HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION

GRADUATION PROJECT

DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION

LÊ HOÀNG TIẾN
Student ID: 21145617

LÊ HỒ MINH KHOA
Student ID: 21145019

Major: AUTOMOTIVE ENGINEERING

Supervisor: TRẦN VŨ HOÀNG, PhD.

Ho Chi Minh City, May 2025

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION

GRADUATION PROJECT

DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION

LÊ HOÀNG TIẾN
Student ID: 21145617

LÊ HỒ MINH KHOA
Student ID: 21145019

Major: AUTOMOTIVE ENGINEERING

Supervisor: TRẦN VŨ HOÀNG, PhD.

Ho Chi Minh City, May 2025

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, January 18th, 2025

GRADUATION PROJECT ASSIGNMENT

Student name: LÊ HOÀNG TIẾN — Student ID: 21145617

Student name: LÊ HỒ MINH KHOA — Student ID: 21145019

Major: Automotive Engineering Technology — Class: 21145FIE4 & 21145FIE1

Supervisor: TRẦN VŨ HOÀNG, Ph.D. — Phone number: 0988757515

Date of assignment: January 18th, 2025 — Date of submission: May 30th, 2025

1. Project title: DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION
2. Initial materials provided by supervisor: project scope document; key research papers on camera–LiDAR fusion, evaluation criteria
3. . Content of the project:
   - Conduct a comprehensive review of 3D object-detection techniques in traffic scenarios and identify key requirements.
   - Design and implement an end-to-end pipeline for multimodal data fusion, from data preprocessing through model inference and performance testing.
   - Develop a visualization and analysis application to present detection results, perform quantitative evaluation, and refine the overall workflow.
4. Final product:
   - A prototype system comprising a proposed end-to-end multimodal data-fusion pipeline for 3D object detection
   - An interactive application for visualizing, evaluating, and exporting 3D object-detection results.

CHAIR OF THE PROGRAM
(Sign with full name)

Vũ Đình Hoàn

SUPERVISOR
(Sign with full name)

Trần Vũ Hoàng

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDU
FACULTY OF INTERNATIONAL EDUCATI
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom- Happiness

SUPERVISOR'S EVALUATION SHEET

Student name: LÊ HOÀNG TIẾN Student ID: 21145617

Student name: LÊ HỒ MINH KHOA Student ID: 21145019

Major: Automotive Engineering Technology

Project title: DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION

Supervisor: TRẦN VŨ HOÀNG, Ph.D.

EVALUATION

1. Content of the project: the content and workload of the project is suitable for an undergraduate thesis.
2. Strengths:

The student has appropriately selected and designed the system. The student has also conducted evaluations on multiple aspects of the system.

3. Weaknesses:

The system still requires a relatively long processing time and has high demands on processing hardware.

4. Approval for oral defense? *(Approved or denied)*

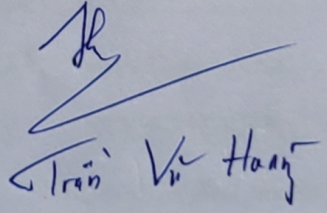
   Approved
5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

   Good
6. Mark: 9 (*in words*: nine)

Ho Chi Minh City, May 27th, 2025

SUPERVISOR

(Sign with full name)



HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDU
FACULTY OF INTERNATIONAL EDUCATI
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom- Happiness

PRE-DEFENSE EVALUATION SHEET

Student name: LÊ HOÀNG TIẾN Student ID: 21145617

Student name: LÊ HỒ MINH KHOA Student ID: 21145019

Major: Automotive Engineering Technology

Project title: DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION

Name of Examiner: .....

EVALUATION

1. Content and workload of the project

.....

Good for undergraduate student requirements

.....

.....

2. Strengths:

Evaluted the system in varias cases

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? *(Approved or denied)*

Approved.

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

Good

6. Mark: 8.8 (*in words*: Eight point eight)

Ho Chi Minh City, June ... , 2025

EXAMINER

(Sign with full name)

Nguyễn Trọng Thắng

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDU
FACULTY OF INTERNATIONAL EDUCATI
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom- Happiness

EVALUATION SHEET OF
DEFENSE COMMITTEE MEMBER

Student name: LÊ HOÀNG TIẾN Student ID: 21145617

Student name: LÊ HỒ MINH KHOA Student ID: 21145019

Major: Automotive Engineering Technology

Project title: DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION

Name of Defense Committee Member:

..... Vũ Đình Huân .....

EVALUATION

1. Content and workload of the project

.....

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

5. Mark: ..... (*in words*:.....)

Ho Chi Minh City, June ... , 2025

COMMITTEE MEMBER

(Sign with full name)

Vũ Đình Huân

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN CAM KẾT VÀ XÁC KIỂM TRA ĐẠO VĂN (DÀNH CHO BÁO CÁO NGHIÊN CỨU KHOA HỌC SINH VIÊN, KHÓA LUẬN, LUẬN VĂN, LUẬN ÁN)

Mẫu 1
(Kèm theo QĐ số 1047/QĐ-ĐHSPKT ngày 14 tháng 3 năm 2022

I. Thông tin chung

1. Tên sản phẩm học thuật: DEVELOPMENT OF A MULTIMODAL 3D OBJECT RECOGNITION AND VISUALIZATION SYSTEM FOR TRAFFIC ENVIRONMENTS USING CAMERA-LIDAR FUSION

2. Loại hình sản phẩm học thuật (Báo cáo nghiên cứu khoa học sinh viên/khóa luận tốt nghiệp/luận văn thạc sĩ/luận án tiến sĩ): Khoá luận tốt nghiệp

2. Mã số sản phẩm học thuật (nếu có):

3. Thông tin tác giả (ghi tất cả tác giả của sản phẩm)

Stt	Họ và tên	MSSV/MSHV	Vai trò (Chủ nhiệm/thành viên/tác giả chính/đồng tác giả…)
1	Lê Hoàng Tiến	21145617	Đồng tác giả
2	Lê Hồ Minh Khoa	21145019	Đồng tác giả

4. Thông tin giảng viên hướng dẫn

Họ và tên: TRẦN VŨ HOÀNG MSCB: 7042

Khoa: Khoa Điện - Điện tử

II. Kết quả kiểm tra đạo văn

Ngày nộp sản phẩm	Ngày kiểm tra đạo văn	% trùng lặp toàn nội dung	% trùng lặp cao nhất từ 1 nguồn
27/05/2025	27/25/2025	1%	1%

Lưu ý: % trùng lặp nêu ở bảng trên không tính % trùng lặp của danh mục tài liệu tham khảo.

III. Cam kết

Nhóm tác giả sản phẩm học thuật và giảng viên hướng dẫn cam kết rằng:

1. Nội dung trong sản phẩm học thuật nêu trên không vi phạm đạo đức và liêm chính khoa học.
2. Kết quả % trùng lặp nêu tại mục II là hoàn toàn chính xác và trung thực.
3. Bằng việc ký xác nhận vào mẫu này, nhóm tác giả và giảng viên hướng dẫn cam kết chịu hoàn toàn trách nhiệm có liên quan đến sản phẩm học thuật nói trên.

Xác nhận của đại diện nhóm tác giả
(ký ghi rõ họ và tên)

Lê Hoàng Tiến Lê Hồ Minh Khoa

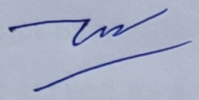
Xác nhận của giảng viên hướng dẫn
(ký ghi rõ họ và tên)

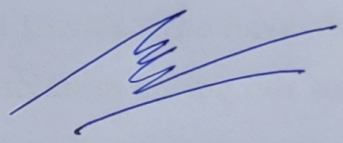
Trần Vũ Hoàng

DISCLAIMER

We affirm that this project represents our original research conducted under the supervision of our advisor, Vu Hoang Tran, Ph.D. All findings and statements presented herein are the outcome of our own dedicated, independent efforts, grounded in the study, analysis, and synthesis of scientific literature. We fully commit to upholding academic integrity by not copying or reproducing the content or results of other works. All referenced sources have been properly acknowledged and cited.

The graduation project implementation group.





Lê Hoàng Tiến

Lê Hồ Minh Khoa

ACKNOWLEDGEMENTS

First and foremost, our group would like to express our sincere gratitude to our advisor, Vu Hoang Tran, Ph.D., for his invaluable contributions and support throughout the completion of this graduation project. Thanks to his dedication and meticulous guidance, we were able to accomplish our project in the best possible way.

We would also like to extend our heartfelt appreciation to the Faculty of International Education - Ho Chi Minh City University of Technology and Education, which has provided us with the environment to acquire valuable lessons and experiences, forming the foundation for this project.

In addition, we cannot forget to mention our families and friends, who have always accompanied, supported, and served as a vital source of motivation during our studies and work at the university. Their encouragement and belief in us have helped us overcome difficulties and continuously improve ourselves.

The knowledge, skills, and support from our supervisor and everyone involved have been the solid foundation that enabled us to overcome challenges and successfully complete the project titled “Development of a Multimodal 3D Object Recognition and Visualization System for Traffic Environments Using Camera-LiDAR Fusion.”

We would like to express our deepest gratitude!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Definition
ITS	Intelligent Transportation Systems
ADAS	Advanced Driver Assistance Systems
LiDAR	Light Detection and Ranging
GPS	Global Positioning System
IMU	Inertial Measurement Unit
Hz	Hertz
CaDDN	
RADAR	Radio Detection and Ranging
CAM	Camera
AI	Artificial Intelligence
CMOS	Complementary Metal-Oxide-Semiconductor
CCD	Charge-Coupled Device
IoU	Intersection over Union
TP	True Positives
FP	False Positives
FN	False Negatives
AP	Average Precision
mAP	mean Average Precision
mATE	mean Average Translation Error
mASE	mean Average Scale Error
mAOE	mean Average Orientation Error
mAVE	mean Average Velocity Error
mAAE	mean Average Attribute Error
NDS	NuScenes Detection Score
FMCW	Frequency Modulated Continuous Wave
.pkl	Pickle file format

BEV	Bird’s-Eye View
CMT	Cross-Modal Transformer
MOAD	Modality-Agnostic Decoding
PME	Proximity-based Modality Ensemble
LC	LiDAR and Camera
L	LiDAR only
C	Camera only
VPS	Virtual Private Server
GPU	Graphics Processing Unit
RAM	Random Access Memory
UI	User Interface
USB	Universal Serial Bus
RGB	Red Green Blue

ABSTRACT

In the context of the rapid development of intelligent transportation and autonomous vehicles, accurate 3D object detection and recognition on roadways plays a vital role in ensuring operational safety and efficiency. LiDAR and camera technologies are two essential sources of data; however, integrating them to achieve optimal performance still faces significant algorithmic and computational challenges.

This study focuses on developing a 3D object detection and visualization system for road traffic applications using fused data from Cameras and LiDAR. We applied the MeFormer model based on the MMDetection3D framework and evaluate its performance on the nuScenes dataset - a benchmark with diverse real-world traffic scenes.

The system aims to accurately predict the position and category of objects such as vehicles, pedestrians, and obstacles, while visualizing the results on both camera images and LiDAR data. This contributes to enhancing safety and supporting intelligent transportation systems in handling complex road scenarios.

This research offers practical value in the fields of autonomous driving and urban traffic management by exploring the integration of camera and LiDAR data for 3D object recognition and visualization. The outcomes of this study may serve as a foundation for future research on multi-sensor integration in modern traffic systems.

CHAPTER 1: INTRODUCTION

1.1. Problem Statement

Road traffic accidents remain a critical global issue, particularly in Vietnam, where traffic density is increasing due to rapid urbanization. According to the Traffic Police Department of the Ministry of Public Security, in 2024, the national traffic safety situation recorded notable statistics: 23,484 traffic accidents occurred nationwide, resulting in 10,944 fatalities and 17,342 injuries [1]. One major contributing factor is the lack of early warning and driver assistance systems capable of accurately detecting objects such as pedestrians, bicycles, or road obstacles.

Developing intelligent transportation systems (ITS) with 3D environmental perception capabilities has become an urgent need - not only to reduce accidents but also to optimize traffic management and improve urban quality of life.

Traditional object detection methods often rely on a single data source, either a camera or LiDAR. Cameras provide 2D images rich in color and detail but lack accurate depth and size information. Conversely, LiDAR offers precise 3D depth data but is sparse, lacks color details, and is sensitive to adverse weather conditions such as rain or fog. Integrating data from both sources presents significant potential but also poses multiple challenges:

- **Data synchronization:** Cameras and LiDAR operate at different frequencies and formats, requiring complex algorithms for temporal and spatial alignment.
- **Big data processing:** The fusion of these modalities generates a massive amount of data, demanding high computational capacity.
- **Model optimization:** Deep learning models must be designed to effectively utilize both modalities while maintaining fast processing speed especially critical in real-time applications like autonomous vehicles.

The nuScenes dataset plays a key role in this research, providing over 1,000 real-world traffic scenes collected in Boston and Singapore. It includes data from 6 cameras, 1 LiDAR, and 5 radars, with detailed annotations for over 1.4 million 3D objects. This enables the

simulation of complex and diverse traffic scenarios, allowing for real-world model evaluation and improved generalization.

The MeFormer model, built on the MMDetection3D framework, leverages transformer architecture to effectively fuse LiDAR point cloud data and camera images. This improves the accuracy of object detection and recognition, paving the way for optimizing future ITS and autonomous driving technologies.

This study contributes not only to the development of advanced deep learning algorithms but also provides practical benefits in fields such as:

- **Advanced Driver Assistance Systems (ADAS):** Detecting hazardous objects and issuing timely alerts to drivers.
- **Urban traffic management:** Monitoring and optimizing traffic flow based on 3D data.
- **Autonomous vehicles:** Providing accurate environmental perception as a foundation for further research in transportation automation.

1.2. Research Objectives

This research aims to develop a 3D object detection system that integrates camera and LiDAR data using the MeFormer model on the nuScenes dataset, aiming to enhance accuracy and efficiency in traffic environment perception. To achieve this goal, the research will focus on several specific objectives, including:

- **Apply a pre-trained MeFormer model:** Utilize the MeFormer architecture to perform 3D object detection using multimodal data (camera + LiDAR).
- **Evaluate model performance:** Assess the model's detection capability using standard metrics such as mAP, mAOE, and mATE.
- **Visualize detection results:** Display 3D bounding boxes and detected objects on camera images and LiDAR point clouds with prediction results for intuitive evaluation.
- **Explore real-world applicability:** Analyze the potential of using MeFormer in real-world scenarios such as ADAS, ITS, and autonomous driving.

1.3. Scope and Limitations

Data Limitations: The study uses the nuScenes mini dataset (10 traffic scenes) instead of collecting real-world data, to save time and cost. While this allows for algorithm development, it limits the model's evaluation under specific traffic conditions in Vietnam, such as mixed traffic flow or tropical weather.

Hardware Limitations: Due to the high cost of LiDAR 3D and the need for powerful computing resources (e.g., NVIDIA Tesla T4 GPU), the study does not implement physical hardware. Instead, the model is trained and tested on a Virtual Private Server (VPS) using cloud resources, limiting its real-time testing capability.

Scope Limitations: The system focuses on 3D object detection and visualization on static (offline) data and has not yet been integrated into real-time applications such as autonomous driving or live traffic monitoring. This aligns with the primary goal of evaluating model performance but does not meet deployment requirements.

Time Constraints: Due to limited research duration, the study concentrates on developing and evaluating the model using available data, without expanding to optimize processing speed or integrate with specific hardware.

Development Potential: Despite its limitations, this research lays the groundwork for future advancements, including:

- Collecting real-world data in Vietnam for localized evaluation.
- Deploying the system on embedded hardware for autonomous vehicles.
- Expanding the model to handle complex scenarios such as adverse weather or dense traffic.

1.4. Methodology and System Pipeline

This research project will be carried out through a series of key tasks to ensure comprehensive implementation and evaluation of the MeFormer model:

- Study the structure, data modalities, and annotation format of the nuScenes dataset, focusing on camera and LiDAR data relevant to 3D object detection.

- Set up the MMDetection3D framework and integrate the pre-trained MeFormer model for inference on nuScenes data.
- Preprocess nuScenes data (e.g., calibration, formatting, normalization) to ensure compatibility with the model's input requirements.
- Apply the MeFormer model on the nuScenes mini dataset to perform 3D object detection without full-scale training.
- Evaluate the model's inference results using standard metrics such as mean Average Precision (mAP), mean Average Orientation Error (mAOE), and mean Average Translation Error (mATE).
- Develop a visualization module to overlay 3D bounding boxes on camera images and LiDAR point clouds for intuitive analysis.
- Analyze the results, summarize key findings, and compile a comprehensive report along with a research presentation.

1.5. Research Timeline

- **January:** Literature review, explore nuScenes dataset, and set up MMDetection3D environment.
- **February:** Preprocess data and integrate pre-trained MeFormer model; run inference on VPS.
- **March:** Evaluate results using mAP, mAOE, mATE.
- **April:** Develop visualization module for camera and LiDAR outputs.
- **May:** Finalize report and prepare research presentation.

1.6. Organization of the thesis

Our thesis layout included 5 chapters:

Chapter 1: Introduction

Introduces the research problem, objectives, and scope. Defines 3D object detection using Camera and LiDAR in traffic environments and highlights its role in smart transportation and autonomous vehicles.

Chapter 2: Literature Review

Covers the fundamentals of multimodal 3D object detection. Briefly compares Camera and LiDAR sensors, outlines fusion strategies, and introduces key tools like nuScenes, MMDetection3D, and MEFormer.

Chapter 3: System Design

Presents system requirements and architecture. Summarizes the data processing pipeline, model inference, and core interface components.

Chapter 4: Results And Evaluation

Outlines the experimental setup and shows MEFormer's performance on the nuScenes mini dataset by class, range, and confidence, with baseline comparisons.

Chapter 5: Conclusion And Development

Summarizes MEFormer's effectiveness in short-range detection of common objects. Notes limitations and suggest future work on rare classes and threshold optimization.

CHAPTER 2: LITERATURE REVIEW

This chapter provides a comprehensive theoretical background on multimodal 3D object detection using LiDAR-camera fusion, emphasizing its significance in autonomous vehicle perception. It explores the fundamentals of LiDAR and camera sensors, fusion strategies (early, late, intermediate), and key challenges like calibration and synchronization. The nuScenes dataset is examined for its structure and evaluation metrics, while the MMDetection3D framework is analyzed in terms of architecture and supported models. Finally, the MEFormer model is reviewed with a discussion on its limitations and future research directions for enhancing fusion-based detection performance.

2.1. Fundamentals of 3D Perception in Autonomous Driving Systems

2.1.1. The Importance of 3D Object Detection in Autonomous Systems

3D object detection is one of the fundamental pillars in the development of autonomous systems, particularly self-driving vehicles. Accurately identifying the position, size, and category of objects in three-dimensional space is a prerequisite for safe and effective operation in real-world environments. It serves as a critical component in any advanced perception system, providing essential input for high-level tasks such as object tracking, collision avoidance, and motion planning.

The ability to perceive the 3D environment with high precision plays a pivotal role in ensuring reliable navigation, especially under complex traffic scenarios. The transition from 2D to 3D perception systems has marked a significant leap forward, enabling autonomous vehicles to make more sophisticated and context-aware decisions.

The six levels of driving automation, as defined by SAE J3016, are categorized based on the degree of control between the human driver and the vehicle system, as illustrated in [2, Fig. 2.1]. Levels 0 to 2 require the human driver to remain primarily responsible for driving tasks. From Level 3 onward, the vehicle begins to take over control functions. Level 5 achieves full automation, operating without any human intervention under all conditions.

The need for highly accurate and reliable 3D object detection is becoming increasingly critical, especially as vehicles progress through higher levels of automation - specifically

from Level 3 to Level 5 according to the SAE classification. As human involvement decreases, the perception system must ensure absolute accuracy to maintain safety standards.

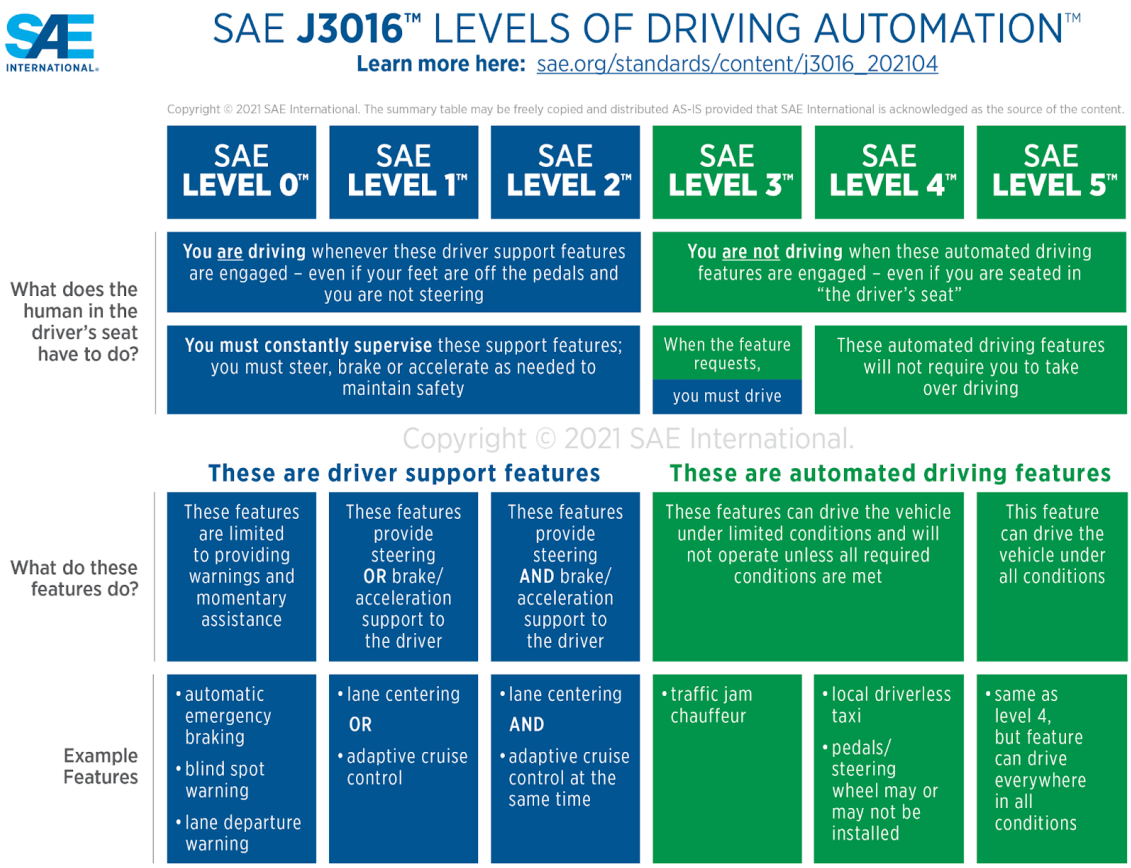


Figure 2.1 SAE Levels of Driving Automation™ Refined for Clarity and International Audience

For this reason, 3D object detection has evolved into a safety-critical component of autonomous driving systems, significantly driving research and development in the field. Any failure in the detection pipeline can result in severe consequences, including threats to human life and a loss of public trust in autonomous vehicle technology.

2.1.2. Complementary Properties of LiDAR and Camera Sensors

Among the various sensors equipped on autonomous vehicles, LiDAR (Light Detection and Ranging) and cameras are particularly important and are often used in tandem due to their complementary characteristics.

LiDAR operates by measuring distances using laser pulses, generating precise spatial and depth information in the form of a 3D point cloud. In contrast, cameras capture high-resolution 2D images rich in semantic content, including color, surface texture, and fine visual details, elements that LiDAR cannot provide.

The complementarity between these sensors goes beyond simple data aggregation; it lies in their ability to mitigate each other's inherent weaknesses. For instance, LiDAR's inability to capture visual semantics is compensated by the camera, while the camera's unreliable depth estimation, especially under poor lighting or adverse weather conditions, is reinforced by the accurate depth data from LiDAR.

This fusion enables the construction of a more robust, reliable, and comprehensive perception system capable of operating effectively under diverse and dynamic real-world driving conditions.

2.1.3. Theoretical Basis for LiDAR-Camera Fusion

The primary objective of fusing data from LiDAR and camera sensors is to enhance the accuracy and reliability of the perception system in autonomous vehicles. By leveraging the distinct advantages of each sensing modality, sensor fusion techniques aim to improve the overall robustness of the system while offering a more comprehensive understanding of the surrounding environment.

In recent years, multimodal fusion has emerged as a key trend in autonomous vehicle perception. The motivation behind sensor fusion extends beyond academic interest; it addresses practical demands for effectively managing complex and uncertain traffic scenarios in real-world conditions.

By combining spatial precision from LiDAR with rich semantic cues from camera images, fusion-based perception systems can achieve higher levels of situational awareness. This, in turn, enables safer and more intelligent decision-making processes

essential components for advancing vehicle autonomy from experimental prototypes to large-scale deployment in public transportation systems.

Figure 2.2 Jaguar I-PACE equipped with Waymo autonomous driving technology

The Waymo Robotaxi, utilizing the Jaguar I-PACE platform, which utilizes a fusion of LiDAR, radar, cameras, and deep learning-based AI to perform real-time localization, object detection, and decision-making, is shown in [3, Fig. 2.2]. The system is classified as SAE Level 4 [3], meaning it is capable of fully autonomous driving within designated geo-fenced areas without requiring human intervention.

Figure 2.3 Mercedes-Benz EQS equipped with Drive Pilot

According to Mercedes-Benz, the Drive Pilot system as illustrated in [4, Fig. 2.3] enables “conditionally automated driving” allowing the driver to delegate control to the vehicle under specific circumstances. As described by the company, such conditions include being stuck in “dense or congested traffic on suitable highway sections in Germany at speeds of up to 60 km/h” [4].

Systems that rely on a single type of sensor often exhibit limitations, especially when exposed to adverse environmental conditions such as poor weather (e.g., rain, fog) or occlusion. For instance, cameras may perform poorly under low-light or harsh weather conditions, while LiDAR can struggle with low-reflectivity or transparent objects.

