftness allows for a more responsive calibration workflow, where developers can iteratively refine the input parameters and immediately assess their impact on the output.

In this context, histogram-based analysis becomes a powerful tool. By comparing histograms of real and simulated point clouds, particularly in terms of intensity, number of points, and distribution, differences can be quantified and used to guide calibration adjustments. This comparison helps ensure that the simulation not only replicates the geometry of the scene but also captures the nuanced behavior of the sensor under specific conditions. The integration of visualization tools and analytical metrics within the closed-loop framework further accelerates the calibration process, leading to a more accurate alignment between simulated and real sensor data.

3.5 Expected Benefits and Outcomes

The adoption of a closed-loop calibration system offers multiple advantages for LiDAR simulation and sensor development. Most notably, it enables real-time tuning, allowing for quicker convergence between simulated and real data. The ability to simulate artefacts such as blooming and ghost points supports a more comprehensive calibration process, which ultimately improves the reliability and accuracy of the sensor model.

By reducing reliance on time-intensive open-loop simulations and offline analysis, the closed-loop system fosters a more efficient development cycle. This translates into reduced costs, faster iteration, and enhanced performance of LiDAR systems used in autonomous vehicles. Through this approach, the simulation and calibration tool contributes to the creation of safer and more capable autonomous driving technologies.

3.6 Importance of Analytical Features in Calibration

To effectively tune the LiDAR sensor model, mere visualization of output data is not sufficient. What truly drives the calibration process are the specific features extracted

from the output data. Whether examining histograms or raw point clouds, attributes such as Peak, Area, Width, and the spatial distribution of points (via XYZ coordinates) provide critical information about how closely the simulated data aligns with real-world sensor behavior. These features serve as objective metrics to quantify and compare the intensity profiles, reflectivity behavior, and structural outlines of objects detected by the sensor. By translating visual information into measurable parameters, the calibration process becomes more systematic and data-driven.

The following sections delve deeper into each of these key attributes, explaining their significance, how they are derived, and their role in refining the fidelity of the simulated LiDAR output.

3.6.1 Peak in LiDAR Signal Processing

A LiDAR sensor emits a laser pulse and measures the time it takes for the pulse to return after reflecting off a surface. This reflected signal is recorded as a time-resolved signal, where the power of the received light is plotted against time. The laser pulse has a finite width, typically around 5 nanoseconds (ns). Since light travels at approximately 30 cm per nanosecond, an object that generates a reflection appearing 1 ns later is approximately 15 cm away, considering that the light has traveled to the object and back. The laser pulse is not a perfect impulse but has a specific shape due to the natural spread of photon emission. This results in a distributed signal rather than a sharply defined peak.

When the reflected signal reaches the sensor, it is detected by a SPAD array. SPADs are highly sensitive photodetectors capable of detecting single photons and generating an electrical signal upon activation. However, the received signal undergoes broadening due to the dead time characteristics of the SPAD array and electronic processing. The reflected signal is a combination of the actual return from the object and background noise. To extract meaningful information, a thresholding technique is applied. Any signal component above the defined threshold is considered a valid detection (peak), whereas the components below the threshold are classified as noise and discarded.

The peak amplitude is a key metric used to determine the strength of the reflected signal. In the case of Valeo's LiDAR technology, two variants are employed: SMART and SLIM. The SMART variant utilizes an array of 3×9 SPADs (three vertical and nine horizontal), while the SLIM variant employs a denser arrangement of 3×21 SPADs. Each SPAD operates in a binary manner, meaning it can either be in an ON or OFF state depending on whether it detects a photon. If a photon strikes a SPAD, it remains activated for a duration of 9 ns. When multiple photons are received within a short time frame, they contribute to an increased signal amplitude, forming the characteristic peak shape in the time-resolved signal.

To improve signal robustness, the LiDAR system does not rely on a single laser pulse but instead fires five consecutive pulses. The detected signals from these pulses are

then summed to form the final signal. In the SMART variant, the peak amplitude is capped at 135, which corresponds to all 27 SPADs (3 × 9) being activated across all five pulses. The SLIM variant, having a larger SPAD array, allows for a higher peak value and enhanced detection quality due to its increased sensitivity and dynamic range.

Unlike conventional cameras that illuminate and capture an entire two-dimensional FOV, Valeo LiDAR sensors operate using a vertical line illumination system. The emitted laser illuminates a narrow vertical strip in the environment, and the sensor itself consists of a vertically aligned SPAD array. This method significantly reduces cost and improves efficiency compared to traditional LiDAR approaches that require scanning across the entire FoV. In the resulting point cloud, each point is not derived from a single SPAD but rather from a collection of SPADs that contribute to the overall measurement.

The peak amplitude plays a critical role in object detection, classification, and material differentiation within the LiDAR framework. A higher peak value indicates a stronger reflection, which typically occurs for highly reflective surfaces such as traffic signs, while lower peak values are associated with darker or more absorbent materials. The precise calibration and interpretation of peak values are essential for ensuring the accuracy and reliability of LiDAR-based perception systems.

3.6.2 Width in LiDAR Signal Processing

The width of a LiDAR signal represents the temporal span of the detected return pulse above the defined detection threshold. Once the emitted laser pulse reflects off a surface, the returning signal is processed in a time-resolved manner, with time on the x-axis and received laser power on the y-axis. The width is measured as the time interval between the two points where the signal crosses the threshold, marking the beginning and end of the detected pulse.

The width of the signal is typically measured in nanoseconds (ns), with one tick on the time axis corresponding to 1 ns, which defines the temporal resolution of the LiDAR system. Since light travels approximately 30 cm per ns, a width measurement in nanoseconds can provide insights into the characteristics of the reflecting surface. In most cases, the detected width is around 15 ns, though this value can vary depending on the reflectivity and shape of the object.

The primary factors influencing width include:

- **Surface Material and Reflectivity** – Highly reflective surfaces tend to produce broader return signals, while darker or absorbent surfaces result in narrower signals.
- **Incident Angle** – If the laser strikes an object at a steep angle, the returned signal may be more spread out, increasing the width.
- **LiDAR System Characteristics** – The properties of the emitted laser pulse and the response of the SPAD array influence how the width is recorded.

While width provides useful information, it has limited resolution due to the discrete nature of the time-resolved signal. At 15 ns, there are often not enough significant signal variations, making it difficult to extract detailed material properties solely from width. Additionally, width can increase even if the peak is saturated, as more photons contribute to the signal beyond the saturation point. However, because width does not offer fine-grained differentiation between different reflection characteristics, it is often used in conjunction with other parameters particularly area, to analyze LiDAR data more effectively.

3.6.3 Area in LiDAR Signal Processing

The area of a LiDAR signal above the threshold is not exactly linear to the total energy but is often used as a measure of the return energy as it follows the energy of the received pulse closer than peak or width values. Mathematically, it can be considered as the integral of the time-resolved signal over the duration in which the signal remains above the threshold. Unlike the peak value, which only measures the maximum amplitude, the area accounts for the entire signal shape, making it a more reliable indicator of the energy in the reflected pulse.

Area is particularly important in point cloud analysis because it provides a more stable and detailed representation of the returned signal compared to the peak value. One of the key limitations of peak amplitude is that it can saturate quickly—for instance, when reflecting off highly reflective objects such as traffic signs. In such cases, even though the true reflection might be much stronger, the peak value is capped by the hardware (e.g., at 135 for SMART LiDAR). However, since width continues to increase when peak saturates, the area (which is a function of both peak and width) does not saturate as quickly.

This makes area a preferred metric for LiDAR analysis over peak and width, as it provides a more continuous and less saturation-prone measure of signal strength. The area is particularly useful for:

- **Distinguishing Material Properties** – Different surfaces reflect light differently, and area helps differentiate between reflective and absorptive materials more accurately than peak.
- **Handling Peak Saturation** – When the peak value reaches its limit, area continues to provide meaningful information, ensuring that strong reflections are still quantifiable.
- **Improving Object Detection and Classification** – Since area captures the total received energy rather than just the peak intensity, it allows for better differentiation between objects with similar peak values but different reflection profiles.

In summary, while peak and width provide valuable insights, area is the most robust parameter in LiDAR signal processing. It offers a more detailed and less saturation-sensitive measurement, making it the preferred choice for most Point Cloud Data (PCD) analysis applications.

3.6.4 XYZ Coordinates in LiDAR Data

The XYZ coordinates represent the spatial position of each detected point in the LiDAR-generated point cloud. These coordinates define the three-dimensional (3D) structure of the scanned environment, forming the fundamental basis of LiDAR-based perception and mapping systems.

When a LiDAR sensor emits a laser pulse, it measures the ToF, its the time taken for the pulse to travel to an object, reflect back, and be detected by the sensor. This ToF measurement is then used to compute the distance to the object using the equation:

$$d = \frac{c \cdot t}{2}$$

where:

- d is the measured distance,
- c is the speed of light (approximately 3.0×10^8 m/s),
- t is the total round-trip time of the pulse. The factor $\frac{1}{2}$ accounts for the fact that the laser pulse travels to the object and back.

Once the distance is known, the XYZ coordinates are computed based on the sensor's orientation and the scanning mechanism. These coordinates are derived using the following transformations:

$$X = d \cdot \cos(\theta) \cdot \cos(\phi)$$
$$Y = d \cdot \sin(\theta) \cdot \cos(\phi)$$
$$Z = d \cdot \sin(\phi)$$

where:

- d is the computed distance from the sensor to the object,
- θ is the horizontal (azimuth) angle, which depends on the scanning mechanism,
- ϕ is the vertical (elevation) angle, which varies based on the LiDAR sensor's vertical resolution.

3.6.5 LiDAR Scanning Mechanism and Point Cloud Formation

The method of capturing XYZ coordinates depends on the type of LiDAR sensor. In the case of Valeo's Smart and Slim LiDAR systems, the scanning follows a vertical line illumination pattern. Unlike conventional cameras that illuminate an entire FOV, these LiDAR sensors use a structured laser beam to illuminate only a narrow vertical line at each moment and detect the return signal by focusing the return light on a vertical matrix of SPADs. The advantage of this setup is the possibility to use a single rotating mirror leading to a reduced cost of the sensor. The disadvantage lies in the increased

vertical blooming from highly reflective objects.

Each point in the point cloud is formed by aggregating data from multiple SPADs. In the SMART variant, each point is derived from a 3×9 SPAD group, while in the SLIM variant, a larger 3×21 SPAD group is used. This means that each recorded point is not a single measurement, but rather a processed and aggregated signal from multiple SPADs, improving measurement stability.

Since the scanning system follows a vertical-line-by-line illumination, the full 2D scan is constructed by combining multiple vertical slices while moving the sensor or rotating its scanning mechanism. When the sensor completes its full scanning cycle, the result is a dense 3D representation of the environment, composed of thousands to millions of XYZ points.

Significance of the attributes

In both histogram-based and point cloud visualizations, attributes like distance, peak, area, and width play a pivotal role in shaping our understanding of the scene. Distance determines the spatial positioning of objects and is fundamental to depth perception and segmentation. The peak of a return signal reflects the strongest surface interaction, often corresponding to the most reflective or dominant object in the beam's path. Area under the signal curve is the closest value to the total energy returned, helping distinguish between diffuse and strong reflectors, while width indicates the spread or sharpness of the signal, offering insights into surface roughness or multilayered reflections. Together, these attributes form the foundation for accurate feature extraction, object classification, and calibration when comparing real and simulated LiDAR data.

4 Implementation

The goal of this implementation is to develop a calibration tool that enables effective comparison and fine-tuning of simulated LiDAR sensor data against real-world measurements. Accurate calibration is essential to ensure that the simulated sensor behaves as close to reality as possible, particularly for applications in autonomous driving where reliability and precision are critical.

To support this objective, the tool features a modular interface organized into panels, each dedicated to a specific type of analysis. Users can select the type of artefact they wish to simulate or calibrate, such as blooming, noise characteristics, or ghost points, and also choose SLIM or SMART sensor variants to match real-world configurations. The tool allows histogram-based comparisons to be performed directly within the GUI, enabling immediate visual and statistical evaluation of simulation accuracy, as illustrated in Figure 4.1.

Point clouds are generated after applying the selected artefact parameters and are automatically saved to a user-defined location. This user-friendly design ensures flexibility, clarity, and efficiency in the simulation and calibration process, making it easier to isolate specific sensor behaviors and improve the fidelity of the simulation model. An example comparison between real and simulated point cloud data for a vehicle is shown in Figure 4.2.

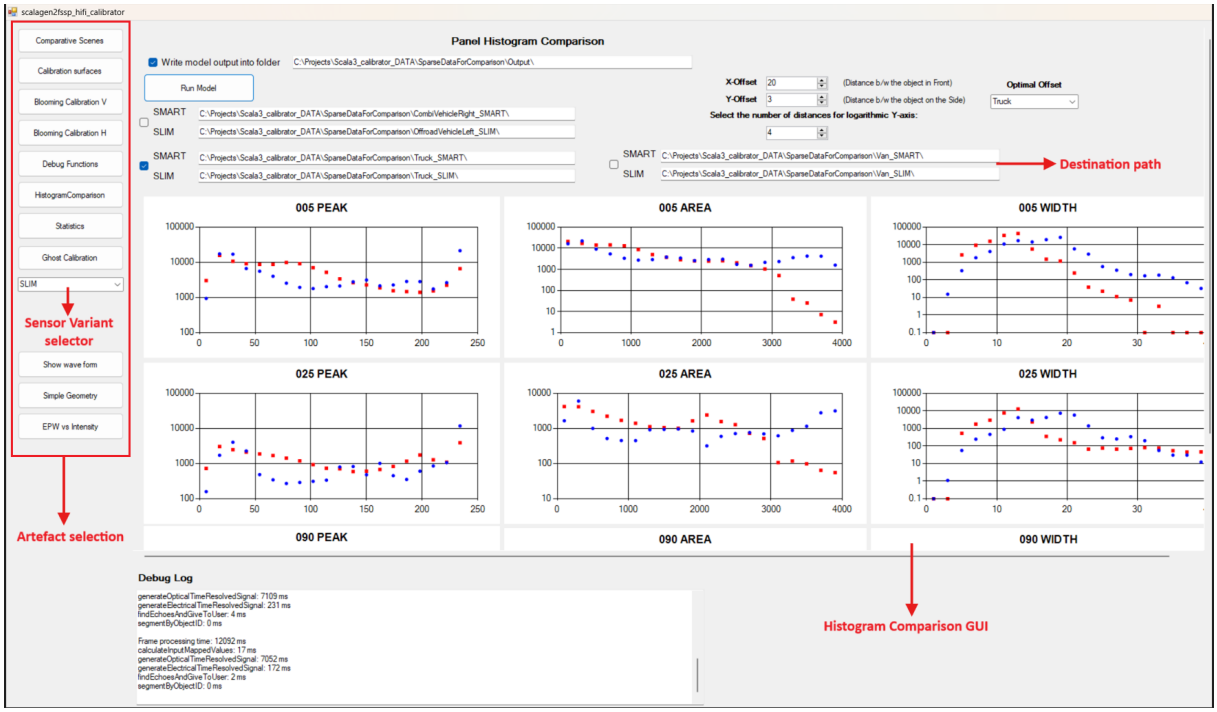

Figure 4.1: Calibration tool GUI

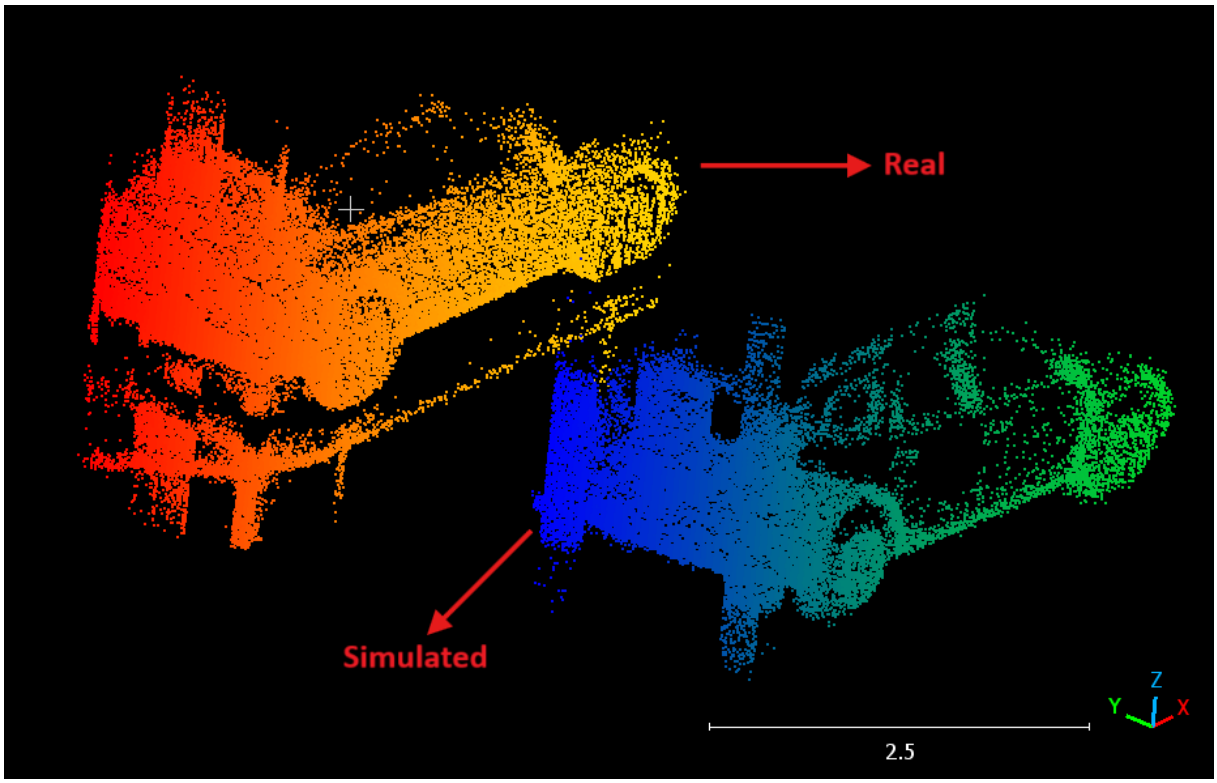

Figure 4.2: Real vs Simulated Point cloud data of a car

4.1 Histogram-Based Metrics for LiDAR Simulation Evaluation

As part of calibrating the LiDAR sensor, histograms were used to better understand how the sensor responds to various objects and surfaces at different distances. To generate the input for the sensor model, an ideal synthetic point cloud is created that closely resembles the shape, size, and position of the real-world object being analyzed in most cases, a vehicle. This synthetic model is placed at the same location as the real vehicle in the scene, ensuring that both real and simulated data can be directly compared under equivalent spatial conditions.

These histogram-based analyses help visualize how point returns are distributed and how dense they are under various conditions. This makes it easier to compare simulated sensor data with actual real-world measurements. To support this, a dedicated "Histogram Comparison" panel was added to the GUI, allowing both visual and numerical analysis of key features such as Peak, Area, and Width of the distribution.

Histogram Parameters and Their Significance

Each histogram generated corresponds to a cross-sectional profile of LiDAR returns for a chosen vehicle type (e.g., Truck, Van, or Car) at different distances. The shape and characteristics of these histograms reveal important calibration cues.

Area is a value that has the closest correlation with the total energy of the reflected laser pulse for a single point, computed as the area under the time-resolved signal above the threshold. This metric has more dynamic range than peak amplitude and helps evaluate reflection strength. When aggregated across a set of points (e.g., a vehicle or traffic sign), it can indicate energy loss or scattering effects such as ghosting.

Peak represents the maximum amplitude of the time-resolved signal above a defined threshold for a single point. It indicates the strength of the reflected laser signal and can be affected by surface reflectivity, angle of incidence, blooming, and the number of active SPADs. Peak values are capped (e.g., 135 in SMART), and therefore may saturate in strong reflections.

Width measures the duration (in nanoseconds) for which the time-resolved signal remains above the detection threshold for a single point. It reflects how stretched or blurred the return signal is, and can indicate effects such as blooming, multipath interference, or increased incidence angle. Unlike peak, width does not saturate quickly, making it useful for analyzing signal quality in extreme conditions.

Graphical User Interface for Histogram Comparison

To make the process of comparing real and simulated LiDAR data more accessible and interactive, a dedicated GUI was developed as part of this work. This GUI is designed to help users visualize important statistical features of LiDAR point clouds and evaluate the alignment between real-world measurements and simulation results.

Choosing the Vehicle Type

The GUI allows users to begin by selecting the type of the target vehicle: Car, Truck, or Van along with the corresponding sensor variant, which can be either SMART or SLIM. These selections determine which data is loaded for the histogram analysis and point cloud comparison. As shown in Figure 4.3, the dataset selection interface is displayed.

Figure 4.3: Dataset selection interface

Merging Real and Simulated Point Clouds

Another useful feature is the option to generate a combined PCD file. This feature places the real and simulated point clouds of the selected vehicle side by side into one PCD file, allowing for easy visual comparison. This helps quickly spot any differences in shape, density, or overall distribution. A detailed explanation of this process is provided in Section 4.1.1.
To allow more flexibility, the GUI includes controls to adjust the alignment of these point clouds:

- X-offset lets the user shift the vehicle forward or backward
- Y-offset adjusts the spacing between the real and simulated point clouds

As shown in Figure 4.4, the alignment offset controls are displayed. These settings help fine-tune the layout of the comparison to better observe overlaps or mismatches.

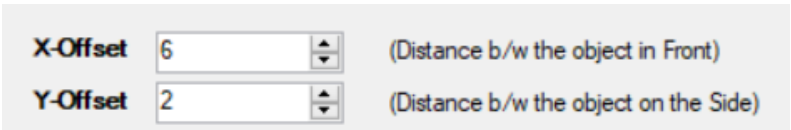

Figure 4.4: Offset alignment

Suggesting Optimal Offsets

To simplify the setup further, the GUI also includes a drop-down menu labeled ”Optimal Offset”. Based on the selected vehicle type, this feature suggests a set of X and Y offset values that provide a good visual alignment between the real and simulated point clouds. This helps users quickly get started without manually adjusting the offsets every time as shown in Figure 4.5.

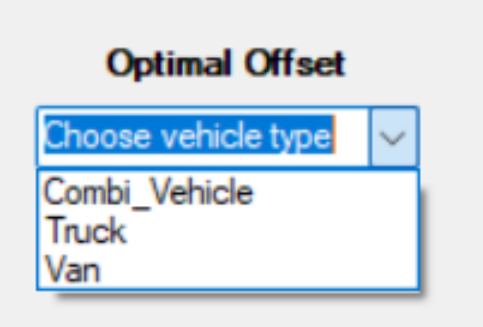

Figure 4.5: Optimal offset interface

Histogram Comparison Panel
A major feature of the GUI is the Histogram Comparison panel as seen in Figure 4.6. When the user clicks the ”Run Model” button, the application plots histograms for the parameters Peak, Area and Width.

These histograms help to visualize how the sensor responds to different vehicle types at different distances. While the GUI displays results for three key distances to keep the interface clean and responsive, it still generates and saves an image covering all four distances in the ”Output” folder as seen in Figure 4.7. This ensures that a full comparison is available for further analysis or reporting.

To ensure that variations across the full dynamic range are clearly visible, all histograms are plotted using a logarithmic scale. This enhances the readability of both subtle and significant differences in the data.