Each sensor on an autonomous vehicle such as a LiDAR or a camera has its own local coordinate system, defined by its physical mounting position and orientation on the vehicle. To combine their data meaningfully, it is necessary to align or transform the data from one sensor’s coordinate frame to another. This process, known as coordinate transformation, requires precise knowledge of the relative positions and orientations of the sensors, which is typically obtained through a calibration process. Without accurate alignment, any attempt to fuse the data would result in misinterpretations of object locations and shapes.

Coordinate Transformation:

The transformation from the LiDAR coordinate frame to the camera coordinate frame is mathematically described by a rotation and a translation. If a point in the LiDAR frame is denoted as P_{LiDAR}, its coordinates in the camera frame P_{cam} can be defined as Formula (2.1).

$$P_{cam} = R_{calib} \cdot P_{LiDAR} + t_{calib}, \tag{2.1}$$

where R_{calib} is the rotation matrix and t_{calib} is the translation vector, both derived from sensor calibration. This transformation ensures that the 3D points from the LiDAR are correctly positioned relative to the camera.

Projection to Image Plane:

The intrinsic matrix (often denoted as K) describes the internal parameters of a camera that affect how 3D points in the camera coordinate system are projected onto the 2D image

plane. We can use Formula (2.2) to define K. It encodes properties such as the focal length, the optical center (principal point), and sometimes skew and pixel scaling factors.

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where f_x, f_y are focal lengths in pixels along the x and y axes, s is skew coefficient and c_x, c_y are coordinates of the principal point in the image.

Once the LiDAR points are expressed in the camera's coordinate frame, they can be projected onto the 2D image plane of the camera. The projection is calculated using Formula (2.3).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \tag{2.3}$$

where (X_c, Y_c, Z_c) are the 3D coordinates in the camera frame, and (u, v) are the corresponding pixel coordinates in the image. This projection allows the system to overlay LiDAR data onto camera images, facilitating sensor fusion and visualization (as illustrated in [5, Fig. 2.4]).

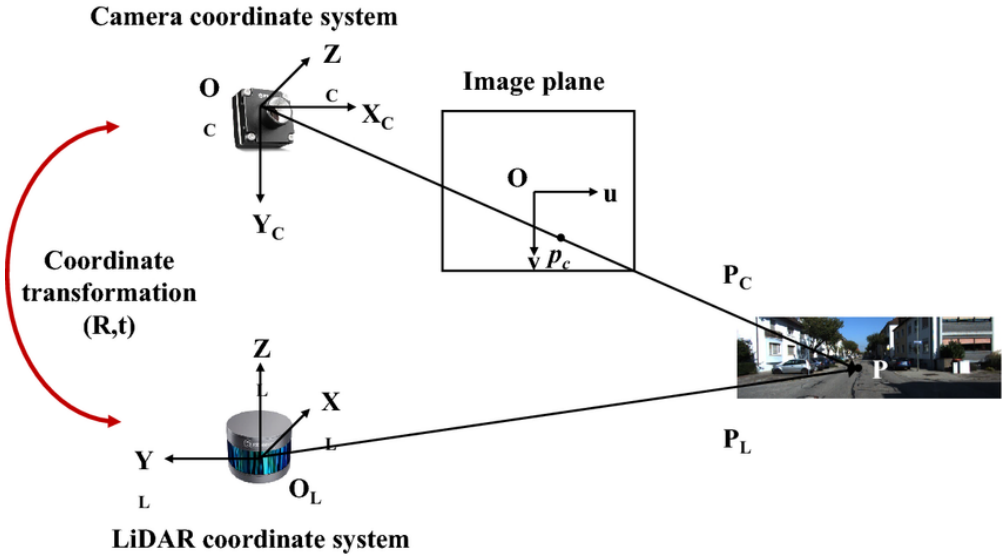


Figure 2.4 Diagram showing the transformation between the camera and the LiDAR using extrinsic parameters.

2.1.4. Overview of MMDetection3D and the NuScenes Dataset

Research on sensor fusion for 3D object detection has made significant progress in recent years, driven by the development of open-source frameworks and large-scale datasets. MMDetection3D is an open-source toolbox built on the PyTorch framework, specifically designed for 3D object detection tasks. It supports both unimodal and multimodal models and includes implementations compatible with several popular datasets, including NuScenes.

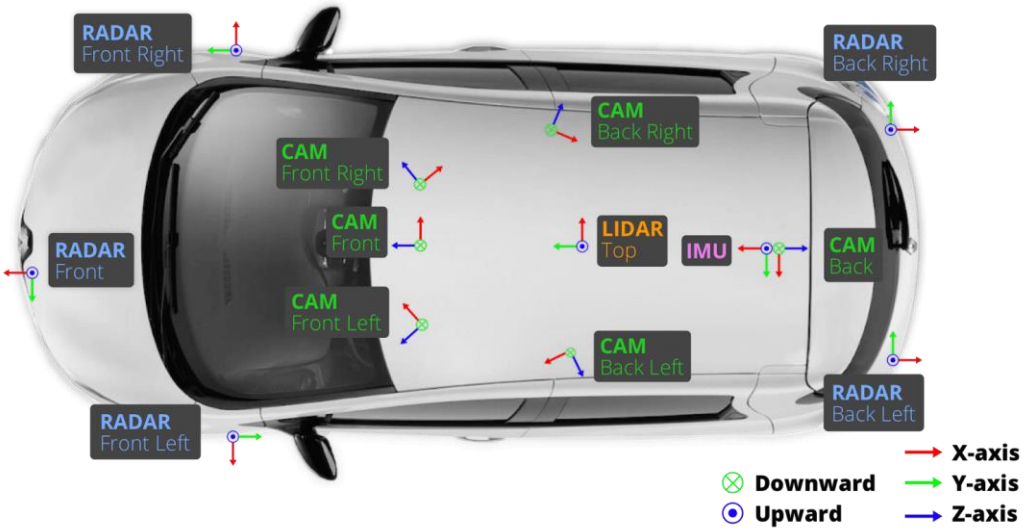


Figure 2.5 Sensor Layout Diagram of the Autonomous Vehicle in the nuScenes Dataset

NuScenes is a large-scale dataset created for autonomous driving applications. It provides data collected from a full sensor suite comprising one 32-beam LiDAR, six cameras, five radars, as well as GPS and IMU, enabling full 360-degree environmental coverage. The dataset contains 1,000 scenes, each lasting 20 seconds, and includes 3D bounding box annotations for ten object classes at a sampling rate of 2 Hz.

As illustrated in [6, Fig. 2.5], the sensor layout on the autonomous vehicle used in the nuScenes dataset is depicted. The vehicle is equipped with a strategically arranged suite of sensors, including RADAR, CAM (cameras), LiDAR, and IMU. The LiDAR unit is mounted on the rooftop to provide detailed 3D spatial mapping, while the cameras and radar modules are distributed around the vehicle to enable object recognition and support

autonomous navigation. This configuration enables full 360-degree perception of the environment, enhancing situational awareness and operational accuracy [6].

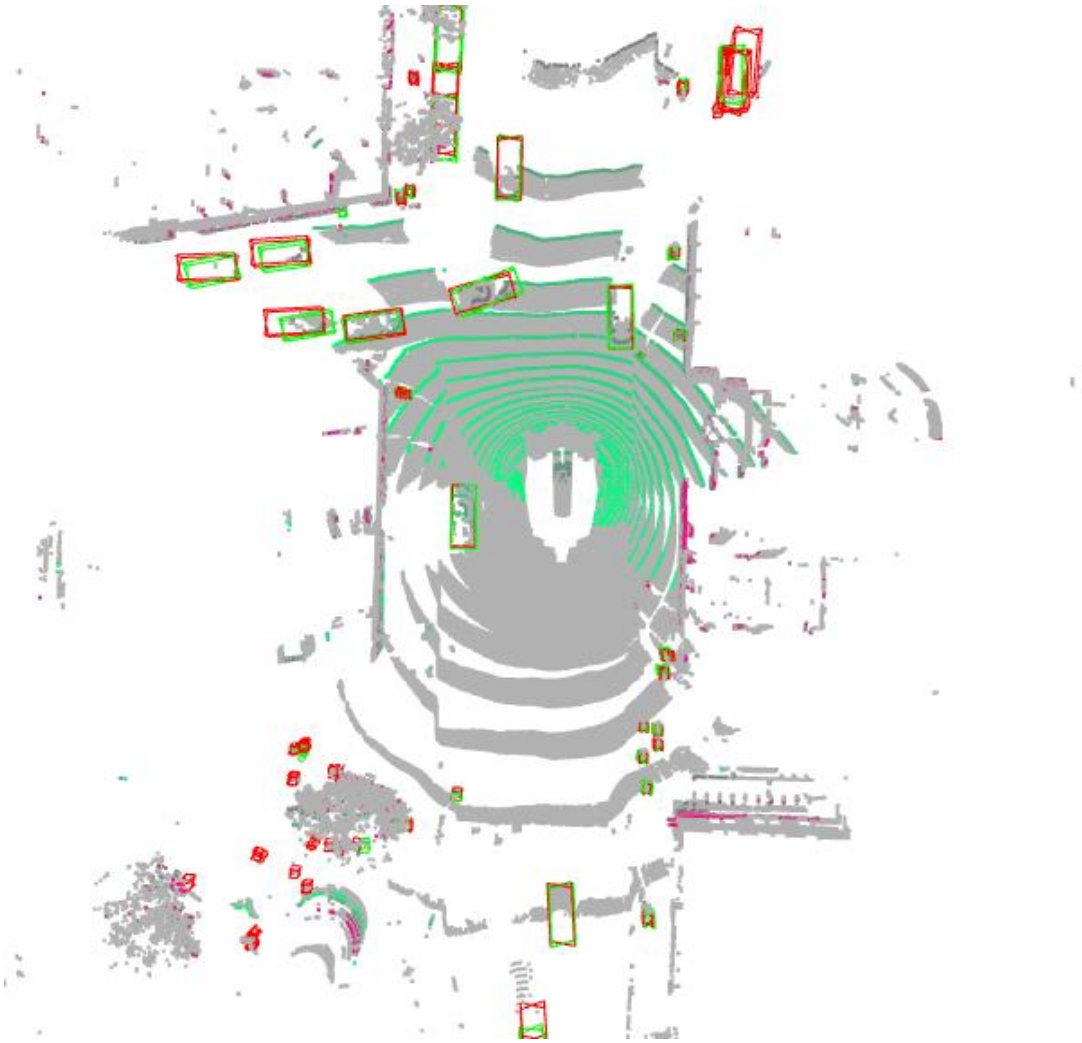


Figure 2.6 3D object detection results visualized with MMDetection3D using the project's custom tool.

The Figure 2.6 illustrates the results of 3D object detection using MMDetection3D, an advanced framework for processing LiDAR sensor data. In this sample display generated by the project's custom visualization tool, objects in the scene are detected, classified, and visualized as 3D bounding boxes. This visualization enables autonomous vehicles to accurately perceive distances, shapes, and spatial relationships of surrounding entities. Such capability is a critical component of navigation and collision avoidance systems in autonomous driving.

The availability of comprehensive frameworks like MMDetection3D, along with rich multimodal datasets such as nuScenes, has been instrumental in accelerating research progress. These resources offer a standardized platform that streamlines model development and evaluation workflows, lowering the entry barrier for new researchers and allowing them to focus on algorithmic innovation rather than building systems from scratch.

2.2. Sensor Modalities for 3D Perception

2.2.1. Working Principle, Advantages, and Limitations of LiDAR

Working Principle: LiDAR is a distance measurement technology that operates by emitting laser pulses and calculating the time it takes for each pulse to return after reflecting off objects in the environment. By repeating this process at high frequencies and from multiple angles, the sensor constructs a dense set of three-dimensional data points commonly referred to as a point cloud, which accurately captures the geometric structure of the surrounding environment.

Advantages of LiDAR:

- **High accuracy in depth measurement:** LiDAR provides direct 3D depth information with significantly greater accuracy than depth estimation from monocular or stereo image data.
- **Robust performance under varying lighting conditions:** LiDAR operates independently of ambient light, maintaining effectiveness in bright daylight, darkness, tunnels, shadows, or rapidly changing illumination making it invaluable for consistent spatial awareness in autonomous navigation and mapping.
- **Instantaneous distance and direction estimation:** Laser reflections allow for the direct measurement of object distance and orientation, reducing the computational complexity required for inferring depth from 2D images.

Limitations of LiDAR:

- **Non-uniform point cloud density:** Point clouds become sparse at long range or for small objects, limiting fine detection; reflectivity also affects density.

- **Lack of semantic information:** LiDAR cannot capture color or texture, making it hard to distinguish elements like traffic lights or signs.
- **High cost and computational demands:** LiDAR is expensive and produces large data volumes, requiring significant processing resources.
- **Performance degradation in adverse weather:** Fog, rain, and snow degrade LiDAR performance by scattering or absorbing laser pulses.
- **Susceptibility to mutual interference:** Overlapping signals from multiple LiDARs can reduce signal quality.
- **Aesthetic and integration constraints:** Bulky, exposed LiDAR units complicate seamless vehicle integration and affect aerodynamics.

As illustrated in [7, Fig. 2.7], LiDAR sensor manufacturers utilize the NVIDIA DRIVE platform to enhance autonomous vehicle perception. LiDAR sensors produce 3D point clouds, enabling precise object detection and safe navigation. This technology significantly improves the accuracy and autonomy of modern self-driving systems.

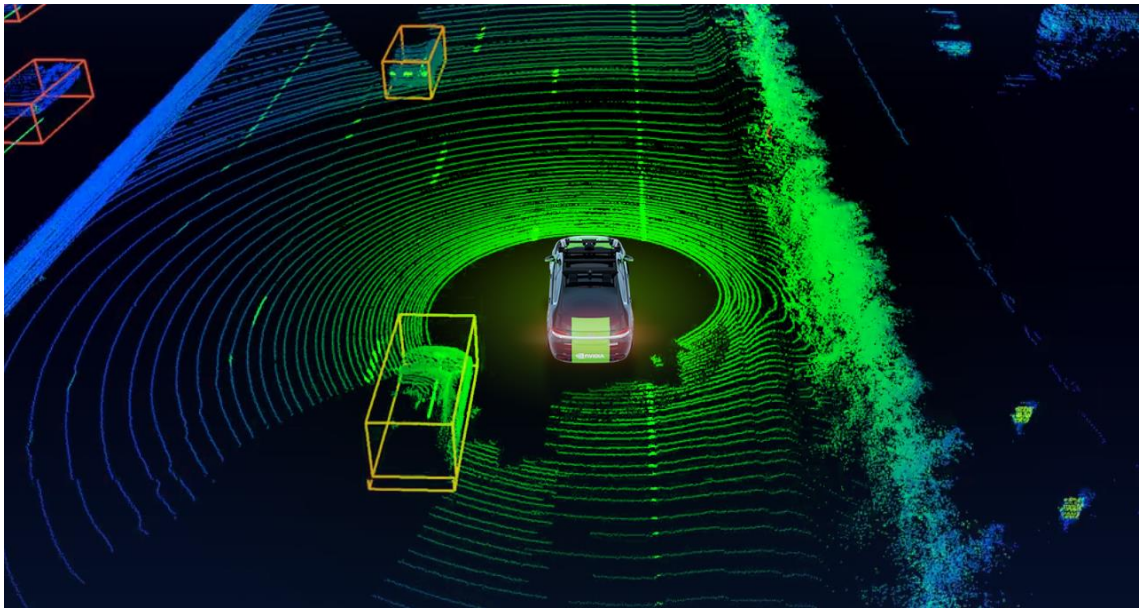


Figure 2.7 A Sense of Responsibility Lidar Sensor Makers Build on NVIDIA DRIVE

2.2.2. Working Principle, Advantages, and Limitations of Cameras

Working Principle: Cameras capture light reflected from the environment using lenses and image sensors (such as CMOS or CCD) to generate 2D images. In autonomous driving systems, multiple cameras are often employed to expand the field of view or to create stereo configurations, which are useful for estimating depth.

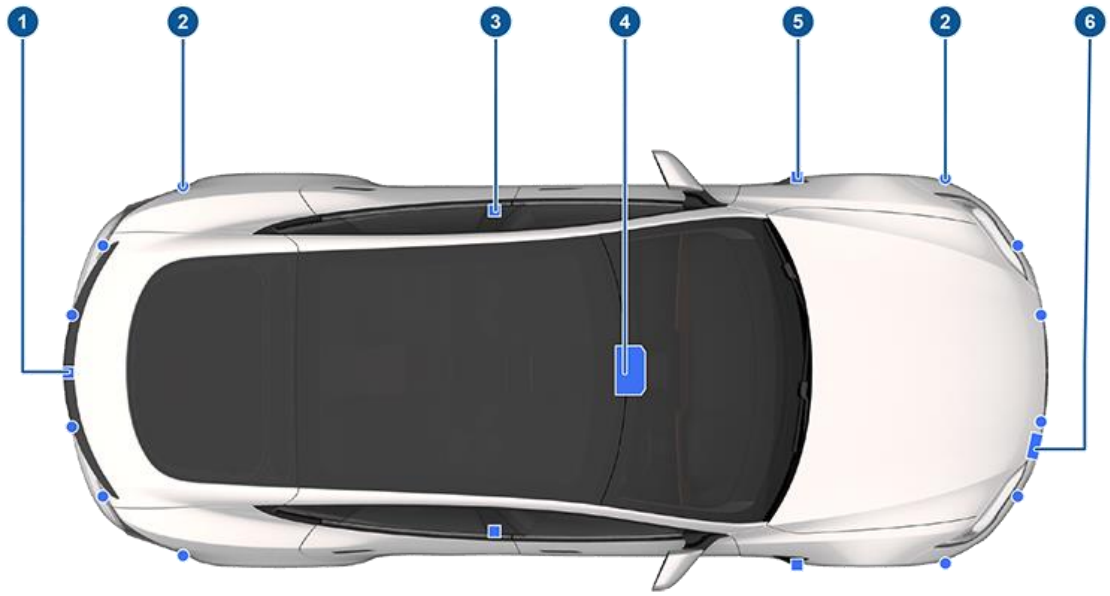


Figure 2.8 Tesla Vision system equipped on the Tesla Model S 2025

As illustrated in [7, Fig. 2.8], the autonomous driving system integrates multiple sensors and cameras for enhanced vehicle perception:

1. A rear camera mounted above the license plate.
2. Ultrasonic sensors (if equipped) located in the front and rear bumpers.
3. Cameras installed in each door pillar.
4. Two cameras mounted to the windshield above the rearview mirror.
5. Cameras placed on each front fender.
6. A radar unit (if equipped) behind the front bumper.

This Model is also equipped with High-precision electronically assisted braking and steering systems.

Tesla Vision is a camera-based perception system that replaces LiDAR and radar in Tesla's newer models. Using eight strategically placed cameras and deep learning, it

enables real-time lane monitoring, object detection, and traffic sign recognition for Autopilot and Full Self-Driving (FSD). While this approach reduces hardware costs and simplifies processing, it raises concerns about performance in low-light or adverse weather conditions..

Advantages:

- **Rich semantic visual information:** Cameras capture high-resolution images with meaningful semantic details such as color, texture, shape, and contextual cues. This allows for effective differentiation between objects with similar geometry but distinct functions, for example, distinguishing pedestrians from lamp posts.
- **Superior object recognition and classification:** With semantically rich image data, cameras excel in tasks like object classification, traffic sign reading, signal recognition, and lane marking detection.
- **Low cost and easy integration:** Compared to LiDAR, cameras are significantly more cost-effective and easier to integrate into the vehicle's exterior design, meeting both technical and aesthetic requirements.

Limitations:

- **Limited depth estimation:** Inferring depth from 2D images especially monocular is an inherently ill-posed and unstable problem. Even with stereo camera systems, accuracy depends on camera baseline, environmental conditions, and object distance. While advanced techniques such as Categorical Depth Distribution Network (CaDDN) have improved reliability, they still fall short of the precision offered by LiDAR-generated depth data.
- **Sensitivity to lighting and weather conditions:** Camera performance can degrade significantly under adverse conditions such as low light (nighttime), glare, fog, heavy rain, or snow, where image quality is severely compromised.
- **Scale ambiguity:** Cameras inherently lack the ability to determine an object's true size or distance from a 2D image without additional depth data or complex processing algorithms, leading to uncertainty in 3D spatial reconstruction.

2.2.3. Comparative Analysis of LiDAR and Camera in 3D Object Detection

In the domain of 3D object detection, both LiDAR and camera technologies play crucial yet distinct roles. LiDAR stands out for its superior spatial localization capabilities through direct distance and depth measurements, while cameras provide exceptional object classification performance thanks to their semantically rich visual data, including color and texture.

Designing a perception system for autonomous vehicles involves a trade-off between performance, reliability, and practical feasibility. The key characteristics of LiDAR and camera technologies in the context of 3D object detection are summarized in Table 2.1.

Table 2.1 LiDAR and Camera: 3D Detection Comparison

Characteristic	**LiDAR**	**Camera**
Depth Accuracy	High (direct measurement)	Low/Medium (inferred from 2D images)
Semantic Detail	Low	High
Color/Texture Information	Not available	Available
Low-Light Performance	Good	Poor
Adverse Weather Performance	Moderate/Poor (e.g., rain, fog)	Poor
Typical Operating Range	~70-100 m	Variable (depends on lens, may exceed LiDAR with telephoto lens)
Field of View (FOV)	Typically 360° horizontal	Typically narrower (multiple cameras needed for full coverage)
Cost	High	Low
Computational Load (Raw Data)	High (large point cloud processing)	Medium/High (image processing)
Classification Accuracy	Medium/Low (geometry-based)	High (semantic and texture-based)

The choice of sensor configuration must align with the specific objectives, operating environment, and cost constraints of the application. For instance, a high-density sensor system like Waymo's featuring multiple LiDARs and cameras offers comprehensive spatial coverage and rich multimodal data but increases integration complexity and deployment cost. Conversely, simpler systems may reduce costs and computational demands but face performance limitations in complex or adverse environments.

2.3. LiDAR-Camera Fusion Strategies

The integration of data from LiDAR and camera sensors can be implemented at different stages of the processing pipeline, leading to three main fusion strategies: **early fusion**, **late fusion**, and **deep/intermediate fusion**. Each strategy is characterized by its operational principle, architectural design, and specific advantages and limitations.

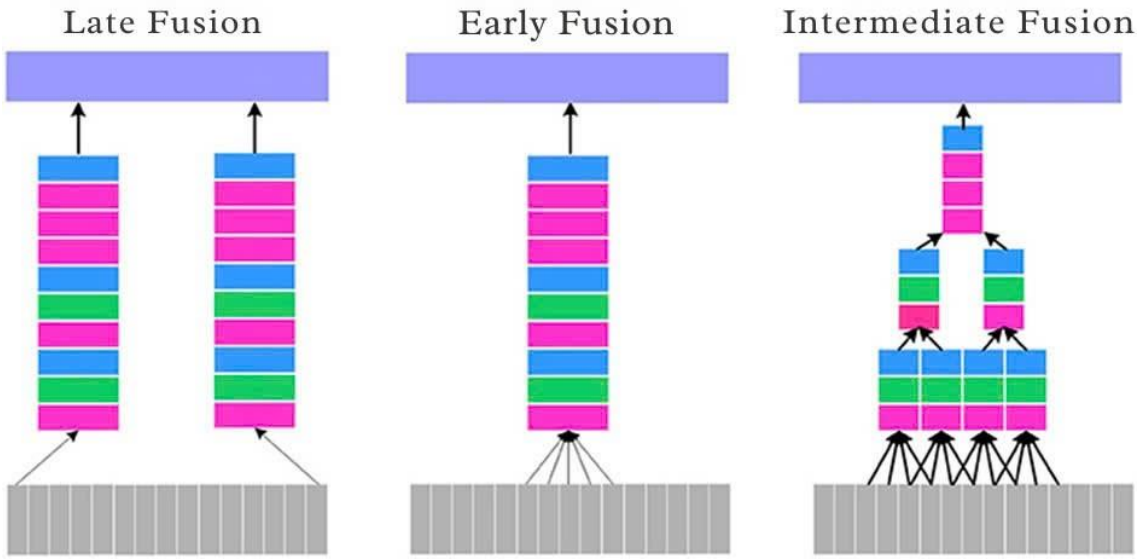


Figure 2.9 Pipeline overview with three main fusion strategies

Three neural network fusion strategies - early fusion, intermediate fusion, and late fusion are illustrated in [8, Fig. 2.9]. Red, green, blue, and purple boxes represent convolutional, pooling, normalization, and fully connected layers, respectively. Early fusion merges features at the input level, intermediate fusion integrates them at multiple intermediate stages, and late fusion combines features at the final stage.

2.3.1. Early Fusion (Data-Level Fusion)

Early fusion integrates raw data from different sensors at the input level, prior to any deep feature extraction. Typical approaches include projecting LiDAR points onto the image plane to enrich each pixel with depth information, or converting both modalities into a unified representation (e.g., voxel grid or bird's-eye view) before concatenation.

Advantages: This strategy enables the network to learn low-level correlations between modalities, maximizing complementary information. The architecture supports end-to-end training, which can improve performance in complex scenarios where early cross-modal interactions are beneficial. Early fusion is especially useful for exploiting fine-grained spatial relationships between sensor.

Limitations: It requires high preprocessing cost and precise sensor calibration and synchronization. The differences between dense image data and sparse LiDAR point clouds make direct fusion challenging. Projecting 3D data onto 2D can also cause geometric distortions and reduce accuracy if calibration is imperfect.

Common Architectures: Input concatenation; Point painting or decoration (coloring LiDAR points with image information).

2.3.2. Late Fusion (Decision-Level or Object-Level Fusion)

In this strategy, data from each sensor is processed independently through unimodal detection networks. The outputs (e.g., bounding boxes, classification scores) are then fused at the decision level to produce final predictions.