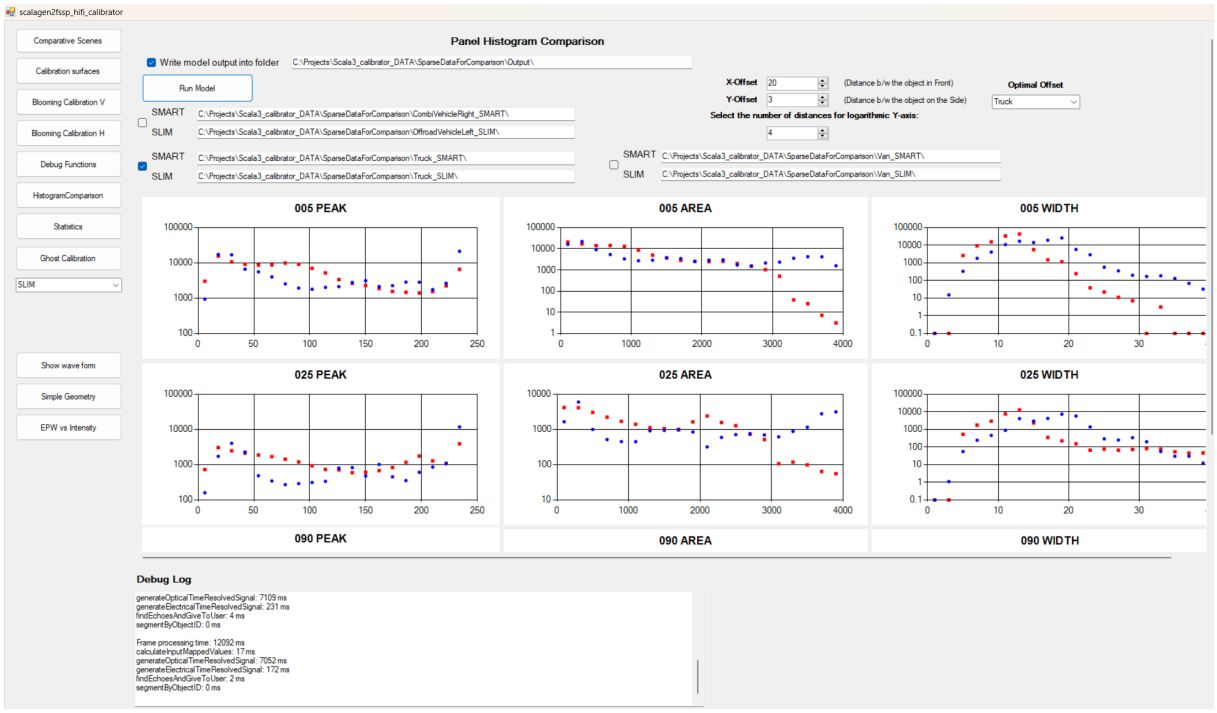

Figure 4.6: Histogram comparison GUI

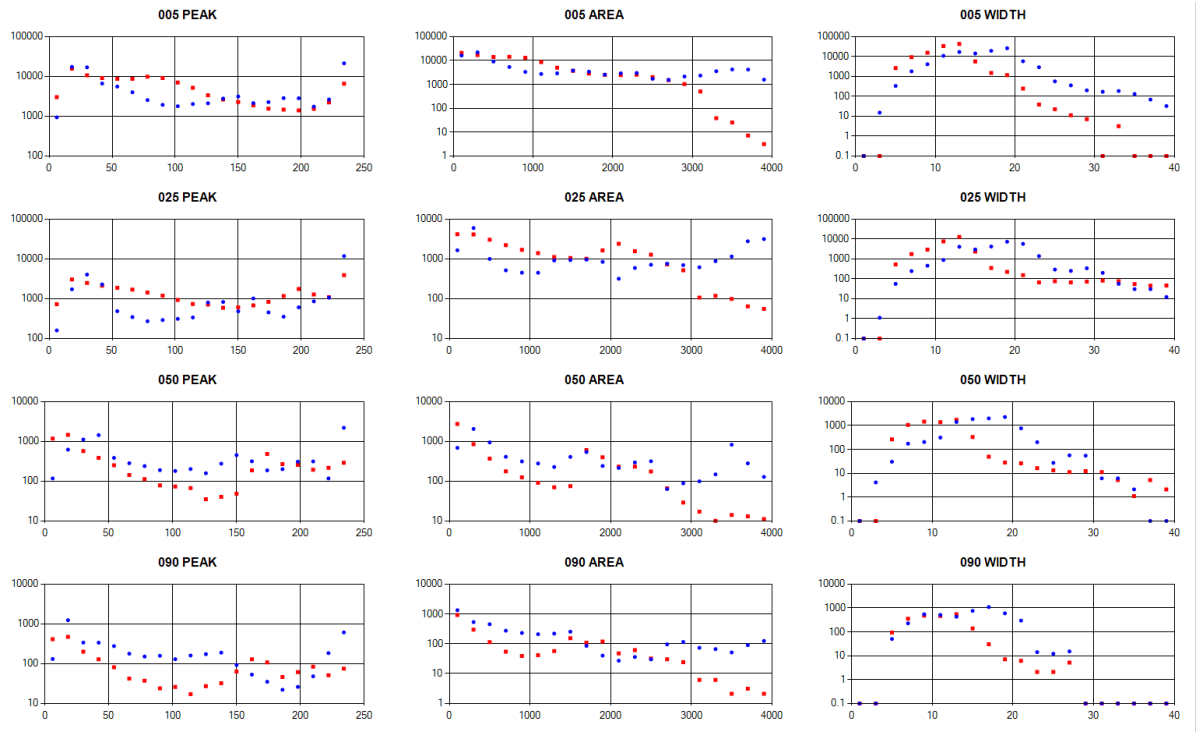

Figure 4.7: Saved histograms of four distance values

4.1.1 Parallel Visualization of Real and Simulated LiDAR Data for Comparison

As part of this thesis, a task was undertaken to merge real and simulated PCD into a unified file. The primary objective of this task was to create a single PCD file that combines real and simulated point cloud data such that the real data would be positioned on the left-hand side and the simulated data on the right-hand side, with a defined offset separating the two. This side-by-side visualization provides clear visual and quantitative comparison between the datasets, highlighting areas requiring calibration improvements and helping bridge the gap between simulation and reality.

Data Preparation and Merging Strategy

The task of merging real and simulated PCD required a systematic approach to ensure proper alignment and visualization. The implementation was designed to iterate through multiple files of real data, each representing measurements at different distances (10m, 15m, 50m, etc.). For each file, the corresponding simulated points were generated and aligned accordingly. This data was then merged and visualized side by side in one PCD file.

As shown in Figure 4.8, the side view of the merged Point Cloud Data provides a clear visual comparison. The top view is illustrated in Figure 4.9, which further highlights the alignment of the two datasets.

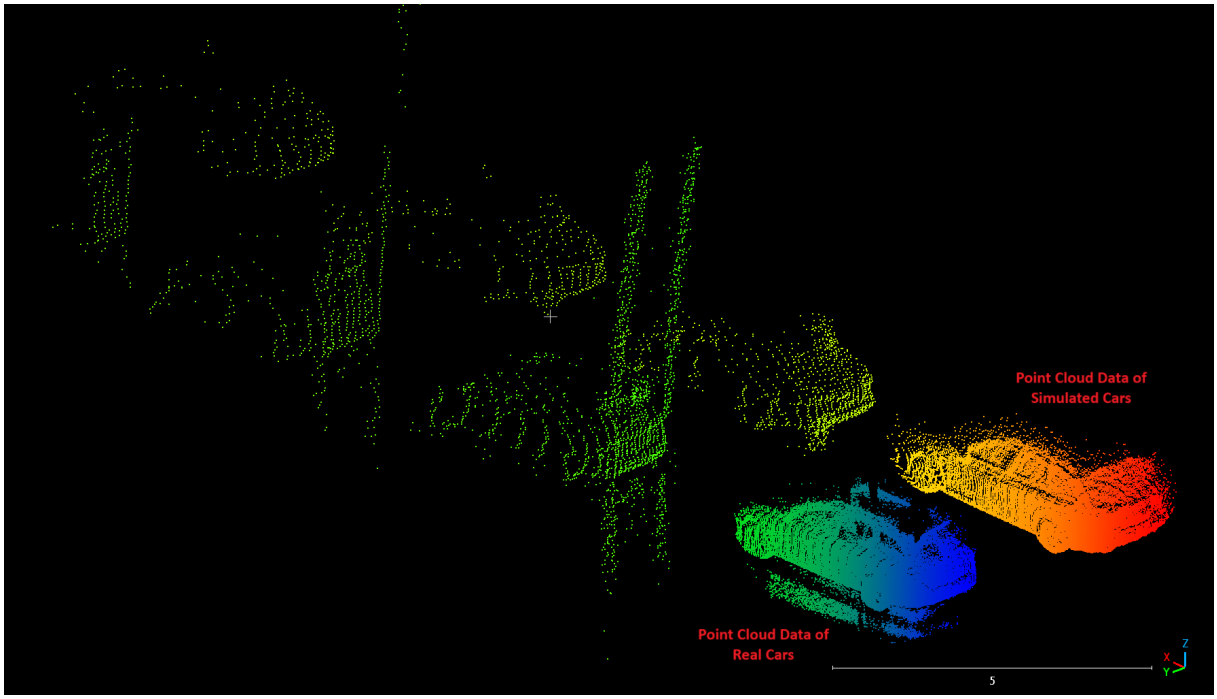

Figure 4.8: Side view of merged Point Cloud Data

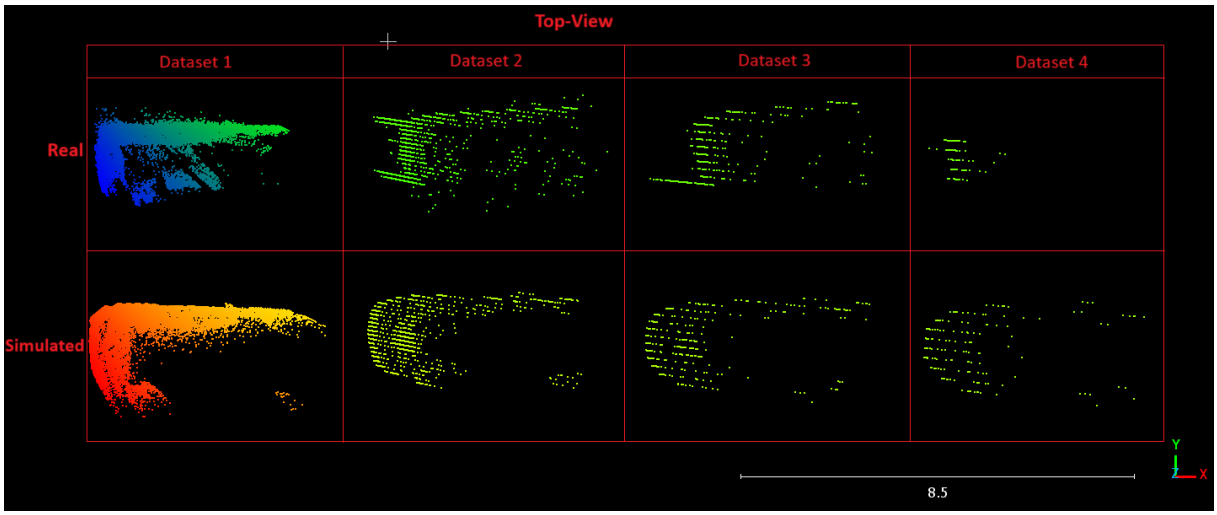

Figure 4.9: Top view of merged Point Cloud Data

Input Data Description

The task involved two primary inputs:

Real Point Cloud Data: Captured using an actual LiDAR sensor, this dataset represents the real-world data with only the points of the vehicle at a specific distance and everything else removed. It includes attributes such as X, Y, Z coordinates and intensity values, with different files placed at varying distances (10m, 15m, 50m, etc.).

Simulated Point Cloud Data: Generated using the simulation framework, this dataset models the expected output of the LiDAR under similar conditions. The simulated points were processed and prepared for comparison with the real data.

Implementation Strategy

Centroid Calculation: For each file in both real and simulated datasets, the x and y coordinates of all points were summed and divided by the total number of points. This step determined the geometric center (centroid) of each dataset:

$$x_{\text{center}} = \frac{\sum x}{\text{total points}}$$

$$y_{\text{center}} = \frac{\sum y}{\text{total points}}$$

The centroids helped center the data, ensuring consistent alignment across datasets.

Offset Application: Once the centroids were calculated, the data points were translated such that the centroid of each snippet was shifted to a predefined position.

- **Real Data:** Positioned on the left-hand side with an incremental offset along the X-axis for each snippet.
- **Simulated Data:** Positioned on the right-hand side with a similar X-axis offset and a mirrored shift along the Y-axis.

Coordinate Adjustment: Each point's position was updated by subtracting the dataset's centroid coordinates and adding the offsets. The formula used was:

$$X_{\text{adjusted}} = X - X_{\text{center}} + i \times \text{XAxisOffset}$$

$$Y_{\text{adjusted}} = Y - Y_{\text{center}} - \text{YAxisOffset}$$

The `XAxisOffset` provided horizontal spacing between snippets, while `YAxisOffset` created separation between real and simulated data.

Data Write-Out: After adjustments, all data points (including attributes like PEAK, AREA, WIDTH, and others) were written to the unified PCD file. This structure ensured side-by-side visualization with the real data appearing on the left and simulated data on the right.

Significance

This merging approach allowed for direct comparison of real-world sensor data against simulated outputs. By aligning the datasets spatially, differences in attributes like intensity, point density, or Blooming became visually apparent, aiding calibration improvements. This methodology also supports quantitative analysis by highlighting misalignments or inconsistencies between datasets.

4.1.2 Comparative Scenes for Scenario-Level Analysis

While the histogram comparisons and parallel visualization discussed in the previous sections focus on individual vehicles analyzing their peak, area, and width characteristics, and visualizing them side by side for alignment, it is equally important to evaluate how well the simulation performs in complete driving scenarios. To address this, the GUI includes another dedicated panel titled ”Comparative Scenes”.

This panel is designed to facilitate the comparison between entire real-world scenes captured by the LiDAR sensor and their corresponding simulated reconstructions. Instead of focusing solely on isolated vehicles, the Comparative Scenes module enables validation of full-scene fidelity, including dynamic elements such as different vehicle types, distances, road markings, terrain variations, and other environmental features.

Scenario Setup and Simulation Input

The first step is to select a real point cloud scene captured during a specific recording session involving traffic. Then, the vehicles, their respective positions and distances from the ego vehicle, as well as terrain characteristics and road markings, are noted from the real data as seen in Figure 4.10.

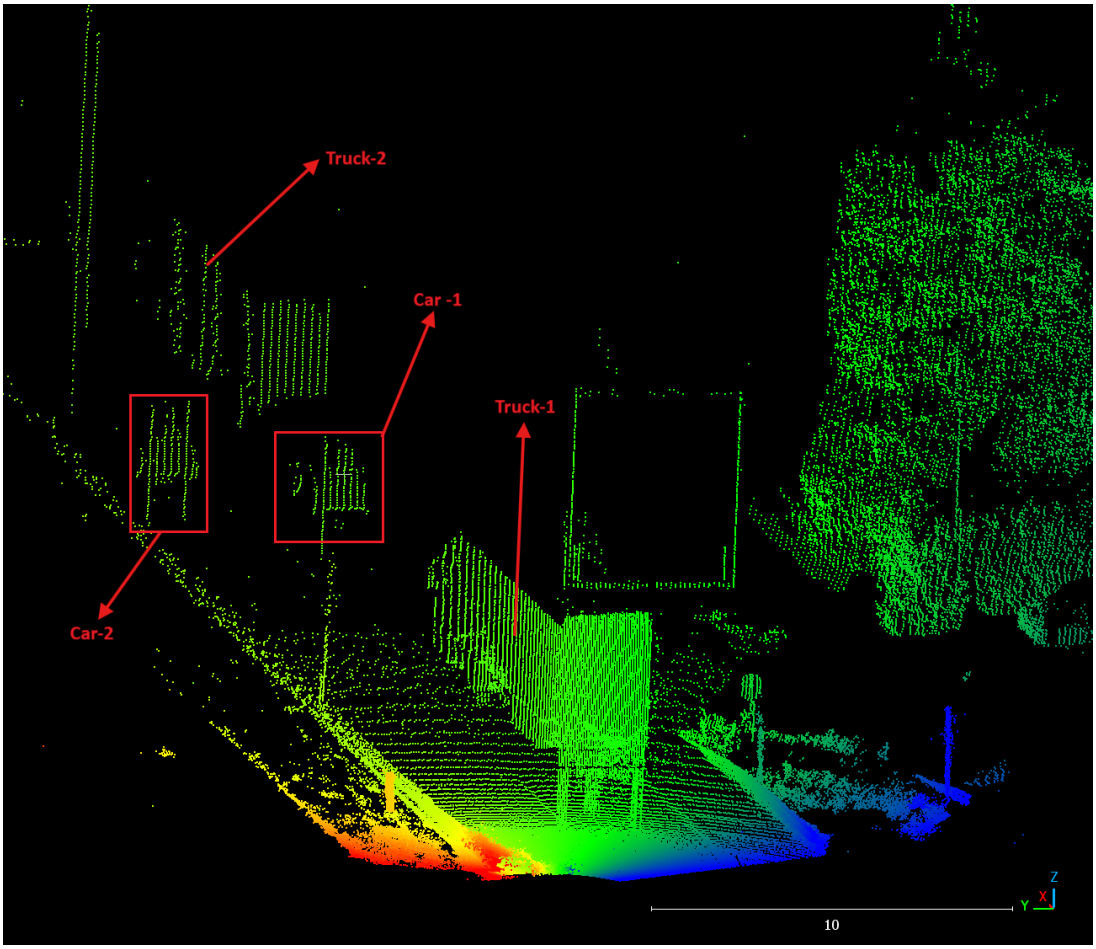

Figure 4.10: Point Cloud Data of a Real scene

Using this information, a realistic simulation of the scene is created in CarMaker as seen in figure 4.11, carefully replicating the identified parameters. One frame of this scene is then recorded using the ideal LiDAR sensor, which usually generates the input for the sensor model. The result is a post-processing (*.postpro) file as seen in figure 4.12, which contains this ideal point cloud with distance, material id, and incidence angle values for each point. This postpro file serves as the input to the Comparative Scenes panel in our tool.

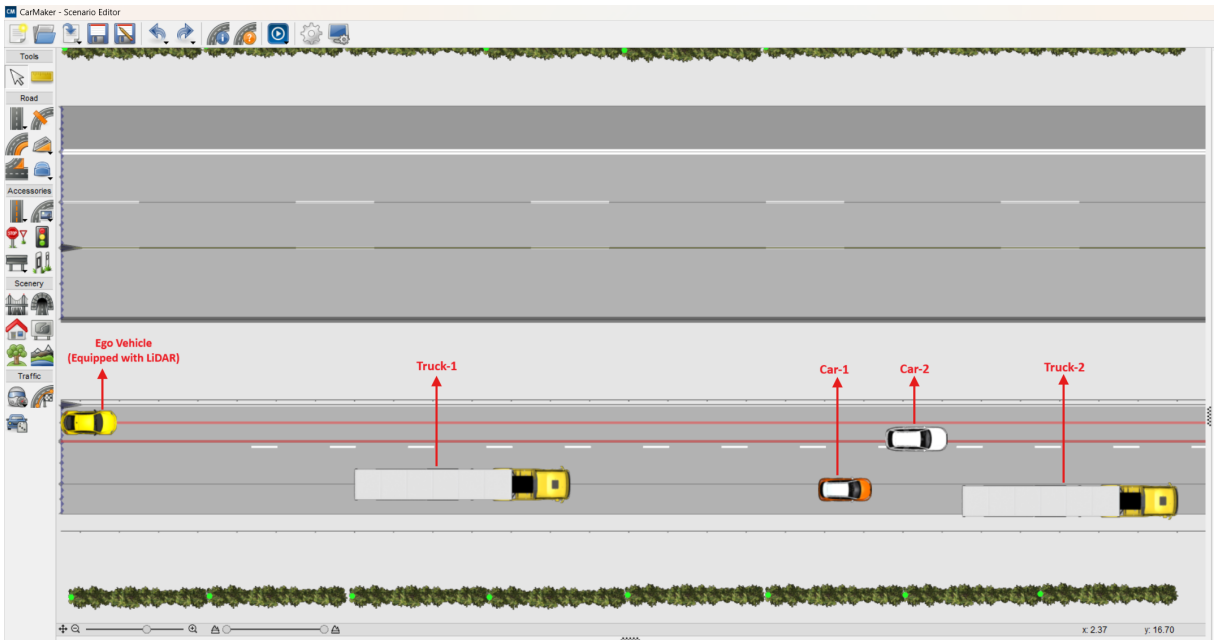

Figure 4.11: CarMaker setup of the Real scene

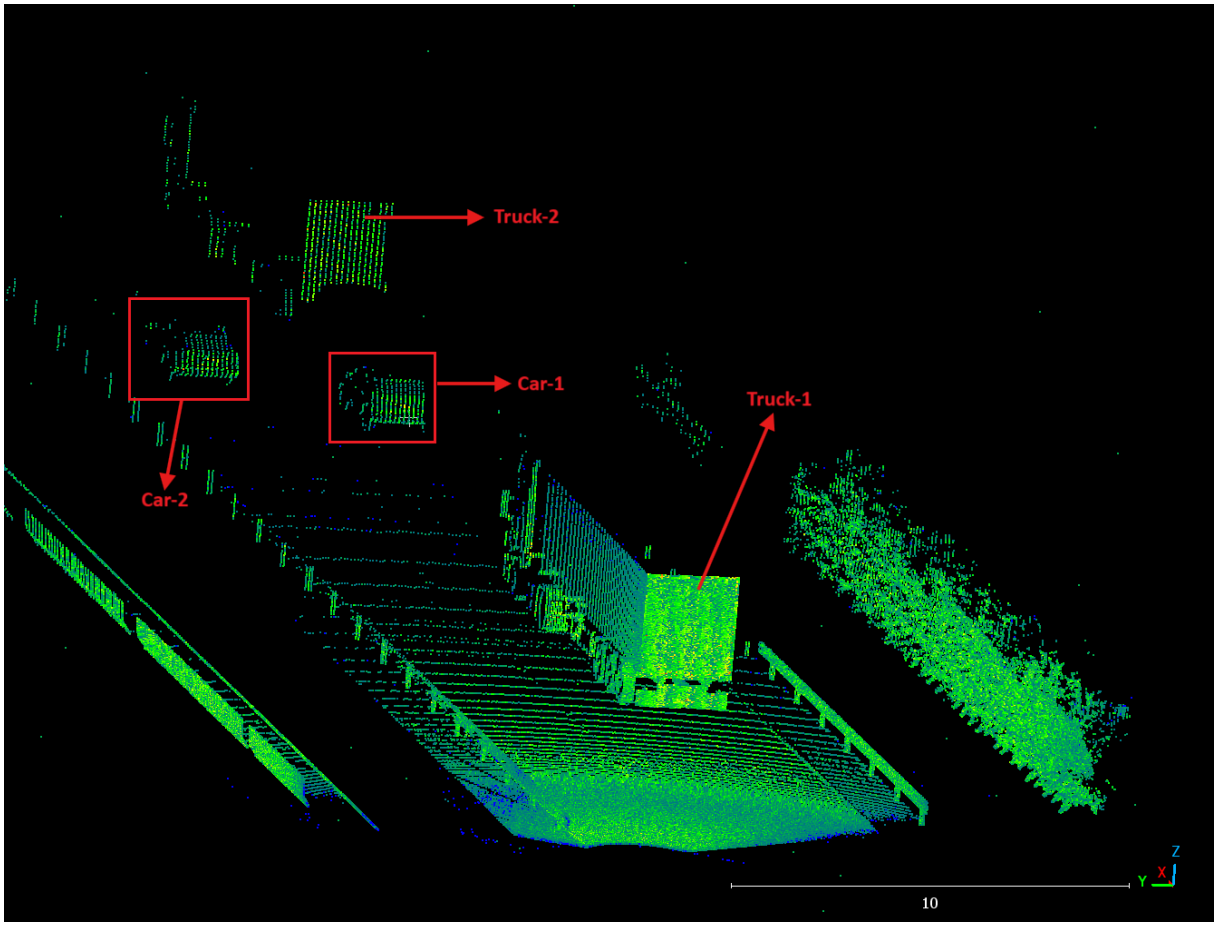

Figure 4.12: CarMaker + Sensor model output of the Real scene

Generation of Simulated Point Clouds with Artefacts

Our main goal is to simulate LiDAR data not just at the geometric or object level, but also including relevant sensor artefacts such as Noise, Blooming effect, Ghost particles and other occlusion and shadowing from terrain or nearby objects.

By feeding the postpro file into the Comparative Scenes pipeline, the system generates a simulated point cloud that aims to closely resemble the real-world scene, not only in content, but also in the imperfections and artefacts that occur naturally during real sensor operation.

This allows for a much more realistic and comprehensive comparison, offering insights into how the simulation environment can replicate real-world sensor behavior in complex scenarios as shown in Figure 4.13. It often serves as a sanity test for the sensor model, quickly revealing problems after the sensor model was updated. It also acts as a valuable tool for identifying areas where sensor models can be further improved, especially when deviations are observed consistently in particular conditions.