Advantages: High flexibility and computational efficiency. Easy to deploy in practical systems due to the ability to leverage pre-trained unimodal models. The modular design allows for independent maintenance and upgrades of each component.

Limitations: Limited or no cross-modal interaction reduces the ability to exploit complementary information. Fine-grained features from deeper layers are discarded. Errors from unimodal detectors may propagate and degrade the final result. Overall performance heavily depends on the reliability of each sensor; for example, poor camera performance in low-light conditions can negatively affect the fusion unless a confidence-aware correction mechanism is used.

Common Architectures: Bounding box merging and matching; Score-weighted averaging; Kalman filters or object tracking methods

2.3.3. Deep/Intermediate Fusion (Feature-Level Fusion)

This strategy extracts intermediate or high-level features from each modality using separate backbones, transforms them into a shared representation space (such as Bird's-Eye View), and then fuses them via dedicated integration modules.

Advantages: It offers a balance between the interaction of early fusion and the flexibility of late fusion, enabling learning of high-level semantic features with less reliance on precise calibration. This approach also leverages modern deep learning architectures for both feature extraction and fusion.

Limitations: Increased model complexity and design challenges related to selecting an appropriate intermediate representation and efficient feature fusion mechanisms.

Common Architectures: Feature fusion in BEV space (e.g., BEVFusion); Cross-attention mechanisms; Transformer-based fusion.

The use of BEV as a shared representation has become a popular trend due to its ability to bridge the *distributional modality gap* between perspective-view camera images and sparse 3D point clouds from LiDAR. However, mapping features into BEV requires careful handling to avoid information loss or distortion. Attention-based models and Transformers are increasingly favored for their capacity to model long-range dependencies and selectively fuse relevant cross-modal features.

2.4. The nuScenes Dataset: A Benchmark for Multimodal 3D Object Detection

2.4.1. Overview: Scale, Collection Process, and Context

The nuScenes dataset is a large-scale, publicly available benchmark developed by Motional (formerly nuTonomy) to support research in autonomous driving. It comprises 1,000 scenes, each approximately 20 seconds long, recorded in two urban environments with complex traffic conditions: Boston (USA) and Singapore. The scenes were manually selected to ensure a wide variety of driving scenarios, traffic conditions, times of day, and weather encompassing both routine situations and challenging or unexpected behaviors.

The nuScenes dataset stands out for its comprehensive sensor suite, designed to emulate the perception systems commonly deployed on modern autonomous vehicles. Data is collected using multiple synchronized sensors, including LiDAR, cameras, radar, GPS, and IMU, enabling full 360-degree environmental coverage around the ego vehicle.

Figure 2.10 Map of the Boston Seaport District as a Contextual Background for the nuScenes Dataset

A detailed map of the Boston Seaport District is presented in [9, Fig 2.10], representing one of the primary locations where the nuScenes dataset was collected. Key roadways, intersections, port facilities, and transportation hubs are highlighted, with roads color-coded to enhance visual differentiation of infrastructure elements. Serving as a contextual reference, the map offers geographic and traffic-related information critical for the development and evaluation of autonomous driving technologies.

A map of the Queenstown area in Singapore is shown in [10, Fig. 2.11], representing one of the key urban locations featured in the nuScenes dataset for autonomous vehicle research. The map includes detailed information on the local transportation network, highlighting major roads such as Commonwealth Avenue and Tanglin Road, along with notable landmarks. This geographic data supports the simulation of complex urban environments, contributing to the improvement of perception and navigation capabilities in autonomous driving systems.

Figure 2.11 Map of Queenstown, Singapore as a Contextual Background for the nuScenes Dataset

LiDAR (Velodyne HDL-32E):

NuScenes employs a single rotating LiDAR sensor Velodyne HDL-32E to capture three-dimensional spatial data.

Specifications: Scanning frequency: 20 Hz

- **Number of channels**: 32
- **Field of view (FOV**): 360° horizontal; +10° to -30° vertical
- **Operating range:** 80-100 m; effective range approximately 70 m
- **Accuracy**: ±2 cm
- **Point generation rate: ~1.39 million points per second**
- **Data format:** Point cloud data is stored in binary .bin files. Each point includes five attributes: 3D coordinates (x, y, z), intensity, and ring index, all represented as 32-bit floating-point numbers (float32).

Figure 2.12 LiDAR Velodyne HDL-32E

The Velodyne HDL-32E LiDAR sensor is depicted in [11, Fig. 2.12], representing an advanced laser-scanning system commonly used in autonomous vehicles. Capable of generating high-resolution 3D point clouds, the sensor enables accurate object recognition and detailed environmental mapping. This spatial data helps autonomous vehicles better interpret their surroundings. The technology is essential for improving the precision and safety of autonomous navigation systems.

In frameworks such as MMDetection3D, the input point cloud data typically includes only four components by default: x, y, z, and intensity; or x, y, z, and timestamp difference when using the LoadPointsFromMultiSweeps process. These features serve as the foundation for further processing in 3D object detection pipelines.

The use of a 32-beam LiDAR sensor results in relatively sparse point clouds in the nuScenes dataset compared to more modern LiDAR systems (e.g., 64 or 128 beams). This sparsity makes it harder to detect small or distant objects, especially in complex traffic scenes. The reduced vertical resolution limits the amount of structural detail captured from

the environment. This can lead to incomplete object representations and decreased detection accuracy. As a result, integrating camera data is crucial for enhancing perception capabilities.

The Basler ace Classic acA1600-60gc

The camera is shown in [12. Fig 2.13], representing an industrial-grade imaging device capable of capturing up to 60 frames per second. Widely utilized in computer vision and data acquisition systems, particularly in automation and autonomous vehicle applications, the camera offers stable image quality and high performance, supporting accurate object recognition in complex environments.

Figure 2.13 Basler ace Classic acA1600-60gc

The camera is equipped with an Evetar Lens N118B05518W (F1.8, f=5.5mm, 1/1.8") and utilizes a 1/1.8" CMOS sensor with a native resolution of 1600×1200 pixels. Images are encoded in Bayer8 format, using 1 byte per pixel, which provides a compact yet information-rich representation. To reduce processing and transmission bandwidth, a 1600×900 region of interest (ROI) is cropped from the original resolution, focusing on the most relevant part of the scene. The camera supports auto exposure, with the exposure time limited to a maximum of 20 ms to avoid motion blur under dynamic conditions. Captured

images are unpacked to BGR format and compressed to JPEG for storage and transmission, striking a balance between image quality and data size. These specifications ensure that the camera delivers reliable visual data in real-time applications.

A key consideration is the difference in data acquisition frequencies between the camera (12 Hz) and the LiDAR sensor (20 Hz), which necessitates precise temporal alignment, especially in studies requiring high-resolution synchronization across modalities. Misalignment can lead to inaccurate fusion results, particularly when tracking fast-moving objects. Therefore, careful calibration and timestamp correction are essential to maintain temporal consistency between sensor streams. For further details on camera orientation and field-of-view overlap, see the figure below.

Radar and Other Sensors

In addition to LiDAR and camera sensors, the nuScenes dataset includes data from five Continental ARS 408-21 long-range radar sensors (operating at 13 Hz, 77 GHz frequency band, with a maximum range of 250 meters), as well as a GPS/IMU system (Advanced Navigation Spatial) for localization and inertial measurement.

Figure 2.14 Long-Range RADAR Sensor (Continental ARS 408-21)

Although this report primarily focuses on LiDAR-camera data fusion, the inclusion of these additional sensors highlights the richness and multimodality of the dataset, enabling future research into comprehensive sensor fusion approaches.

The Continental ARS 408-21 long-range RADAR sensor, operating at a frequency of 77 GHz, provides high-precision distance and velocity measurements, as illustrated in [13, Fig. 2.14]. Leveraging Frequency Modulated Continuous Wave (FMCW) technology, the sensor is capable of independently calculating motion parameters within a single measurement cycle.

Sensor Calibration

The nuScenes dataset provides full calibration for all sensors, including precise LiDAR extrinsics (via laser scanner) and camera extrinsics (using a calibration cube with fiducial markers, aligned to LiDAR and ego frame). Detailed camera intrinsics are also included, ensuring accurate sensor alignment.

These calibration parameters are essential for performing accurate coordinate transformations and projections, enabling effective multi-sensor data fusion. However, the multi-stage calibration process can introduce cumulative error, potentially affecting the performance of multi-modal deep learning models.

2.4.2. Data Organization and Annotation

The data in the nuScenes dataset is organized following a relational database model, where metadata and annotation information are stored in JSON files. Each entity in the database such as scenes, frames, and annotations is uniquely identified by a token, enabling efficient cross-referencing between tables. The nuScenes devkit provides utility classes and functions to facilitate database access and manipulation.

Database Schema

The nuScenes devkit provides utility classes and functions to facilitate database access and manipulation. Each table contains a set of metadata fields that describe its entities in detail. The key fields for each table are summarized in [14, Tab. 2.2]. The main tables include:

- **log**: Contains information about the data collection session, including log file names, the vehicle used, timestamps, and geographic locations.

- **scene**: Represents a 20-second driving segment, linking to all associated data via tokens.
- **sample**: A keyframe annotated at 2 Hz.
- **sample_data**: Raw sensor data at a specific timestamp, linked to sensor information, ego vehicle pose, and the actual data files (e.g., .bin for LiDAR or .jpg for camera).
- **ego_pose**: The global position and orientation of the vehicle at a given time.
- **sensor**: Describes the type and channel of a sensor (e.g., CAM_FRONT, LIDAR_TOP).
- **calibrated_sensor**: Contains both extrinsic and intrinsic calibration data, including intrinsic matrices for cameras.
- **instance**: Represents a tracked object instance throughout a scene.
- **category**: Classifies objects into categories (e.g., vehicle, pedestrian).
- **attribute**: Describes dynamic properties of an object, such as its motion status.
- **visibility**: Indicates the visibility level of an object in camera images.
- **sample_annotation**: Provides 3D bounding box annotations for each object within a sample, including position, size, rotation, and the number of LiDAR/RADAR points observed.
- **map**: Top-down binary masks representing semantic map information.
- **lidarseg**: Semantic LiDAR segmentation maps associated with each sample_data.

This relational structure supports complex queries such as: “Retrieve all LiDAR points belonging to pedestrian-type objects in a nighttime scene.”

Fields such as num_lidar_pts and num_radar_pts in the sample_annotation table are particularly important for filtering valid ground truth annotations. They ensure that only objects observable by the sensors are used during model training and evaluation, thereby promoting fairness and consistency in benchmarking.

Table 2.2 Overview of metadata in nuScenes

Table Name	Key Fields	Description
sample	token, timestamp, scene_token, next, prev	Annotated keyframe sampled at 2Hz.
sample data	token, sample_token, ego_pose_token, calibrated_sensor_token, filename, is_key_frame, timestamp	Sensor data at a given timestamp; filename points to a .bin or .jpg file.
sample annotation	token, sample_token, instance_token, attribute_tokens, visibility_token, translation, size, rotation, num_lidar_pts	3D bounding box annotation for a single object in a keyframe.
instance	token, category_token, nbr_annotations, first_annotation_token, last_annotation_token	An object instance tracked across a scene.
category	token, name, description	Object classification (e.g., vehicle.car).
attribute	token, name, description	Object attributes (e.g., vehicle.moving).
calibrated sensor	token, sensor_token, translation, rotation, camera_intrinsic	Calibration parameters of the sensor, including camera intrinsics.
ego_pose	token, translation, rotation, timestamp	Ego vehicle pose in the global coordinate frame.
visibility	token, level, description	Visibility level of the object in the camera image.
map	token, log_tokens, category, filename	Semantic map layers (e.g., drivable surface).
sensor	token, channel, modality	Sensor type and configuration

3D Bounding Box Annotation, Attributes, and Object Classification

LiDAR sensor data annotated with color-coded bounding boxes is shown in [6, Fig. 2.15], where concentric circles represent radial distance measurements from the sensor. The yellow, orange, blue, and purple boxes indicate the spatial locations of detected objects in 3D space. A thumbnail image positioned in the top-right corner presents the corresponding camera view, allowing visual comparison between LiDAR data and the real-world scene.

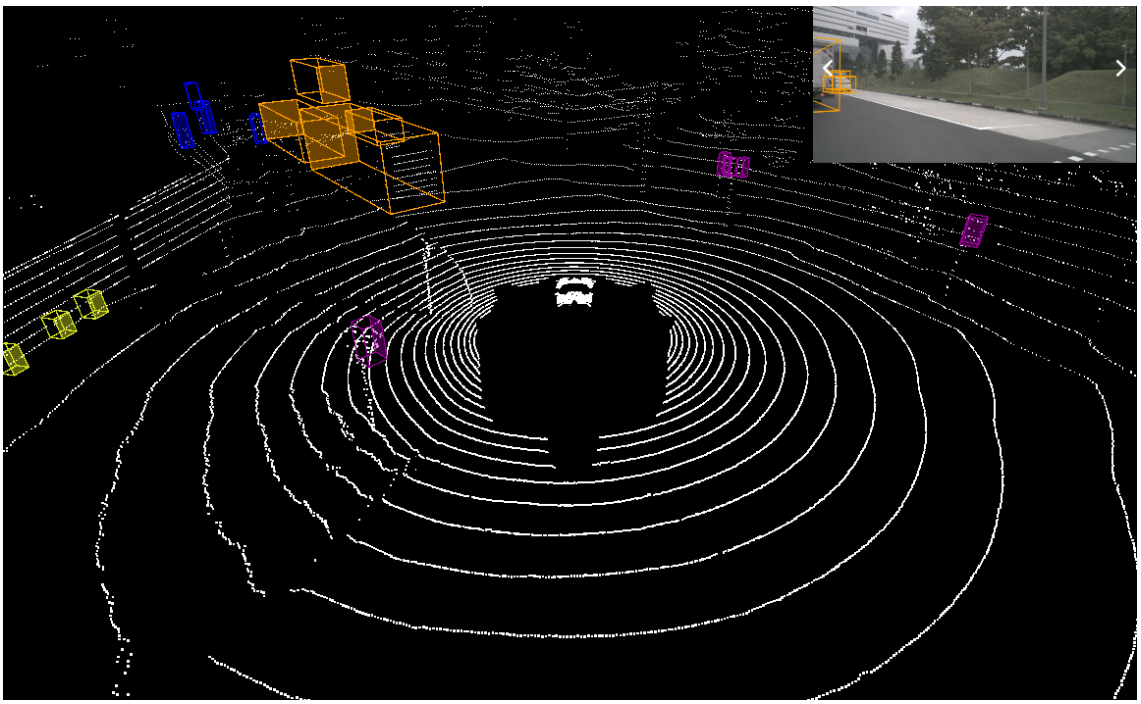


Figure 2.15 Environmental Analysis Using LiDAR Bounding Boxes

Each object in the nuScenes dataset is annotated with a 3D bounding box in the global coordinate system, defined by the following parameters:

- "translation": <float> - The center of the bounding box in meters, specified as [center_x, center_y, center_z].
- "size": <float> - The dimensions of the bounding box in meters, ordered as [width, length, height].
- "rotation": <float> - The orientation of the bounding box represented as a quaternion [w, x, y, z].

The 3D object detection challenge in nuScenes focuses on ten major object classes: **car, truck, bus, trailer, construction_vehicle, pedestrian, motorcycle, bicycle, traffic_cone, and barrier.**

In addition to spatial and categorical data, annotations also include:

- **Attributes**: Indicating object state at the time of observation, such as vehicle.moving, vehicle.parked, pedestrian.standing, or cycle.with_rider.
- **2D Velocity**: The velocity components along the x and y axes (vx, vy) of the object.

These additional fields support advanced perception tasks such as behavior prediction and scene understanding.

Class imbalance is a notable issue in nuScenes. For example, the car class constitutes 43.7% of all annotations approximately 40 times more frequent than the bicycle class. This severe imbalance necessitates the application of specialized training strategies, such as:

- Re-sampling to equalize class distributions
- Data augmentation techniques

These methods help mitigate model bias toward frequent classes and enhance detection performance on rare but safety-critical object types.

2.4.3. Evaluation Protocol and Metrics

The nuScenes dataset provides an official evaluation protocol along with a comprehensive set of metrics to assess performance in the 3D object detection task. Evaluation is conducted on keyframes (samples) at a frequency of 2 Hz [6].

Mean Average Precision (mAP) Using Center Distance

The primary metric for evaluating object detection and classification performance is mAP. Unlike many other datasets that employ the 3D Intersection over Union (IoU) criterion, nuScenes utilizes the 2D center distance on the ground plane to determine matches between predicted and ground truth bounding boxes.

A prediction is considered a true positive (TP) if the Euclidean distance between centers of the predicted and ground truth boxes is below a given threshold. The Average Precision (AP) is computed for each class as the integral of the Precision-Recall curve, constrained to the region where both precision and recall exceed 0.1. The mAP is then calculated as the

mean of AP across multiple distance thresholds (0.5 m, 1 m, 2 m, and 4 m), followed by averaging across all object classes.

By relying on center distance instead of IoU, the nuScenes mAP metric becomes more robust to small errors in size and orientation estimation of bounding boxes, thereby emphasizing accurate localization of object centers.

Evaluating 3D object detection models requires robust metrics that reflect both detection accuracy and spatial alignment. The most common metrics include:

Precision and Recall:

Precision and recall are fundamental metrics for evaluating detection performance. Precision is defined as the ratio of true positives *(TP)* to the sum of true positives and false positives *(FP)*, while recall is the ratio of true positives to the sum of true positives and false negatives *(FN)*, as defined in Equation (2.4).

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Mean Average Precision (mAP):

mAP summarizes detection performance across all classes. It is computed as shown in Equation (2.5)

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{2.5}$$

where AP_i is the average precision for class i, and N is the number of classes.

Intersection over Union (IoU):

IoU quantifies the overlap between predicted and ground-truth 3D bounding boxes. It is calculated as the ratio of their intersection to their union, as described in Equation (2.6).

$$IoU = \frac{Volume\ of\ Intersection}{Volume\ of\ Union} \tag{2.6}$$

TP Metrics: mATE, mASE, mAOE, mAVE, mAAE

In addition to mAP, nuScenes defines five TP metrics to capture specific aspects of detection quality. These metrics are computed only on TP predictions i.e.; predictions matched with ground truth under a 2D center distance threshold of 2 meters.

The TP metrics include:

- **mATE (mean Average Translation Error):** Measures average positional error as the 2D Euclidean distance (in meters) between the predicted and ground truth centers.
- **mASE (mean Average Scale Error)**: Measures average scale error, computed as $mASE = 1 - IOU$ after aligning both the centers and orientations of the predicted and ground truth objects.
- **mAOE (mean Average Orientation Error):** Measures the average yaw angle error (in radians), defined as the smallest absolute difference in yaw between the predicted and ground truth boxes. The orientation is treated as a periodic angle with a period of π for the *barrier* class and 2π for all other classes. This metric is not applicable to the *traffic_cone* class.
- **mAVE (mean Average Velocity Error):** Measures the average velocity error, computed as the L2 norm of the difference between the predicted and ground truth 2D velocity vectors (in meters per second). This metric is **not applicable** to the *traffic_cone* and *barrier* classes.
- **mAAE (mean Average Attribute Error):** Measures the average classification error of object attributes, calculated as

$$mAAE = 1 - accuracy_{attribute} \quad (2.7)$$

In Formula (2.7), **accuracy** refers to the correctness of attribute classification. This metric is **not applicable** to the *traffic_cone* and *barrier* classes. The TP metrics provide a more nuanced evaluation of model performance. For instance, a model may achieve a high mAP while exhibiting a large mAOE, which could significantly hinder its ability to predict future motion. Such multi-dimensional evaluation is particularly important in safety-critical applications.

NuScenes Detection Score (NDS): Formula, Components, and Weighting

The NDS is a composite metric designed to summarize overall detection performance. It is computed as a weighted sum of mAP and the five transformed TP metrics.

The aforementioned metrics mAP, mATE, mASE, mAOE, mAVE, and mAAE are consolidated through a weighted summation to provide an overall assessment of detection performance. Specifically, the true positive score (TP_{score}) is defined as follows:

$$TP_{score} = max(1 - TP_{error}, 0,0), \tag{2.8}$$

where TP_{error} denotes the error associated with each true positive instance. The use of the maximum function ensures that TP_{score} remains non-negative, with a lower bound of zero. This formulation penalizes higher errors while preventing negative scoring, thereby maintaining the interpretability and stability of the evaluation metric.

First, the TP errors (mATE, mASE, mAOE, mAVE, mAAE) are converted into TP scores using Formula (2.8).

NDS Formula:

The NDS metric combines mean Average Precision (mAP) with several mean error metrics including translation (mATE), scale (mASE), orientation (mAOE), velocity (mAVE), and attribute (mAAE) to provide a comprehensive evaluation of detection performance. It is calculated as shown in Formula (2.9), where each error term TP_{error} represents the mean error for its respective metric.

$$NDS = \frac{1}{10}\left(5 \times m\ A\ P + \sum_{TP \in \{mATE, mASE, mAOE, mAVE, mAAE\}} \left(1 - TP_{error}\right)\right), \tag{2.9}$$

where, **mAP** is assigned a weight of 5, while each TP metric contributes a weight of 1, resulting in a **total weight of 10**.

This weighting scheme reflects the fact that **mAP** representing overall detection and classification performance is considered the most critical factor. Nevertheless, specific aspects such as localization, orientation, velocity, and attribute prediction also play essential roles. A model aiming to achieve a **high NDS** must perform well across all

metrics, thereby promoting the development of **comprehensive and robust 3D detection systems**.

2.5. MMDetection3D Framework for LiDAR-Camera Fusion Research

MMDetection3D is a powerful and flexible open-source framework built on top of **PyTorch**. It plays a central role in the research and development of **3D object detection algorithms**, including approaches that integrate **LiDAR and camera data** [15].

2.5.1. Framework Architecture and Core Components

MMDetection3D inherits the modular architecture of MMDetection (a 2D object detection framework), enabling easy customization and extension[15]. The main components include:

- **Backbones**: Deep neural networks responsible for feature extraction from input data, such as PointNet++, ResNet, or VoxelNet.
- **Necks**: Modules that bridge the backbone and detection head, typically used to aggregate or refine multi-scale features, such as the Feature Pyramid Network (FPN).
- **Heads**: Perform tasks like classification and bounding box regression, with representative architectures including CenterHead and VoteHead.
- **Data Pipelines**: Handle input data through a series of steps including loading, preprocessing, and data augmentation.
- **Datasets**: Classes that define how specific datasets are loaded and processed, such as NuScenes [15].
- **Models**: Describe the overall model architecture, including the connections between backbone, neck, and head components.

This modular design allows researchers to easily swap and experiment with individual components (e.g., replacing the backbone or designing a new fusion module) without rewriting the entire codebase. Seamless integration with MMDetection [15] further enables the use of pre-trained 2D backbones, which is particularly beneficial for image-processing branches in LiDAR-Camera fusion models.

2.5.2. NuScenes Dataset Integration: Data Preparation, Info Files, Loaders, and Augmentation Pipeline

MMDetection3D provides standardized tools and workflows for processing the NuScenes dataset:

- **Data Preparation**: The script tools/create_data.py nuscenes is used to process raw data and generate necessary info files for training and evaluation. This process creates an object database (nuscenes_database) and .pkl files containing training and validation metadata (e.g., nuscenes_infos_train.pkl).
- **Info Files (.pkl)**: Store metadata for each frame, including LiDAR data paths, intermediate sweeps, camera calibrations, object annotations (e.g., gt_boxes, gt_names), LiDAR/Radar point counts, and other fields. The data must be transformed into MMDetection3D's internal coordinate system.
- **Data Loaders and Pipelines**: Defined in configuration files, including steps such as LoadPointsFromFile, LoadAnnotations3D, data transformations, and formatting modules like DefaultFormatBundle3D, Collect3D.
- **LoadPointsFromMultiSweeps**: A crucial module for working with NuScenes, enabling the aggregation of point clouds from multiple LiDAR sweeps (e.g., 10 sweeps) to increase point density and improve detection performance. However, it requires careful handling of motion compensation and timestamp alignment.
- **Data Augmentation**: Supports techniques such as global rotation, scaling, and translation (GlobalRotScaleTrans), random flipping (RandomFlip3D), point filtering (PointsRangeFilter), and sample mixing strategies like MixUp, CutMix, and PillarMix [16]. Maintaining consistency between LiDAR and camera branches remains a challenge in fusion models.

2.5.3. Configuration System for Experiments with NuScenes

MMDetection3D utilizes a flexible configuration system based on Python .py files. Each configuration file defines the entire workflow of an experiment, including:

- Data paths and info file references
- Model architecture: backbone, neck, head, fusion modules, loss functions

- Training schedule: optimizer, learning rate, number of epochs
- Evaluation settings and other runtime parameters

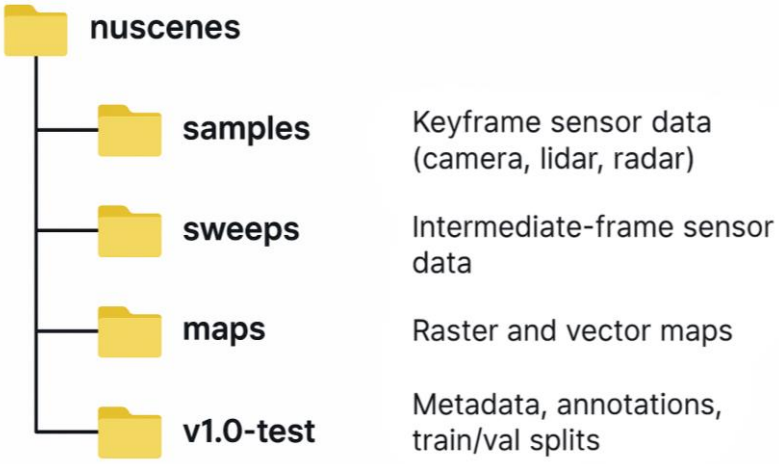


Figure 2.16 nuScenes Data Organization Diagram

The configuration inheritance mechanism enables the creation of experimental variants by overriding only the necessary parameters. This approach is particularly useful for systematic ablation studies and hyperparameter tuning. The structure of this data organization is illustrated in Figure 2.16.

2.5.4. Supported LiDAR-Camera Fusion Models for NuScenes

Six distinct urban scenes annotated with color-coded bounding boxes are depicted in [17, Fig. 2.17], highlighting detected objects such as pedestrians, vehicles, and other traffic elements relevant to autonomous driving.

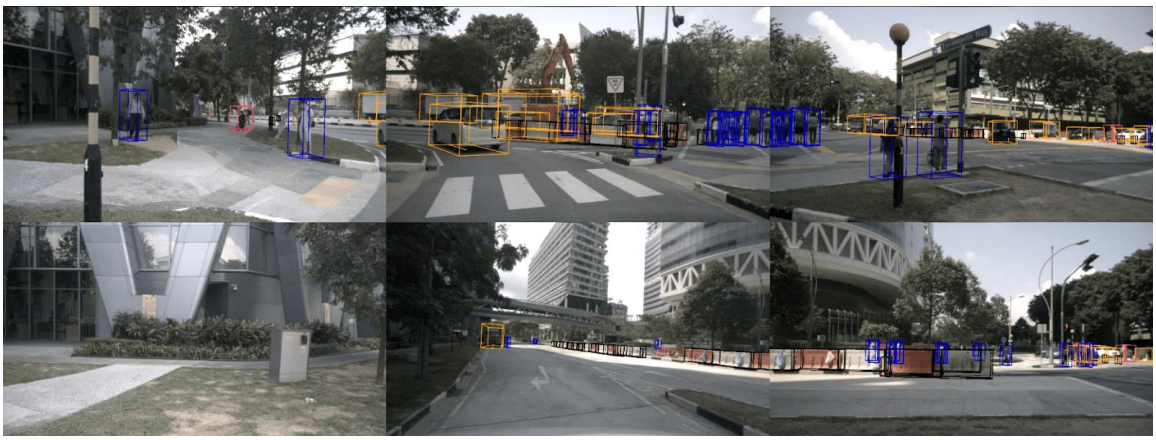


Figure 2.17 LiDAR-Camera Fusion Models in NuScenes: Urban Object Perception

MMDetection3D supports a diverse range of LiDAR-Camera fusion and multimodal learning models, each tailored to address the unique challenges of the NuScenes dataset. These models leverage different fusion strategies and architectural innovations to maximize the complementary strengths of LiDAR and camera data. The details of the selected models are summarized in Table 2.3.

Prominent models include:

- **BEVFusion**: A state-of-the-art fusion model that integrates multi-sensor features in the Bird's Eye View (BEV) space [18]. By projecting both LiDAR and camera features into a unified BEV representation, BEVFusion enables efficient spatial reasoning and facilitates the use of 2D convolutional networks for 3D scene understanding. This approach has demonstrated superior performance in complex urban environments, where accurate spatial alignment of multimodal data is critical.
- **mmFUSION**: Employs multi-dimensional attention mechanisms for intermediate-level fusion between LiDAR and camera modalities [19].
- **CenterPoint**: A strong LiDAR-only baseline that can be extended for image integration [20]. Its anchor-free, center-based detection paradigm allows for flexible integration of additional modalities, making it a popular choice for multimodal research.
- **DETR3D**: A Transformer-based model that uses 3D queries to interact with image features from multiple views [21]. This enables the model to reason about object locations in 3D space by directly attending to relevant image regions, leveraging the global context provided by the Transformer architecture.
- **PETR**: Similar to DETR3D but utilizes a distinct 3D positional encoding [22].
- **TransFusion**: Combines LiDAR query representations with image features using a Transformer-based decoder[23]. By fusing information at the feature level, TransFusion can capture complex relationships between modalities, leading to more accurate and reliable object detection.

The increasing prevalence of BEV-based models (e.g., BEVFusion) and Transformer architectures within the MMDetection3D ecosystem highlights the BEV representation as

a dominant intermediate space for LiDAR-Camera fusion. This is attributed to its intuitive spatial representation of 3D scenes and the efficient application of 2D convolutional operations. Moreover, the attention mechanisms of Transformers offer high flexibility for integrating information from multiple sensor modalities.

Table 2.3 Selected LiDAR-Camera Fusion Models in MMDetection3D for NuScenes

Model Name	**Core Fusion Strategy**	**Key Architectural Components**	**NDS on NuScenes**
BEVFusion [18]	BEV Feature Fusion	Camera Encoder (LSS), LiDAR Encoder (Voxel), BEV Fusion Module, BEV Detection Head	71.4%
mmFUSION [19]	Intermediate Feature Fusion (Attention)	Camera Encoder, LiDAR Encoder, Cross-Modal and Multi-Modal Attention, 3D Detection Head	69.75%
DETR3D [21]	3D Object Queries + Image Features	2D Image Backbone, Transformer Decoder with 3D Queries, Camera Positional Encoding	48%
PETR [22]	3D Object Queries + Image Features	2D Image Backbone, Transformer Encoder-Decoder with 3D Queries, 3D Positional Encoding	44.2%
TransFusion [23]	LiDAR Queries + Image Features (Transformer)	LiDAR Backbone (Voxel), Image Backbone (2D), Transformer Decoder combining queries and features	73%

Note: NDS performance may vary depending on the MMDetection3D version, specific configurations, and dataset splits (val/test).

2.6. In-depth Analysis of the MEFormer Model

MEFormer (Multi-Level Enhancement Transformer) is a Transformer-based 3D object detection model tailored for autonomous driving. It leverages the Transformer's ability to capture long-range dependencies and integrates multi-level features to improve detection accuracy and efficiency from point cloud data. The model's multi-level enhancement module enables effective processing of both local and global information, significantly boosting performance in complex environments.

2.6.1. Overall Architecture of MEFormer

An overview of the proposed multimodal 3D object detection architecture is presented in [24, Fig. 2.18], which employs two distinct modalities: images from cameras and point clouds from LiDAR. Feature maps are concurrently extracted from both data sources using separate backbone networks. These features are then processed through three decoding branches, each initialized with the same object query Q. All branches share a transformer decoder f but differ in the combinations of modality inputs used as keys and values specifically, LiDAR and camera (LC), LiDAR only (L), and camera only (C) resulting in intermediate box features Z_{LC}, Z_L, and Z_C, respectively. Each branch's predicted boxes are directly supervised using ground truth annotations. A cross-attention-based Prediction Modality Ensemble (PME) module follows, using Z_{LC} as the query and Z_{LC}, Z_L, and Z_C as keys and values to generate the final ensembled box features Z_e. The output boxes from Z_e are also supervised to refine detection accuracy.

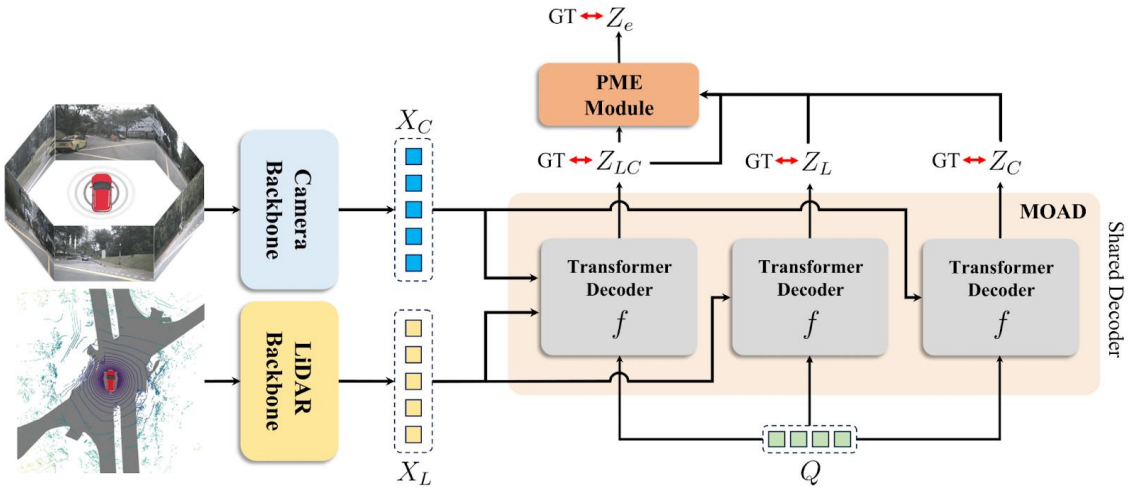


Figure 2.18 The overall architecture of MEFormer

The architecture of MEFormer comprises three main components: **Cross-Modal Transformer (CMT)** backbone [25], a **Modality-Agnostic Decoding (MOAD)** scheme, and a **proximity-based modality ensemble (PME)** module. The detailed descriptions of each component are as follows:

- **CMT:** This component is responsible for extracting features from the input data typically point clouds from LiDAR at multiple resolution levels. MEFormer may adopt a backbone such as PointNet++ or a similar 3D network to process the data and generate features across various scales. This enables the model to capture both local information (e.g., fine-grained object details) and global information (e.g., overall scene layout). The output is a set of multi-level features that serve as the foundation for the subsequent processing stages.
- **MOAD:** This module forms the core of the MEFormer architecture and utilizes Transformer layers to enhance the features extracted in the previous stage. The MEFormer Module applies attention mechanisms to process and refine feature representations, enabling the model to learn inter-feature relationships across different levels. MEFormer creates three decoding branches from the same shared decoder:

- **LiDAR-only branch**: decoder takes only LiDAR features X_L
- **Camera-only branch**: decoder takes only camera features X_C
- **Multi-modal branch**: decoder takes both features X_{LC}

Formally, if f denotes the shared transformer decoder, the branch outputs are:

$$Z_L = f(Q, X_L) \tag{2.10}$$

$$Z_C = f(Q, X_C) \tag{2.11}$$

$$Z_{LC} = f(Q, [X_L; X_C]) \tag{2.12}$$

Using modality-specific positional embeddings for Z_L, Z_C, Z_{LC}. The resulting Z values, as defined in Equations (2.10), (2.11), and (2.12), correspond to sets of decoded box features produced by their respective branches.A **single box prediction head** $Head_{box}$ (classification + regression) is then applied to each branch's output. Thus for branch $m \in \{L, C, L+C\}$, the predicted boxes B_m and scores S_m are:

$$(B_m, S_m) = Head_{box}(Z_m) \quad (2.13)$$

This head is shared across branches, making it "modality-agnostic"

During training, each branch is supervised independently: Hungarian matching is performed between the ground-truth objects and the predictions of each branch. A loss $\mathcal{L}_m$ (e.g. focal + L1) is computed for each branch, and all branch losses are summed, as formulated in Equation (2.14):

$$\mathcal{L}_{MOAD} = \sum_{m \in \{L, C, L+C\}} \mathcal{L}_m \quad (2.14)$$

This forces the decoder (and modality backbones) to learn full geometric and semantic decoding from each input type [24]. This module plays a crucial role in improving object detection performance by effectively combining local and global contextual information.

- **PME:** After feature enhancement, the representations are fused to generate a unified feature representation. The feature fusion process integrates information across levels to ensure a comprehensive understanding of the scene. This fused representation is then passed to the Task Head, which performs specific detection tasks such as object classification (e.g., vehicle, pedestrian) and bounding box estimation. The Task Head may include fully connected layers or specialized modules to accurately predict object properties.

Concretely, let $\{B_L, B_C, B_{L+C}\}$ be the sets of boxes (and associated box features) output by each branch. PME first **projects** each branch's box features via a learned linear transform. Then it performs a *cross-attention* among all boxes: each box from one branch can attend to boxes from all branches (including its own) as "keys"/"values". However, to discourage mismatches, PME adds a **distance-based attention bias**. It computes the BEV-center distance $d_{ij} = |c_i - c_j|$ between the centers of any two predicted boxes *i, j* (from possibly different branches). These distances are linearly mapped to bias terms. This relationship is formalized in Equation (2.15):

$$b_{ij} = \alpha \cdot d_{ij} + \beta, \quad (2.15)$$

where α, β are learned scalar and bias. Larger center separations yield larger biases. This bias is added to the raw attention score before softmax, so that boxes far apart (unlikely to be the same object) attend less to each other [24].

Formally, if Q_i is the query vector of box i and K_j, V_j are the key/value of box j, the (biased) attention is given by Equation (2.16):

$$Attn(Q_i, K_j) = softmax(Q_i K_j^{\top} + b_{ij}) \quad (2.16)$$

So nearby boxes boost each other's contribution. Through this biased cross-attention layer, each box's feature is updated by pooling information from nearby boxes across modalities. In effect, corresponding detections from LiDAR and camera reinforce each other, while contradictory ones (far apart in BEV) are downweighted.

2.6.2. Advantages and Limitations of MEFormer

MEFormer offers several advantages over existing 3D object detection models:

- **Capability to capture long-range dependencies:** The attention mechanism of Transformers enables MEFormer to model complex spatial relationships, especially in scenes with occlusions.
- **Efficient multi-level feature processing**: Integrating information across different levels allows the model to accurately detect objects at varying scales.
- **Optimized computational efficiency**: MEFormer is designed to balance detection accuracy with computational cost, making it suitable for real-time applications.
- **High generalization ability**: The model performs robustly across diverse datasets and environmental conditions, owing to its strong feature learning capacity.

Despite these strengths, MEFormer has several limitations:

- **Computational cost**: Transformer-based architectures are resource-intensive, especially when applied to large-scale 3D data.
- **Training data requirements**: The model demands large and diverse datasets to achieve optimal performance.
- **Scalability**: Extending MEFormer to more complex tasks may require further architectural enhancements.

2.6.3. Comparison with Other 3D Object Detection Models

The performance of MEFormer can be evaluated by comparing it with other state-of-the-art 3D object detection models on standard benchmark datasets such as **nuScenes**. Table 2.4 presents a comparison of MEFormer against several leading models on the nuScenes validation set, using two widely adopted metrics: **NDS** and **mAP**.

Table 2.4 Performance Comparison on the nuScenes Validation Set

Model	**NDS**	**mAP**	**Year**
EA-LSS [26]	0.78	0.77	2023
MEFormer [24]	**0.74**	**0.72**	**2024**
UniTR [27]	0.73	0.71	2023
SparseFusion [28]	0.73	0.71	2023
BEVFusion [18]	0.71	0.69	2023
mmFusion [19]	0.7	0.65	2022
CenterPoint [20]	0.67	0.60	2021

Based on the results above, MEFormer achieves strong performance metrics (NDS: 0.74, mAP: 0.72), demonstrating competitiveness with leading multi-modal 3D object detection models. It matches or outperforms established models like BEVFusion and is only marginally behind newer models such as EA-LSS. These results indicate that MEFormer's strategies for reducing LiDAR over-reliance and mitigating negative fusion are both effective and robust.

Architecturally, MEFormer stands out through its use of MOAD and PME modules. While BEVFusion transforms and fuses image and point cloud features in BEV space, MEFormer employs a shared decoder for both modalities and leverages a proximity-based fusion mechanism for final predictions.

CHAPTER 3: SYSTEM DESIGN

3.1. Design Requirements

This section presents the design criteria for a 3D object detection and visualization system in urban traffic environments, utilizing the MEFormer model on the nuScenes mini dataset. The requirements are categorized into four main groups: functional, performance, robustness, and computational aspects.

3.1.1. Functional Requirements

The 3D object detection system is designed to perform inference on the nuScenes mini dataset, aiming for efficient operation in real-world urban traffic scenarios. The system must accurately detect and localize key traffic object classes, including cars, motorcycles, bicycles, buses, trucks, and pedestrians. Each object is represented by a 3D bounding box, characterized by absolute spatial coordinates (x, y, z), physical dimensions (length, width, height), and orientation angle.

To leverage the advantages of different sensor types, the system integrates data from both RGB cameras and LiDAR sensors through a multi-modal fusion strategy. Specifically, the MEFormer model is implemented with an architecture optimized for multimodal 3D object detection tasks, incorporating modality-agnostic decoding and proximity-based modality ensemble techniques.

In addition to detection accuracy, the system must maintain robust operation under complex urban scenarios such as traffic congestion, partial occlusion of objects, and high object density. Efficient handling of these cases is essential to ensure system usability in practical applications, including autonomous driving assistance and traffic monitoring.

Regarding visualization, the system should provide an interactive 3D interface on personal computers, enabling the display of detection results in .obj (3D data) and .json (structured data) formats. Detected objects are visualized with 3D bounding boxes overlaid on camera images and clearly marked within the LiDAR point cloud. The visualization system must support operations such as rotation, zooming, object filtering, and include performance analysis tools to facilitate comprehensive model evaluation.

3.1.2. Performance Requirements

The selection of the 3D object detection model is based on specific quantitative criteria to ensure accuracy and efficiency for deployment in real urban environments. The primary evaluation metrics include:

- **Mean Average Precision (mAP):** Measures the overall detection accuracy across multiple object classes, such as cars, motorcycles, and pedestrians.
- **Mean Average Translation Error (mATE):** Assesses spatial accuracy by calculating the average positional error between predicted and ground-truth object locations in 3D space.
- **Mean Average Orientation Error (mAOE):** Reflects the average deviation in orientation angles of detected objects compared to their actual orientations.

These metrics will be computed and analyzed on the nuScenes mini dataset to ensure reliable performance evaluation. The MEFormer model is selected based on its superior results and is directly compared with several state-of-the-art 3D detection methods, including PointPillars and SECOND. The performance of several widely used 3D object detection models, including metrics such as mAP (%), mATE (m), and mAOE (rad), has been systematically compared and presented in Table 3.1.

Table 3.1 Performance Comparison of 3D Object Detection Models

Models	mAP (%)	mATE (m)	mAOE (rad)
BEVFormer (C) [29]	44.5	0.58	0.38
DETR3D (C) [21]	41	0.64	0.39
BEVFusion (L+C) [18]	71.3	0.25	0.36
MEFormer (L+C) [24]	72	0.27	0.3

MEFormer demonstrates superior performance with mAP = 72%, outperforming BEVFormer by 24% and BEVFusion by 0.7%. The model achieves mATE = 0.27 meters and mAOE = 0.3 radians, indicating higher accuracy in both object localization and orientation estimation.

Expected performance metrics on the nuScenes mini subset:

Based on calibration factors derived from validation set performance, the predicted metrics include **mAP ≥ 53%** on the nuScenes mini test set, **mATE ≤ 0.49 meters** to ensure acceptable localization accuracy, and **mAOE ≤ 0.38 radians** for reliable orientation estimation.

3.1.3. Robustness Requirements

The robustness of the 3D object detection system is a critical factor, especially in real-world urban environments characterized by diverse operating conditions. The system must maintain stable performance across scenarios involving significant variations in lighting, weather, and traffic scene complexity.

The nuScenes dataset is introduced as the first comprehensive multi-sensor dataset designed for autonomous vehicle systems, integrating data from six cameras, five radars, and one LiDAR sensor, with a full 360-degree field of view. This dataset addresses existing gaps in multi-sensor datasets, which often lacked adequate representation of real-world challenges.

Data collection was conducted across multiple carefully selected routes to ensure diversity in geographic locations, time of day (daytime and nighttime), and weather conditions (sunny, rainy, and overcast). Particular emphasis was placed on capturing scenes with varied lighting and weather conditions to enhance the dataset's representativeness and reliability for model evaluation.

nuScenes is the first multi-sensor dataset to provide data under nighttime and rainy conditions, a crucial factor for studying the adaptability and robustness of deep learning models in uncertain environments. According to dataset statistics, keyframes include 19.4% captured during rain and 11.6% during nighttime, demonstrating significant environmental coverage [6].

This design enables nuScenes to effectively support a range of computer vision and deep learning tasks, including 3D object detection, object tracking, and behavior modeling, particularly under complex and unpredictable environmental conditions.

The MEFormer model was selected due to its flexible adaptability to the aforementioned environmental conditions, enabled by its transformer architecture combined with modality-agnostic decoding and proximity-based modality ensemble mechanisms. Specifically, when camera image quality degrades due to poor lighting, the system automatically increases the weighting of geometric features from LiDAR. Conversely, under weather conditions that reduce LiDAR signal reliability, visual features from cameras play a dominant role in the final detection decision.

The ability to handle diverse data is further enhanced through a specialized attention mechanism, allowing the system to effectively adapt to varying scene complexities and object densities. Weighting based on the reliability of each data source according to spatial proximity and model confidence ensures robustness in real urban traffic scenarios, outperforming methods relying solely on single-sensor LiDAR input, such as PointPillars.

3.1.4. Computational Requirements

The 3D object detection system for urban environments must not only ensure high accuracy and robustness in identifying and localizing objects but also operate efficiently under limited computational resources. This requirement becomes especially critical when the system is deployed on standard hardware infrastructures, such as virtual private servers (VPS) or personal laptops that lack dedicated GPUs.

Computational constraints during inference:

When deployed for inference on the nuScenes mini dataset, the MEFormer model requires relatively moderate computational resources compared to current state-of-the-art models.

To ensure efficiency throughout model training, data processing, and result analysis, appropriate hardware selection is critical. This study clearly delineates two main task groups: model training and data processing are conducted on servers or VPS infrastructure; visualization and performance analysis are performed locally on personal computers. A detailed comparison between the proposed hardware configuration and the actual configuration used in this study, clarifying the suitability and performance of each system for the respective task groups, is presented in Table 3.2.

Table 3.2 Comparison of Proposed and Actual Hardware Configurations

Usage Purpose	**Parameter**	**Proposed Configuration**	**Actual Configuration**
Model Training (server)	RAM	16 GB	33 GB
	GPU	NVIDIA RTX 3090	NVIDIA Tesla T4
	Storage	10 GB (including 5 GB data)	15 GB
Analysis and Visualization (local)	RAM	8 GB	16 GB
	GPU	Onboard GPU	NVIDIA GTX 1050
	Storage	20 GB	512 GB

For tasks involving model training and complex data processing such as multimodal feature extraction and deep neural network training, the proposed configuration includes a high-performance GPU (e.g., RTX 3090) and a minimum of 16 GB RAM to meet the high computational demands. In practice, the system is deployed on the Lightning.ai VPS platform equipped with an NVIDIA Tesla T4 GPU and 33 GB of RAM, exceeding the proposed specifications. This setup enables efficient training within a reasonable timeframe without encountering memory bottlenecks.

For visualization and model result analysis tasks, the proposed configuration prioritizes cost efficiency, requiring only 8 GB of RAM and an integrated GPU. However, during experimentation, the actual device used was a laptop with higher specifications, including 16 GB RAM and an NVIDIA GTX 1050 GPU, which accelerated BEV image rendering, detection result analysis, and statistical chart display.

The discrepancy between the proposed and actual configurations reflects flexibility in system deployment and demonstrates that effectively leveraging available resources can enhance performance without compromising result quality.

Optimization Strategy:

To adapt to hardware limitations, various optimization strategies have been implemented. Additionally, batch processing techniques are applied to optimize performance when handling multiple consecutive data samples. During the visualization phase, level-of-detail rendering and occlusion culling techniques help maintain smooth interactive experiences even on standard hardware.

The effective integration of MEFormer within the MMDetection3D framework is also a key factor, as this framework supports hardware acceleration and optimization techniques on non-specialized hardware, outperforming heavier models such as SECOND, which require significantly greater computational resources.

Local Visualization Computational Constraints:

The desktop application for result visualization is developed using PyQt5 combined with Open3D for 3D rendering. The system requires moderate configuration, with a minimum of 8 GB RAM to simultaneously handle point clouds, 3D models, and the user interface. Computers using integrated graphics can still perform interactive visualization at an acceptable level; however, equipping a discrete GPU significantly improves performance when rendering dense point clouds.

3.2. System Pipeline

The proposed method encompasses a detailed pipeline from data collection to result visualization, utilizing the MEFormer model on the nuScenes mini dataset.

The entire pipeline is designed with a hybrid architecture combining cloud-based and local computing, as illustrated in Figure 3.1. Specifically, preprocessing and inference steps utilize the computational resources of a VPS, leveraging MEFormer's batch processing capabilities. Inference results are exported in .json and .pkl formats, then transferred to a personal computer for analysis and visualization. The visualization component is developed using PyQt5 integrated with the Open3D library, supporting the display of 3D bounding boxes (in .obj format), data overlays on camera images, and interactive navigation within the point cloud..