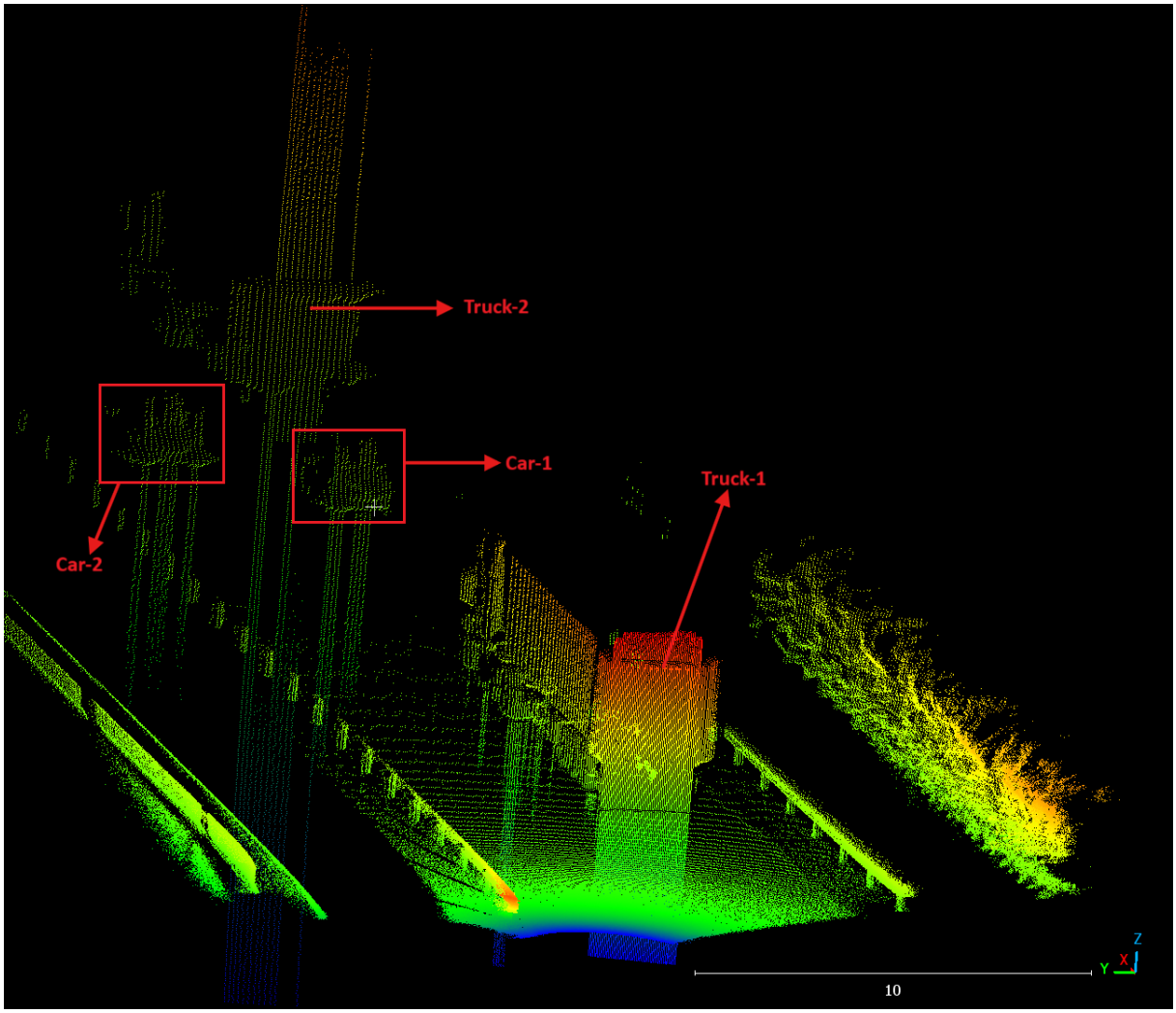

Figure 4.13: Simulated output of the Real scene

Bridging Individual and Scene-Level Comparison

Together with the histogram comparison of object-level features and the side-by-side visualization of isolated vehicles in the Parallel Visualization panel, the Comparative Scenes tool provides a higher-level perspective. It allows the developers to validate not just the individual point clusters, but also assess how well the entire scene matches reality. This helps in making the simulated LiDAR data more accurate and closer to what the sensor would capture in the real world.

4.2 Modeling Noise in LiDAR Data for Realistic Simulation

LiDAR is widely used in autonomous driving, robotics, and mapping applications due to its ability to provide high-resolution spatial information. However, real-world LiDAR data is often affected by various noise factors, leading to inaccuracies in the measured point

cloud visible as statistical variation of distance, peak, area and width values. These inaccuracies arise due to a combination of sensor limitations, environmental influences, and target material properties.

For a simulation to be realistic and reliable, it must not only generate an idealized version of LiDAR data but also incorporate noise characteristics that reflect real-world behavior. This is critical because autonomous systems rely on sensor data for perception, and an inaccurate simulation cannot be used for algorithm development and testing.

4.2.1 Sources of Noise in LiDAR Data

Noise in LiDAR data originates from multiple factors, which can be broadly categorized into the following:

a. Environmental Factors

- **Atmospheric Conditions:** Fog, rain, and dust can scatter or absorb the laser pulses, leading to inaccurate range measurements.
- **Lighting Conditions:** Intense sunlight can cause background illumination noise, affecting the sensor's ability to detect returns.
- **Surface Reflectivity:** Highly reflective surfaces (e.g., road signs) may create oversaturation or blooming effects, while dark surfaces (e.g., asphalt) may result in missing data points.

b. Sensor-Specific Limitations

- **Beam Divergence:** LiDAR beams spread over distance, affecting accuracy at farther ranges.
- **Detection Thresholds:** If a return signal is too weak, it may be discarded, introducing gaps in the point cloud.
- **Intrinsic SPAD operation:** Detection of photons by SPADs is an inherently statistical process leading to noise even under ideal conditions.

c. Motion-Induced Noise

- **Vehicle Vibration:** When mounted on a moving platform (e.g., a car), LiDAR measurements may experience jitter or misalignment.

4.2.2 Simulating Noise in LiDAR Data: Need and Baseline Estimation

LiDAR sensors, both in real-world and simulated environments, exhibit noise that can influence perception and object detection algorithms. However, while real-world LiDAR data inherently contains various types of noise, a simulated LiDAR system initially generates an idealized version of the scene, free from such disturbances. This discrepancy makes simulation results appear cleaner and less realistic than real-world measurements.

To ensure that LiDAR simulation is a useful tool for testing and development, it must accurately replicate the noise characteristics of real sensors. Simulating LiDAR noise serves multiple objectives:

- **Improved Realism:** Introducing noise ensures that the simulated sensor mimics real-world conditions, making it a more reliable testbed for autonomous driving applications.
- **Algorithm Validation:** Object detection and perception algorithms must be robust against noise. Testing within a realistic simulation ensures they generalize well to real-world data.
- **Simulation Calibration:** By comparing simulated and real-world data, we can refine parameters such as blooming effects, range-dependent noise, and reflectivity variations to improve the accuracy of the simulation.

A key step in noise simulation is establishing a baseline noise reference under controlled conditions. In a laboratory environment, we have precise knowledge of:

- Reflectivity values of test panels
- Material properties
- Environmental factors (e.g., lighting, humidity, air clarity)

This controlled setup allows us to isolate and characterize the base noise of the LiDAR sensor, independent of external disturbances. In contrast, real-world LiDAR data contains two distinct types of noise:

- **Base Sensor Noise** – The fundamental noise inherent to the LiDAR system, even under ideal conditions, caused by electronic and optical limitations.
- **Environmental Noise** – Additional disturbances introduced by real-world factors such as varying surface properties, atmospheric interference, and lighting conditions.

By first simulating only the base sensor noise, we establish a reference baseline, which helps in:

- Understanding the intrinsic limitations of the sensor

- Distinguishing unavoidable base noise from real-world environmental effects
- Ensuring that noise models in simulation are correctly implemented before introducing external complexities

This approach ensures that LiDAR noise modeling is accurate and reliable before extending the simulation to account for real-world variations.

4.2.3 Data Collection and Preprocessing

Laboratory Setup for Baseline Noise Estimation

The data used for this analysis was collected from a controlled lab environment at the Valeo site as seen in figure 4.14, specifically designed to study LiDAR sensor performance. The laboratory setup consists of homogenous panels positioned in the shape of a “C” at fixed intervals. These panels are arranged from 5 meters to 80 meters, spaced equidistantly at 5-meter intervals. This structured configuration allows for a controlled and repeatable testing environment, making it ideal for analyzing the inherent noise characteristics of the sensor.

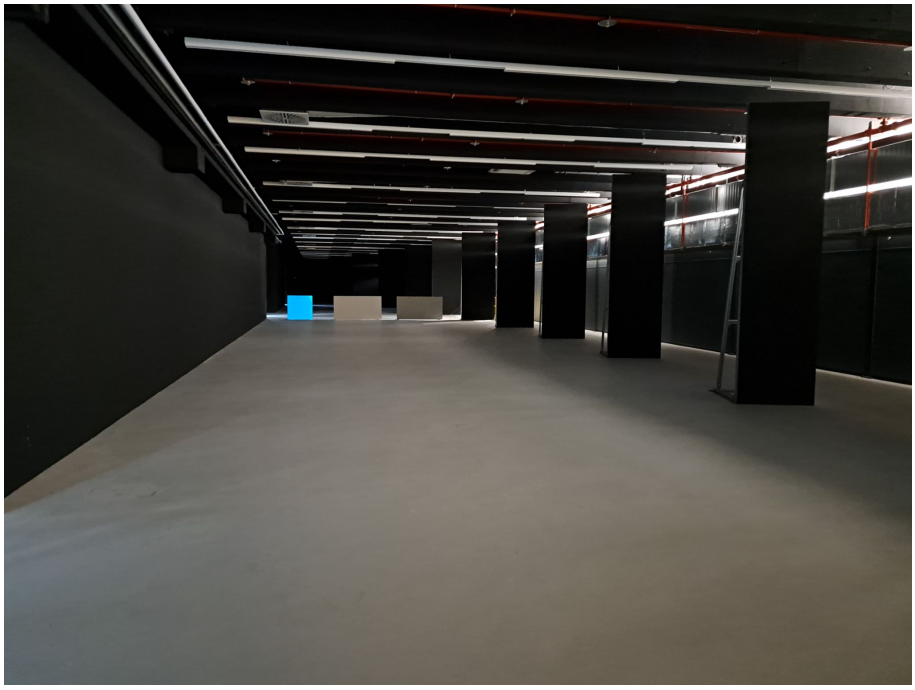

Figure 4.14: Laboratory Setup

Selection and Merging of Point Cloud Data (PCD)

A large dataset was available from the lab experiments, consisting of multiple point cloud files capturing the scene under identical conditions. For statistical robustness, a subset of 30 PCD files (each containing 1 frame) was selected and merged into a single dataset as shown in figure 4.15. The primary motivation for merging these files was to increase the sample size, thereby reducing random fluctuations and improving

the statistical significance of noise analysis. By aggregating multiple scans, a more accurate estimation of LiDAR noise distribution could be achieved.

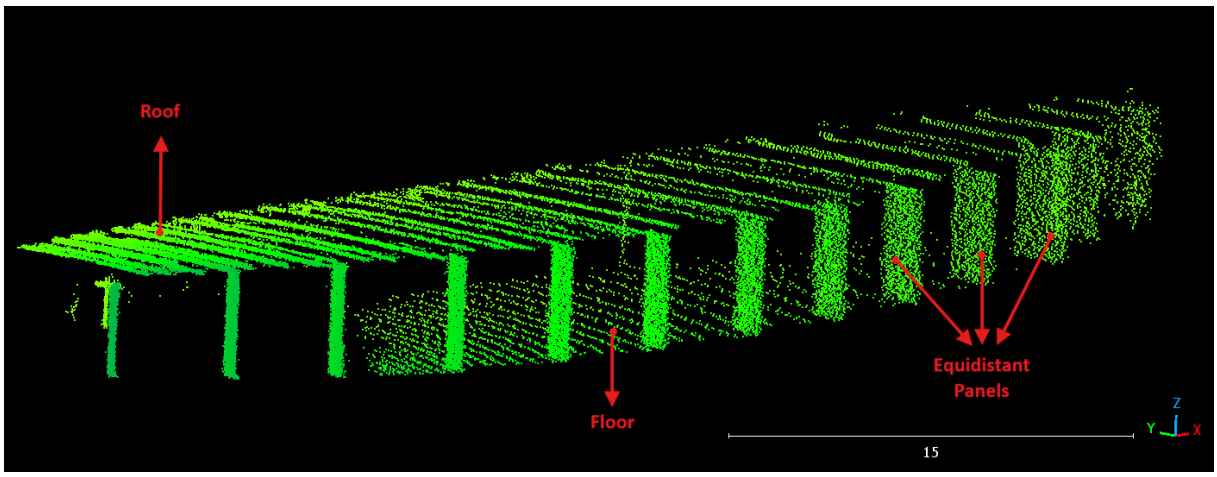

Figure 4.15: Point Cloud Data of Lab

Preprocessing: Trimming Roof and Floor Data

Once the 30 PCD files were merged, the next step involved trimming unwanted regions, specifically the roof and floor points as shown in figure 4.16. In our analysis we only want to consider surfaces with a known reflectivity which are perpendicular to the sensor.

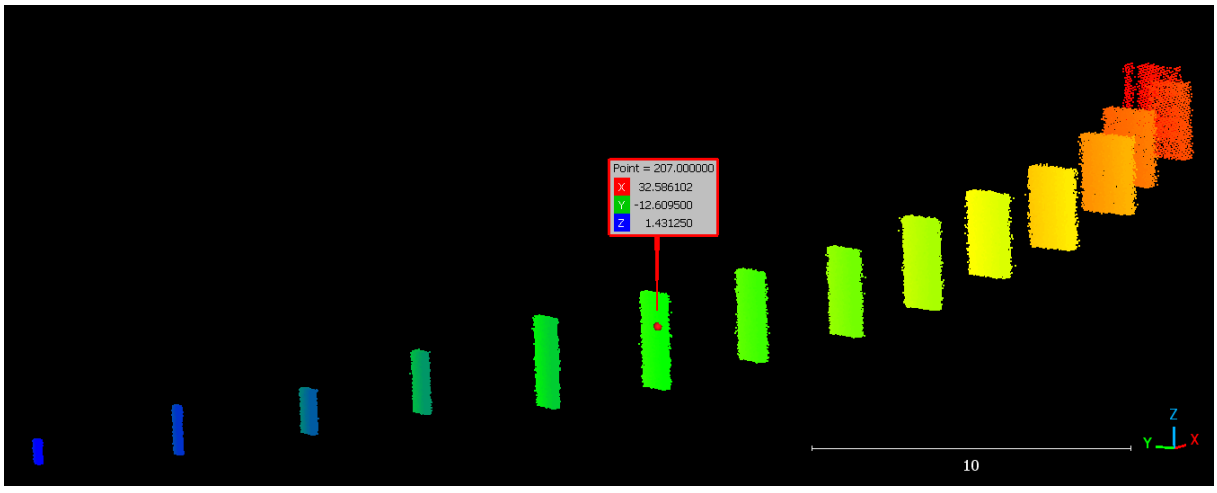

Figure 4.16: Point Cloud Data of Panels only

By removing these extraneous points, the dataset is refined to focus solely on the panels of interest.

Panel-wise Trimming and Segmentation

Each individual panel in the lab setup was then segmented and saved as a separate file. This was done to facilitate panel-wise noise analysis, allowing a detailed study of

how noise characteristics vary with distance. However, before saving, each panel was carefully trimmed to exclude all four borders. This step is necessary because:

- **Edge Effects:** The borders of the panels often exhibit inconsistent LiDAR returns due to partial beam reflection, which can lead to erroneous noise measurements.
- **Sensor Reception Artifacts:** The laser beam may experience only partial reflection at the edge of a surface leading to an incorrect reading of distance and amplitude.
- **Avoiding Contamination from Neighboring Panels:** Border points might include reflections from adjacent objects or scattered noise, which could affect the statistical accuracy of the analysis.

Figures 4.17 and 4.18 show the results of this panel-wise segmentation and trimming process, highlighting the difference between the real and simulated data.

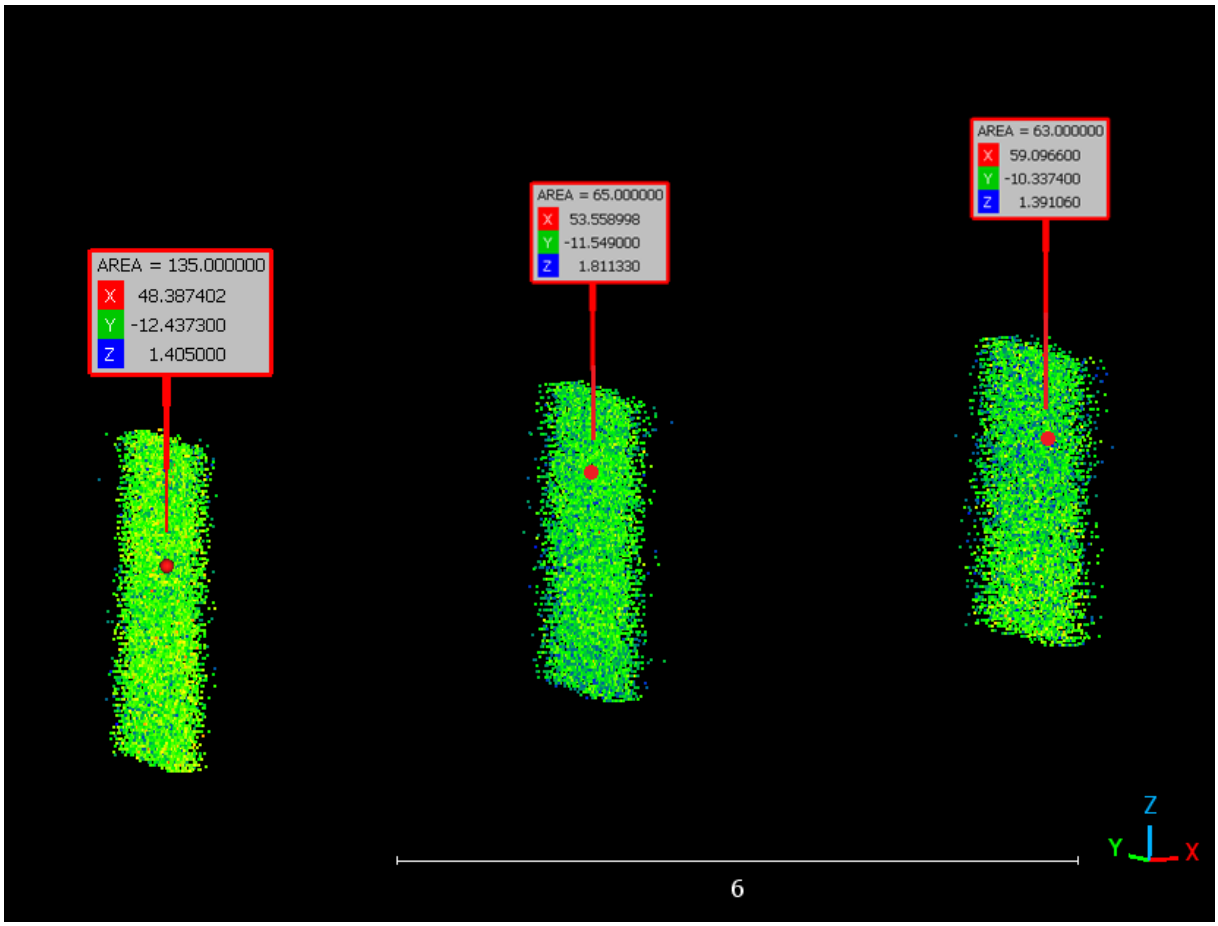

Figure 4.17: Point Cloud Data of Trimmed Panels

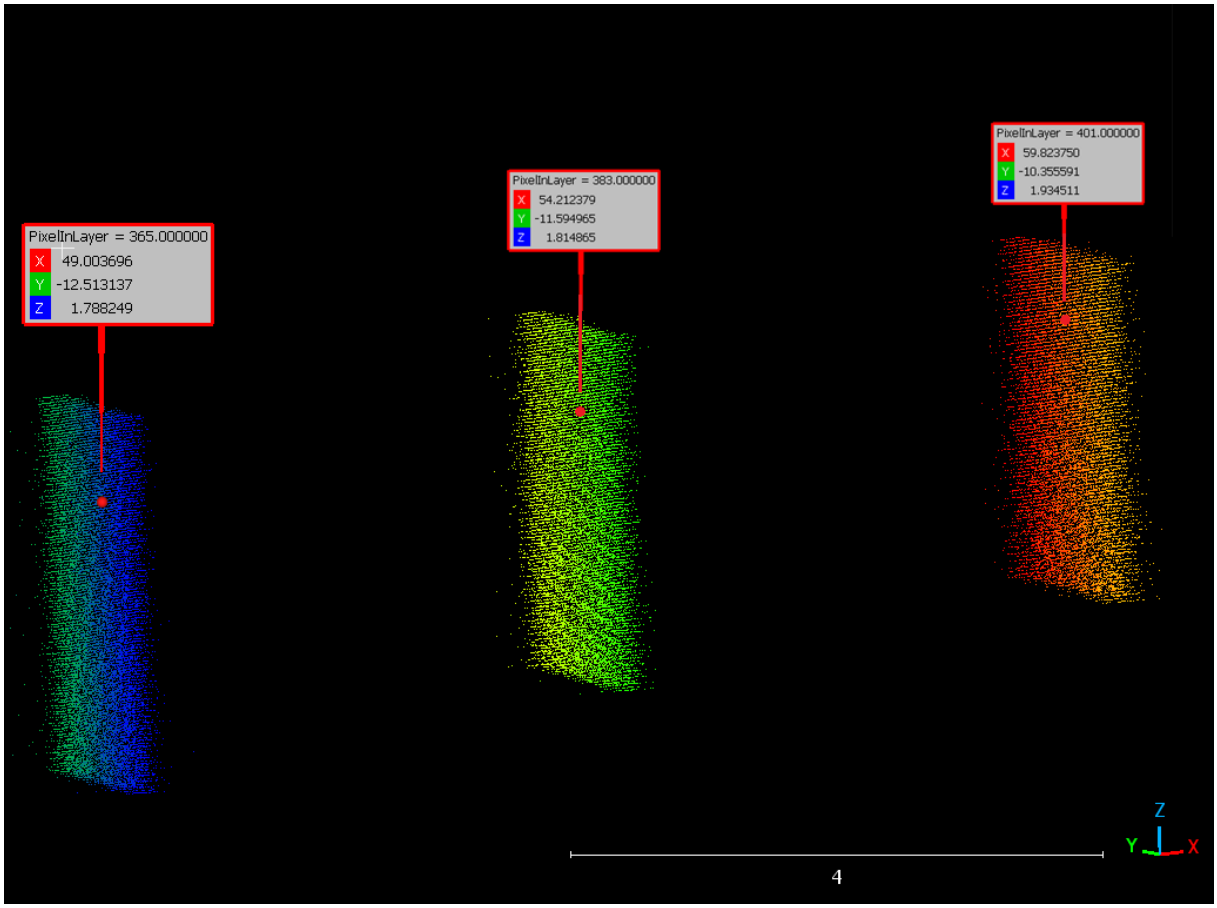

Figure 4.18: Point Cloud Data of Simulated Panels

By isolating only the core region of each panel, the analysis ensures that noise estimation remains consistent and unaffected by external distortions.