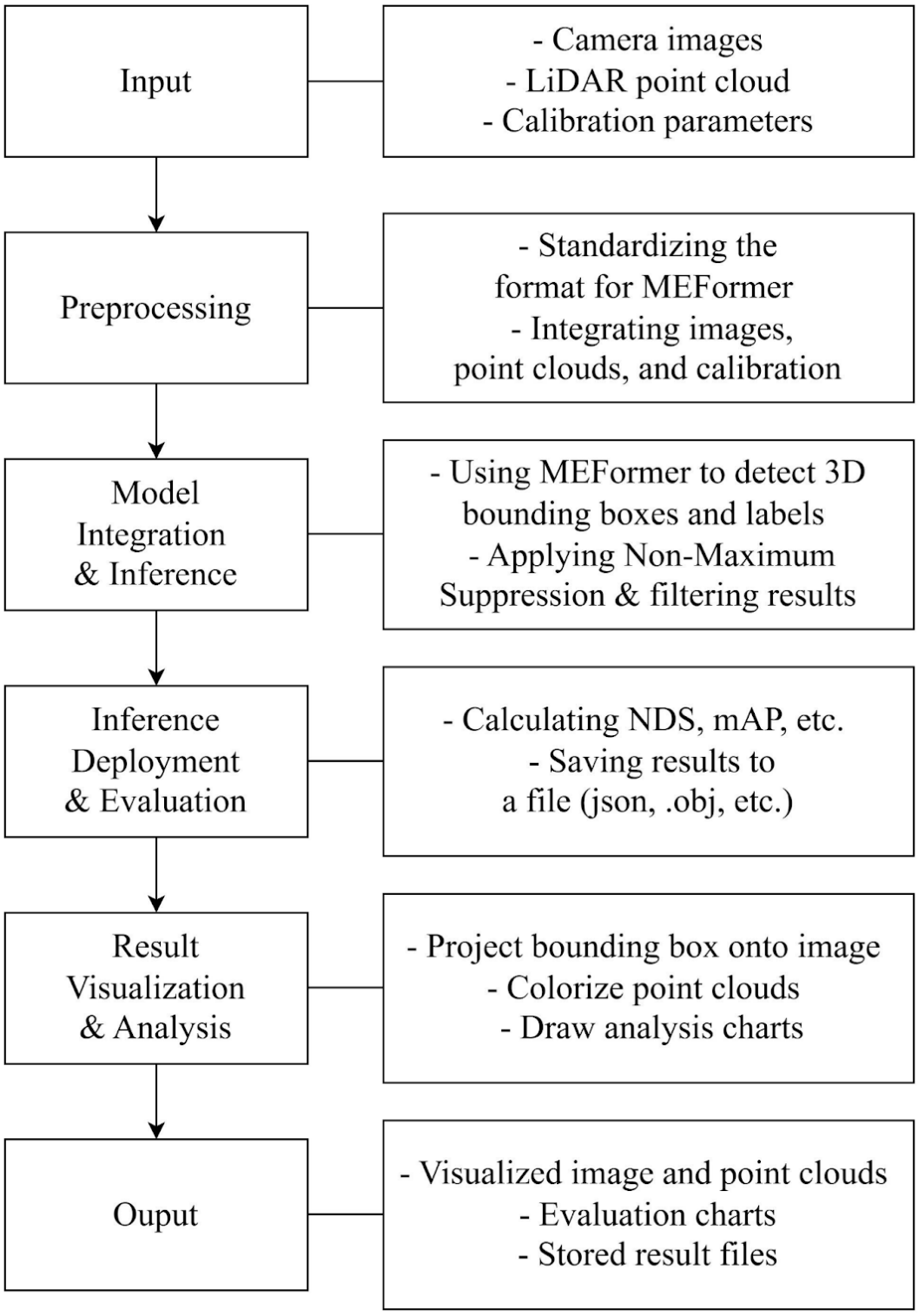


Figure 3.1 Pipeline for Applying the MEFormer Model to 3D Object Detection in Traffic Environments

MEFormer was selected due to its prominent characteristics that meet the requirements of a multi-sensor 3D object detection system. First, MEFormer effectively leverages fused data from cameras and LiDAR, enhancing object recognition capabilities in urban environments characterized by high noise levels and diverse object sizes and shapes. Second, the model demonstrates the potential to achieve competitive results in terms of mAP, mAOE, and mATE, owing to its unified feature space architecture and flexible deep learning mechanisms. Additionally, MEFormer exhibits robustness across various operating conditions including changing lighting, partial occlusions, and heterogeneous sensor data-attributable to its multi-layer deep learning structure and robustness enhancement strategies.

From an implementation perspective, MEFormer is optimized for the MMDetection3D ecosystem, enabling efficient operation on both remote servers (VPS) and academic laptops with limited hardware configurations. This optimization contributes to reduced infrastructure costs during experimentation and evaluation phases.

The evaluation system extends beyond aggregate quantitative metrics by providing detailed analyses segmented by object category, environmental conditions (rain, nighttime, heavy traffic), and detection difficulty (small objects, occluded objects, distant objects). Statistical analyses elucidate performance trends relative to object size, occlusion level, and input sensor data quality, thereby identifying the system's strengths and weaknesses.

Evaluation results are dynamically visualized through integrated charting functions within the PyQt5 interface. Users can interactively explore relationships among performance metrics or directly compare predicted outputs against ground truth annotations on a frame-by-frame basis. This functionality effectively supports in-depth behavioral analysis of the model under complex scenarios present in the nuScenes mini dataset.

3.2.1. Data Collecting and Preprocessing

The system is built upon the nuScenes mini dataset, which includes comprehensive information from RGB cameras, LiDAR point clouds, and 3D annotations of objects in urban traffic environments. This dataset is a condensed version of the full nuScenes dataset

but maintains representativeness in terms of scene diversity, weather conditions, and object categories, making it suitable for research and experimentation with multi-modal 3D object detection models.

The data processing pipeline consists of three main stages: calibration, format conversion, and normalization. During calibration, camera and LiDAR data are aligned using calibration parameters provided by the dataset to ensure spatial and temporal synchronization across sensors. Next, the data are converted into formats compatible with the MEFormer model, including tensors and data structures supported by the MMDetection3D framework. The final stage involves data normalization, which includes pixel value normalization for images and coordinate normalization for point clouds to enhance stability and efficiency during inference.

After preprocessing, the data are saved as .pkl files to serve subsequent inference steps within the MMDetection3D pipeline. Compared to traditional manual preprocessing approaches (e.g., using the PCL library), this automated pipeline leverages MMDetection3D's tools to reduce processing time and minimize errors caused by manual operations.

The clear separation between computationally intensive tasks (inference on VPS) and user-interactive tasks (local visualization) optimizes overall system performance and enhances flexible deployment across common hardware platforms.

3.2.2. Model Integration

After completing the data preprocessing stage, the system proceeds with model integration and performs inference to detect 3D objects in urban traffic environments. This process is implemented within the MMDetection3D framework, an open-source toolbox specialized for 3D object detection developed by OpenMMLab. The choice of MMDetection3D over other frameworks such as MMSegmentation ensures high compatibility with the MEFormer architecture and allows full utilization of available optimization tools tailored for 3D object detection tasks.

The MEFormer model is loaded as a pre-trained checkpoint from the official GitHub repository. The model requires input data to be formatted consistently with the parameters

used during its original training. Therefore, the data preprocessing module must guarantee full compatibility with MEFormer's input format.

For camera image data, preprocessing involves resizing images to a standard resolution of 1600×640 pixels to match the input specifications of the original training. For LiDAR point cloud data, the input spatial region is constrained to a cubic volume of 108 m × 108 m × 8 m centered around the ego vehicle to reduce noise and focus on objects with potential direct interaction. The data are then voxelized at a resolution of 0.075 m to generate input tensors of appropriate size, balancing spatial detail with memory and inference time efficiency. All camera and LiDAR coordinate systems are unified via calibration matrices provided by the nuScenes dataset, ensuring precise alignment across sensor modalities.

Inference is conducted on a VPS, where the MEFormer model processes batches of input data to produce 3D object detection results. These results include 3D bounding boxes with information on object position, size, and orientation, which are saved in .json or .pkl formats for subsequent analysis and visualization.

In summary, the model integration and inference pipeline not only ensures technical compatibility between data and model architecture but also optimizes system performance through the use of a specialized framework, precise preprocessing techniques, and efficient computational pipeline organization. This contributes to achieving the accuracy and inference speed required for practical deployment in real-world traffic environments.

3.2.3. Inference Deployment and Evaluation of the MEFormer Model

The inference process is conducted using the MEFormer model to detect 3D objects from multi-modal sensor data in the nuScenes mini dataset. MEFormer features an architecture comprising independent encoders for the two primary data sources: camera images and LiDAR point clouds. Extracted features from each modality are fused through a proximity-based ensemble mechanism, followed by a modality-agnostic decoder, enabling the system to flexibly adapt to varying input data quality [24].

Inference optimization is performed to ensure processing efficiency and resource conservation on non-specialized hardware. The model employs mixed precision computation techniques, reducing GPU memory usage by approximately 30-40% without

significant accuracy loss. Batch sizes are flexibly optimized within a range of 2 to 4 samples per iteration, depending on the available GPU memory. Additionally, dynamic batching is applied to adjust batch size based on scene complexity, maintaining stable throughput.

Inference outputs are extracted and processed into a standardized output structure, including model evaluation metrics, object classification across the 10 classes defined in the nuScenes dataset, 3D bounding box parameters (center coordinates, dimensions, orientation), and corresponding confidence scores. To eliminate duplicate detections, a non-maximum suppression (NMS) algorithm with a 3D IoU threshold of 0 is applied, retaining all detected object instances.

3.2.4. Result Standardization, Visualization and Analysis

Upon completion of inference, the system performs result standardization and stores outputs in two primary formats to support both automated analysis and visual inspection. Specifically, the .pkl format is used to store comprehensive detection information, including 3D bounding box coordinates as NumPy arrays, confidence score matrices, and metadata related to the scene and sensors.

Concurrently, the .json format is structured according to the standardized prediction result format of the nuScenes dataset, facilitating integration with external analysis tools or manual verification.

Each result file includes key metadata such as scene identifier, timestamp, sensor configuration, and processing parameters to ensure traceability and enable reanalysis across different conditions. Beyond data standardization, the system offers visualization tools for qualitative assessment: 3D bounding boxes are projected onto camera images for contextual review, while LiDAR point clouds with annotated detections support geometric and accuracy evaluation.

3.3. System Architecture and Implementation of Visualization and Analysis Application

3.3.1. Architecture Overview

The visualize_nuscenes_app system is designed based on a layered architecture model, comprising three main layers that interact with each other through clearly defined interfaces. This architectural approach ensures a high degree of modularity, facilitating ease of extension and maintenance

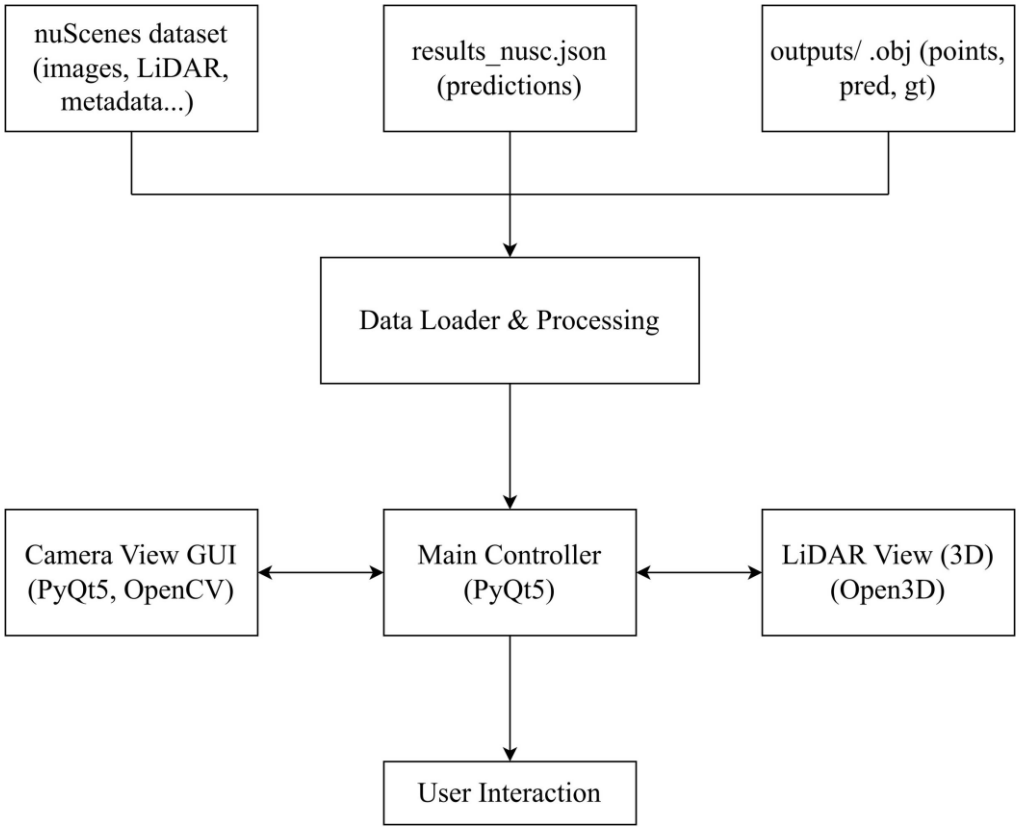


Figure 3.2 System Architecture Overview of the Visualization Application

Each layer has distinct responsibilities and communicates with other layers through clearly defined interfaces, as illustrated in Figure 3.2. User Interaction is facilitated through the Main Controller, allowing users to navigate scenes, adjust visualization parameters, and trigger exports or analyses. The system is designed to ensure smooth communication between modules, enabling real-time updates and responsive feedback to user actions.

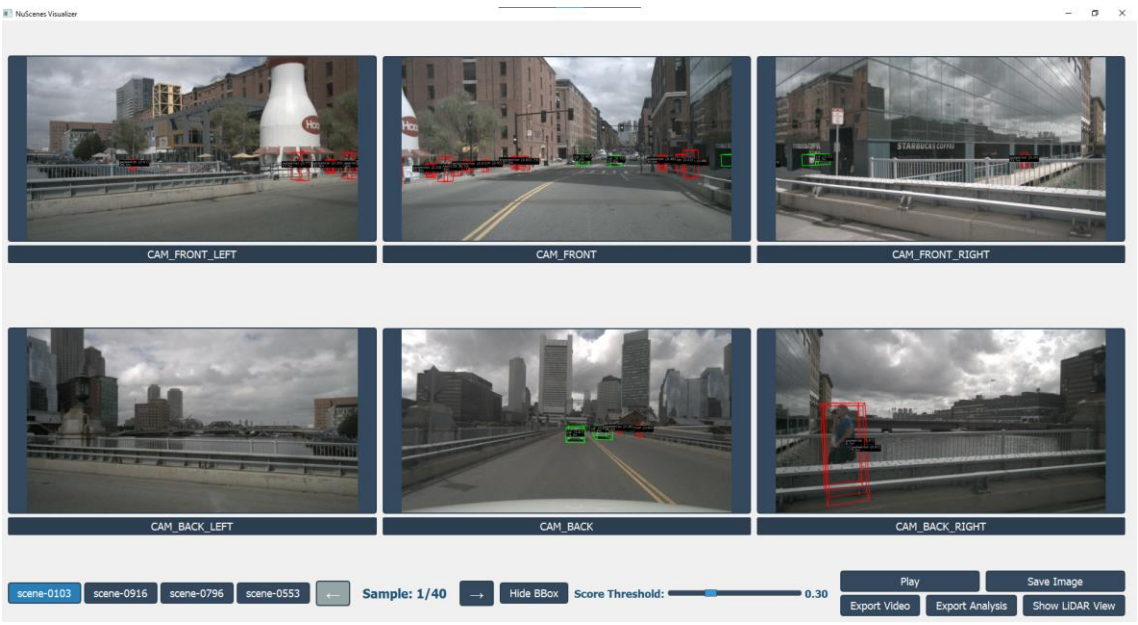


Figure 3.3 Main User Interface of the Application, Displaying Key UI Components

The Main Window serves as the entry point of the application, managing the overall layout and coordinating other UI components, as shown in Figure 3.3. The Camera Views display data from multiple cameras with various viewing angles, while the LiDAR View focuses on visualizing point cloud data with interactive 3D capabilities. The Control Panel provides tools for navigation control, visualization settings, and export management.

The Data Processing Layer acts as an intermediary, responsible for data handling and management. This layer includes the **Dataset Manager**, **Scene Manager**, **Frame Manager**, and **Box Manager**. The **Dataset Manager** oversees loading and processing the NuScenes dataset, ensuring efficient data loading and appropriate caching. The **Scene Manager** manages scene information and scene selection, whereas the **Frame Manager** handles frame data and frame navigation. The **Box Manager** is responsible for managing bounding boxes and annotations, ensuring their accurate display in both 2D and 3D views.

The Visualization Layer is the final layer, dedicated to data rendering. It consists of the 2D Visualizer, 3D Visualizer, View Synchronizer, and Interaction Handler. The 2D Visualizer is responsible for displaying camera images and 2D bounding boxes, while the 3D Visualizer focuses on rendering point clouds and 3D bounding boxes. The View

Synchronizer ensures synchronization across different views, and the Interaction Handler manages user interactions.

3.3.2. Technologies Used

The system is developed using Python 3.8+ as the primary programming language, combined with modern libraries and frameworks. PyQt5 is employed for the user interface, providing necessary widgets and tools for UI construction. Open3D is utilized for 3D visualization, enabling efficient rendering of point clouds and 3D bounding boxes. OpenCV is integrated for image processing and operations on camera images. The key technologies and their respective versions utilized in this study, along with their specific purposes, are systematically summarized in Table 3.3.

Table 3.3 Key Technologies Utilized

Technology	Version	Purpose
Python	3.8	Primary programming language
PyQt5	5.15.0	User interface
Open3D	0.17.0	3D visualization
OpenCV	4.7.0	Image processing
NumPy	1.23.0	Numerical data processing

The system processes the NuScenes dataset using the **nuscenes-devkit** for efficient data loading and parsing. **Pandas** handles tabular data such as annotations and evaluation results, while **JSON** manages metadata for compatibility and easy exchange. **NumPy** supports fast numerical computations, and **Matplotlib** is used for visualizing and exporting performance analysis and statistics.

3.3.3. Data Flow and Processing

The data flow within the system begins with loading the NuScenes dataset. The **Dataset Manager** is responsible for loading the dataset from the specified directory, parsing and validating the metadata, and initializing the necessary data structures.The overall data flow and processing pipeline of the visualization system, including input sources, processing stages, and output generation, is illustrated in Figure 3.4.

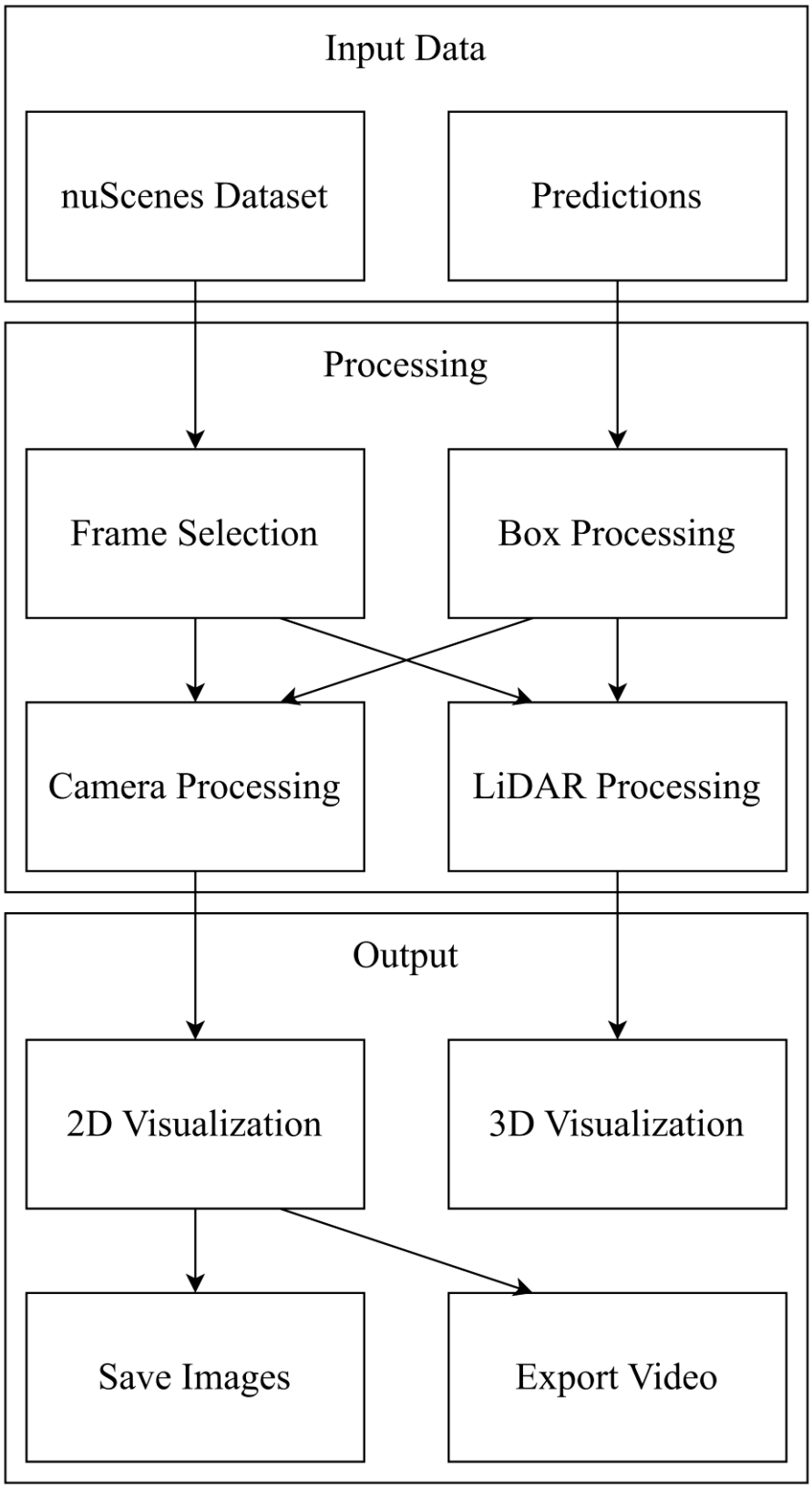


Figure 3.4 Data Flow Diagram in the Visualization System

After the dataset is loaded, the **Scene Manager** oversees the selection of scenes and frames. When a scene is selected, the **Frame Manager** loads and validates the frame data, preparing the necessary data structures for visualization. The **Box Manager** processes bounding boxes from annotations, performs the required coordinate transformations, and filters boxes based on predefined criteria.

The data is then processed by the Camera Processing and LiDAR Processing modules. Camera Processing loads camera images, applies necessary image transformations, and prepares images for 2D visualization. LiDAR Processing loads point cloud data, conducts filtering and coordinate transformations, and prepares points for 3D visualization.

3.3.4. Interaction Between Components

The interactions among system components are designed to ensure both consistency and efficiency throughout the user experience. The **Main Window** functions as the central coordinator, managing the flow of information between various **UI components** and ensuring that the **Camera Views** and **LiDAR View** remain synchronized in real time. The **Control Panel** serves as the primary interface for user input, sending commands to other components and facilitating seamless navigation, parameter adjustment, and data export. This architecture enables users to interact with the system intuitively while maintaining coherent and responsive visualization across all views.

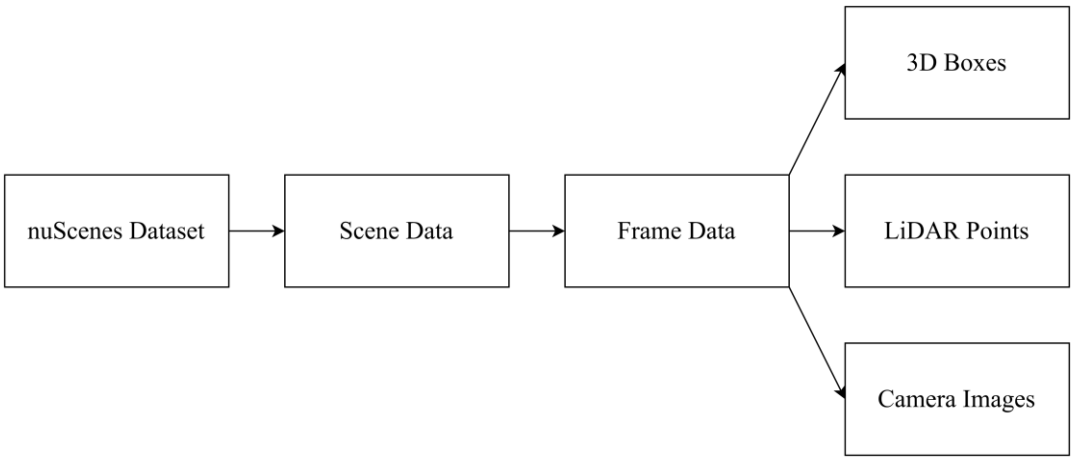


Figure 3.5: Dataset Manager Diagram

The **Dataset Manager** provides data to other components, ensuring efficient data access as illustrated in Figure 3.5. The **Scene Manager** handles scene selection, while the **Frame Manager** manages frame navigation. These components work collaboratively to guarantee accurate and consistent data visualization.

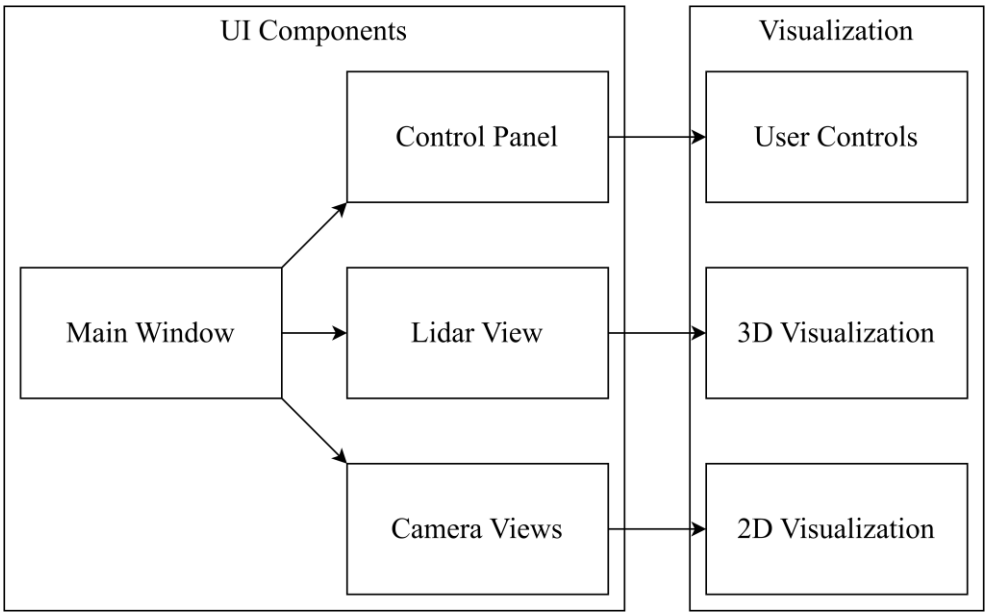


Figure 3.6 UI Components and Visualization

The **Visualization Layer** ensures synchronization between 2D and 3D visualizations. **User controls** affect both views, allowing flexible interaction with the data. **Export operations** handle outputs from both views, enabling users to save and share visualization results.

3.3.5. Statistical Analysis and Graphical Representation for Evaluation

Class Distribution Statistics

The primary purpose of class-based statistics is to evaluate the model's effectiveness in detecting individual object categories (e.g., cars, pedestrians, trucks, motorcycles). Performance analysis by class enables identification of trends or biases within the model. This analysis is crucial for detecting classes with low detection rates, which can guide improvements through data augmentation, additional training samples, or adjustment of training weights. Furthermore, comparing predicted class distributions with ground truth

distributions in the dataset aids in assessing model balance, informing adjustments to better meet practical application requirements.

Distance-based Performance Statistics

The distance between sensors and objects significantly influences detection performance. Evaluating performance across distance intervals (e.g., 0-10 m, 10-20 m, 30-40 m, 40-50 m, >50 m) allows assessment of model robustness under varying real-world conditions.

Key metrics include:

- **Number of Detections:** Indicates detection density at each distance, helping to identify optimal operating ranges and areas needing improvement.
- **Average Confidence Score:** Reflects the mean confidence of detections per distance interval; higher values suggest greater model certainty within that range.
- **Median Confidence Score:** Represents typical confidence levels per distance, less affected by outliers, useful for assessing model reliability.
- **Standard Deviation:** Measures variability of confidence scores; lower values indicate stable model performance at the given distance.

Threshold-based Statistics

Confidence threshold-based statistics evaluate model performance under different confidence score cutoffs, directly impacting the trade-off between precision and recall. Important indicators include:

- **Detection Rate > 0.3:** Reflects the proportion of detections with moderate confidence, representing overall detection capability.
- **Detection Rate > 0.7:** Represents the proportion of high-confidence detections, serving as a basis for assessing model precision and stability under stricter conditions.

This analysis facilitates the evaluation of the model's capabilities across varying confidence levels, serving two primary objectives: improving the model by identifying the optimal operating threshold and adjusting confidence thresholds to meet the specific

requirements of different applications (e.g., prioritizing high precision in autonomous driving systems or high recall in early warning systems).

The aforementioned statistics provide a multidimensional perspective on model performance, supporting several goals:

- **Model Improvement:** Based on class-wise and distance-based statistics, weaknesses can be identified, guiding targeted enhancements in architecture, training, or data preprocessing.
- **Threshold Adjustment:** Threshold-based statistics assist in setting confidence levels appropriate for particular use cases, such as prioritizing precision in safety monitoring systems or recall in support systems.
- **Performance Comparison:** These statistics enable comparisons between different model versions or multi-sensor fusion strategies.
- **System Optimization:** By leveraging optimal distance ranges or confidence thresholds, the system can be configured for more effective operation under real-world conditions.

3.4. Key Features and Functions of Visualization and Analysis System

The nuScenes data visualization system is developed to comprehensively support the processes of testing, evaluation, and result presentation on the nuScenes dataset. The system's key features are designed with a focus on usability, efficiency, and scalability, addressing the needs of both researchers and engineers involved in autonomous system development.

3.4.1. User-Friendly Interaction

The system interface is designed to be user-friendly and easy to operate. Upon launch, users can input or select the paths to the data directory, result files, and output folder through intuitive dialog boxes. All interface operations provide clear feedback, enabling users to effectively control the entire visualization and analysis workflow. The startup interface, as illustrated in Figure 3.7, guides users through the initial setup process.

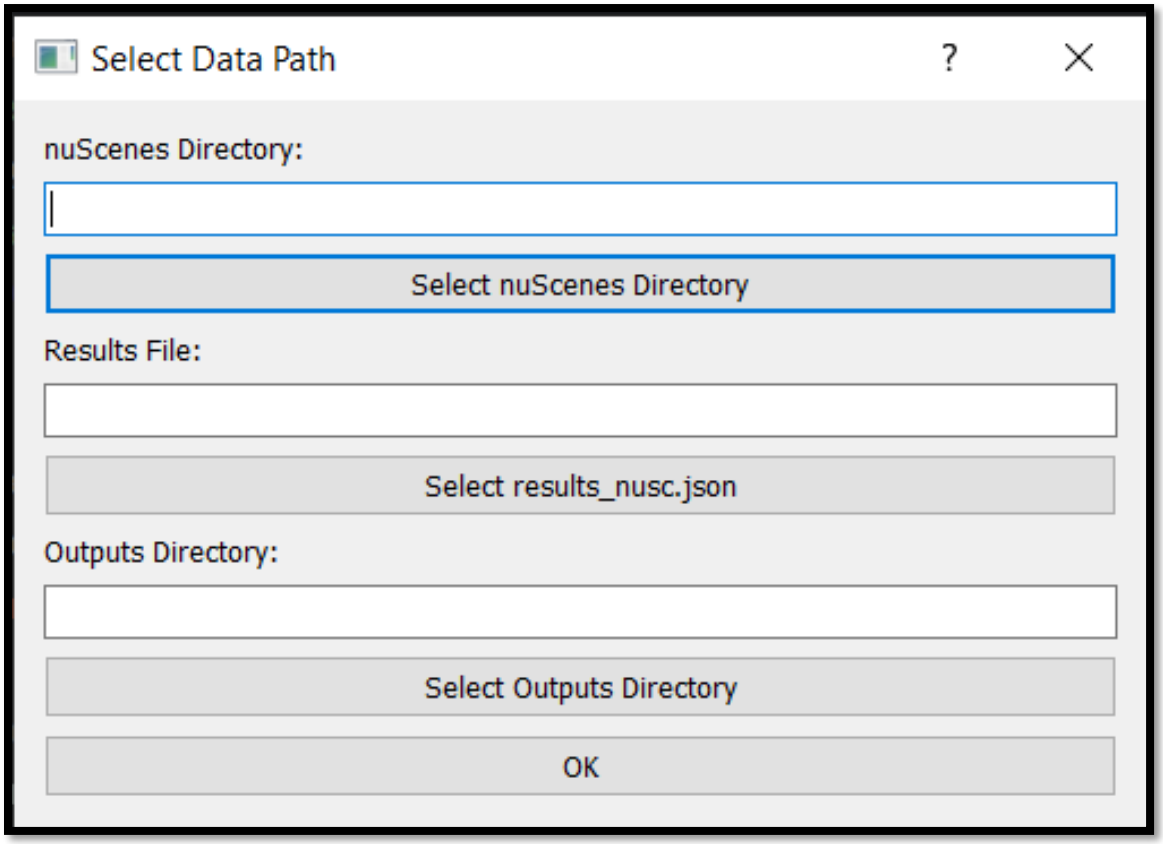


Figure 3.7 Startup Interface

User Instructions:

Upon launching the application, a dialog box appears prompting the user to input or select the paths to the nuScenes dataset directory, the results file (results_nusc.json), and the outputs folder. Once completed, the main interface is displayed, allowing users to utilize the features previously described.

3.4.2. Multi-Camera Visualization

A key feature of the system is its ability to simultaneously visualize six camera views, including CAM_FRONT, CAM_FRONT_LEFT, CAM_FRONT_RIGHT, CAM_BACK, CAM_BACK_LEFT, and CAM_BACK_RIGHT. Each camera frame is displayed within the main interface, enabling users to observe a comprehensive view of the vehicle's surrounding environment. The system supports interactive operations such as zooming, panning, and resetting the view, facilitating detailed inspection of detected objects within each frame. This functionality is particularly valuable for evaluating the prediction quality

of object detection models in complex real-world environments. The camera view region, as depicted in Figure 3.8, illustrates the layout and interactive capabilities of the multi-camera display.

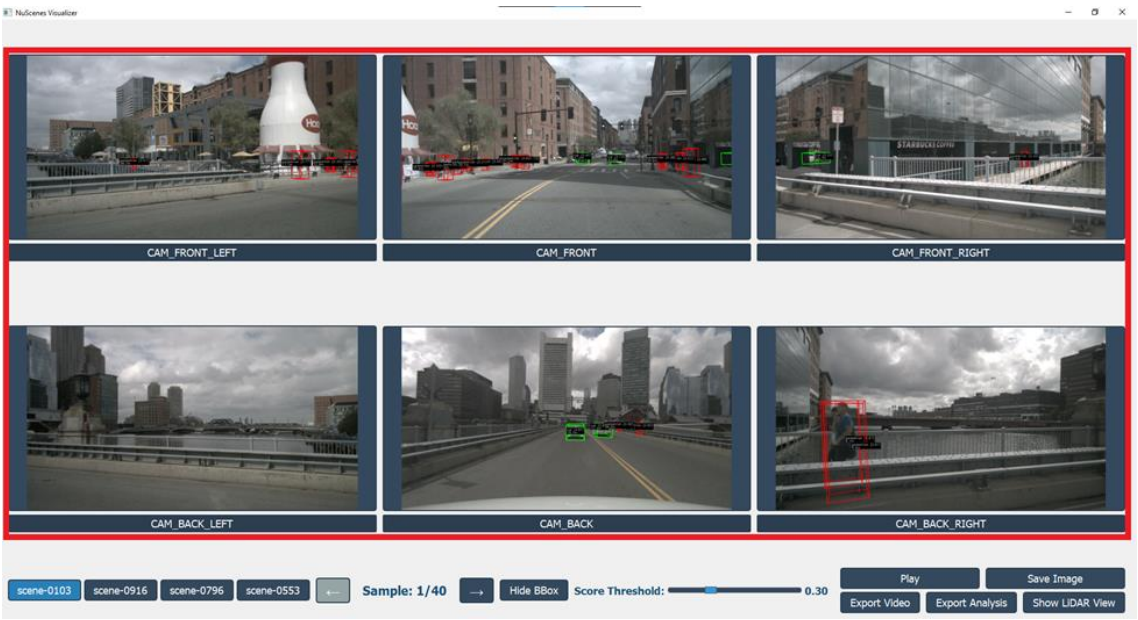


Figure 3.8 Camera View Region

User Instructions:

Users can hover the mouse over a camera frame and use the scroll wheel to zoom in or out of the image. When zoomed in, holding the left mouse button and dragging allows panning across the view. Right-clicking on the frame resets it to the original state. Switching between scenes and samples automatically updates the images for all cameras.

3.4.3. 3D LiDAR Visualization

In addition to the camera views, the system provides a 3D LiDAR data visualization function for each sample. The LiDAR window displays the point cloud, predicted bounding boxes, and ground truth annotations within a three-dimensional space, supporting interactive operations such as rotation, zooming, scaling, and changing display modes. Users can utilize keyboard shortcuts to quickly adjust display attributes, facilitating efficient inspection, comparison, and analysis of model prediction results on 3D spatial data. In the visualization, red bounding boxes represent model predictions, green bounding boxes indicate ground truth annotations, and gray points correspond to the LiDAR point

cloud. The visualization of point cloud data and detected objects within the LiDAR view window, as implemented in the system, is depicted in Figure 3.9.

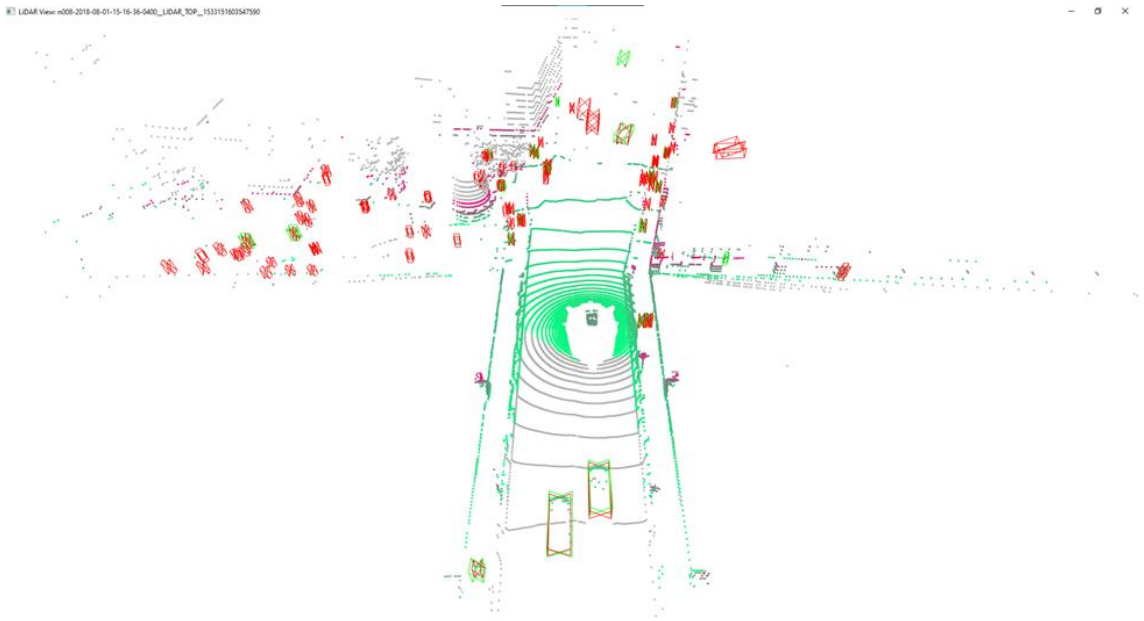


Figure 3.9 LiDAR View Window

User Instruction:

To open the LiDAR window, users click the “Show LiDAR View” button on the main interface. Within the LiDAR window, the mouse is used to rotate, zoom, or pan the 3D view. Keyboard shortcuts such as R (reset view), A (toggle coordinate axes), P (adjust point size), and L (toggle lighting) enable quick adjustments of display modes. Pressing Q or closing the window exits the LiDAR view mode.

3.4.4. Scene and Sample Navigation

The system supports users in selecting scenes and navigating between samples (frames) within each scene quickly and conveniently. Navigation buttons are intuitively arranged on the interface, letting sequential frame switching while displaying the current position within the entire scene. This feature facilitates users in tracking the temporal evolution of objects and verifying the consistency of predictions across continuous data sequences. The multi-camera visualization interface with scene navigation controls and detection bounding boxes, allowing for comprehensive monitoring of the vehicle's surroundings, is presented in Figure 3.10.

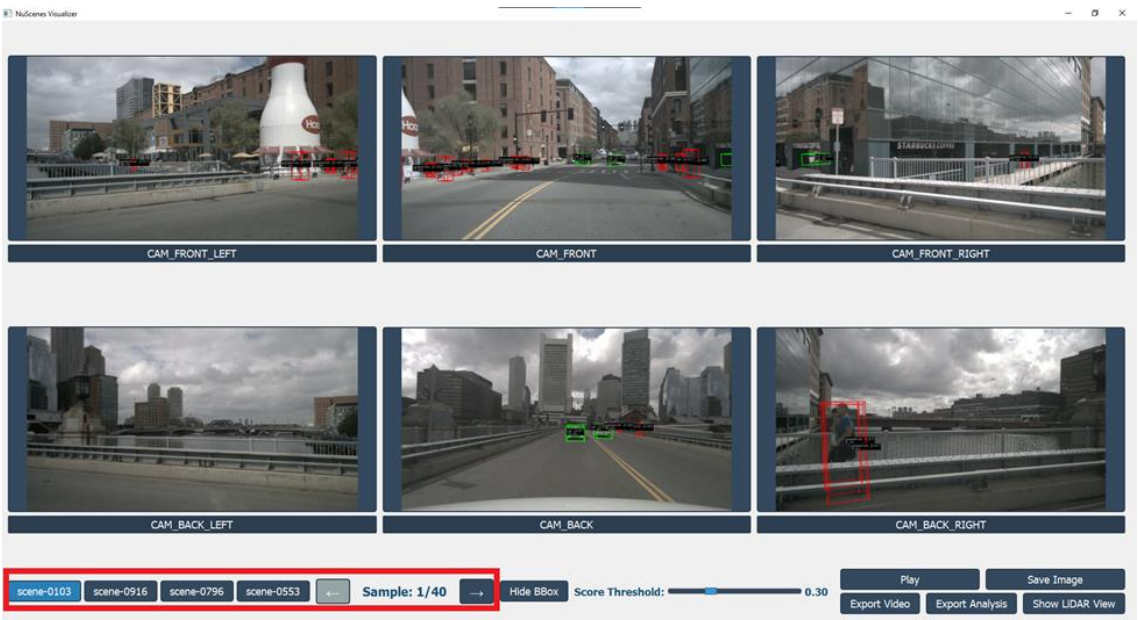


Figure 3.10 Sample Navigation Area

User Instruction: Users select the desired scene by clicking on the scene name buttons located at the top of the interface.

To switch between samples, the ← (previous) and → (next) arrow buttons are used. The label "Sample: x/y" displays the current sample's position within the scene.

3.4.5. Bounding Box Display Customization

To enhance flexibility during the evaluation process, the system allows users to customize the display of bounding boxes on camera images, as shown in Figure 3.11. Users can easily toggle the bounding box overlay on or off with a single action and adjust the confidence threshold via a slider. This functionality helps filter out low-confidence predictions, focusing analysis on significant objects, thereby improving the effectiveness of testing and result interpretation.

User Instruction:

Click the "**Hide BBox**" button to temporarily remove all bounding boxes from the camera views or click the "**Show BBox**" button to display them again. To filter detections based on confidence, use the "**Score Threshold**" slider: adjusting this slider sets the minimum confidence score required for a bounding box to be shown, with a range from 0.0 to 1.0. Any bounding boxes with scores below the selected threshold will be hidden

from view. The current threshold value is displayed next to the slider, providing immediate feedback as you make adjustments.

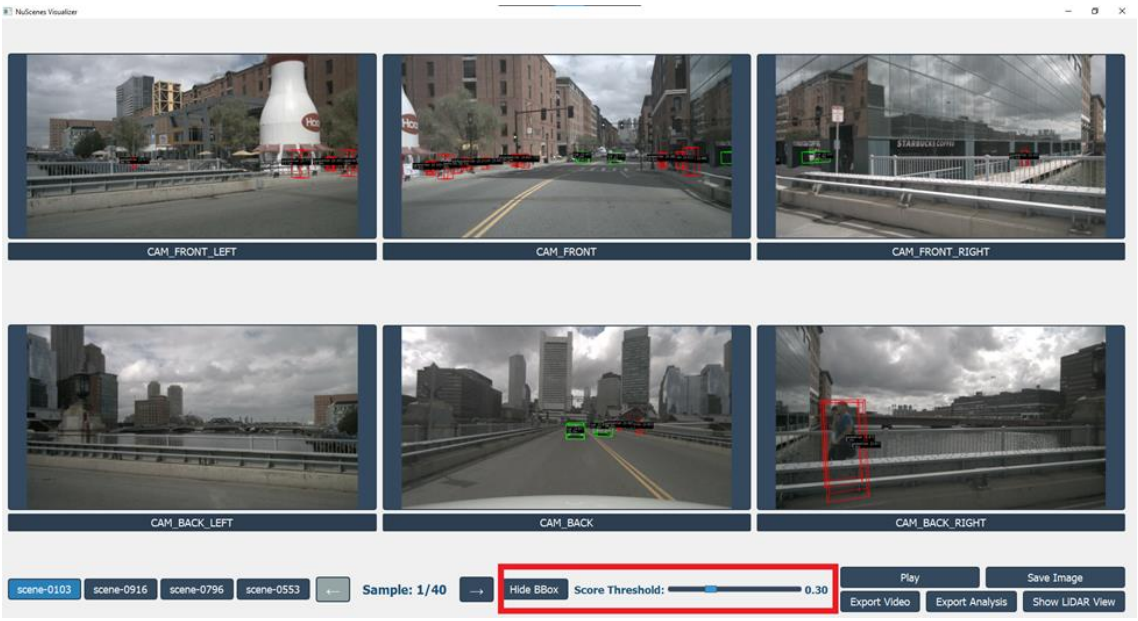


Figure 3.11 Bounding Box Display Function Buttons

Bounding Box Visualization Details

The system displays prediction data through bounding boxes with the following characteristics:

Color Coding by Object Category:

- Pedestrian: Red (255, 0, 0)
- Car: Green (0, 255, 0)
- Truck: Blue (0, 0, 255)
- Bus: Yellow (255, 255, 0)
- Motorcycle: Purple (128, 0, 128)
- Bicycle: Orange (255, 165, 0)
- Construction vehicle: Pink (255, 192, 203)
- Traffic cone: Cyan (0, 255, 255)
- Barrier: Brown (165, 42, 42)
- Trailer: Gray (128, 128, 128)
- Other: White (255, 255, 255)

Information Display:

The system presents prediction information through a carefully designed two-line label system that provides essential data for each detected object. The first line displays the object's category name followed by its confidence score, formatted as “category_name (score)”.

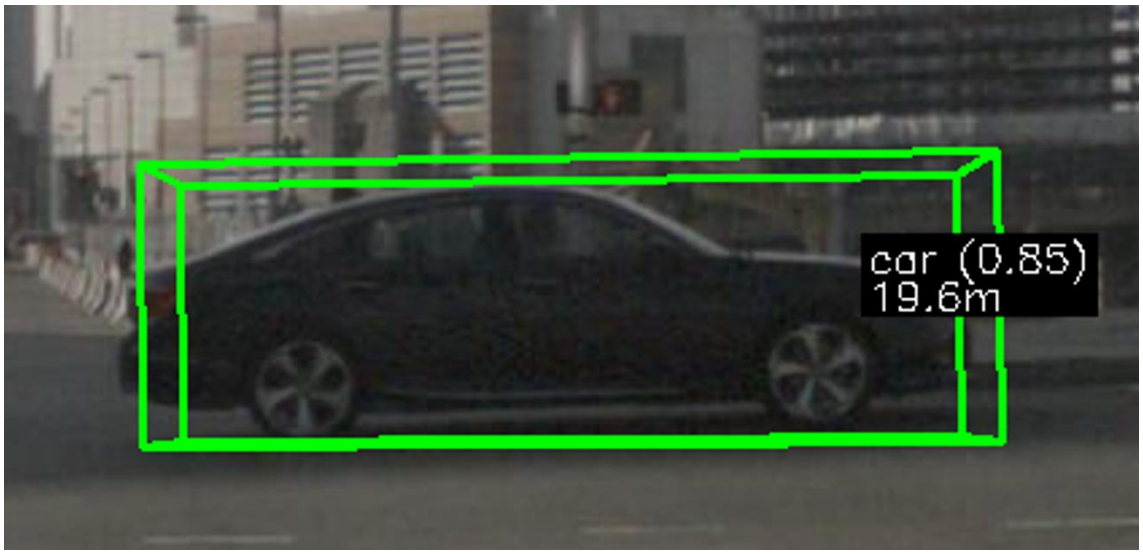


Figure 3.12 Detailed information of a bounding box

For example, in Figure 3.12, “car (0.85)”. This immediate visual feedback helps users quickly assess both the type of object and the model's confidence in its detection. The second line shows the calculated distance from the camera to the object's center in meters, such as “19.6m”, providing crucial spatial context for understanding the detection's location in the scene. To ensure optimal readability in various lighting conditions and backgrounds, the text is displayed on a semi-transparent black background that extends slightly beyond the text boundaries. The label is strategically positioned above the top edge of the bounding box, maintaining a consistent 10-pixel offset to prevent overlapping with the box itself while ensuring the information remains visually connected to its corresponding object.

Bounding Box Structure: the bounding box visualization employs a sophisticated 3D-to-2D projection system to accurately represent the spatial position and orientation of detected objects. Each bounding box is constructed from eight vertices that form a

complete 3D cuboid, representing the object's actual dimensions and orientation in 3D space.

The system uses the camera's intrinsic parameters to project these 3D coordinates onto the 2D image plane, ensuring accurate representation of the object's position and perspective. To maintain visual clarity and relevance, the system implements several filtering mechanisms: boxes with centers behind the camera ($z \leq 0.5$) are automatically excluded to prevent confusing rear projections, and boxes that would appear wider than 80% of the image width are filtered out to avoid overwhelming the display with oversized detections.

The box structure is rendered using a combination of 12 line segments: four lines forming the bottom face, four lines forming the top face, and four vertical lines connecting corresponding vertices between the top and bottom faces. This complete wireframe representation allows users to better understand the object's 3D orientation and spatial relationship with other elements in the scene, while the consistent 2-pixel line thickness ensures visibility without obscuring the underlying image details.

3.4.6. Camera Image and Video Export

The system provides functionality for exporting images and videos from the cameras, supporting reporting, presentation, or offline analysis. Users can save all images from the six cameras of the current sample, including bounding box overlays, or export videos compiling frames from the front camera (CAM_FRONT) with predicted bounding boxes for the entire scene. This feature is particularly useful for archiving case studies, illustrating results in scientific publications, or presenting findings at technical conferences. The user interface elements that enable the export of images and videos from the visualization system are highlighted in Figure 3.13.

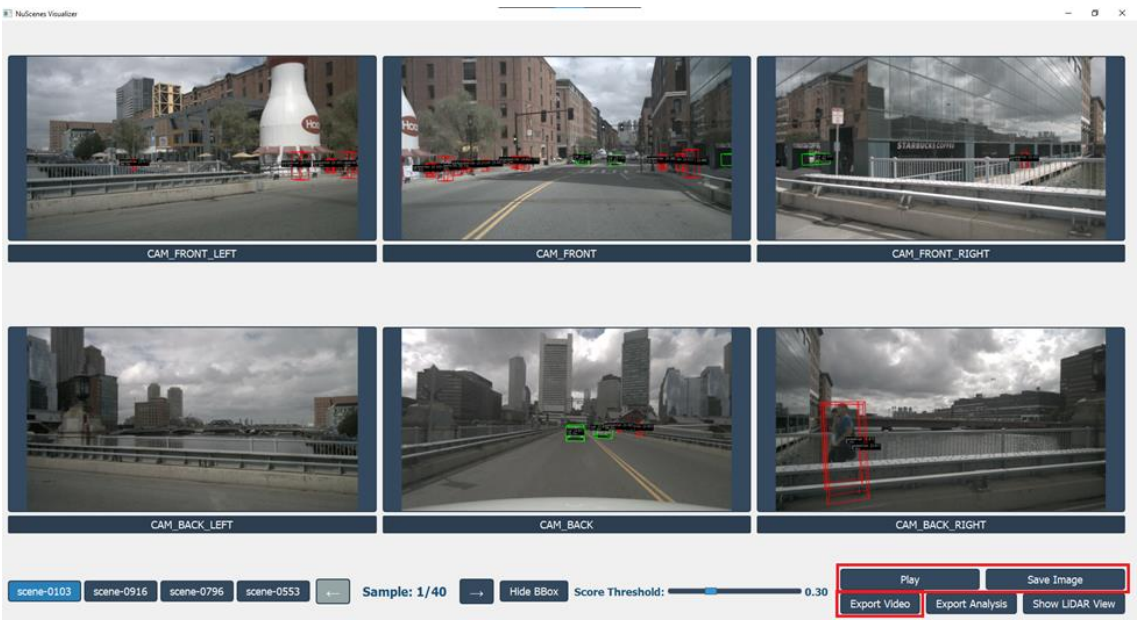


Figure 3.13 Image and Video Export Buttons

User Instruction:

To export images, click the "**Save Image**" button and select a destination folder in the dialog box. The system will then save images from all cameras to the chosen folder, each named by camera.