4.2.4 Noise Characterization: Area and Distance Histograms and Detection Probability

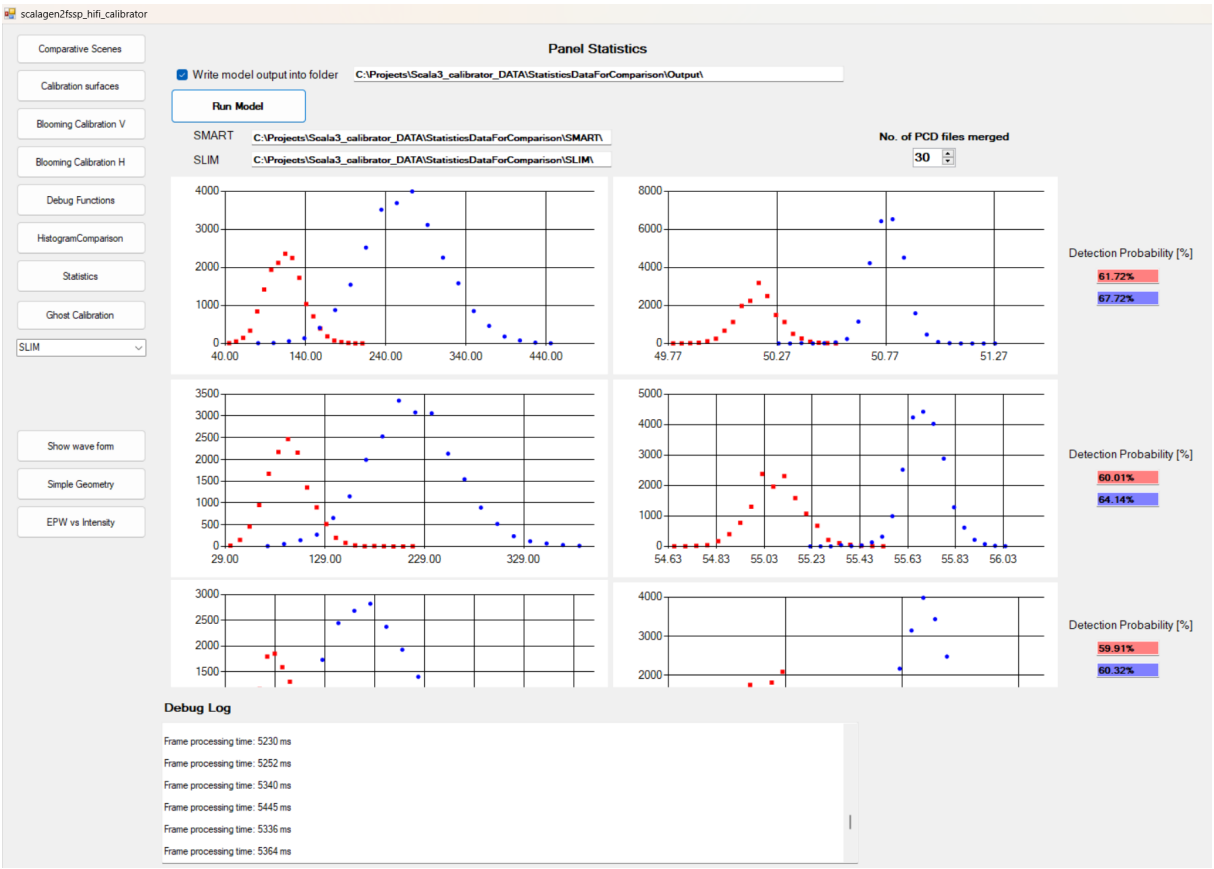

Figure 4.19: GUI of Noise Characteristics panel

Following the preprocessing of LiDAR data, which involves segmentation and refinement of individual panels, the subsequent critical step is to statistically analyze the inherent noise characteristics of the sensor. Among the most informative techniques for this purpose are area histograms and distance histograms. These statistical tools serve to quantify how consistently and accurately the LiDAR sensor captures surface information, particularly under varying conditions and distances.

Importance of Area and Distance Histogram Analysis

Evaluation of Point Distribution Variability

LiDAR sensors do not yield a uniform point distribution across a surface. Instead, the distance and amplitude values of detected points are influenced by the sensor's optical properties. The area histogram reveals systematic noise patterns or inconsistencies in sensor performance.

Investigation of Range-Dependent Noise Behavior

As the distance between the sensor and the target increases, signal attenuation be-

comes prominent. This often results in lower point density and elevated noise levels. The distance histogram captures this trend, highlighting how detection reliability deteriorates with increasing range. This is particularly relevant in autonomous driving applications, where accurate long-range perception is critical for safety.

Identification of Systematic Sensor Errors and Artifacts
Ideally, all panels should yield a uniform distribution of LiDAR points. However, deviations may occur due to hardware limitations. By inspecting both area and distance histograms, systematic errors can be detected, such as consistent loss of returns at specific ranges or spatial clustering anomalies.

Validation of Simulated Sensor Models
Comparative analysis between histograms derived from real-world data and those obtained from simulations is essential for model validation. Discrepancies between these datasets suggest a need to refine simulation parameters.

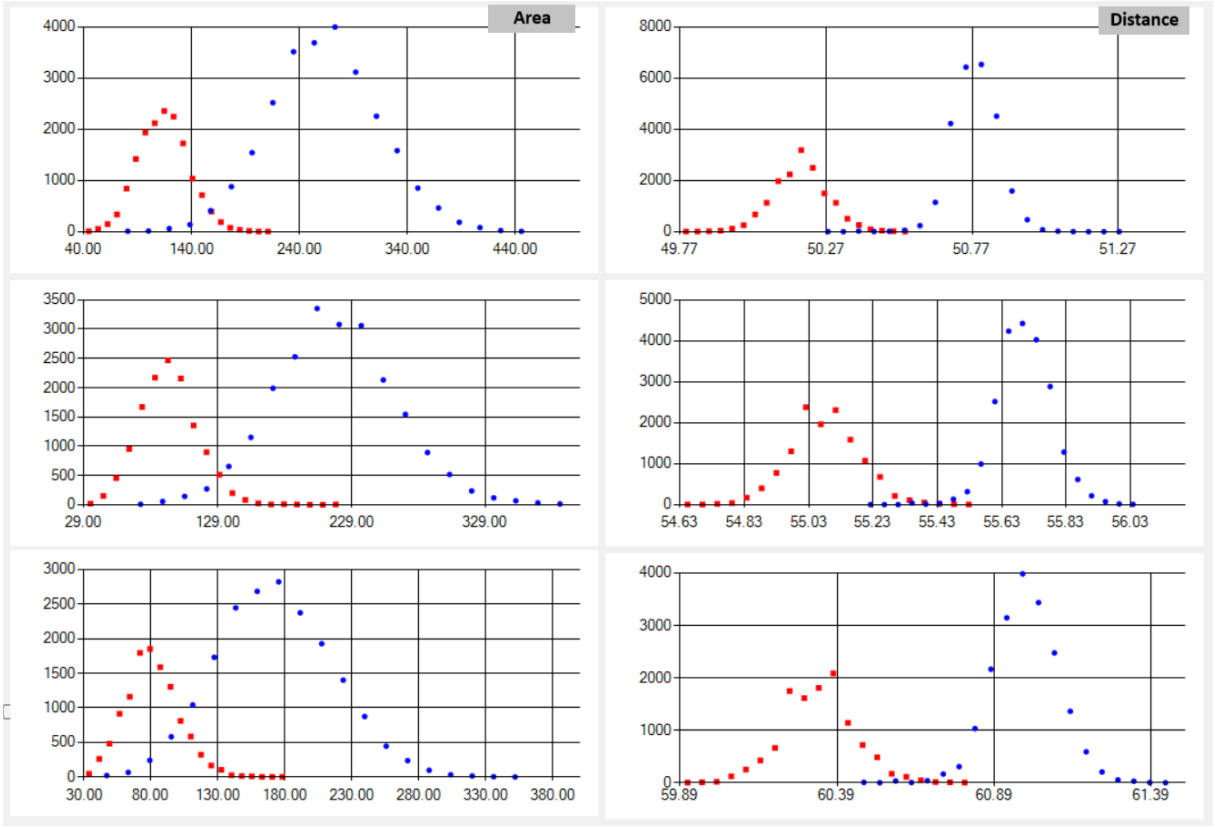

Figure 4.20: Histogram comparison of Area and Distance

Detection Probability and Its Significance

In addition to histograms, a critical component of LiDAR noise characterization is the analysis of detection probability, defined as the likelihood that the sensor successfully detects a surface point at a given distance. This metric offers a practical understanding of the sensor's performance and its limitations under varying operational conditions.

Calibration and Refinement of the Simulation Model
An accurate simulation of LiDAR behavior necessitates close alignment between simulated and real detection probabilities. Disparities between the two datasets signal the need for model refinement. Adjustments to simulation parameters, such as reflectivity functions, noise models, and environmental attenuation can be informed by these comparisons. This calibration process ensures that the simulated sensor environment remains representative and reliable for developing and validating perception algorithms.

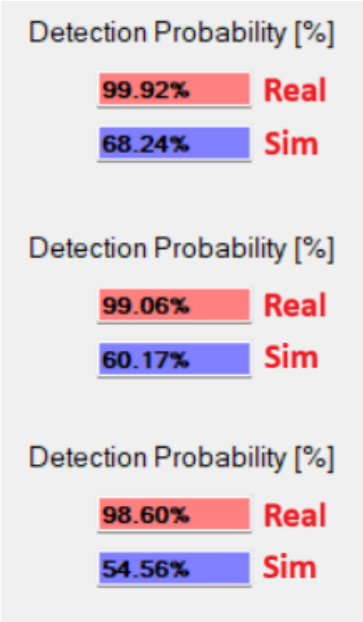

Figure 4.21: Detection Probability of Real and Simulated points

4.2.5 Spatial Misalignment Between Real and Simulated LiDAR Data

During the noise characterization analysis of real and simulated LiDAR data, a consistent spatial discrepancy was observed in the alignment of corresponding panels. As illustrated in Figure 4.22, which shows comparative plots at distances of 50m, 55m, and 60m, the simulated panels are consistently positioned slightly behind the real-world panels.

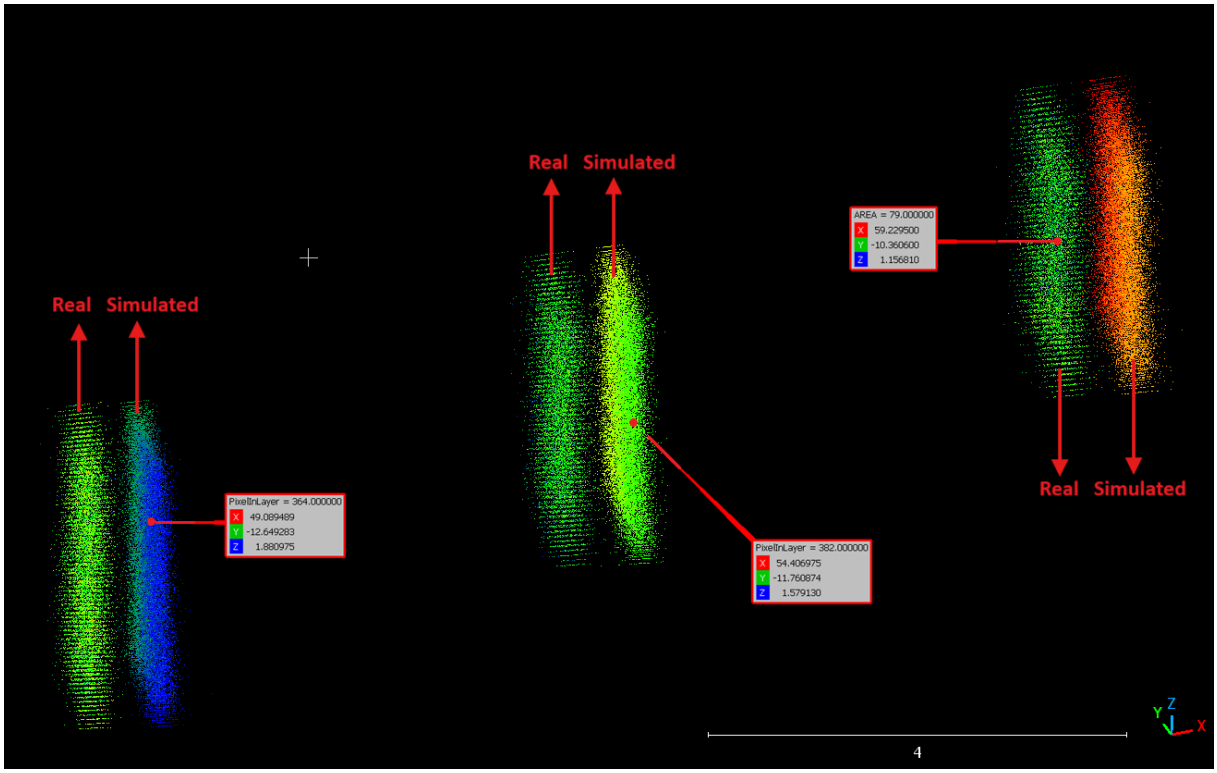

Figure 4.22: Comparison of Real and Simulated panels at 50m, 55m, and 60m

Observation of Misalignment
The histogram analysis of distance and peak distributions of real and simulated panels were expected to coincide. However, a noticeable horizontal shift can be observed in Figure 4.20, with the Simulated point cloud as compared to Real point cloud.

Also, the detection probability analysis observed in Figure 4.21, revealed a stark contrast in point density. While the real panels achieved a detection rate of approximately 99%, the simulated panels exhibited a lower detection probability, around 60%. This divergence, while partially attributable to noise modeling, is also exacerbated by the spatial misalignment between real and simulated data. This discrepancy between the real and the simulated data needs to be addressed in the future.

Walk Error
The primary source of this offset is known as “Walk Error.” Walk error refers to a consistent and systematic inaccuracy in real ToF measurements, which tends to become more pronounced when sensing surfaces at oblique angles or under varying reflective conditions. In simulation environments, this error can introduce a positional bias in the reconstructed point cloud, causing simulated objects to appear slightly behind their real-world counterparts.

This effect arises because the detected signal corresponds to the weighted average of the time-resolved return pulse, which can shift depending on the pulse amplitude and shape. As a result, stronger or earlier reflections disproportionately influence the measured time-of-flight, leading to a systematic shift in the perceived position. In prac-

tice, this error is typically corrected using a dedicated walk error correction function that compensates for these amplitude-dependent timing offsets.

Implications and Future Correction
The presence of walk error highlights the importance of precise sensor modeling in simulation environments. To improve the realism and reliability of the LiDAR simulation, the following corrective strategies may be considered:

- Incorporation of a walk error correction model within the simulation pipeline.
- Calibration of ToF parameters based on empirical measurements from real sensors.
- Temporal synchronization of simulated point generation with real-world signal propagation dynamics.

Addressing this issue will not only improve the spatial alignment between real and simulated panels but also enhance the fidelity of histogram-based comparisons and detection probability measurements.

4.3 Blooming

Blooming in LiDAR refers to an artifact in point cloud data where certain objects appear larger or more spread out than they actually are. This effect is most noticeable around highly reflective surfaces such as traffic signs, lane markings, or retroreflective materials. Blooming causes these objects to have exaggerated dimensions in the LiDAR representation, leading to inaccuracies in perception and localization. Figure 4.23 shows a highly reflective traffic sign and the blooming effect is particularly visible around highly reflective objects. Figure 4.24 illustrates the point cloud data of the traffic sign without blooming, while Figure 4.25 shows the same point cloud data with the blooming effect applied, highlighting the increased size of the traffic sign due to the artifact.

Figure 4.23: Traffic Sign [26]

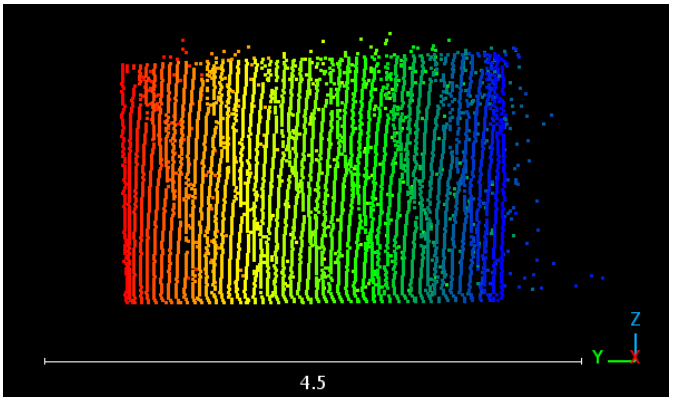

Figure 4.24: Point Cloud Data of the Traffic Sign

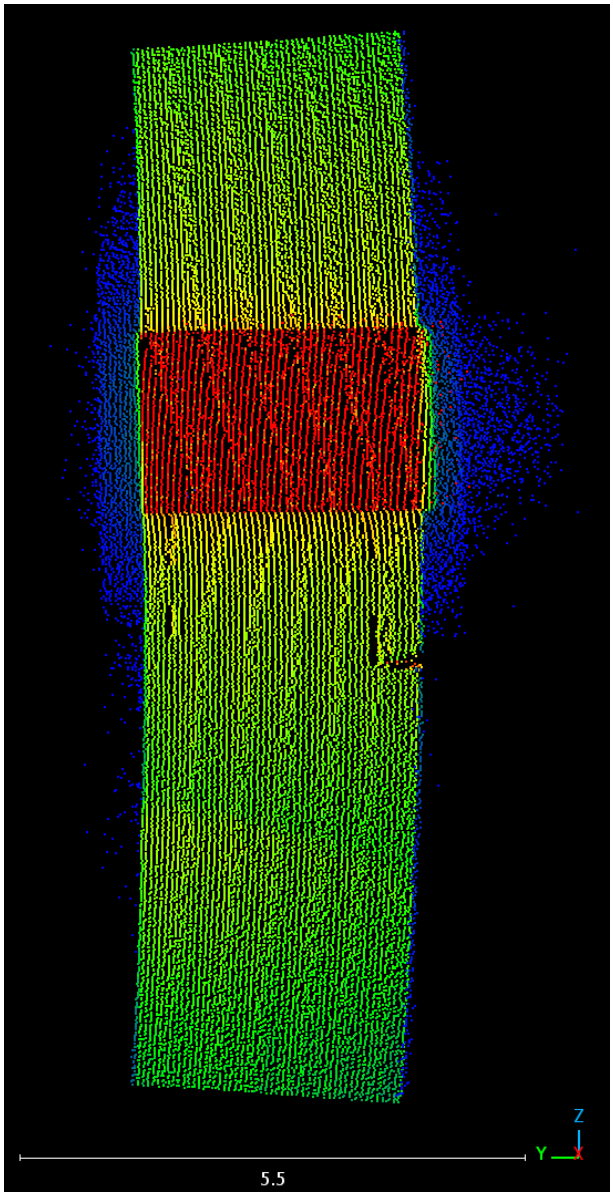

Figure 4.25: Point Cloud Data of the Traffic Sign with Blooming

Blooming in LiDAR sensors primarily occurs due to the following factors:

1. **Beam Divergence and Optical Effects**
   LiDAR beams are not perfectly collimated; they have some divergence (spreading). At greater distances, the beam covers a larger area. If a part of the beam reflects from a highly reflective surface the power is enough to trigger a detection.

2. **Scattering in the optical path of the sensor**
   The amount of light from retro-reflectors is very high. When entering the sensor it is scattered in the optical path and hits areas of the SPAD array that are not directly looking at the reflective surface.

4.3.1 Effects of Blooming in LiDAR Applications

Blooming can cause several issues in autonomous driving and perception systems:

- **Inaccurate Object Dimensions:** Reflective traffic signs may appear larger than they are, leading to errors in object recognition and classification.
- **Localization Errors:** Vehicles relying on LiDAR-based mapping may misinterpret bloomed objects as different structures, causing slight deviations in positioning.
- **False Positive Detections:** Ghost reflections or bloomed regions may be mistakenly identified as obstacles, triggering unnecessary braking or navigation corrections. Traffic signs over a bridge create blooming that looks like a big wall and may cause emergency breaking.
- **Inconsistent Data Fusion:** When combining LiDAR data with camera images, the bloomed LiDAR points may not align properly with the actual object edges detected in images.

4.3.2 Types of Blooming in LiDAR

Blooming in LiDAR can be categorized into two types based on the direction in which the points spread due to overexposure or saturation:

- **Vertical Blooming** (blooming along the height axis)
- **Horizontal Blooming** (blooming along the width axis)

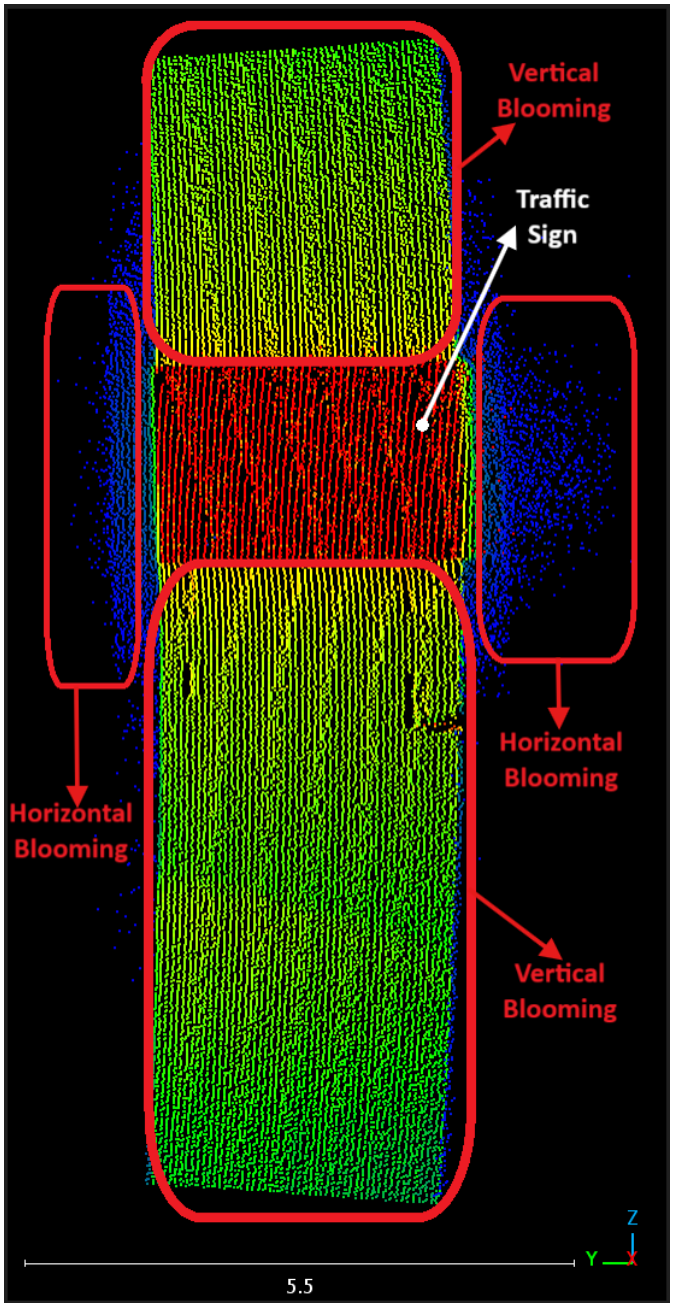

Figure 4.26: Vertical and Horizontal Blooming

Both types of blooming occur due to similar underlying causes: high reflectivity, internal scattering, and beam divergence. However, their prominence depends on the architecture of the LiDAR sensor, particularly how the laser beams are arranged and emitted.

Vertical Blooming

Vertical blooming refers to the excessive spreading of LiDAR points along the height axis (Z-axis in the point cloud). It occurs when the sensor registers additional returns in the vertical direction due to high reflectivity of the target.

Causes of Vertical Blooming

One primary cause of vertical blooming is the distribution of laser beams and photodetectors in the vertical direction. Many LiDAR sensors, including ours, use optics to spread laser light vertically. When a highly reflective surface returns an intense signal, the light can scatter across the entire vertical array of photodetectors. This results in multiple false returns being registered at different heights, as each beam may capture reflections incorrectly due to this vertical spread.

Effects of Vertical Blooming

- **Overestimation of Object Height:** Objects appear taller than they actually are.
- **False Obstacles in Perception Systems:** Autonomous driving systems may detect obstacles where none exist. For example under a bridge with big traffic signs.
- **Degraded Point Cloud Accuracy:** Points from a single reflective surface may be misclassified across different heights.

Horizontal Blooming

Horizontal blooming refers to the spreading of LiDAR points along the width axis (X-axis in the point cloud). This effect makes objects appear wider than they actually are.

Causes of Horizontal Blooming

A key cause of horizontal blooming is beam divergence in the horizontal direction. Some LiDAR sensors exhibit slight horizontal divergence, which causes the laser footprint to expand in width over long distances. This leads to a wider distribution of points and contributes to the blooming effect. The issue can be further exacerbated by environmental factors such as dirt, moisture, or condensation on the sensor cover, resulting in increased variability of horizontal blooming under different weather conditions.

Effects of Horizontal Blooming

- **Overestimation of Object Width:** Traffic signs, lane markers, or other reflective objects may appear wider in LiDAR data than in reality.
- **Errors in Object Classification:** The AI system may misinterpret bloomed objects, affecting vehicle detection or road sign recognition.
- **Point Cloud Distortion in Mapping:** If horizontal blooming is not corrected, it can introduce inaccuracies in HD maps used for autonomous driving.