To export a video, click the "**Export Video**" button and specify the output file location and name. The system will generate a video from the current scene's CAM_FRONT frames with predicted bounding boxes and save it to the selected location.

3.4.7. Statistical Analysis and Result Export

Beyond visualization functions, the system integrates a quantitative analysis module that enables performance statistics based on distance ranges, class distributions, and exports results in tabular or graphical formats. Users can easily perform these analyses and save the results as CSV files or images, facilitating quantitative evaluation and comparison of different models. The functionality of the statistical analysis module and the process for exporting graphical results from the visualization system are illustrated in Figure 3.14.

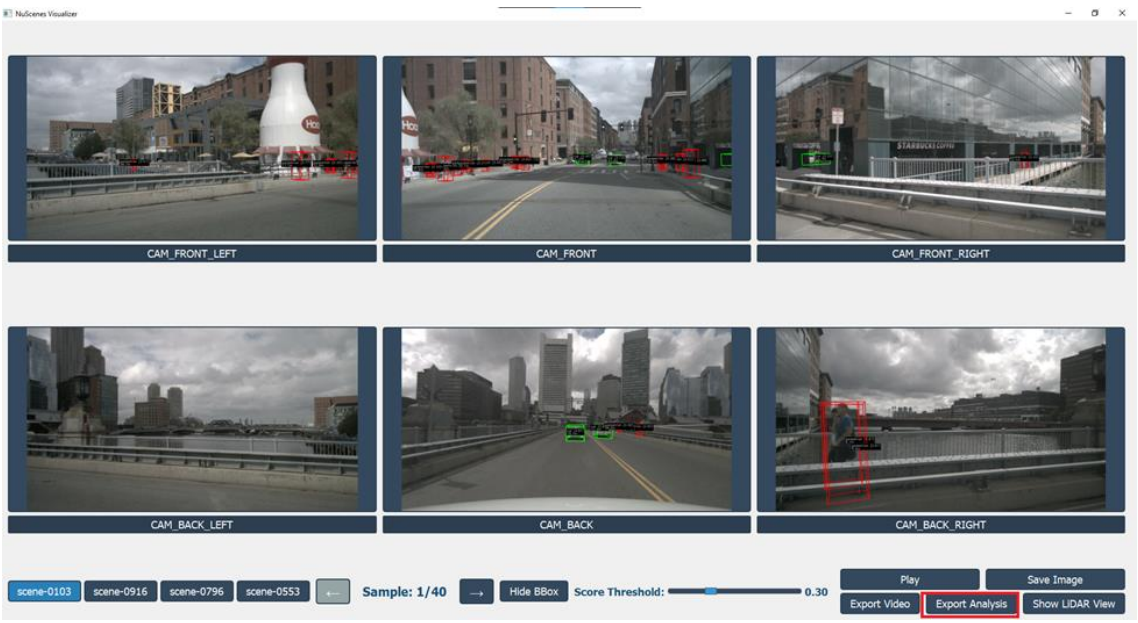


Figure 3.14 Statistical Analysis and Graph Export Function

User Instruction:

Click the "**Export Analysis**" button on the interface. The system will prompt the user to select a folder for saving the analysis results. Upon confirmation, the statistical files (CSV and chart images) will be automatically stored in the selected folder.

CHAPTER 4: RESULTS AND EVALUATION

4.1. Environment

In the project developing a 3D object detection and visualization system, two primary environments were utilized to ensure efficient deployment. The first is a VPS provided by Lightning.ai, selected due to its student support policy. The second is a personal computer, used for visualization and result analysis.

VPS Configuration:

The VPS plays a critical role in computationally intensive tasks such as model training and inference. Its configuration includes an NVIDIA Tesla T4 GPU with 16 GB of VRAM, CUDA version 11.1.1, 33 GB of RAM, and up to 300 GB of storage. The MMDetection3D framework is installed on the VPS, integrated with the MeFormer model from publicly available source code, supporting 3D object detection. The VPS handles model training, inference, and stores prediction results.

Personal Computer Configuration:

The personal computer is employed for visualizing prediction results, such as displaying 3D bounding boxes on images and point clouds and analyzing system performance. Its specifications include an NVIDIA GTX 1050 GPU, 16 GB of RAM, and 500 GB of storage. Key libraries installed on this machine include PyQt5 5.15.0 (user interface), Open3D 0.17.0 (3D point cloud visualization), OpenCV 4.7.0 (image processing), and NumPy 1.23.0 (numerical computation).

4.2. Dataset

4.2.1. Overview of the nuScenes Mini Dataset

The dataset employed in this project is the **nuScenes mini**, a condensed version of the full nuScenes dataset, designed to support research projects with limited computational resources. This dataset comprises **10 scenes**, each lasting **20 seconds**, capturing complex urban traffic scenarios. The data is collected from a diverse sensor suite, including:

- **6 cameras:** front, front-left, front-right, rear, rear-left, and rear-right views.
- **1 LiDAR sensor:** a 32-channel device.

- **5 radars:** providing supplementary information.

However, in this project, the **MEFormer** model utilizes only **camera and LiDAR** data for 3D object detection and prediction. Keyframes (including images, LiDAR, and radar data) are sampled at a frequency of 2 Hz, corresponding to **40 keyframes per scene** (calculated as 20 seconds per scene × 2 samples per second = 40 keyframes).

4.2.2. Scenes Used for Evaluation and Prediction

Among the 10 scenes in the nuScenes mini dataset, this project focuses on evaluation and prediction using **4 specific** scenes for valuating:

- **Scene 0103**
- **Scene 0916**
- **Scene 0796**
- **Scene 0553**

The selection of these four scenes ensures diversity in traffic scenarios, thereby enabling the assessment of the model's performance under varied real-world conditions.

4.2.3. Severe Class Imbalance

The nuScenes mini dataset annotates **23 object classes** but suffers from a severe **class imbalance problem**. The distribution of object classes is highly uneven, with the ratio between the rarest and the most common classes reaching up to **1:10,000**. This issue is known as the **"long tail problem"**, characterized as follows:

- **Common classes:** Objects such as cars, trucks, and pedestrians appear frequently. On average, each keyframe contains approximately **20 vehicles and 7 pedestrians**.
- **Rare classes:** Objects such as bus stops, construction vehicles, and trailers occur very infrequently, sometimes almost negligible in certain scenes.

4.3. Evaluation Approach

Impact of Dataset on Model Evaluation

Class imbalance introduces significant challenges in evaluating model performance, specifically:

- mAP dominated by common classes:

- When calculating mAP across all 23 classes, common classes like cars and pedestrians dominate the overall results.

- This can lead to a situation where the model achieves a high mAP but fails to detect rare classes effectively. Consequently, the overall evaluation does not accurately reflect the model's true capability in detecting infrequent object classes.

• **Poor evaluation of rare classes:**

- Rare classes have too few samples for reliable evaluation.

- **mAP** or **recall** metrics for rare classes often exhibit high variance and instability.

- In some cases, no **ground truth** instances of rare classes appear in the test batch, making it impossible to compute evaluation metrics for those classes.

As a result, **per-class Average Precision (AP)** for rare classes tends to have very high noise, reducing the reliability of the evaluation.

To address the issue of class imbalance and ensure a more comprehensive assessment of model performance, the project adopts three primary evaluation metrics, along with proposed performance thresholds when evaluating on the dataset: **mAP>53%, mATE ≤ 0.49 meters, mAOE ≤ 0.38 radians**.

Furthermore, employing **per-class evaluation** provides a more granular perspective on the model's performance across individual object classes, particularly the rare classes. This approach facilitates the clear identification of the model's weaknesses in detecting infrequent classes, thereby guiding future improvements.

4.4. Results

4.4.1. Inference Process

Following the successful setup of the experimental environment, data preparation, and conducting inference experiments with the MEFormer model on the nuScenes mini dataset, the pipeline executed smoothly without any program interruptions. The inference results and 3D point predictions were successfully saved as the file *results_nusc.json* and an output folder containing *.obj* files. The model's performance will be evaluated using standard 3D detection metrics.

Processing Speed:

- 162 samples were processed in approximately 96 seconds (processing speed ~1.7 tasks/second).
- The result formatting process was very fast (~7 seconds for 162 samples).

Table 4.1 Comparison of MEFormer model results: Published vs. Experimental

Metric	**MEFormer (Published, nuScenes full)**	**MEFormer (Experimental, nuScenes mini)**
NDS	0.74	0.72
mAP	0.72	0.74
mATE	0.27	0.27
mASE	0.24	0.31
mAOE	0.30	0.21
mAVE	0.27	0.36
mAAE	0.11	0.33

The comparative results between the MEFormer model on the full nuScenes dataset (as reported) and the experimental results on the nuScenes mini dataset, presented in Table 4.1, demonstrate a strong correlation between the two datasets. Specifically, the nuScenes Detection Score (NDS) of the model reported in the publication is **0.74**, while the experimental result is **0.72**. The mAP values are **0.72** and **0.74**, respectively, both significantly exceeding the minimum required threshold of **53% (mAP > 0.53)**. This confirms that the model not only meets but also surpasses the proposed standards for object detection accuracy.

Regarding the mean Average Translation Error (mATE), both results are very close, with values of **0.27** (reported) and **0.27** (experimental), which are substantially lower than the required threshold of 0.49 meters **(mATE ≤ 0.49 m)**. This indicates that the model has a high capability to localize objects accurately in 3D space, making it suitable for practical applications such as autonomous driving or traffic monitoring.

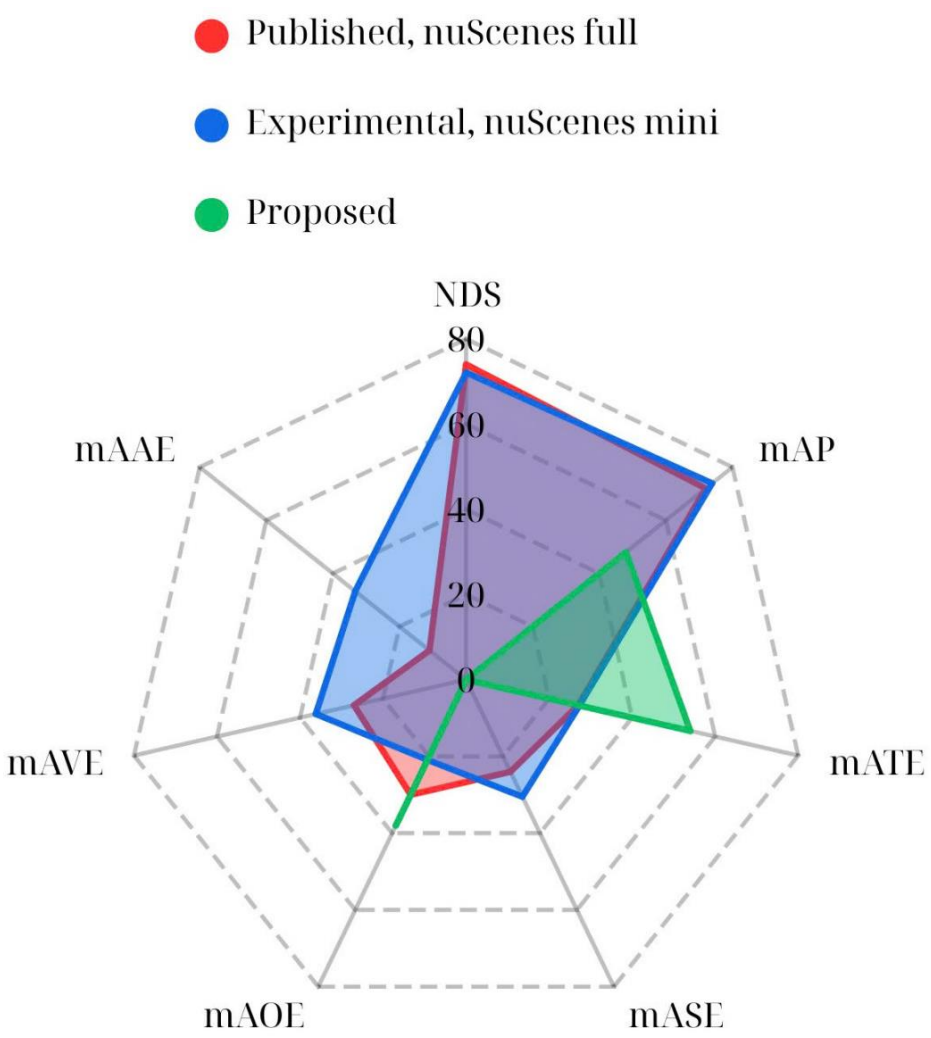


Figure 4.1 Spider chart for comparison of MEFormer model results

The mean Average Orientation Error **(mAOE)** also indicates stable model performance, with values of **0.30** (reported) and **0.21** (experimental), both below the required threshold of **0.38 radians (mAOE ≤ 0.38 rad)**. This demonstrates that the model not only correctly detects objects but also accurately determines their movement direction, a critical factor in traffic safety systems.

Other metrics such as mean Average Scale Error **(mASE)**, mean Average Velocity Error **(mAVE)**, and mean Average Attribute Error **(mAAE)** also show consistency between the two result sets. Although on the mini dataset some metrics, like mASE and mAVE, tend to be higher due to the smaller sample size and reduced diversity, these values remain within acceptable ranges and do not significantly affect the overall model performance.

In summary, the MEFormer model fully satisfies the proposed standards for accuracy and error metrics, demonstrating stability and reproducibility when deployed in practical scenarios. This confirms the model's potential for application in modern 3D detection systems, particularly within intelligent urban traffic environments.

Per-Class Object Evaluation

Table 4.2 Results of per-class object evaluation

Object Class	AP	ATE	ASE	AOE	AVE	AAE
car	0.867	0.167	0.137	0.057	0.186	0.116
truck	0.589	0.188	0.162	0.023	0.125	0.268
bus	0.865	0.236	0.126	0.023	0.687	0.137
trailer	0.989	0.227	0.264	0.010	0.011	1.000
construction_vehicle	0.000	1.000	1.000	1.000	1.000	1.000
pedestrian	0.935	0.143	0.237	0.200	0.199	0.085
motorcycle	0.825	0.251	0.269	0.392	0.091	0.055
bicycle	0.644	0.194	0.346	0.146	0.600	0.001
traffic_cone	0.811	0.054	0.315	nan	nan	nan
barrier	0.862	0.266	0.189	0.028	nan	nan

When analyzing the results by individual object classes, it is evident that the performance of the MEFormer model varies significantly across different object categories within the nuScenes mini dataset, as shown in Table 4.2. For common vehicle classes such as **car, bus**, and **truck**, the model achieves very high Average Precision (AP) scores of 0.867, 0.865, and 0.589, respectively. Notably, the **car** and **bus** classes not only exhibit high AP but also demonstrate very low errors in translation (ATE), scale (ASE), and orientation (AOE), indicating that the model detects and localizes these vehicles with high accuracy and stability. For example, for the **car** class, the translation error is only 0.167 and the orientation error is 0.057, reflecting excellent capability in localizing and determining the movement direction of vehicles.

For classes such as **trailer** and **barrier**, the model also shows outstanding performance with AP scores of 0.989 and 0.862, respectively. However, it is important to note that

although the **trailer** class achieves near-perfect AP, the scale error (ASE = 0.264) and attribute error (AAE = 1.000) are relatively high, suggesting that while the model accurately detects the trailer's position, it struggles to precisely estimate its size or attributes. For the **barrier** class, error metrics remain low except for some NaN values where metrics are not applicable.

The **pedestrian** class also attains a very high AP of 0.935, with low errors in translation, scale, and orientation, demonstrating the model's strong ability to detect and localize pedestrians. This is particularly critical for applications in traffic safety and autonomous driving.

Nevertheless, certain object classes remain challenging for the model. Most notably, the **construction_vehicle** class yields the worst results across all metrics (AP = 0, all errors = 1.000), indicating that the model fails to detect any instances of this class in the test dataset. This may be due to the extremely limited number of samples or the highly diverse morphology of these objects compared to other classes.

Classes such as **motorcycle** and **bicycle** achieve AP scores of 0.825 and 0.644, respectively, indicating moderate detection performance. However, their scale and orientation errors are higher compared to larger vehicle classes. This reflects the practical difficulty in accurately detecting and localizing small, variably shaped objects like motorcycles and bicycles within complex 3D environments.

For small static objects such as **traffic cones**, the model achieves an AP of 0.811 and a very low Average Translation Error (ATE = 0.054), indicating a strong capability to detect small, stationary objects. However, velocity and attribute metrics are not applicable to this class, resulting in NaN values.

In summary, the MEFormer model demonstrates outstanding performance for common object classes with stable shapes and sizes, such as cars, buses, barriers, and pedestrians. Nevertheless, the model still faces challenges with less frequent or morphologically diverse classes like construction vehicles, motorcycles, and bicycles. These results suggest that to improve overall effectiveness, additional training data for difficult classes should be

incorporated, alongside research into methods that enhance detection of small and morphologically diverse objects in 3D environments.

4.4.2. Experimental Results of Using the Visualization Application

Experimental Procedure:

We employed the nuScenes mini dataset for our experiments, utilizing the following output files post-evaluation:

- *results_nusc.json*: This file contains the model's prediction information, including translation, size, rotation, detection_name and detection_score for each object in every sample.
- *.obj files*: These store the LiDAR point cloud data, the 3D ground truth bounding boxes, and predicted bounding boxes for each frame, facilitating 3D spatial visualization.

The application usage workflow consisted of the following steps:

- Application Initialization: Users input or select the path to the nuScenes dataset folder, the results_nusc.json file, and the directory containing the .obj files.
- Data Loading and Synchronization: The application automatically loads the data, maps samples, scenes, and camera views, and synchronizes prediction information with the original dataset.
- Visualization and Analysis: Users interact with the interface to inspect and evaluate prediction results in both 2D (camera images) and 3D (LiDAR point cloud) spaces.

Evaluation of the 3D LiDAR Visualization Functionality:

Since the model integrates both camera and LiDAR data sources for object detection, the prediction results are promising, as clearly illustrated in Figure 4.2. Red bounding boxes represent predictions, blue bounding boxes denote ground truth, and gray points correspond to the LiDAR point cloud.

The results indicate that the predicted bounding boxes for large objects such as cars and trucks exhibit a high degree of correspondence with the ground truth in terms of position, size, and orientation. This demonstrates that the model effectively leverages spatial information from LiDAR and semantic information from the camera to enhance detection

accuracy. However, for smaller objects such as motorcycles or pedestrians, some predicted bounding boxes show deviations in position, incorrect sizing, or are even missed entirely, particularly when these objects are located at the edges of the observation area or occluded by larger objects.

Figure 4.2 LiDAR view window of sample 34 from scene-0103

Overall Evaluation: The system's 3D LiDAR visualization effectively supports testing and evaluation of the camera-LiDAR fusion model. By displaying predictions, ground truth, and point cloud data together, users can readily assess detection accuracy and spot calibration or synchronization issues. Results indicate strong performance for large, nearby objects and reasonable accuracy even with partial occlusion. However, detecting small, distant, or heavily occluded objects remains challenging, highlighting the need for further advances in sensor fusion and data augmentation.

Evaluation of the Camera Image Visualization Functionality: During the use of the application for visualizing prediction results on camera images, we conducted tests across multiple samples from various scenes within the nuScenes dataset. Below are selected observations and analyses based on representative cases:

a) Urban environment with high traffic density (Scene-0103, Sample 34)

This sample captures a highly congested intersection, as shown in Figure 4.3. The scene includes a dense mixture of vehicles, motorcycles, and pedestrians, presenting a complex scenario for object detection and visualization evaluation.

Figure 4.3 CAM_FRONT frame of sample 34 from scene-0103

The visualization results demonstrate that the predicted bounding boxes are accurately overlaid on the corresponding objects in the camera images when using a detection score

threshold greater than 0.3, as shown in Figure 4.4. This accuracy is particularly evident for vehicles situated at distances between 10 and 40 meters from the sensor, where the model consistently provides reliable detections and localization.

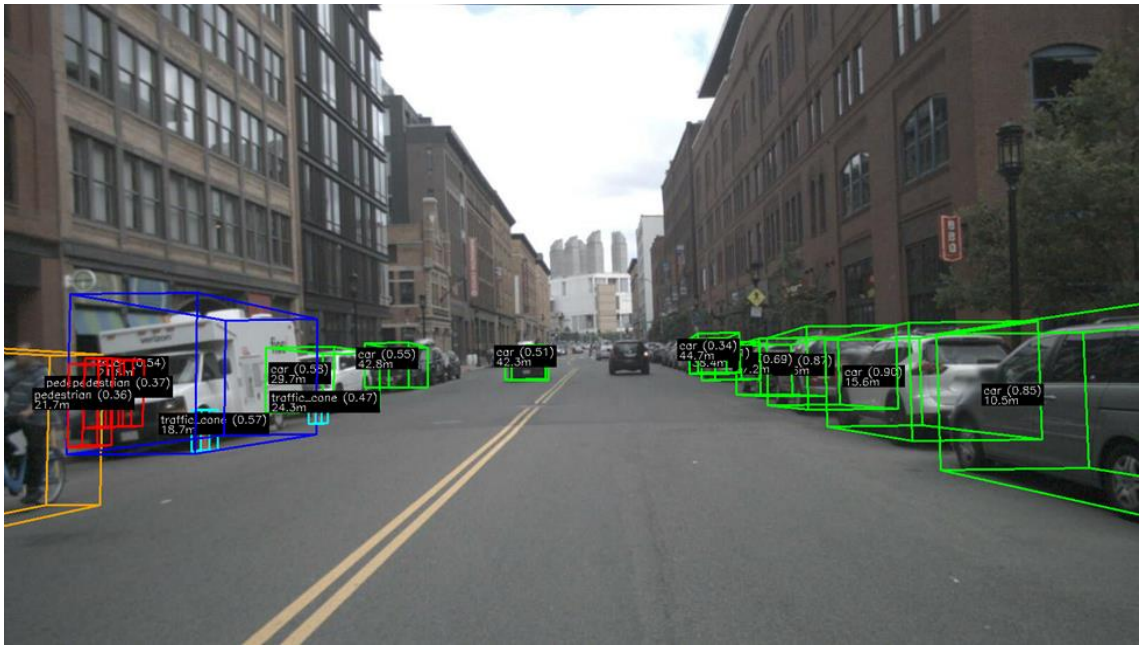


Figure 4.4 CAM_FRONT frame of sample 34 from scene-0103 displaying prediction results with a threshold score > 0.3

However, some small objects, such as motorcycles or pedestrians located at the edges of the frame, were occasionally missed or misclassified. This observation highlights a typical characteristic of the model when operating in complex environments: while it demonstrates strong recognition capabilities for large, nearby objects, it still exhibits limitations in detecting small, distant, or partially occluded objects. These challenges are especially pronounced in crowded urban scenes, where visual clutter and occlusions are common.

b) Low Lighting Conditions (Scene-0916, Sample 8):

In this case, the lighting condition is low, occurring near dusk, as illustrated in Figure 4.5. Under these circumstances, the ambient illumination is significantly reduced, which can impact both the quality of camera images and the reliability of object detection.

Figure 4.5 CAM_FRONT frame of sample 8 from scene-0916

The system maintains a relatively stable capability to overlay bounding boxes on primary objects. However, the confidence scores of the predictions tend to decrease slightly under low lighting conditions, resulting in some bounding boxes being suppressed when the score threshold is increased, as illustrated in Figure 4.6.

Figure 4.6 CAM_FRONT frame of sample 8 from scene-0916 displaying prediction results with a threshold score > 0.3

This indicates that the model and visualization system can effectively reflect the uncertainty of predictions under adverse conditions, thereby enabling users to easily recognize and adjust parameters accordingly.

c) Distant Objects (Scene-0796, Sample 11)

In the frame shown in Figure 4.7, the majority of objects are located beyond 30 meters, and the number of predicted bounding boxes decreases significantly.