4.3.3 Vertical Blooming in Our LiDAR Sensor

In our LiDAR sensor, vertical blooming is more pronounced due to the specific arrangement of laser beams within the scanning array. The beams are stacked vertically, meaning each channel is positioned directly above another. When a highly reflective object causes saturation, multiple vertically aligned photo-detectors are affected simultaneously, leading to a stronger vertical blooming effect compared to horizontal blooming.
This phenomenon is a crucial consideration in sensor design and performance optimization, influencing LiDAR accuracy and reliability in environments with reflective surfaces.

4.3.4 Implementation Approach

Step 1: Selection of Blooming Scenario
To analyze and calibrate the blooming effect in LiDAR data, the first step was to identify a real-world object that consistently exhibits strong blooming artifacts as seen in Figure 4.27. After carefully examining the real PCD dataset, a traffic sign was chosen for the following reasons:

- **High Reflectivity:** Traffic signs are coated with retroreflective material designed to return maximum light to the source (such as vehicle headlights or LiDAR beams). This high reflectivity often causes saturation in the LiDAR sensor, leading to the blooming effect.
- **Consistent Structure:** Unlike vehicles, pedestrians, or other dynamic objects, traffic signs are static and have a well-defined shape, making them ideal for controlled analysis. The expected dimensions are known, making it easier to compare actual vs. bloomed shapes.
- **Distance-dependence:** When a vehicle is driving on the highway it is recording the data of the traffic sign at different distances and we can use this data for analysis.
- **Real-World Relevance:** In autonomous driving, traffic signs are critical for perception and navigation. A misinterpretation due to blooming (e.g., detecting a sign as being larger than it actually is) can lead to incorrect decision-making by the vehicle.

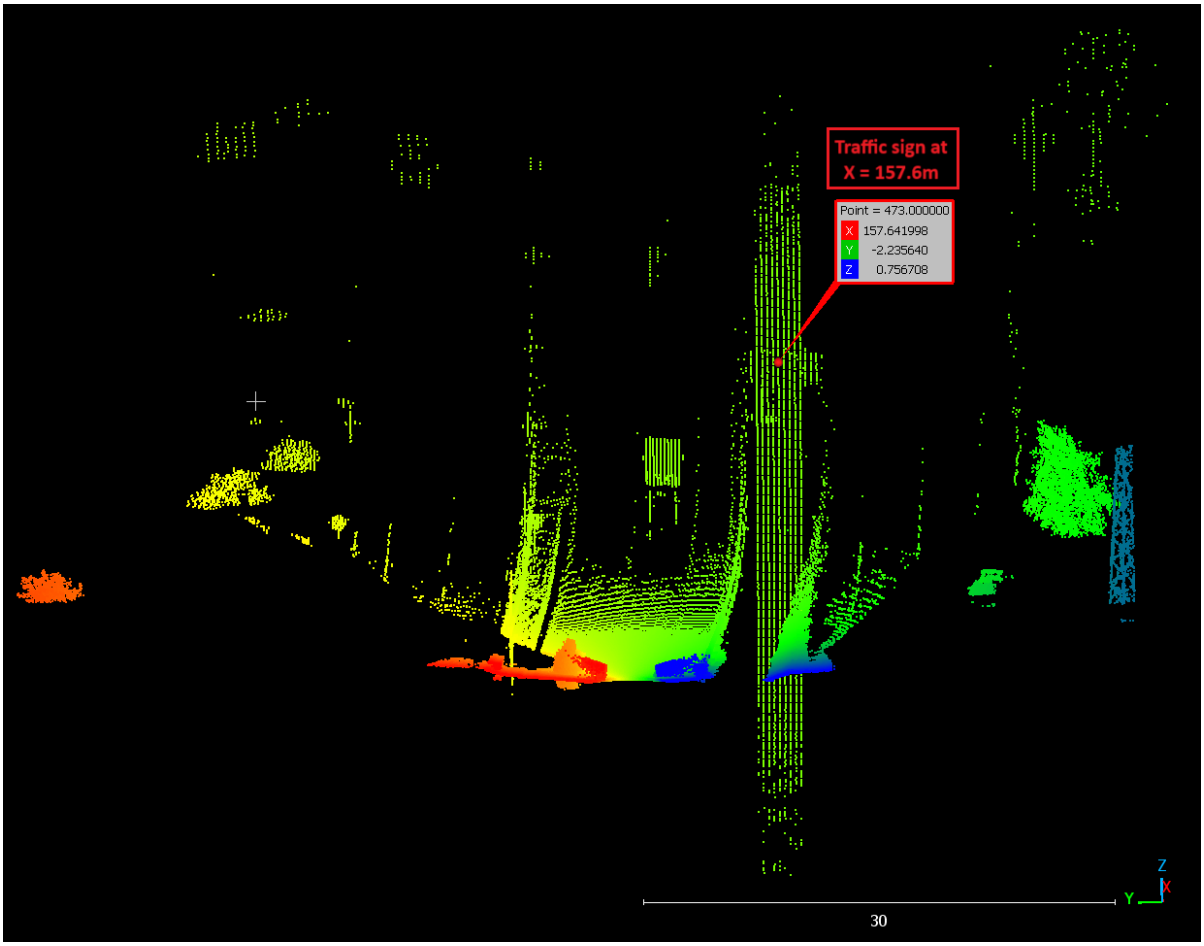

Figure 4.27: Point Cloud of Real-World with Blooming effects

Step 2: Selecting Multiple Distances (70m, 150m, 200m)
After deciding on traffic signs as the object of interest, three LiDAR point cloud files were selected, each containing the same traffic sign at different distances—70m, 150m, and 200m. It is advantageous to select multiple distances because:

- **Blooming Behavior Changes with Distance:** At longer distances, the complete vertical blooming effect can fit within the field of view (FOV) of the sensor; however, the return signal tends to be weaker. At closer distances, although the entire blooming may not fully appear within the sensor's FOV, the traffic sign is captured with stronger signal intensity and greater detail.
- **Understanding how Blooming Evolves Over Distance:** By analyzing blooming across multiple distances, we can quantify how the effect scales with range. This helps in calibrating blooming correction functions that adjust for different distances.
- **Validating a Distance-Based Correction Approach:** If blooming characteristics follow a pattern across distances, a (LUT (e.g., LUT_blooming_X, LUT_blooming_Y) can be designed to compensate for these variations dynamically.

Step 3: Snipping the PCD Files (Extracting Only the Traffic Sign and Blooming Points)
After selecting the relevant PCD files, the next step was to filter out unnecessary points to focus exclusively on the traffic sign and the associated blooming points as shown in Figure 4.28. This involved:

- **Isolating the Region of Interest:** Using bounding box selection or clustering, a region around the traffic sign was extracted. The goal was to remove other environmental elements (e.g., road, vehicles, vegetation) and retain only the sign and its bloomed reflections.
- **Retaining Blooming Points:** When extracting the traffic sign, it was important to preserve the bloomed points rather than applying strict filtering. These additional points (false returns caused by blooming) needed to be included for accurate calibration analysis.
- **Standardizing the Output Format:** The extracted data was saved in a format suitable for direct comparison between real and simulated LiDAR data. This ensured that only relevant points were used in histogram and point cloud comparisons.

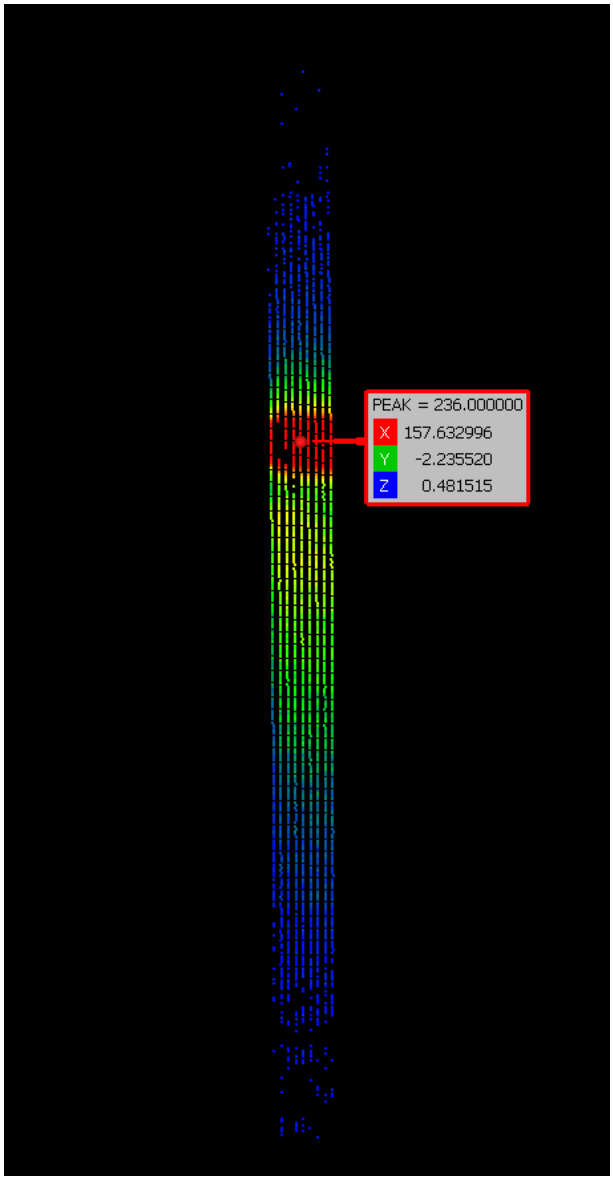

Figure 4.28: Point Cloud of Traffic sign with Blooming effects

4.3.5 GUI Enhancement for Multi-File Processing

Initial GUI Implementation: Single Dataset Processing
The initial version of the GUI was designed to process only one dataset at a time. While this allowed for detailed analysis of blooming effects, it had several limitations:

- Users had to reload data manually when switching between different files.
- Each dataset required a separate processing run, leading to redundant computations.
- Comparing blooming effects at different distances was tedious because we had to replace the files of interest at the source path.

Extending the GUI for Multi-Dataset Handling
To improve efficiency and usability, the GUI was extended to handle multiple datasets simultaneously as shown in Figure 4.29. This enhancement was achieved through the following key modifications:

- **Implementing a Global Data Storage System:** A global variable was introduced to store processed data from multiple files at once. This eliminated the need to reload or reprocess data when switching between datasets.
- **Enabling Instant Dataset Switching:** Dedicated buttons were provided in the GUI, allowing users to switch between different datasets dynamically. Instead of re-running simulations, the GUI now dynamically loads the preprocessed real and simulated LiDAR graphs upon selection. Since all processed data was stored in memory, switching between files was instantaneous without requiring reprocessing, which significantly reduced computation time.

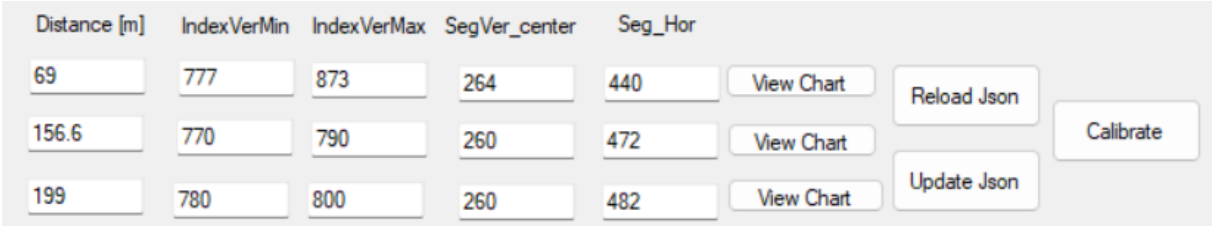

Figure 4.29: Dataset of Traffic Sign for 3 Distances

4.3.6 Calibration of Blooming Effect Using Lookup Tables (LUTs)

Initially, the calibration of the blooming effect in the LiDAR simulation was performed manually by modifying the `ValeoScaLaPostprocessingConfig_x.ini` file. The calibration required updating specific (LUT parameters for both vertical and horizontal blooming:

Vertical Blooming Calibration:

- `PostPro.LUT_blooming_X`
- `PostPro.LUT_blooming_Y`

The LUT for vertical blooming is shown in Figure 4.30, which defines the intensity of the blooming effect across different angular offsets.

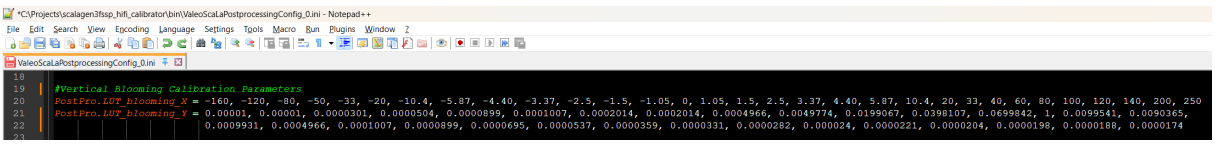

Figure 4.30: Look-Up-Table to calibrate Vertical Blooming

Horizontal Blooming Calibration:

- `PostPro.LUT_bloomingH_X`
- `PostPro.LUT_bloomingH_Y`

The LUT for horizontal blooming is shown in Figure 4.31, where similar intensity values are defined for the horizontal axis.

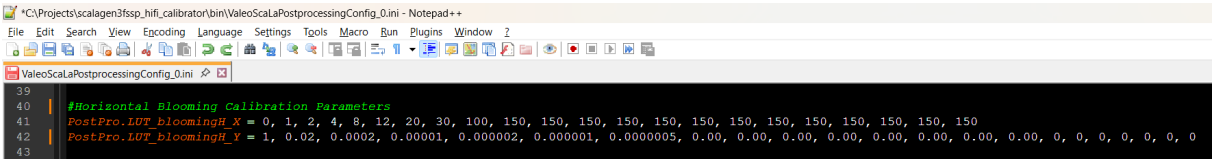

Figure 4.31: Look-Up-Table to calibrate Horizontal Blooming

These parameters define the intensity of the blooming effect across different angular offsets. A typical set of values for vertical blooming is as follows:

Example LUT Parameters for Vertical Blooming:

- `PostPro.LUT_blooming_X` = -160, -120, -80, -50, -33, -20, -10.4, -5.87, -4.40, -3.37, -2.5, -1.5, -1.05, 0, 1.05, 1.5, 2.5, 3.37, 4.40, 5.87, 10.4, 20, 33, 40, 60, 80, 100, 120, 140, 200, 250
- `PostPro.LUT_blooming_Y` = 0.00001, 0.00001, 0.0000301, 0.0000504, 0.0000899, 0.0001007, 0.0002014, 0.0002014, 0.0004966, 0.0049774, 0.0199067, 0.0398107, 0.0699842, 1, 0.0099541, 0.0090365, 0.0009931, 0.0004966, 0.0001007, 0.0000899, 0.0000695, 0.0000537, 0.0000359, 0.0000331, 0.0000282, 0.000024, 0.0000221, 0.0000204, 0.0000198, 0.0000188, 0.0000174

Challenges in Manual Calibration

The primary difficulty in manually adjusting these values was due to the very small numerical values in `PostPro.LUT_blooming_Y`. Many values contain multiple decimal places (e.g., 0.00001 or 0.0000899), making it extremely difficult to modify, interpret, and fine-tune. Even small changes could lead to unpredictable effects, requiring multiple iterations and significant patience to achieve optimal calibration.

Development of a Helper Calibration Tool

To address these challenges, an additional helper calibration tool was developed and integrated into the GUI as shown in Figure 4.32, providing an intuitive way to modify LUT values.

Key Features of the Helper Tool:

- **Table-Based Visualization:** Instead of manually editing numerical values in a configuration file, the tool loads the LUT values in a tabular format within the GUI, making them easier to read and adjust.
- **Logarithmic Scaling for Y-Values:** Since the Y values in the LUT contain extremely small decimal numbers, they were transformed into whole numbers ranging from 1 to 1000 using a logarithmic scale. This transformation significantly improves the readability and ease of adjustment, allowing users to work with numbers that are more intuitive.
- **Real-Time Calibration & Direct File Writing:** A Save button was implemented to directly write the updated LUT values back into the configuration file (`.ini` file). When saving, the values are automatically converted back into their fractional format, ensuring compatibility with the existing simulation framework. This eliminates the need for manual copying and pasting, streamlining the calibration workflow.

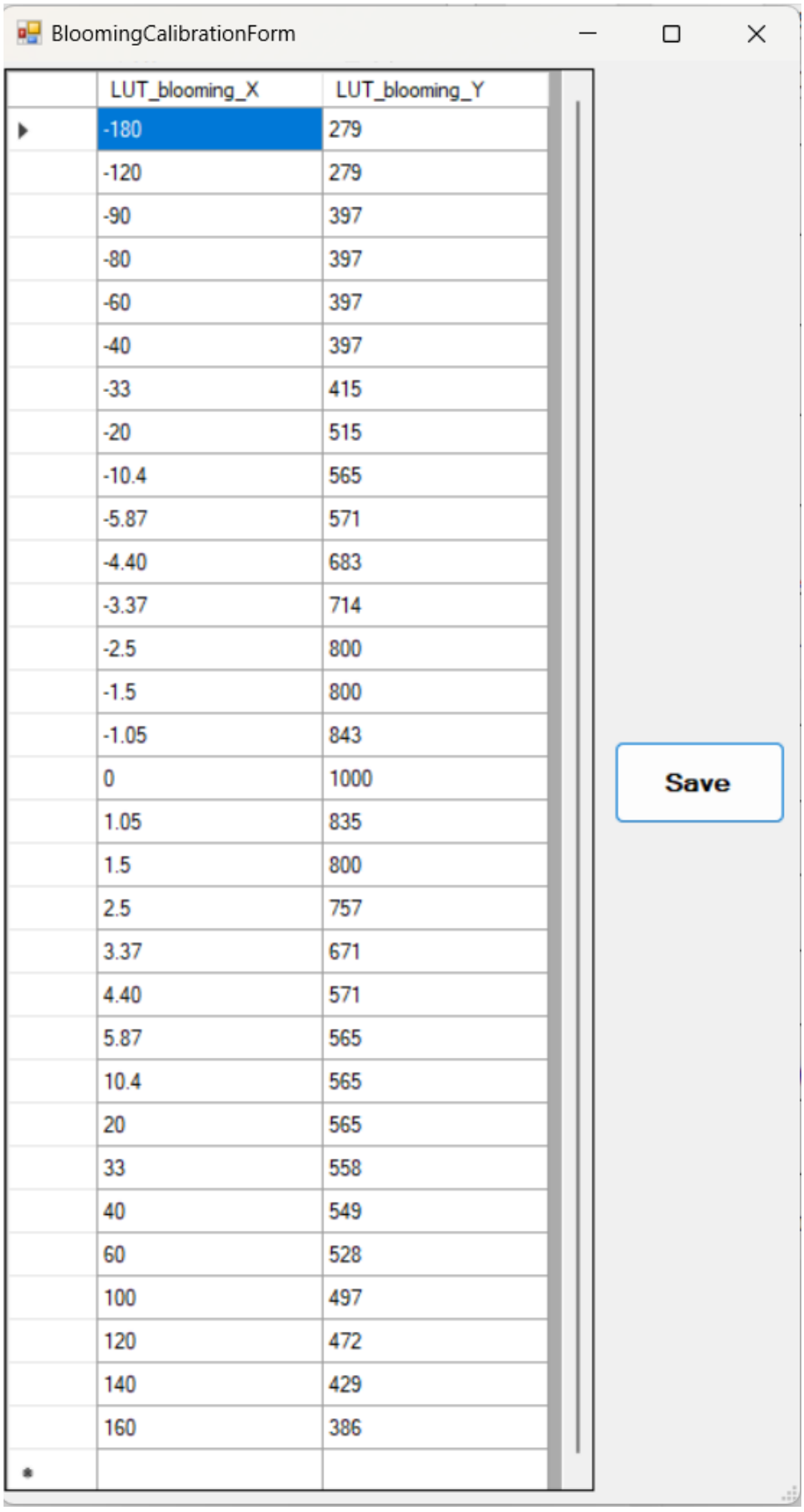

Figure 4.32: Helper tool for Calibrating Parameters

Benefits of the Helper Calibration Tool:

- **Faster Calibration:** Adjustments can be made rapidly without the need for manual fine-tuning of small decimal values.
- **Improved Accuracy:** Logarithmic scaling prevents errors due to misinterpretation of numerical values.
- **User-Friendly Workflow:** The graphical table representation provides an intuitive way to modify parameters.
- **Automated Data Handling:** Eliminates the need for manually entering values into the `.ini` file, reducing potential mistakes.

By integrating this tool into the GUI, the calibration process was greatly simplified, making it both faster and more efficient while improving the overall accuracy of the blooming effect adjustments.

Evaluation of Calibration Results

The primary goal of the calibration process was to fine-tune the blooming effect parameters so that the simulated LiDAR data closely matches the real-world point cloud data. Since blooming is an artifact that depends on multiple factors including distance, reflectivity, and beam divergence ensuring consistency across multiple distances was a critical part of the evaluation.

Calibration Strategy:

To systematically evaluate the effectiveness of the calibration:

- **Tuning of Calibration Coefficients and Parameters:** All relevant LUT coefficients (`LUT_blooming_X`, `LUT_blooming_Y`, `LUT_bloomingH_X`, `LUT_bloomingH_Y`) were iteratively adjusted. The adjustments aimed to reduce the deviation between the real and simulated blooming effect while maintaining physical plausibility.
- **Multi-Distance Validation:** Three different distances (e.g., 70m, 150m, 200m) were selected to ensure that the calibration was not overfitting to a single case. Each dataset was processed independently, and the parameters were tuned to achieve a uniformly accurate calibration across all three distances.
- **Graphical Comparison & Visual Analysis:** The real and simulated data were overlaid on top of each other to visually assess the accuracy of the blooming effect reproduction. Overlays of the real and simulated curves allowed precise identification of discrepancies, ensuring a quantitative and qualitative evaluation.
- **Iteration and Refinement:** The calibration process was performed iteratively, adjusting the parameters and analyzing the results after each modification. The

helper calibration tool significantly accelerated this process by allowing direct parameter tuning through the GUI, reducing manual effort. A balance was maintained to avoid overfitting, ensuring that changes improved the overall realism rather than compensating for specific cases.

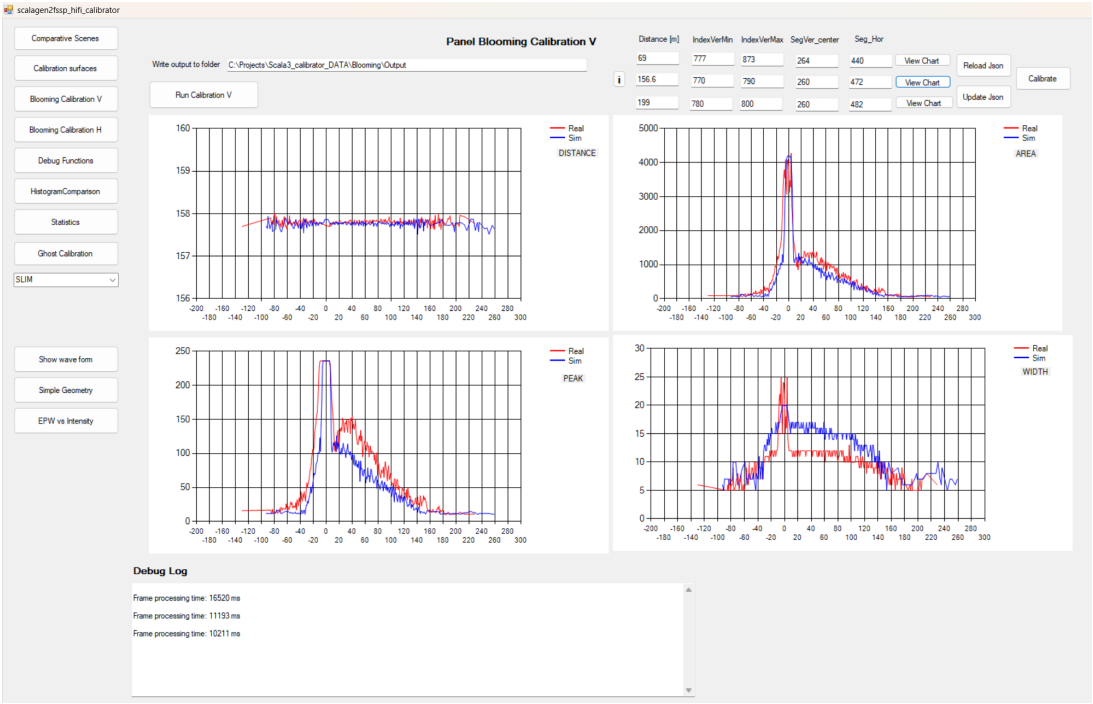

Figure 4.33: Histogram of Vertical Blooming at 157m

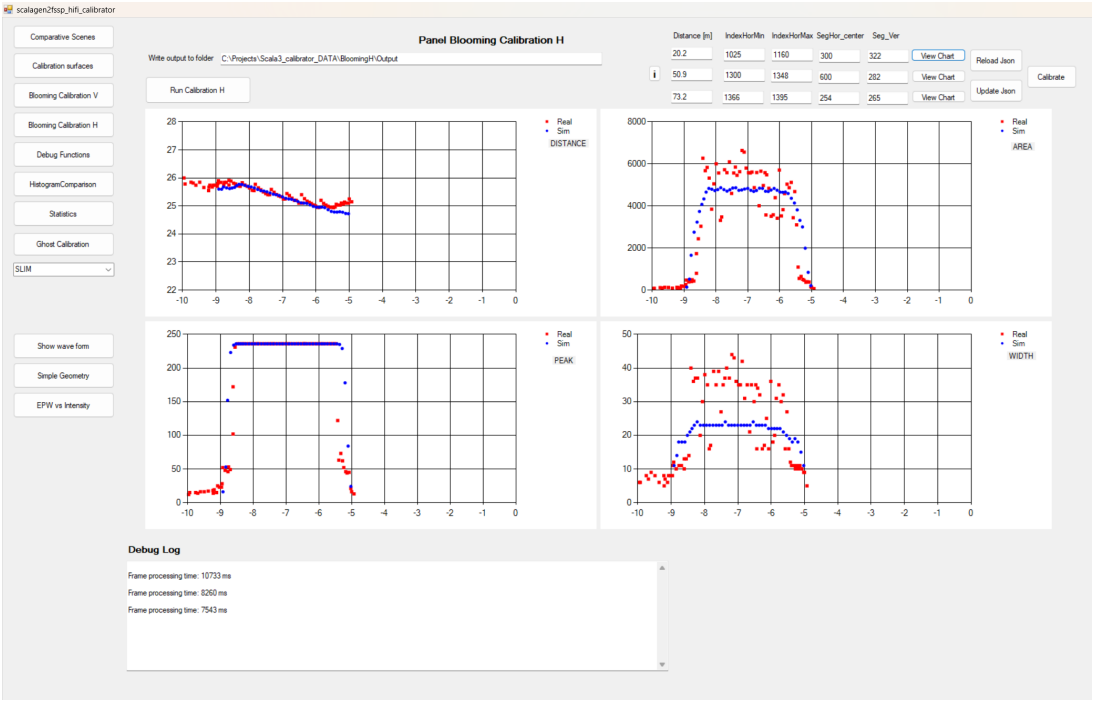

Figure 4.34: Histogram of Horizontal Blooming at 20m

4.4 Ghost Points

Ghost particles in LiDAR refer to spurious points that do not correspond to actual objects in the environment but appear in the point cloud due to unintended signal interactions as shown in Figure 4.35. These artifacts distort the accuracy of LiDAR-based perception systems, leading to incorrect object detection and depth estimation. Ghost particles arise due to various optical and environmental factors, making them a critical challenge in LiDAR simulation and sensor validation.

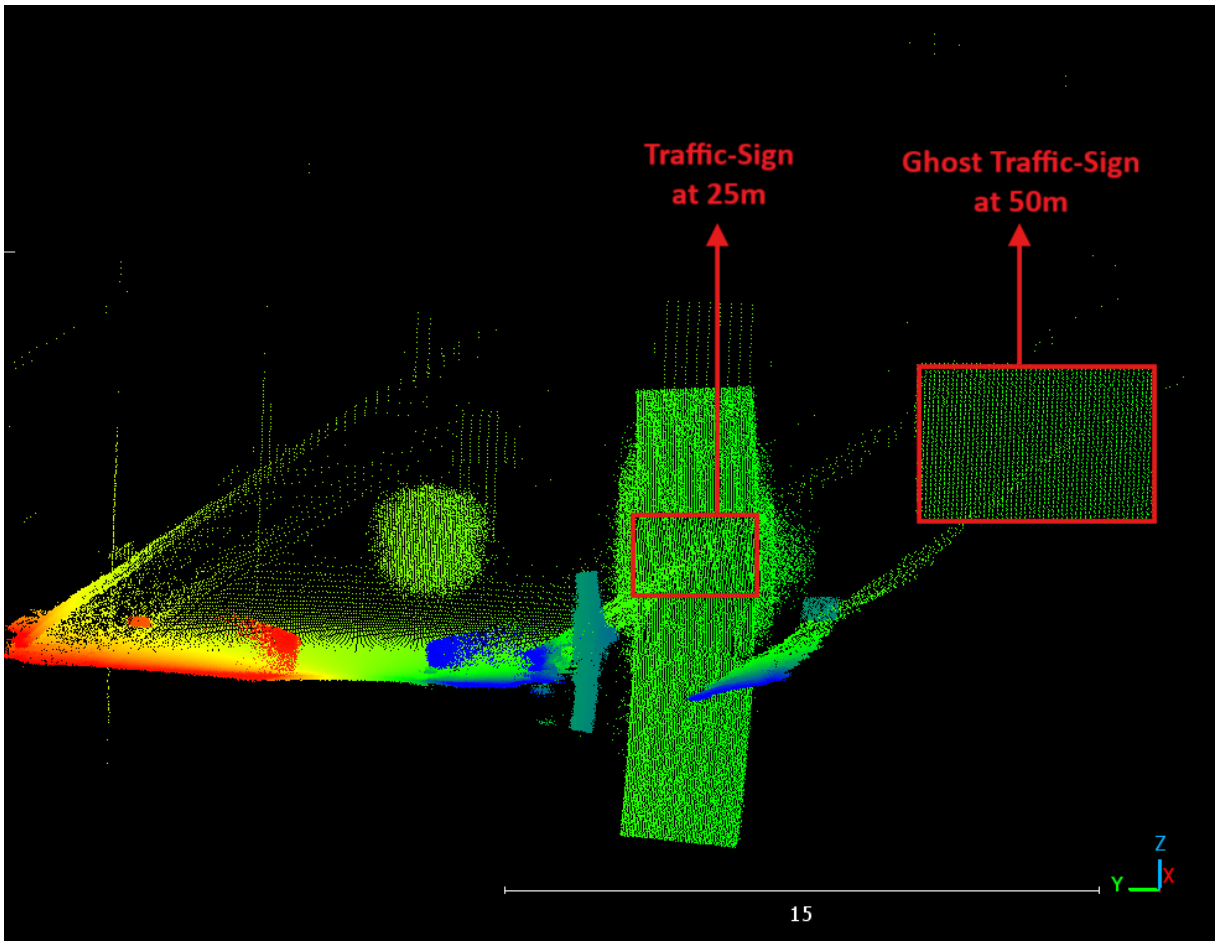

Figure 4.35: Point cloud of Traffic sign and Ghost points with surroundings

4.4.1 Factors Causing Ghost Particles

The occurrence of ghost particles in LiDAR systems is primarily attributed to the following factors:

- **Multiple Reflections:** When a LiDAR pulse encounters a highly reflective surface, it may undergo multiple bounces before returning to the sensor. This results in secondary points appearing at incorrect depths, leading to ghost particle formation [9]. Common sources of multiple reflections include metallic objects, road signs, and smooth surfaces.

- **Multi-Path Propagation:** In environments with complex geometries, LiDAR pulses can take indirect paths due to diffraction or scattering. This results in ghost points appearing at unexpected locations, which can affect depth perception and object recognition [39].
- **Transparent and Semi-Transparent Surfaces:** Materials such as glass and water partially transmit LiDAR pulses, causing secondary reflections that create ghost particles [13]. These reflections can lead to false detections behind transparent objects, affecting the reliability of LiDAR-based scene understanding.

4.4.2 Effects of Ghost Particles in LiDAR Applications

Ghost particles, which are unintended reflections captured by the LiDAR sensor, pose a unique challenge in many perception and mapping tasks. These artifacts often appear due to reflective surfaces, multipath propagation, or sensor-specific limitations and can introduce false spatial information in the point cloud. Their presence compromises the accuracy of downstream applications by distorting the geometric and semantic understanding of the environment. This section outlines the primary domains affected by ghost particles, illustrating their impact on both system performance and safety in real-world deployments.

Ghost particles can significantly impact various LiDAR-based applications, including:

- **Autonomous Driving:** Ghost particles can cause incorrect object detections, leading to potential safety risks in autonomous driving systems [21]. False positive detections may result in unnecessary braking, while false negatives can lead to missed obstacles.
- **Object Detection and Tracking:** The presence of ghost particles in LiDAR point clouds affects the performance of object detection algorithms by introducing spurious points. This can lead to misclassification or a loss of tracking accuracy in dynamic environments [45].
- **Simulated vs. Real-World Data Alignment:** The absence of ghost particles in simulated LiDAR data creates a discrepancy between real and synthetic datasets [37]. This affects the effectiveness of machine learning models trained on simulated data, reducing their ability to generalize to real-world scenarios.
- **Environmental Mapping and SLAM:** In Simultaneous Localization and Mapping (SLAM) applications, ghost particles introduce errors in 3D reconstructions, leading to inaccuracies in localization and map generation [44]. This is particularly problematic in robotics and geospatial applications where precision is critical.

4.4.3 Implementation Approach

Selection of Ghost Particle Scenario

To analyze and understand the formation of ghost particles in LiDAR data, it is essential to begin by selecting a real-world scenario where these artifacts are consistently

observed. The choice of object for analysis significantly influences the relevance and accuracy of the evaluation. In this context, traffic signs have been selected as the primary object of interest for the following reasons:

- **High Reflectivity of Traffic Signs:** Traffic signs are specifically designed with retroreflective coatings to maximize visibility under various lighting and weather conditions. These coatings are engineered to return incident light, whether from vehicle headlights or active LiDAR pulses directly back to the source. While this characteristic is beneficial for human and machine perception, it also increases the likelihood of intense secondary reflections in LiDAR systems. The strong return signal from the traffic sign is reflected again by the surface of the vehicle towards the traffic sign. This secondary reflection is reflected from the traffic sign again producing a faint ghost image of the traffic sign at double the distance of the original traffic sign.

  Compared to other reflective surfaces such as metallic vehicle bodies, road guardrails, or wet pavement, traffic signs provide a controlled, repeatable, and high-reflectivity target ideal for systematic study of ghosting effects. The strong and consistent returns from traffic signs allow for reliable identification of primary and secondary echoes, making them suitable for calibration and simulation validation.

- **Environmental Factors Enhancing Ghosting Effects:** Although traffic signs themselves are the primary focus, the environment in which they are situated also contributes to the formation of ghost particles. For instance, signs positioned near glass bus stops, enclosed intersections, or metallic signposts may further exacerbate multipath propagation. In such cases, the reflective nature of surrounding materials can redirect the LiDAR pulse in unintended ways, leading to complex return patterns and the generation of ghost points. Therefore, scenes involving traffic signs in highway scenarios are particularly valuable for analysis because they provide an isolated occurrence of the ghost image.

- **Relevance to Real-World Applications:** The accurate simulation and calibration of ghost particles have direct implications for autonomous driving and ADAS. Traffic signs are critical elements in the decision-making pipeline of autonomous vehicles. Misidentification or false detection due to ghost particles can lead to navigation errors or safety risks. By focusing on traffic signs, this study aims to enhance the fidelity of LiDAR simulation models and improve the robustness of perception algorithms in real-world driving environments.

In the subsequent sections, a detailed analysis of ghost image behavior around traffic signs will be presented, including comparisons between real and simulated LiDAR data. This will form the foundation for developing and calibrating sensor models that better capture the nuances of high-reflectivity interactions in practical scenarios.

4.4.4 Selecting Multiple Distances

Following the identification of traffic signs as the object of interest for ghost image investigation, the next step involves analyzing how ghosting behavior changes with varying

distances. For this purpose, four representative ranges were selected: 25 meters, 45 meters, 58 meters and 62 meters. These distances were chosen to reflect realistic detection ranges of traffic signs in autonomous driving scenarios and to observe the evolution of ghost particle characteristics over short, medium, and long-range LiDAR interactions.

Importance of Multiple Distance Evaluation

Ghost image formation depends on both the reflectivity of the traffic sign and the distance between the LiDAR sensor and the object. As distance increases, both real and ghost point clouds experience a decline in density, but ghost particles are more sensitive to these effects due to their indirect origin. The behavior across ranges is described below:

- **Short Range:** Both real and ghost particles are clearly visible, with high point cloud density. The LiDAR beam is focused and strong, and even secondary reflections responsible for ghosting are sufficiently intense to be captured. Ghost particles in this range appear well-defined and easy to distinguish, particularly behind reflective surfaces.
- **Medium range:** Beam divergence begins to reduce the density of both real and ghost points. Although ghosting is still present, it is generally less pronounced than at short range, with fewer points forming the ghost structures. Reflective surfaces and partial enclosures can amplify ghost effects in this range.
- **Long range:** Real and ghost point densities drop significantly due to beam divergence and signal attenuation. However, ghost particles can still be clearly observed, though they are represented by a sparse set of points. These low-density ghost structures are still relevant for perception algorithms, as they can resemble distant real objects and lead to false positives if not properly filtered or modeled.

4.4.5 Preprocessing LiDAR Data: Isolating Traffic Sign and Ghost Particles

To effectively analyze ghost particle behavior in LiDAR data, the raw point cloud must undergo a targeted preprocessing step. The objective is to isolate the traffic sign, which serves as the primary object of interest along with any associated ghost particles, while eliminating irrelevant background information. As shown in Figure 4.35, the original point cloud includes not only the traffic sign and its ghost reflections but also surrounding objects such as vegetation and road infrastructure.

The first phase of preprocessing involves defining a Region Of Interest (ROI) around the traffic sign and its surrounding space. Later, the points corresponding to the traffic sign and potential ghost reflections are segregated from unrelated elements like road surfaces, vegetation, or distant background objects. This refinement, which narrows the data to the region seen in Figure 4.36, ensures a more focused analysis compared

to the broader context provided in Figure 4.35.

A crucial aspect of this process is the intentional retention of ghost particles. While standard point cloud denoising techniques aim to eliminate anomalous returns, in this case, such returns are of primary interest. Ghost particles arising from factors such as multipath propagation, retroreflection from the traffic sign, or interactions with semi-transparent or reflective surfaces must be preserved to understand their spatial distribution, intensity, and occurrence patterns.

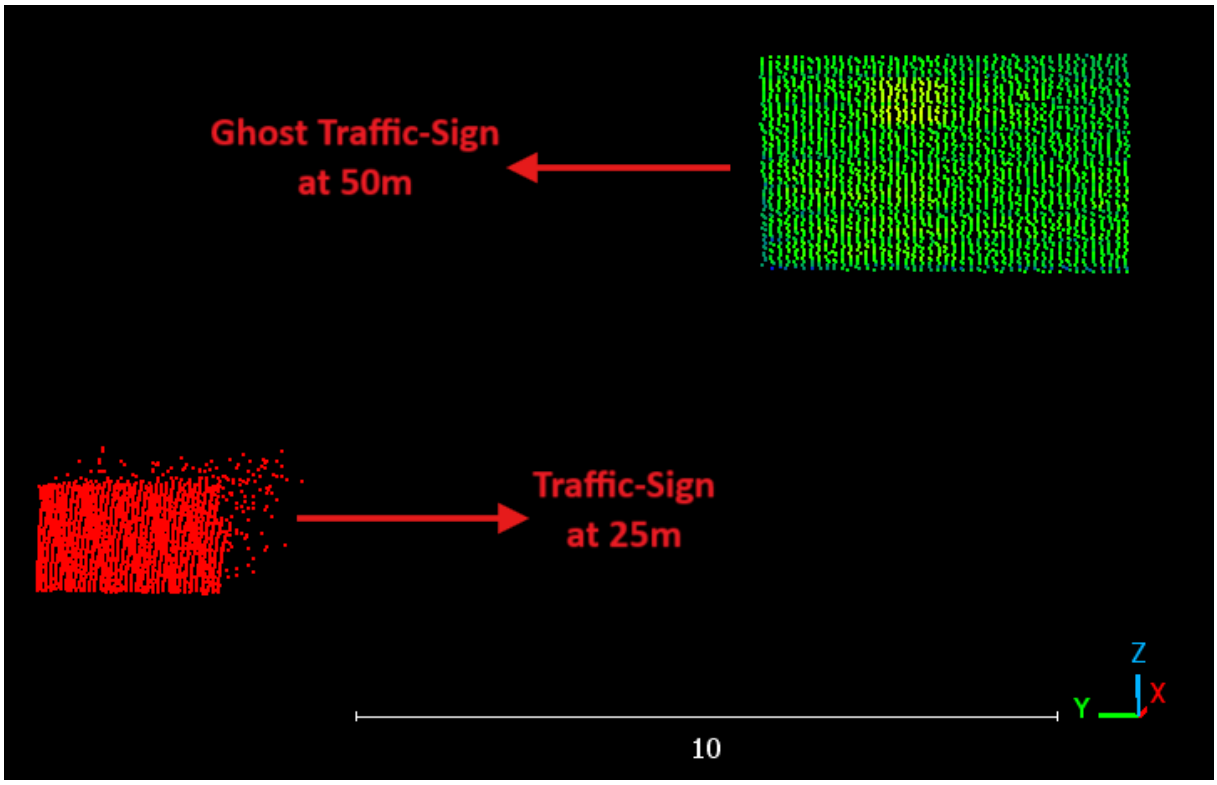

Figure 4.36: Snipped point cloud of retained Traffic sign and Ghost points

Finally, the standardization of the output format plays an essential role in the analysis pipeline. The extracted and filtered point cloud is saved as a .txt point cloud file, which allows for seamless visualization and comparison between real-world and simulated LiDAR outputs. This step enables downstream processes such as histogram generation, distance-based profiling, and algorithmic validation of ghosting behavior, contributing to the development of reliable correction mechanisms for future simulation improvements.

4.4.6 Graphical User Interface for Ghost Point Analysis

The GUI is designed to visualise four datasets of ghost particles. The GUI enables efficient comparison between real and ghost reflections of traffic signs based on three

key metrics: area, peak, and number of points. It is designed to facilitate side-by-side evaluation of these four datasets.

Components of the GUI
There are several key features of the GUI that facilitate efficient analysis, which can be seen in the following components.

Sensor Variant Selection:

- Based on the selected variant of the sensor between SMART and SLIM, the corresponding data path is fetched from the textboxes to select the appropriate datasets.

Area Histogram:

- The area histogram represents the total energy of the detected return pulse above the threshold.
- For the real traffic sign, the area histogram chart displays the distribution using both real and simulated point cloud data, plotted together to enable direct visual comparison.
- For the ghost traffic sign, the area histogram chart presents the distribution based on ghost points, allowing analysis of how the simulated model replicates ghosting behavior compared to the real data.

Peak Histogram:

- The peak amplitude is a key metric used to determine the strength of the reflected signal.
- For the real traffic sign, the peak histogram chart displays the maximum point concentration within the point cloud using both real and simulated data, enabling direct comparison of how sharply the reflections are captured.
- For the ghost traffic sign, the peak histogram chart illustrates the maximum point concentration of the ghost points, providing insight into the strength and clarity of the ghost reflections in both real and simulated datasets.

Point Count Display:

- Displays the total number of LiDAR points associated with the real and ghost reflections numerically.
- Provides a quick overview of data density differences.

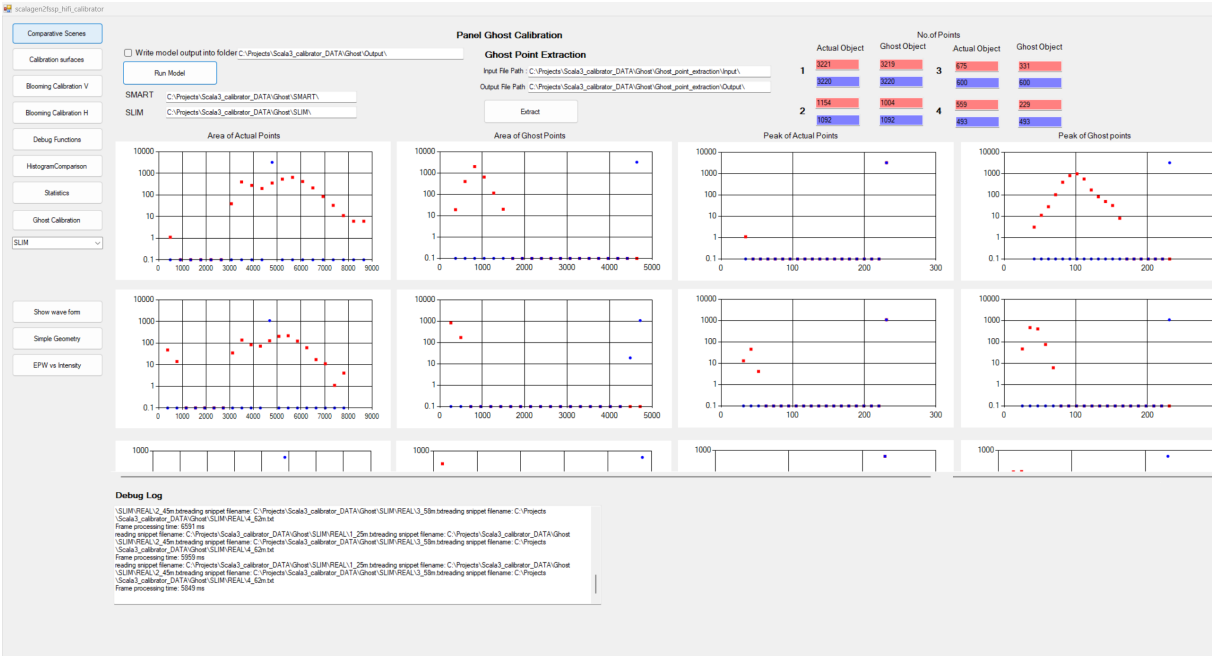

Figure 4.37: Panel Ghost Calibration

As shown in Figure 4.37, the panel layout enables users to compare the various datasets visually through the graphical components discussed above. The side-by-side arrangement allows easy tracking of data from both real and ghost reflections, aiding in accurate calibration analysis.

4.4.7 Future Development: Integration of Ghost Traffic Sign Simulation

The current implementation of the calibration tool includes a fully developed GUI and a complete analysis pipeline designed to support the evaluation of ghost particle behavior. At the moment of writing this thesis, the sensor model does not yet have the functionality to generate ghost images. As per the initial division of tasks, the responsibility for implementing the simulated ghost point generation module was assigned separately, while the focus of this thesis was to develop the GUI and ensure the pipeline was fully prepared for later integration.

At this stage, the GUI and processing pipeline are ready and functional. A stub function has been included in the codebase to represent the ghost traffic sign simulation module. Once the corresponding simulation logic is implemented, it can be integrated seamlessly by replacing the stub with the actual function. This modular structure ensures that no further adjustments are required in the visualization or analysis components.

Currently, when comparing the histograms of area and peak, it is observed that the values for the real traffic sign (both real and simulated point clouds) are accurately reflected in the GUI. However, the values for the ghost traffic sign including its area

histogram, peak histogram, and total point count are derived from the stub function and do not represent actual simulated data. These placeholders serve the purpose of interface testing and layout validation, and will be replaced with real data once the ghost simulation functionality is implemented. This step will complete the end-to-end calibration and analysis tool for ghost particle behavior.