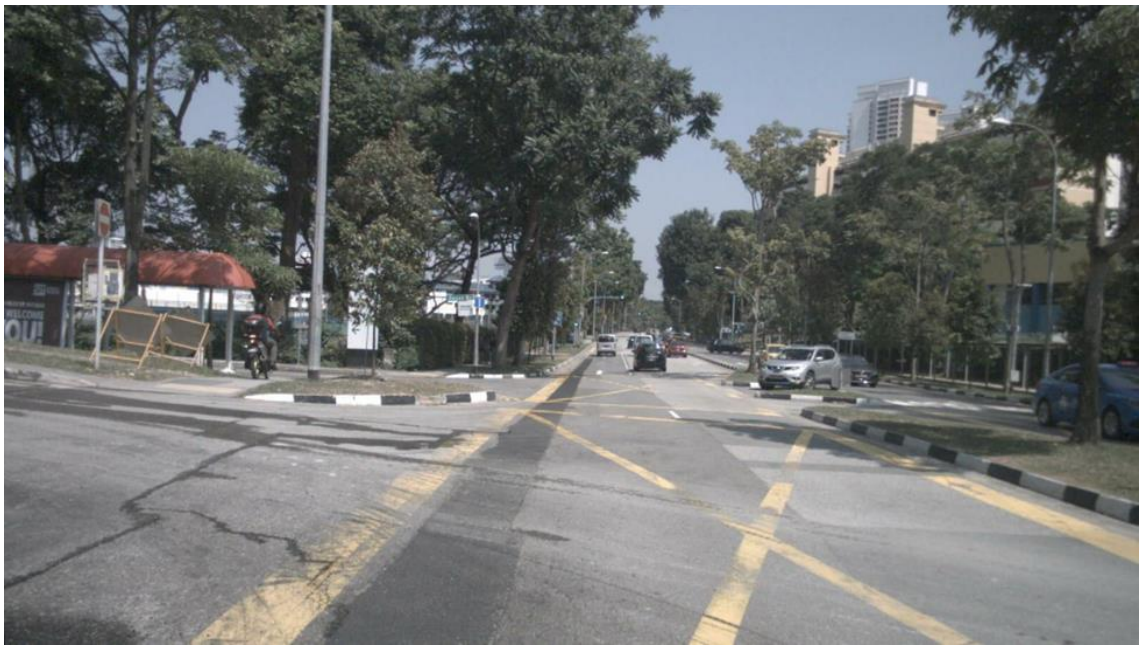


Figure 4.7 CAM_FRONT frame of sample 11 from scene-0796

Large objects such as trucks are still reliably detected by the model at greater distances due to their prominent size and distinct visual features. However, smaller entities particularly compact vehicles, pedestrians, or cyclists are often absent from the detection output, as demonstrated in Figure 4.8.

These objects tend to occupy fewer pixels in the image at long ranges, making them harder to distinguish from background noise or visual clutter. This observation highlights a key weakness of the model: its reduced sensitivity to smaller or less visually prominent targets at long distances, which can pose serious challenges for real-world deployment in complex traffic environments.

Figure 4.8 CAM_FRONT frame of sample 11 from scene-0796 displaying prediction results with a threshold score > 0.3

Overall Evaluation:

- The visualization system enables users to easily inspect and identify cases of incorrect localization, class confusion, or missed detections.
- Features such as zooming, panning, and filtering by score threshold facilitate detailed examination of individual cases, proving particularly useful when analyzing complex samples.
- Overlaying bounding boxes on camera images aids in detecting calibration errors when bounding boxes are misaligned with actual objects, thereby supporting improvements in the data processing pipeline.

Conclusion:

The camera image visualization functionality of the application has demonstrated effectiveness in supporting qualitative testing, evaluation, and error detection of the 3D object detection model on the nuScenes dataset. Analysis of specific samples indicates that the system performs well across diverse environmental conditions while accurately reflecting the model's limitations, laying a foundation for future enhancements.

4.4.3. Evaluation of the Visualization and Analysis System

The visualization and evaluation system for the camera–LiDAR fusion model has demonstrated clear effectiveness in supporting the testing, analysis, and presentation of 3D object detection results on the nuScenes dataset. The integration of multidimensional visualization (2D/3D), flexible interaction capabilities, and quantitative analysis tools enables users to easily identify and assess correct, incorrect, or missed predictions, while also facilitating the detection of calibration issues, data synchronization errors, and model quality concerns.

Advantages:

- Multidimensional Visualization: The system supports the simultaneous visualization of detection results on both two-dimensional (2D) camera images and three-dimensional (3D) LiDAR point clouds. This dual-modality display enables comprehensive visual inspection by allowing users to assess spatial consistency and alignment between predicted outputs and corresponding ground truth annotations across different sensor modalities.
- Flexible Interaction: A range of interactive functionalities is incorporated into the system, including zooming, panning, and filtering based on confidence scores. Users can also switch dynamically between scenes or individual data samples, toggle the visibility of various data layers (e.g., bounding boxes, segmentation masks, point clouds), and export visual content such as annotated images, video sequences, and statistical reports. These capabilities facilitate detailed exploratory testing, result documentation, and streamlined presentation of findings.
- Support for Quantitative Analysis: The system integrates tools for automated chart generation, statistical analysis of object distance and class distributions, and computation of key performance metrics such as mean Average Precision (mAP) and Intersection over Union (IoU). These features enable rigorous, data-driven evaluation of model performance across different object categories and scenarios.

- Error Detection Capability: Overlaying bounding boxes and direct comparison with ground truth facilitate the identification of calibration errors, data synchronization problems, and erroneous predictions.

Limitations:

- Performance Constraints on Non-GPU Devices: When processing large scenes or multiple consecutive samples, loading and rendering speeds may be slow on personal computers without GPU.
- Challenges with Small, Faraway, or Occluded Objects: Both the model and system exhibit limitations in recognizing small, faraway, or occluded objects, leading to some missed detections or inaccurate bounding box size and position predictions.
- Lack of Support for Additional Sensors: Currently, the system primarily focuses on camera and LiDAR data visualization and does not integrate radar, GPS, or other supplementary sensor data.
- Dependence on Calibration Quality: Visualization results can be distorted if input data suffer from calibration errors, complicating the evaluation process.

Recommendations for Improvement:

- Performance Optimization: Enhance data loading and rendering algorithms, introduce fast preview modes or parallel processing to accelerate performance on non-GPU hardware.
- Sensor Support Expansion: Incorporate visualization capabilities for radar, GPS, and other sensor data to increase system comprehensiveness.

4.4.4. Statistical Analysis and Prediction Results by Distance and Confidence Threshold

A comprehensive analysis of the prediction results reveals that the model generated 29,975 predictions across the entire test dataset.

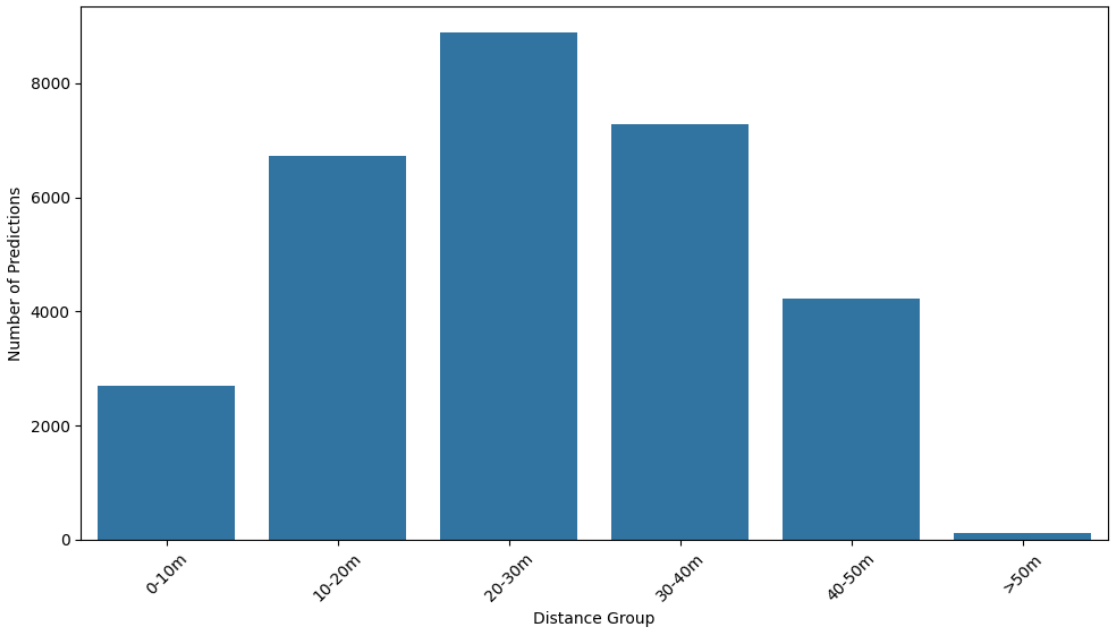


Figure 4.9 Number of Predictions by Distance Group

The distribution analysis by distance reveals that the majority of predictions are concentrated within the range of 10 to 40 meters from the sensor. Among these, the 20-30 meter group accounts for the largest share with 8,896 predictions, followed by the 30-40 meter group (7,284 predictions) and the 10-20 meter group (6,737 predictions). The number of predictions decreases as the distance increases, with only 127 predictions in the beyond 50 meter group. This reflects the practical limitation of the model's ability to detect distant objects, as well as the lower actual occurrence of objects at greater distances. The distribution is visually summarized in Figure 4.9, where a bar chart illustrates the number of predictions by distance group.

A more detailed statistical analysis shows that the average prediction distance is 26.5 meters, with a standard deviation of 11.5 meters. The minimum predicted distance is approximately 3 meters, while the maximum exceeds 51 meters. The 25th, 50th, and 75th percentiles are 17.5 m, 26.1 m, and 35.4 m, respectively, indicating that most predictions are concentrated within the 20-35 meter range, which corresponds to the effective operating zone of both the sensor and the model.

Table 4.3 Prediction Statistics by Distance Group

Distance Range	Number of Predictions	Mean Score	Median Score	Score Std. Dev.	Score > 0.3 (%)	Score > 0.7 (%)
0-10 m	2,694	0.251	0.046	0.352	25.9	22.5
10-20 m	6,737	0.231	0.052	0.323	24.8	18.8
20-30 m	8,896	0.168	0.050	0.246	18.0	9.0
30-40 m	7,284	0.130	0.051	0.185	13.3	3.0
40-50 m	4,237	0.092	0.045	0.132	7.4	0.6
>50 m	127	0.081	0.040	0.129	6.3	1.6

When examining the prediction quality across different distance groups, a clear trend is observed: **the prediction quality progressively decreases as the distance increases.**

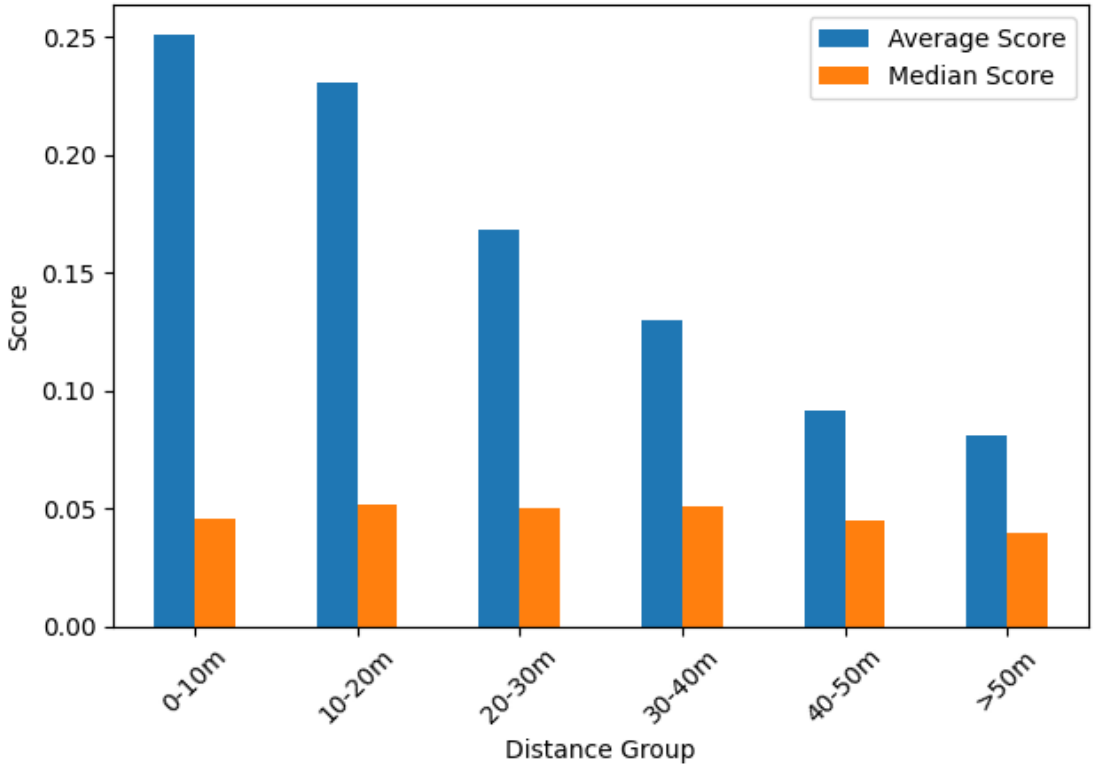


Figure 4.10 Average and Median Scores by Distance Group

In the closest distance group (0-10 m), the average prediction score is 0.251, with 25.9% of predictions having scores greater than 0.3 and 22.5% exceeding 0.7. However, in the

farthest group (>50 m), the average score decreases to 0.081, with only 6.3% of predictions scoring above 0.3 and 1.6% above 0.7. This reflects the practical reality that the model is more confident in predictions made near the sensor, whereas predictions at greater distances generally exhibit lower confidence. Additionally, the standard deviation of scores decreases with distance, indicating that the variability in prediction quality at far distances is less pronounced than at closer ranges.

A noteworthy observation is that the **median score** across all groups remains very low (around 0.04-0.05), as illustrated in Table 4.3, indicating that the majority of predictions have low confidence scores, with only a small fraction achieving high scores. This may be attributed to the model generating many low-confidence predictions, particularly at longer distances, or to the characteristics of the mini dataset, which contains a limited number of actual objects.

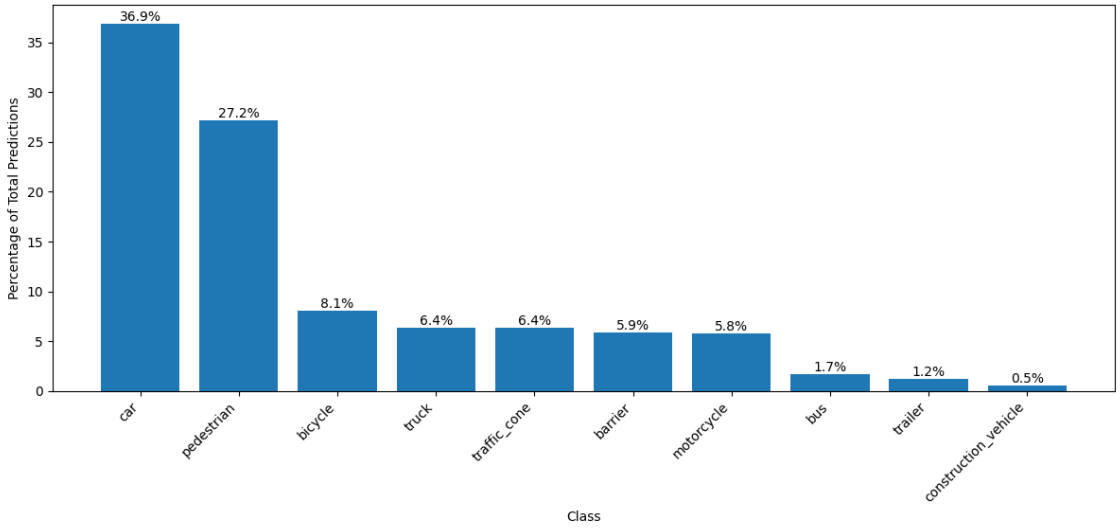


Figure 4.11 Distribution of Predictions by Class

Regarding the distribution of the object classes in the predictions shown in Figure 4.11, the **car** class accounts for the largest proportion at 36.88%, followed by **pedestrian** (27.16%) and **bicycle** (8.08%). Classes such as **truck**, **traffic_cone**, **barrier**, and **motorcycle** each represent approximately 5-6%, while less frequent classes like **bus**, **trailer,** and **construction vehicle** account for less than 2%. This distribution accurately reflects urban traffic realities and indicates that the model tends to generate more

predictions for common classes. This tendency may affect the detection performance of rare classes if appropriate data balancing or threshold adjustments are not applied.

In summary, the above statistics demonstrate that the MEFormer model performs most effectively at close to medium distances, with prediction quality decreasing as objects are located farther from the sensor. The distribution of prediction scores suggests that the model is relatively conservative, exhibiting high confidence only in a small subset of predictions, particularly those near the sensor. The class distribution appears reasonable; however, attention must be paid to rare classes to avoid missing important objects in practical applications. These analyses provide a critical foundation for proposing future improvements, such as augmenting data for rare classes, optimizing thresholds according to distance, or applying post-processing techniques to reduce the number of low-confidence predictions and enhance system reliability.

4.4.5. Analysis on Score Thresholds

Based on detailed statistics of prediction quality across different distance groups, recommendations can be made regarding the appropriate selection of score thresholds for various application purposes:

- **Threshold > 0.3:** This threshold is suitable when the goal is to **maximize detection recall**, accepting a large number of predictions, including those with moderate confidence. This is particularly useful in driver assistance systems, traffic monitoring, or applications requiring early detection without missing objects, even at the expense of some false positives. According to the statistics, within the 0-20 m range, approximately 25% of predictions have scores above 0.3, ensuring that the system does not overlook important nearby objects.
- **Threshold > 0.7:** This threshold is appropriate when the objective is to **optimize precision**, retaining only predictions with very high confidence to minimize false alarms. It is suitable for automated decision-making systems (e.g., automatic emergency braking, autonomous vehicle control), where each decision must rely on the most reliable predictions. However, the proportion of predictions scoring above 0.7 decreases sharply with distance, dropping to 22.5% at 0-10 m and nearly zero beyond

40 m. Therefore, a high threshold should only be applied to nearby objects where the model is most confident.

Relation to "The 3-Second Rule" and Practical Application

"The 3-Second Rule" is a widely accepted traffic safety guideline recommending that drivers maintain a following distance sufficient to reaction for $2.5\ seconds$ at the current speed [30]. At a typical urban speed of approximately $50\ km/h$ (equivalent to $13.9\ m/s$), the minimum safe distance is:

$$Recommended\ distance = 2.5s \times 13.9m/s = 34.75m \quad (4.1)$$

Comparing this with the prediction statistics, it is evident that the model performs most reliably within a range **under 40 meters** (specifically, the 0-10 m, 10-20 m, 20-30 m, and 30-40 m groups), which have both a large number of predictions and significantly better prediction quality compared to farther distances. This range aligns well with the safe following distance suggested by "The 3-Second Rule" in urban traffic conditions.

In conclusion, the MEFormer model fully meets the requirements for detection and recognition of objects within the safe distance range defined by "The 3-Second Rule" at urban speeds (~50 km/h). This ensures that the system can provide early detection of potential hazards ahead, effectively supporting traffic safety applications, autonomous driving, and driver assistance in urban environments.

CHAPTER 5: CONCLUSION AND DEVELOPMENT

5.1. Conclusion

Within the scope of this research, we have successfully implemented the MEFormer model based on the MMDetection3D framework to address the problem of 3D object detection and localization on the nuScenes dataset. Experimental results demonstrate that the model operates stably, with a well-constructed pipeline that ensures scalability and reusability for future research.

Evaluation results on the nuScenes mini dataset show that MEFormer achieves outstanding performance with high mAP and NDS scores, particularly excelling in common object classes such as cars, buses, and pedestrians. Detailed analysis by class and distance groups indicates that the model performs most effectively within a range under 40 meters corresponding to the safe distance defined by the "3-Second Rule" in urban traffic. This confirms that the model is fully suitable for applications in driver assistance systems, traffic monitoring, or autonomous vehicles in real-world environments.

Additionally, statistical analyses reveal that prediction quality decreases as distance increases, and the model faces challenges with rare or morphologically diverse classes such as construction vehicles, motorcycles, and bicycles. These practical issues warrant continued research and improvement.

5.2. Development and Recommendations

Based on the results and in-depth analysis, we propose several directions for future development and improvement of the project as follows:

- **Development of a Hardware Data Collection System and Real-World Testing:**

  - Design and build a hardware system integrating sensors such as LiDAR, cameras, and radar to collect data directly from real-world environments. Proactive data collection enables the model to encounter diverse and specific traffic scenarios characteristic of Vietnam, while also facilitating the creation of a dataset tailored to the application objectives.

- After completing the hardware system, the MEFormer model can be deployed and tested directly on real autonomous vehicle platforms or within urban traffic simulation environments. This process not only evaluates the model's performance under operational conditions but also assesses real-time processing capability, system latency, and adaptability to complex real-world scenarios. This step is crucial for technology transfer from research to practical application and lays the foundation for future research on model optimization, embedded systems, and integrated hardware-software solutions.

- **Data Augmentation for Rare Classes:** Apply data balancing techniques, data augmentation, or transfer learning to improve detection performance for rare, small, or morphologically diverse classes such as construction_vehicles, motorcycles, and bicycles.
- **Application-Specific Threshold Optimization:** Develop dynamic score threshold selection strategies tailored to different distance ranges and application purposes (e.g., early warning, autonomous decision-making) to optimize both precision and recall simultaneously.

REFERENCES

[1] T. Đoàn. "10.944 người chết vì tai nạn giao thông trong năm 2024." Báo Tin Tức. https://baotintuc.vn/xa-hoi/10944-nguoi-chet-vi-tai-nan-giao-thong-trong-nam-2024-20250106103121152.htm (accessed May 14, 2025)).

[2] S. International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806," SAE International, 2018. Accessed: May 14, 2025. [Online]. Available: https://www.sae.org/standards/content/j3016_201806/

[3] A. Magazine, "Waymo to Double the Size of Its Autonomous Vehicle Fleet." [Online]. Available: https://www.assemblymag.com/articles/99248-waymo-to-double-the-size-of-its-autonomous-vehicle-fleet

[4] M. Mercedes-Benz, "Premier constructeur à certifier le système de conduite automatisée de niveau 3 aux Etats-Unis," *Mercedes-Benz Mag,* 2023. [Online]. Available: https://www.mercedes-benz-mag.fr/mobilite/premier-constructeur-a-certifier-le-systeme-de-conduite-automatisee-de-niveau-3-aux-etats-unis/.

[5] X. Li, Y. Xiao, B. Wang, H. Ren, Y. Zhang, and J. Ji, "Automatic targetless LiDAR–camera calibration: a survey," *Artificial Intelligence Review,* vol. 56, no. 9, pp. 9949-9987, 2023.

[6] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025/05/15, 2020. [Online]. Available: https://www.nuscenes.org/publications.

[7] I. Tesla. "Model S Owner's Manual." Tesla. https://www.tesla.com/ownersmanual/models/en_us/GUID-682FF4A7-D083-4C95-925A-5EE3752F4865.html (accessed May 9, 2025).

[8] M. Peng, C. Wang, T. Chen, G. Liu, and X. Fu, "Dual temporal scale convolutional neural network for micro-expression recognition," *Frontiers in psychology,* vol. 8, p. 1745, 2017.

[9] W. Newsroom. "The Seaport: A Boston neighborhood guide." WBUR News. https://www.wbur.org/news/2023/09/01/seaport-boston-massachusetts-locals-field-guide (accessed May 14, 2025).

[10] W. C. contributors, "Map of Queenstown," ed. Wikimedia Commons, 2006.

[11] I. Velodyne LiDAR, "HDL-32E High Definition Real-Time 3D LiDAR Sensor Datasheet," Morgan Hill, CA, 2018. [Online]. Available: https://www.mapix.com/wp-content/uploads/2018/07/97-0038_Rev-M_-HDL-32E_Datasheet_Web.pdf

[12] B. AG. "aCA1600-60gc Product Page." Basler Product Documentation. https://docs.baslerweb.com/aca1600-60gc (accessed May 13, 2025).

[13] C. E. Services. "ARS 408-21 Long Range 77GHz Radar Sensor." Continental Engineering Services. https://conti-engineering.com/components/ars-408 (accessed May 9, 2025).

[14] nuTonomy. "nuImages schema documentation." GitHub. https://github.com/nutonomy/nuscenes-devkit/blob/master/docs/schema_nuimages.md (accessed May 25, 2025).

[15] OpenMMLab. "MMDetection3D v1.0.0rc0 Documentation." OpenMMLab. https://mmdetection3d.readthedocs.io/en/v1.0.0rc0/ (accessed May 13, 2025).
[16] M. Zhang, S. Abdulatif, B. Loesch, M. Altmann, and B. Yang, "Class-Aware PillarMix: Can Mixed Sample Data Augmentation Enhance 3D Object Detection with Radar Point Clouds?," *arXiv preprint arXiv:2503.02687,* 2025.
[17] nuScenes. "nuScenes Object Detection Task." https://www.nuscenes.org/object-detection (accessed May 13, 2025).
[18] Z. Liu, Tang, Haotian, Amini, Alexander, Yang, Xinyu, Mao, Huizi, Rus, Daniela L, Han, Song, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," 2023, vol. 35: IEEE, pp. 2774-2781.
[19] J. Ahmad and A. Del Bue, "mmfusion: Multimodal fusion for 3d objects detection," *arXiv preprint arXiv:2311.04058,* 2023.
[20] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11784-11793.
[21] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," in *Conference on Robot Learning*, 2022: PMLR, pp. 180-191.
[22] Y. Liu, T. Wang, X. Zhang, and J. Sun, "Petr: Position embedding transformation for multi-view 3d object detection," in *European conference on computer vision*, 2022: Springer, pp. 531-548.
[23] X. Bai *et al.*, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1090-1099.
[24] J. Cha, M. Joo, J. Park, S. Lee, I. Kim, and H. J. Kim, "Robust Multimodal 3D Object Detection via Modality-Agnostic Decoding and Proximity-based Modality Ensemble," *arXiv preprint arXiv:2407.19156,* 2024.
[25] J. Yan *et al.*, "Cross modal transformer: Towards fast and robust 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 18268-18278.
[26] H. Hu *et al.*, "Ea-lss: Edge-aware lift-splat-shot framework for 3d bev object detection," *arXiv preprint arXiv:2303.17895,* 2023.
[27] H. Wang *et al.*, "Unitr: A unified and efficient multi-modal transformer for bird's-eye-view representation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 6792-6802.
[28] Y. Xie *et al.*, "Sparsefusion: Fusing multi-modal sparse representations for multi-sensor 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17591-17602.
[29] Z. Li *et al.*, "BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers.(2022)," *URL https://arxiv. org/abs/2203.17270,* 2022.
[30] N. S. Council, *Reference Material for DDC Instructors*, 5th Edition ed. Itasca, IL: National Safety Council, 2005.