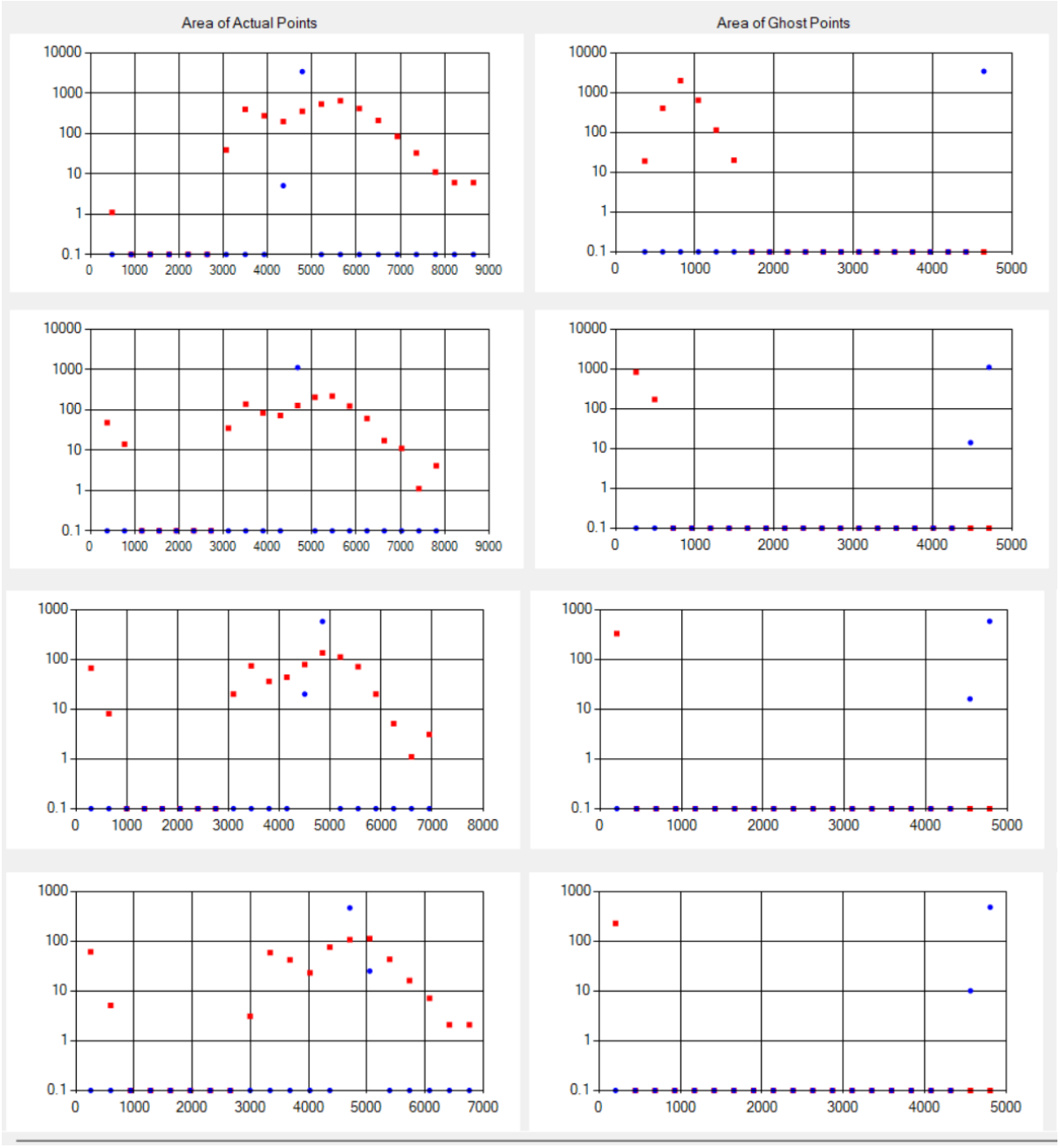

Figure 4.38: Area histogram of Actual and Ghost points across 4 datasets

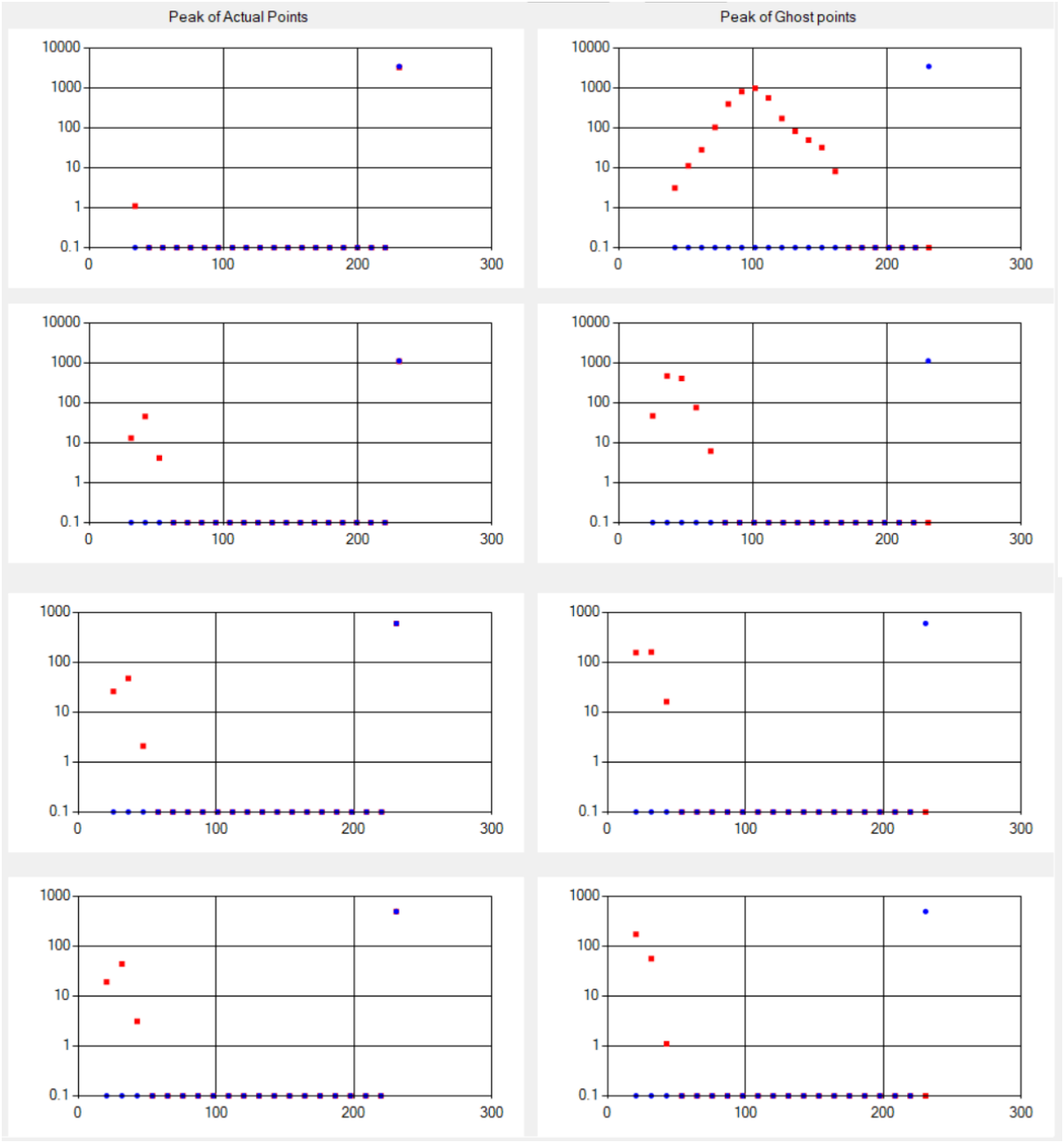

Figure 4.39: Peak histogram of Actual and Ghost points across 4 datasets

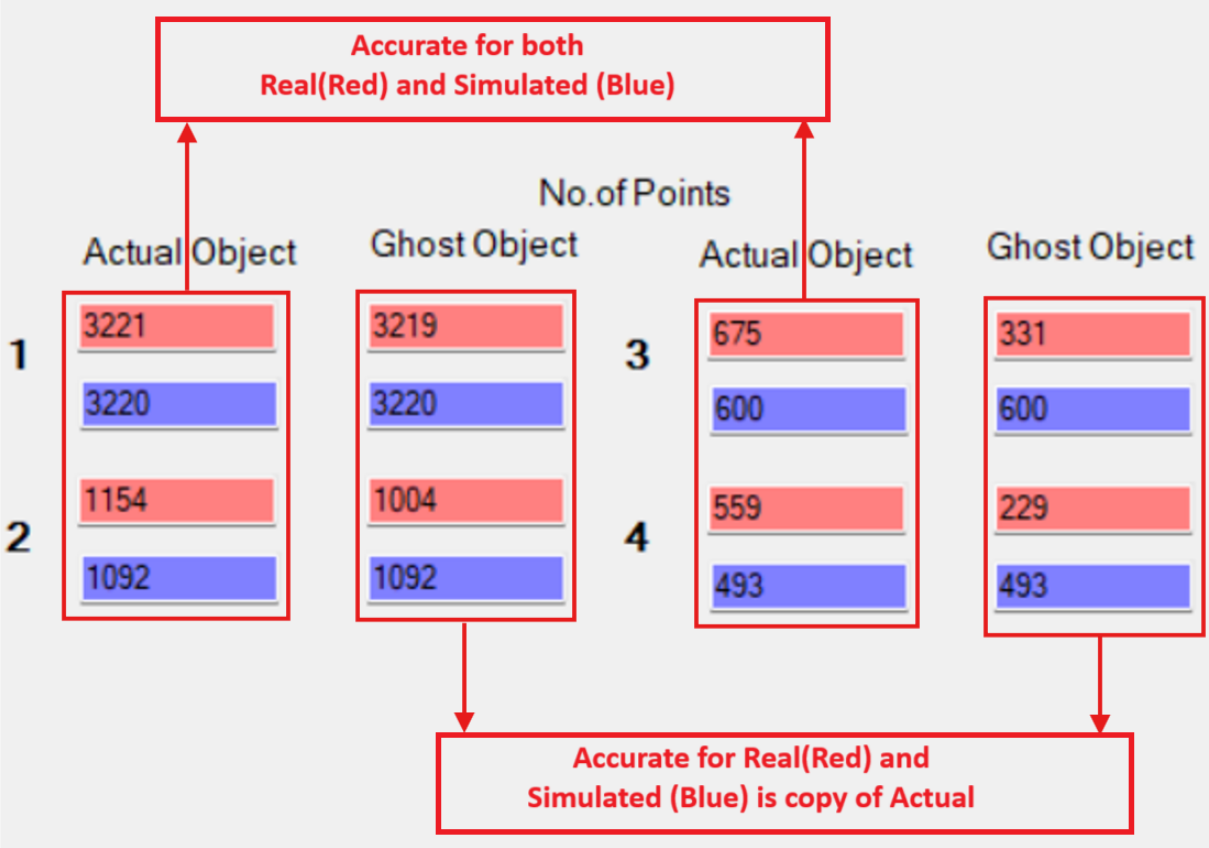

Figure 4.40: Point count of Actual and Ghost objects across 4 datasets

Figure 4.41: Simulated actual traffic sign

As shown in Figure 4.41, the actual traffic sign has been accurately simulated by the tool. However, the simulated traffic sign appears slightly behind the real one. This shift occurs because the distance is computed as the average of all points corresponding to the actual traffic sign, including several points corresponding to noise located behind it. Consequently, the resulting simulated sign is offset slightly behind the actual position. Furthermore, as previously noted, the ghost traffic sign is not present in the simulation since a stub function is currently used and the full functionality has not yet been implemented.

As we can see from the Area histogram in Figure 4.39, the distribution of the real data for the actual traffic sign is more evenly dispersed, whereas the simulated data shows a sharp concentration at a single point. Several factors may contribute to this difference. One reason could be the inaccurate or unknown material ID used in the simulation, which affects the reflectivity and spread of the point cloud. Another possibility is the presence of letters or symbols on the real traffic sign, which may introduce subtle surface variations and irregular reflections that are not captured in the simulated model. Additionally, background noise points behind the traffic sign could broaden the distribution in the real data. These issues will be further explored and addressed in the future scope to improve the fidelity of the simulation and enhance artefact comparison accuracy.

4.4.8 Ghost Point Extraction Algorithm

In order to assess the impact of ghost particles in LiDAR data, it is essential to extract and isolate these artifacts from the actual first reflection. The Ghost Point Extraction Algorithm aims to distinguish genuine traffic sign reflections from ghost artifacts by applying a combination of distance-based detection, clustering, and targeted scene refinement. This selective approach ensures that only the real traffic sign and its associated ghost particles are preserved, allowing for focused and meaningful analysis.

The extraction algorithm is structured into three major stages:

1. Distance-based identification of ghost particles
2. Clustering and classification of real versus ghost objects
3. Refinement of the scene by eliminating unrelated environmental data

Distance-Based Identification of Ghost Particles

In the context of traffic signs, ghost particles exhibit highly predictable behavior: they consistently appear at twice the distance of the actual object and have approximately double the physical dimensions. This repeatable pattern allows for a systematic detection method, enabling efficient isolation of these artifacts from the surrounding point cloud.

To initiate the process, a reference traffic sign is identified within the scene. Its real-world dimensions and location are used to calculate the expected position of the corresponding ghost object precisely double the distance from the LiDAR sensor as seen in Figure 4.42. The algorithm then scans the point cloud for clusters located at this anticipated ghost location. If a candidate cluster exhibits the correct spatial alignment and demonstrates a geometric size approximately twice that of the actual traffic sign, it is flagged as a ghost particle cluster.

In addition to geometric consistency, intensity values are also evaluated. Ghost particles, typically resulting from multipath or retroreflective phenomena, often possess

lower intensity due to energy losses in secondary reflections. This property serves as an auxiliary metric for confirming ghost particle identity.

This stage capitalizes on the structured nature of ghosting in traffic signs, making the detection robust, reproducible, and suitable for downstream quantitative analysis.

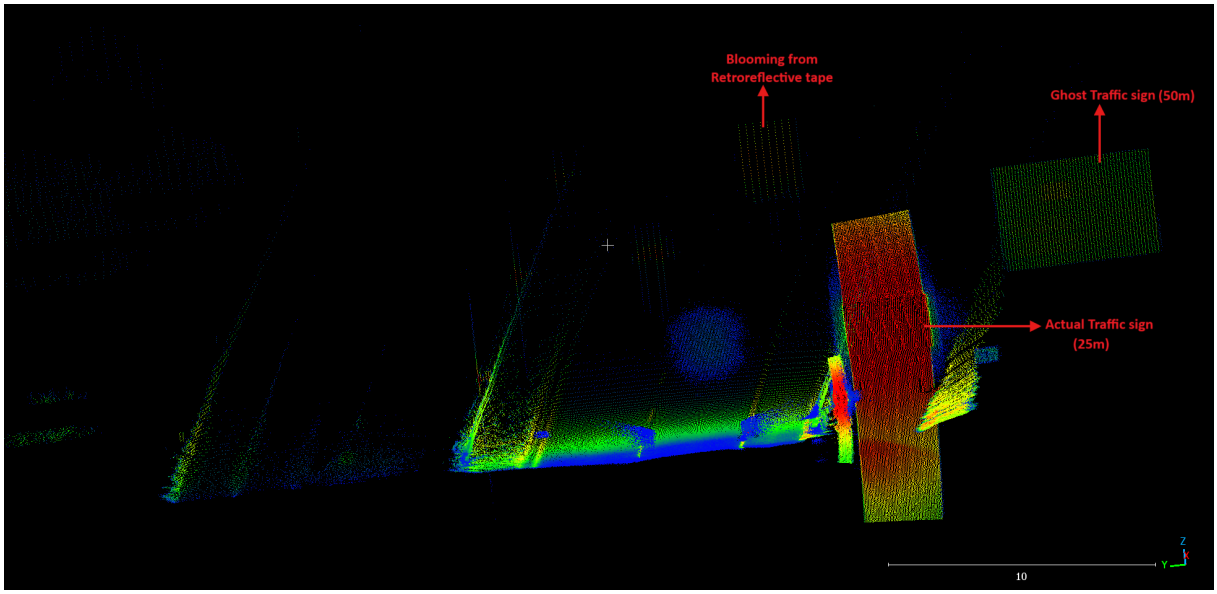

Figure 4.42: Complete Scene with Traffic Sign and Ghost Points

Clustering and Classification of Real and Ghost Objects

Following the identification of potential ghost locations, a clustering process is employed to distinguish the true traffic sign from its ghost counterpart. Given that ghost objects are geometrically scaled duplicates, the classification must consider both shape and location simultaneously.

The method begins by defining bounding volumes for the real and expected ghost regions. A Density Based Spatial Clustering Algorithm (DBSCAN) is applied to the filtered point cloud to form distinct clusters based on point proximity. Each cluster is then compared to the expected dimensions: clusters matching the real traffic sign's size are labeled as real, while those at twice the distance and scale are marked as ghost objects.

Additionally, the clustering process applies thresholding techniques to discard small or noisy clusters that do not conform to the expected shape of a traffic sign. This ensures that only structured, meaningful clusters are retained for further processing.

The use of both geometric validation and intensity-based differentiation further refines the classification, providing a reliable distinction between actual and ghosted objects.

Scene Refinement and Data Structuring

The final phase of the algorithm is aimed at purifying the point cloud scene, retaining only the traffic sign and its associated ghost particles for focused study. All extraneous elements such as vehicles, road textures, vegetation, and distant background objects are removed.

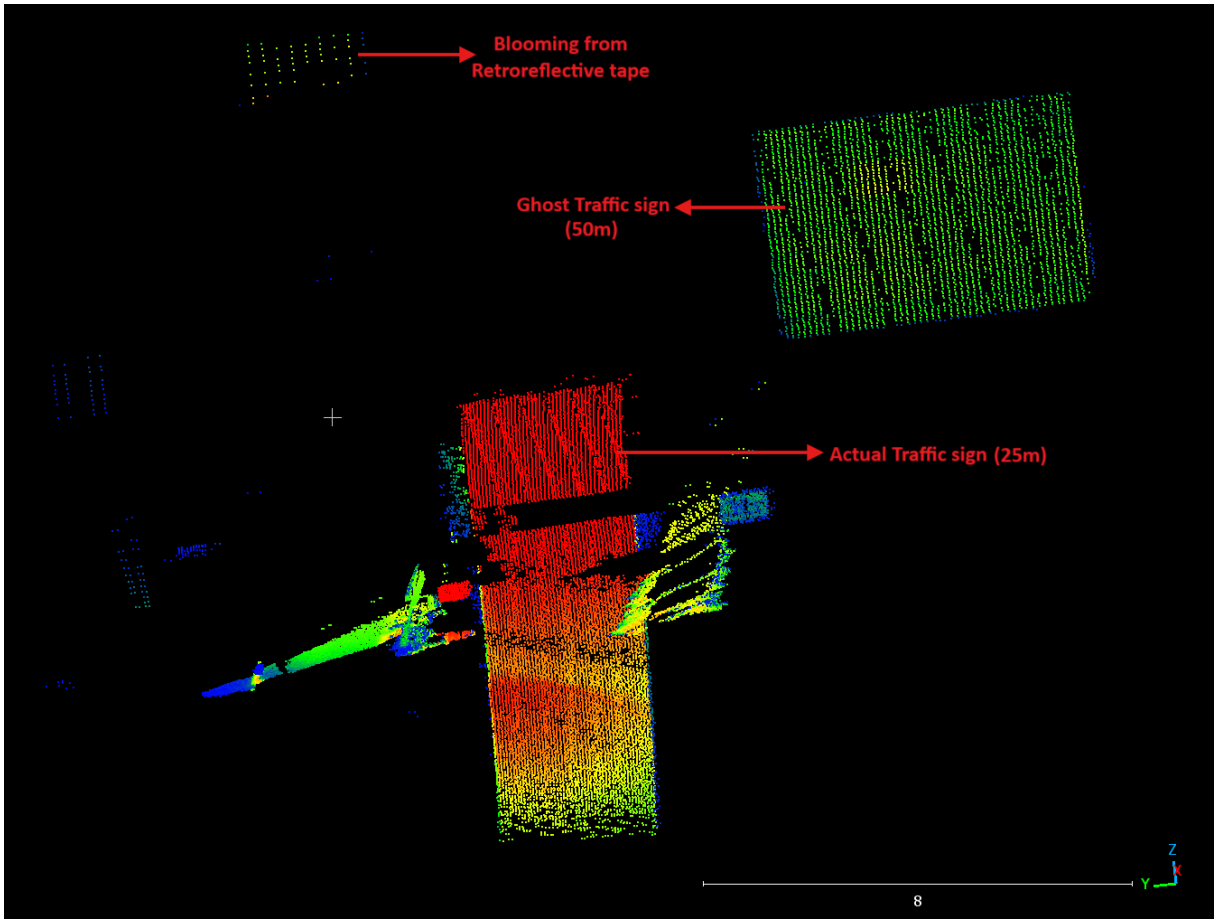

Figure 4.43: Isolated Traffic Sign and Ghost Points After Filtering

This refined point cloud now represents a minimalist dataset containing just the real traffic sign and its ghost image as seen in Figure 4.43. It is exported in a *.pcd format that supports direct visualization, simulation comparison, and histogram-based analysis.

The benefit of this step is twofold: it enhances clarity in the visual and statistical evaluation of ghosting behavior, and it provides a consistent input for validating simulated LiDAR outputs under similar conditions.

4.5 Automation in Calibration Workflow

To streamline the LiDAR sensor calibration process and minimize manual intervention, automation plays a critical role in ensuring consistency, scalability, and reproducibility of

results. As the volume of data and complexity of simulation scenarios increase, manual configuration and data handling can become error-prone and time-consuming. Integrating automation into the calibration workflow enables efficient processing of multiple datasets, adaptive parameter handling, and seamless tool operation across diverse testing environments. This section outlines key automation strategies implemented in the calibration tool to enhance usability and reliability.

4.5.1 Co-Efficient Management and Automation for Blooming Calibration

Initial Implementation: Hardcoded Parameters

In the earlier version of the tool, key parameters for each dataset such as distance, IndexVerMin, IndexVerMax, SegVerCenter, SegHor (for vertical blooming) and distance, IndexHorMin, IndexHorMax, SegHorCenter, SegVer (for horizontal blooming) were hardcoded directly in the code.

Limitations of Hardcoded Parameters:

- **Lack of flexibility:** Any modifications required changes to the source code, making updates time-consuming.
- **Manual effort:** Each time a new dataset was introduced, parameters had to be manually adjusted in the script.
- **Difficult to fine-tune:** Calibration adjustments had to be done externally before re-running the tool, leading to inefficiencies.

Enhancement: Moving Parameters to the GUI

To improve flexibility and usability, all these parameters were moved to the GUI, allowing each dataset (corresponding to different distances) to have its own set of parameters that can be updated directly within the interface.

Key Modifications:

- Each distance now has independent parameter controls in the GUI.
- Users can adjust parameters dynamically instead of modifying the source code.
- Parameter values are stored externally, eliminating the need for hardcoding.

Automation Script for JSON-Based Parameter Management

To further streamline the process, an automation script was developed to store these parameters in a JSON file, ensuring that:

- All parameters are saved persistently and can be reloaded whenever needed.

- Consistency is maintained across SMART and SLIM variant sessions.
- Users can modify parameter values either through the GUI or directly within the JSON file.

When the GUI loads, it checks for the presence of the JSON file:

- If no JSON file is found in the source path, the tool initializes with default parameter values from the GUI.
- If a JSON file exists, the stored values are automatically loaded into the GUI, ensuring that previously saved configurations are used.

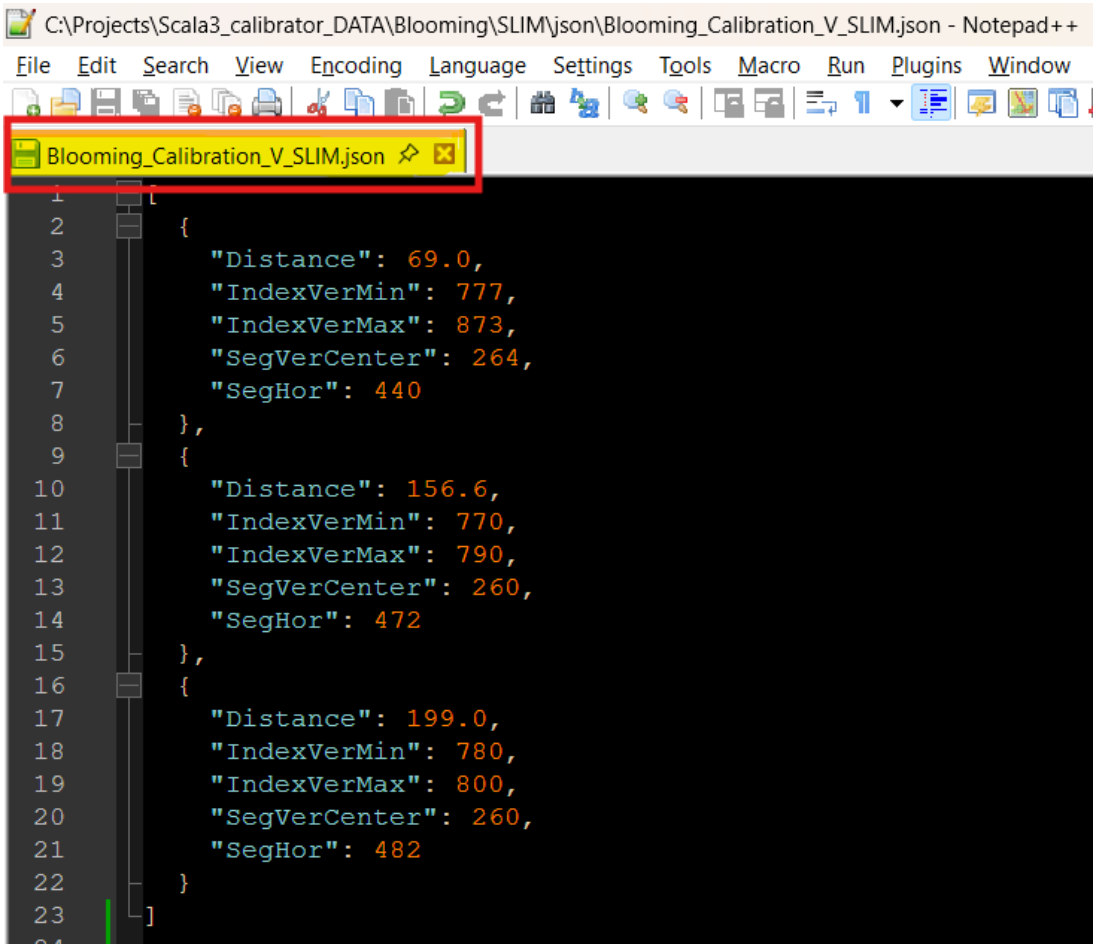

Figure 4.44: JSON file of Vertical Blooming

GUI Functionality: Update JSON and Reload JSON Buttons

To provide user control over the stored parameters, two additional buttons were introduced as seen in Figure 4.45:

1. **Update JSON:**

- Saves the current values of all parameters (for all distances) from the GUI into the JSON file.
- Ensures that any modifications made within the GUI are stored persistently.
- The updated parameters will automatically be loaded the next time the GUI is started, eliminating the need for manual reconfiguration.

2. **Reload JSON:**
   - Reads the latest values from the JSON file and updates the GUI fields accordingly.
   - Useful when parameter values are modified directly in the JSON file, allowing users to refresh and apply changes without restarting the GUI.
   - Enables quick switching between different pre-configured parameter sets stored in the JSON.

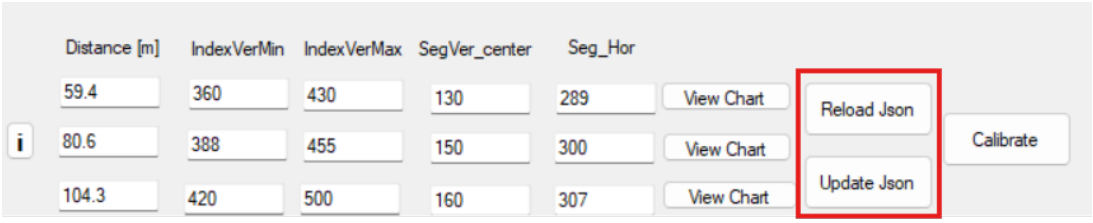

Figure 4.45: Reload and Update JSON buttons in GUI

Outcome and Benefits

- **Dynamic parameter adjustment:** Users can fine-tune parameters without modifying the code.
- **Persistent storage:** Settings are saved and restored automatically, ensuring continuity across sessions.
- **Improved efficiency:** No need to re-enter values manually for every session.
- **User-friendly workflow:** Simplified parameter management through the GUI with the ability to update or reload values as needed.

4.5.2 Automating Header Processing for Efficient Data Handling

LiDAR data used in this thesis was provided as text files containing point cloud information, structured in columns representing attributes such as X, Y, Z, peak, area, and width. These files were generated using the CloudCompare tool, which includes a header in the first line to define the sequence of the data fields. However, the existing code relied on a fixed sequence for the data, and any deviation from this order would result in processing errors.

To address this limitation, an automation task was undertaken to dynamically detect and process headers, ensuring the data could be correctly interpreted regardless of the sequence specified in the header. This approach eliminated dependency on a fixed format, thereby improving robustness and flexibility in handling diverse datasets.

Implementation Approach

The task was approached by designing a flexible algorithm capable of:

Reading and Parsing Headers: The first line of each file was read to identify the order of attributes, such as X, Y, Z, Point, Layer, EchoNumber, WIDTH, AREA and PEAK as seen in Figure 4.46. The algorithm matched these attributes to their corresponding indices, creating a mapping to guide further data processing.

Dynamic Data Parsing: Once the header sequence was extracted, subsequent lines of point cloud data were processed based on the identified order. This ensured that the attributes were correctly aligned, even when the sequence in the header varied as seen in Figure 4.47.

Error Handling: For headers with missing or unexpected attributes, the program logged the discrepancies and either used default values or skipped those fields to maintain continuity.

Outcome and Benefits

The automation successfully eliminated the dependency on a fixed header sequence, allowing seamless processing of diverse point cloud datasets. Key outcomes included:

- **Enhanced robustness** in handling files with varied header formats.
- **Reduction in manual effort and processing errors.**
- **Improved scalability and adaptability** for large-scale data handling.

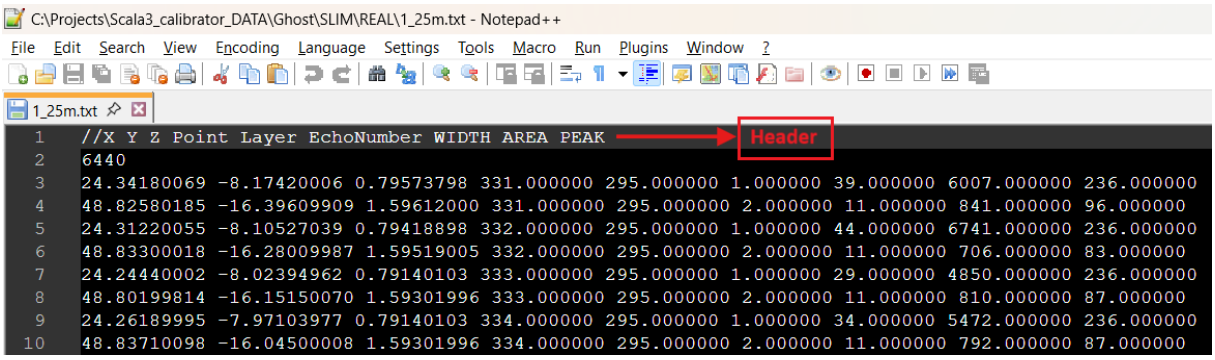

Figure 4.46: Header Type 1

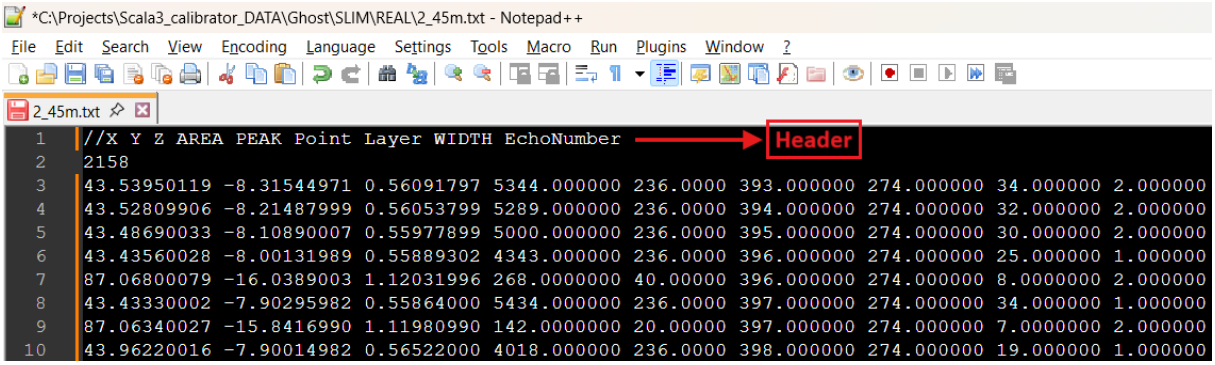

Figure 4.47: Header Type 2

Figure 4.48: Code Snippet to process data based on Header format

5 Conclusion

This thesis focused on the development of a tool that is capable of calibrating and validating simulated LiDAR sensor data in the context of autonomous driving systems at Valeo Schalter und Sensoren GmbH. The primary objective was to bridge the gap between simulated and real-world LiDAR point cloud data by developing a tool that visualizes discrepancies, supports calibration, and validates key sensor behaviors such as noise, blooming, and ghost points. The developed tool offers a practical approach to improving simulation accuracy without altering the sensor model directly, thereby supporting more reliable algorithm development and system validation processes.

The journey of this thesis began with a comprehensive literature review aimed at understanding the necessity of LiDAR in autonomous driving systems, especially when compared to conventional sensors like radar and cameras. This exploration highlighted LiDAR's superior depth perception capabilities and its ability to provide dense 3D environmental data, essential for high-level autonomy. In addition to comparing sensor modalities, the literature review also provided a technical foundation in LiDAR operation, covering topics such as the underlying power budget equation and the functionality of SPAD arrays. These insights were crucial for building a strong understanding of the sensor's performance characteristics and limitations, which informed the later stages of simulation analysis and calibration.

The next phase of the thesis focused on developing the concept of the calibration solution, building upon an existing open-loop setup that allowed only static, one-time adjustments without any feedback mechanism. To overcome the limitations of this approach, a closed-loop setup was introduced, enabling continuous comparison between simulated and real LiDAR data. The core calibration tool was structured around this feedback-driven loop, allowing for iterative parameter adjustments and real-time analysis. This closed-loop design made it possible to systematically refine the simulation outputs, resulting in significantly improved alignment with real-world LiDAR behavior.

Following the establishment of the closed-loop calibration framework, the focus shifted to identifying the most critical artefacts that must be accurately simulated to achieve realistic LiDAR behavior. After careful analysis, noise, blooming, and ghost particles were selected as the primary artefacts for simulation, given their significant impact on point cloud accuracy. To visualize and compare these artefacts effectively, a histogram-based algorithm was employed. This method allowed for a structured representation of point distributions, where key metrics such as peak, area, and width were analyzed to quantify the differences between real and simulated data, providing clear insights into the effectiveness of the calibration.

The thesis then transitioned into the implementation phase of the proposed solution. The first crucial step involved selecting appropriate real-world point cloud scenarios based on the type of artefact under investigation. For analyzing noise, point cloud data captured in a controlled lab environment was used, while for studying blooming and ghost particles, scenarios involving highly reflective surfaces such as traffic signs were chosen due to their tendency to trigger these artefacts. Once the relevant scenarios were identified, the layout of the GUI was designed to accommodate the visualization of key metrics. The implementation focused on enabling intuitive data analysis through histograms, ensuring that peak, area, width and detection probability characteristics could be clearly observed and compared across datasets.

In addition to data visualization, few automation features were integrated into the tool to streamline the calibration process. Notably, for blooming calibration, a helper module was developed and integrated as an extension to the main tool. This addition significantly simplified the calibration workflow by providing user-friendly parameter modifications, ultimately saving time and reducing manual complexity. Another key enhancement involved improving the tool's ability to handle point cloud data in text format. Initially, the tool required the header of the file to follow a predefined order. To make the tool more robust and user-friendly, the functionality was added to dynamically read and interpret the header, allowing it to process data regardless of the column arrangement, thus enhancing flexibility and usability in diverse data scenarios.

In conclusion, this thesis successfully demonstrates that meaningful calibration and validation of LiDAR simulations can be achieved through a software-based tool that emphasizes visual analysis, modular comparison techniques, and automation-driven usability. The tool not only aids in identifying discrepancies between real and simulated data but also streamlines the calibration process, thereby reducing complexity and saving development time. By enhancing simulation fidelity, it supports the development of robust perception algorithms and minimizes dependency on costly real-world testing. These outcomes contribute directly to the validation workflow at Valeo Schalter und Sensoren GmbH and lay a solid foundation for future extensions, including integration with automated calibration pipelines or machine learning-based refinement techniques to further improve efficiency and accuracy of the sensor models.

5.1 Outlook

The current tool offers a strong foundation for calibrating and validating simulated LiDAR data and opens up several promising directions for future enhancement. One significant next step involves completing the simulation and analysis of ghost artefacts. While this thesis laid the foundation for ghost particle detection by establishing an analysis pipeline and identifying the expected behavior of such artefacts, the actual implementation of ghost generation within the simulation is still pending. Incorporating a reliable ghost simulation mechanism would enable end-to-end validation for one of the most challenging artefacts in LiDAR perception.

Another area requiring attention is the presence of walk error in the real LiDAR data, which has not yet been accounted for in the simulation. Walk error, caused by signal detection delays depending on return pulse strength can lead to distance inaccuracies, particularly for weak or slanted returns. Modeling and compensating for this effect in future iterations of the simulation will be essential for achieving higher fidelity and ensuring that simulated data accurately mirrors real-world sensor behavior across varying reflectivity and angle conditions.

While several features of the calibration tool have already been automated, such as dynamic parsing of point cloud headers, loading pre-configured data using JSON scripts and integration of helper modules for blooming calibration, there is still room to make the workflow even more user-friendly. In the future, more steps in the calibration process could be automated, such as adjusting parameters based on observed results. This would reduce the need for manual changes and make the tool easier and quicker to use, especially when working with multiple datasets or scenarios.

This thesis established that histogram-based methods offer a clear and effective way to analyze and compare LiDAR artefacts. However, the tool primarily provides a qualitative means of comparison through histogram visualizations, which rely on user interpretation. To strengthen the analytical accuracy, future versions of the tool could incorporate quantitative metrics or KPIs that numerically express the degree of match between real and simulated data. This would enable a standardized evaluation of calibration quality, facilitate benchmarking across scenarios, and support more objective decision-making during the validation process.

Finally, the simulation framework could be extended to include additional complex artefacts such as multi-path reflections, environmental effects like fog or rain, and dynamic scene elements that introduce motion blur. By enriching the artefact library and simulating edge cases more accurately, the tool would become even more valuable for developing and validating perception algorithms in real-world autonomous driving conditions.

In conclusion, the future work outlined here presents a roadmap for continuous improvement and innovation in the field of LiDAR sensor simulation and calibration. By addressing these areas, the tool can evolve to become more accurate, automated, and adaptable, ultimately enhancing the reliability of perception systems and contributing to the advancement of autonomous driving technologies in the automotive industry.

Bibliography

[1] AUTOMOTIVE, IPG: *CarMaker: Virtual test driving solution*. https://ipg-automotive.com/products-services/simulation-software/carmaker. – Accessed: 2025-02-10

[2] BIMBRAW, KESHAV : *Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology*. https://www.researchgate.net/publication/283757446_Autonomous_Cars_Past_Present_and_Future_-_A_Review_of_the_Developments_in_the_Last_Century_the_Present_Scenario_and_the_Expected_Future_of_Autonomous_Vehicle_Technology/citation/download. – Accessed: 2025-03-09

[3] CONTRIBUTORS, Wikipedia: *Single-photon avalanche diode*. https://en.wikipedia.org/wiki/Single-photon_avalanche_diode. – Accessed: 2025-01-30

[4] EULERPOOL: *Valeo Employees 2025*. https://eulerpool.com/en/stock/Valeo-Stock-FR0013176526/Employees. – Accessed: 2025-03-25

[5] GIRARDEAU-MONTAUT, D.: *CloudCompare - 3D point cloud and mesh processing software*. https://www.danielgm.net/cc. – Accessed: 2025-03-29

[6] GMBH, PicoQuant: *Time-Correlated Single Photon Counting*. https://www.picoquant.com/images/uploads/page/files/7253/technote_tcspc.pdf. – Accessed: 2025-03-05

[7] GOGREEN: *Autonomous Driving*. https://www.gogreenleasing.co.uk/blog/the-5-levels-of-vehicle-automation. – Accessed: 2025-01-19

[8] HEIDARI, Sam: *How Next-Gen LiDAR Technology Is Reshaping Autonomous Systems*. https://www.eetimes.eu/lidar-evolution-how-next-gen-technology-is-reshaping-autonomous-systems/. – Accessed: 2025-04-02

[9] HENLEY, Connor ; SOMASUNDARAM, Siddharth ; HOLLMANN, Joseph ; RASKAR, Ramesh: *Detection and Mapping of Specular Surfaces Using Multibounce Lidar Returns*. https://arxiv.org/abs/2209.03336. – Accessed: 2025-02-28

[10] HEXAGON: *SCALA® 3D LASER SCANNER GEN 1*. https://hexagondownloads.blob.core.windows.net/public/AutonomouStuff/wp-content/uploads/2020/10/valeo-scala-datasheet-whitelabel.pdf. – Accessed: 2025-03-07

[11] HEXAGON: *Valeo SCALA Gen 2*. https://autonomoustuff.com/products/valeo-scala-gen-2. – Accessed: 2025-04-08

[12] JAHROMI, Sahba ; KOSTAMOVAARA, Juha: *Timing and probability of crosstalk in a dense CMOS SPAD array in pulsed TOF applications*. https://pubmed.ncbi.nlm.nih.gov/30119371/. – Accessed: 2025-04-04

[13] LAB, MIT M.: *Detecting and Mapping Transparent or Mirror-like Surfaces with Lidar*. https://www.media.mit.edu/projects/detecting-and-mapping-transparent-or-mirror-like-surfaces-with-lidar/overview/. – Accessed: 2025-01-18

[14] LEE, G. ; CHEON, J. ; LEE, I.: *VALIDATION OF LIDAR CALIBRATION USING A LIDAR SIMULATOR*. https://isprs-archives.copernicus.org/articles/XLIII-B1-2020/39/2020/. – Accessed: 2025-01-08

[15] LI, Wentao ; SHI, Tianyun ; WANG, Rui ; YANG, Jingjie ; MA, Zhen ; ZHANG, Wanpeng ; FU, Huijin ; GUO, Pengyue: *Advances in LiDAR Hardware Technology: Focus on Elastic LiDAR for Solid Target Scanning*. https://www.mdpi.com/1424-8220/24/22/7268. – Accessed: 2025-02-15

[16] LTD, Level Five S.: *Valeo Mobility Kit – SCALA 3D LiDAR gen 2*. https://levelfivesupplies.com/product/scala-3d-laser-scanner-gen-2/. – Accessed: 2025-02-12

[17] MAPIX TECHNOLOGIES: *3D LiDAR sensor developed for automotive industry*. https://www.mapix.com/lidar-scanner-sensors/velodyne/velodyne-vlp-32c/. – Accessed: 2025-04-12

[18] MCMANAMON, Paul: *Field Guide to Lidar*. https://www.lehmanns.de/shop/technik/32231526-9781628416541-field-guide-to-lidar. – Accessed: 2025-04-17

[19] MICROSOFT: *Visual Studio IDE*. https://visualstudio.microsoft.com. – Accessed: 2025-03-05

[20] MICROSOFT: *Windows Forms Overview*. https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview. – Accessed: 2025-05-05

[21] NASSI, Ben ; MIRSKY, Yisroel ; SHAMS, Jacob ; BEN-NETANEL, Raz ; NASSI, Dudi ; ELOVICI, Yuval: *Protecting Autonomous Cars from Phantom Attacks*. https://cacm.acm.org/research/protecting-autonomous-cars-from-phantom-attacks/. – Accessed: 2025-05-02

[22] NIDEC COMPONENTS: *Various Sensors required for Autonomous Driving*. https://www.nidec-components.com/e/featuring/lidar-polygon/various-sensors/. – Accessed: 2025-04-11

[23] OPENPR : *Autonomous Driving*. https://www.openpr.com/news/2861808/at-a-cagr-of-18-2-advanced-driver-assist-systems-adas-market. – Accessed: 2025-05-03

[24] REDDIT: *Valeo announces 2 new major contracts for its third-generation LiDAR*. https://www.reddit.com/r/MVIS/comments/17f2yis/valeo_announces_2_new_major_contracts_for_its/?rdt=46296. – Accessed: 2025-03-13

[25] SENSOREN GMBH, Valeo S.: *Long-range LiDAR Sensors – Valeo's SCALA™ Gen 3*. https://www.valeo.com/en/catalogue/cda/long-range-lidar-sensors-valeo-scala-gen-3/. – Accessed: 2025-03-25

[26] SHUTTERSTOCK: *N/A*. https://www.shutterstock.com/image-photo/green-overhead-road-sign-welcome-germany-533279011. – Accessed: 2025-05-06

[27] SMARTOSC: *The future of Automotive Software Development : AI, IoT and Cloud Integration*. https://www.smartosc.com/the-future-of-automative-software-development/. – Accessed: 2025-03-25

[28] STROUSTRUP, B.: *The C++ Programming Language, 4th ed.* https://www.informit.com/store/c-plus-plus-programming-language-9780321563842. – Accessed: 2025-04-09

[29] VALEO: *SCALA™ LiDAR*. https://smartworkplace.apps.valeo.com/comfort-driving-assistance-portal/ls/content/7082790704640625/our-activities/adas-sensors/adas-and-active-safety/scala. – Accessed: 2025-02-01

[30] VALEO: *Valeo SCALA™ LiDAR*. https://www.valeo.com/en/valeo-scala-lidar/. – Accessed: 2025-01-30

[31] VALEO SCHALTER UND SENSOREN GMBH : *Sustainability in the Automotive Industry*. https://www.valeo.com/en/sustainability/. – Accessed: 2025-02-24

[32] VALEO SCHALTER UND SENSOREN GMBH : *Valeo Brain Division*. https://www.valeo.com/en/brain-division/. – Accessed: 2025-05-01

[33] VALEO SCHALTER UND SENSOREN GMBH : *Valeo Light Division*. https://www.valeo.com/en/light-division/. – Accessed: 2025-01-20

[34] VALEO SCHALTER UND SENSOREN GMBH : *Valeo Power Division*. https://www.valeo.com/en/power-division/. – Accessed: 2025-04-01

[35] VALEO SCHALTER UND SENSOREN GMBH : *Valeo SCALA™ LiDAR*. https://www.valeo.com/en/valeo-scala-lidar/. – Accessed: 2025-04-22

[36] VALEO SCHALTER UND SENSOREN GMBH: *Valeo: A Global Leader in Automotive Solutions*. https://www.valeo.com/. – Accessed: 2025-04-05

[37] WAABI: *A Comprehensive Study of Realistic LiDAR Simulation for Autonomy*. https://waabi.ai/lidar-dg/. – Accessed: 2025-04-10

[38] WANG, Xin ; PAN, HuaZhi ; GUO, Kai ; YANG, Xinli ; LUO, Sheng: *The evolution of LiDAR and its application in high precision measurement*. https://www.researchgate.net/publication/341831431_The_evolution_of_LiDAR_and_its_application_in_high_precision_measurement. – Accessed: 2025-01-25

[39] WIKIPEDIA: *Multipath Propagation*. https://en.wikipedia.org/wiki/Multipath_propagation. – Accessed: 2025-03-20

[40] WIKIPEDIA CONTRIBUTORS : *Advanced driver-assistance system — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Advanced_driver-assistance_system&oldid=1173388522. – Accessed: 2025-04-18

[41] Y, Marc.: *Apple LIDAR Demystified: SPAD, VCSEL, and Fusion*. https://4sense.medium.com/apple-lidar-demystified-spad-vcsel-and-fusion-aa9c3519d4cb. – Accessed: 2025-03-17

[42] YANG, Boquan ; LI, Jixiong ; ZENG, Ting: *A Review of Environmental Perception Technology Based on Multi-Sensor Information Fusion in Autonomous Driving*. https://www.mdpi.com/2032-6653/16/1/20. – Accessed: 2025-02-17

[43] YOLE GROUP: *First and second generation Valeo Scala LiDARs: the technology gap revealed*. https://www.yolegroup.com/technology-outlook/first-and-second-generation-valeo-scala-lidars-the-technology-gap-revealed/. – Accessed: 2025-05-03

[44] YU, Yongtao ; WANG, Yuxin ; HOU, Yifan ; WU, Zhenyu ; ZHANG, Xinyu ; SHI, Zhenwei: *A Review of Dynamic Object Filtering in SLAM Based on 3D LiDAR*. https://www.mdpi.com/1424-8220/24/2/645. – Accessed: 2025-01-22

[45] ZHANG, Zhipeng: *LiDAR-Based 3D Object Detection and Tracking for Autonomous Driving*. https://dr.ntu.edu.sg/bitstream/10356/174239/2/Thesis_Zhipeng_final.pdf. – Accessed: 2025-03-09