3D Object Detection for Autonomous Mobile Robots

Using LiDAR and LiDAR-Camera Fusion

Nordin el Assassi
14595079

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 900
1098 XH Amsterdam

Supervisor
MSc. A. van Schravendijk

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

Semester 2, 2024-2025

Abstract

This thesis investigates the feasibility of 3D object detection on the Avular Origin One, an autonomous mobile robot intended for deployment in a waste-to-energy facility. The objective is to assess whether LiDAR enables reliable truck detection in dynamic industrial scenes and whether fusing LiDAR with a front-facing camera improves detection performance. Development and evaluation were conducted entirely in simulation using the Gazebo environment.

A modular data generation pipeline was developed to extract synthetic sensor data and annotations from simulation and convert them into a custom dataset replicating the nuScenes format. Four detection models were evaluated: three LiDAR-only and one LiDAR–camera fusion model. The models were tested across three evaluation scenarios and nine training configurations, including transfer learning, early fusion, and point cloud filtering.

Results show that LiDAR-only models achieve high detection accuracy, with Centerpoint-Pointpillar offering the best trade-off between speed and detection performance. Fusion using BEVFusion provided no detection improvement and substantially reduced performance due to its computational cost. Transfer learning consistently improved detection outcomes, while early fusion and point cloud filtering showed no clear benefit. These findings highlight LiDAR as the most practical sensing modality for 3D object detection. However, further testing is required to confirm real-time model performance on the physical robot.

Contents

Chapter 1

Introduction

Autonomous mobile robots (AMRs) must be capable of perceiving and interpreting their surroundings to operate safely and effectively in real-world environments. One approach to this problem is real-time 3D object detection, which enables robots to recognise and localise objects in dynamic scenes. This technology is well established in autonomous driving research [26], [1], [21]. However, translating these methods to smaller robots presents new challenges, including limited onboard computational resources, restricted sensor coverage, and suboptimal sensor placement.

In industrial environments, AMRs are increasingly used to monitor activity and assist with local logistics [4]. These settings are often dynamic, with unpredictable truck movements and frequent human presence. Ensuring safety and operational reliability in such environments requires real-time object detection capabilities. One example is the waste-to-energy (WtE) facility operated by Afval Energie Bedrijf (AEB) in Amsterdam, trucks occasionally unload in incorrect zones, leading to inefficiencies that must be resolved manually. An autonomous robot, the Avular Origin One, is being deployed to address this issue by monitoring truck activity and assisting in rule enforcement.

Despite being equipped with several onboard sensors, the Avular Origin One currently supports only basic motion detection within its immediate field of view (FOV). This is insufficient for deployment in dynamic industrial settings where close interaction with vehicles and pedestrians is expected. The robot must instead achieve real-time situational awareness, enabling it to detect trucks, distinguish them from pedestrians, and navigate safely through the facility.

Given the available sensor modalities, this task can be approached using either LiDAR data alone or a multimodal strategy that fuses LiDAR with a front-facing camera. LiDAR provides precise spatial information even in poor lighting conditions, while the camera contributes complementary visual context. Although fusion can improve detection accuracy, it also introduces additional computational demands that must be addressed within the limited hardware available [1].

Due to these hardware constraints, and because collecting real-world data in active industrial environments is both dangerous and time consuming, all development and testing are conducted entirely in simulation. Experiments are carried out using the Gazebo simulator, and synthetic sensor data is captured and converted into a dataset that follows the nuScenes format [5], which is commonly used in autonomous driving research [21].

The need to detect trucks reliably, operate within limited hardware constraints, and develop entirely in simulation leads to the following research question: *How effective is LiDAR for real-time simulated 3D object detection on the Avular Origin One, and how does fusion with a camera affect performance?*

To answer this question, the first step is to design a pipeline for generating and storing synthetic sensor data along with the corresponding object annotations. This leads to the first sub-question: *How can sensor and annotation data effectively be extracted from the simulation environment?*

Once the data pipeline is in place, the next stage involves selecting and implementing a set of object detection models. These must include both LiDAR-based models and models that combine LiDAR with input from a front-facing camera. This raises the second sub-question: *What state of the art object detection models are available for LiDAR-only and LiDAR camera fusion setups?*

Finally, the implemented models must be evaluated in terms of detection performance, inference speed, and computational demands, which prompts the sub-question: *How does the use of mono or stereo camera data affect detection performance and inference speed compared to LiDAR-only input?*

To address the research questions, this thesis begins with a background chapter introducing the robot and industrial context, followed by sections on 3D object detection and fusion. A subsequent chapter describes the simulation design, data generation, datasets, and model evaluation. Results are then presented and assessed. The final chapter summarises the findings and outlines limitations.

Chapter 2

Background

This chapter provides an overview of the core concepts and components relevant to 3D object detection and sensor fusion. It begins by introducing the robot and the industrial environment in which it is deployed, followed by an introduction to object detection, a description of sensing modalities, data representations, and commonly used datasets. Sensor fusion is then discussed, and the chapter concludes with a review of relevant detection models.

2.1 Robot and environment

The Origin One, shown in Figure 2.1, is a research-oriented robotic platform developed by the Dutch company Avular. It is designed for autonomous navigation in both indoor and outdoor environments and supports modular configuration with various sensors and utilities, such as a LiDAR sensor or a robotic arm. The robot was selected by the Maintenance Lab at the Amsterdam University of Applied Sciences (HvA), which was asked by AEB to assist with the development of an autonomous monitoring system at their facility.

Figure 2.1: The Origin One robot developed by Avular [2].

The robot is intended for use at the Hoog Rendement Centrale (HRC), part of the waste to energy facility operated by Afval Energie Bedrijf (AEB) in Amsterdam, as shown in Appendix A. AEB is a municipal waste and energy company based in the city. At the HRC, the robot supports a penalty system by autonomously monitoring truck movements and recording information such as license plates.

2.2 Object detection

Object detection enables AMRs to perceive the environment by localising objects in 3D space and assigning semantic labels [26]. In the context of 3D object detection, objects are represented as 3D bounding boxes, often referred to as ground truth, defined by (x, y, z, w, l, h, ψ). These bounding boxes represent the size and position of objects as oriented cuboids, each labelled with a predefined semantic category, such as trucks or pedestrians. Here, (x, y, z) denotes the centre of the bounding box in the global coordinate frame, (w, l, h) specifies its width, length, and height respectively, and ψ indicates the yaw angle, which defines the orientation of the bounding box relative to the heading of the AMR [21].

In addition to localisation, the task involves semantic classification. This is formulated as a multi-class prediction problem, where the model estimates a probability distribution over predefined categories and selects the most probable class [26].

2.3 Data foundations

2.3.1 Sensor modalities

This section outlines the three sensor modalities commonly employed for object detection with AMRs: LiDAR, RGB cameras, and depth cameras.

LiDAR

LiDAR (Light Detection and Ranging) sensors estimate distance by measuring the time delay between the emission and reception of laser pulses. This time of flight (ToF) principle allows them to generate accurate but sparse 3D point clouds, even under varying lighting conditions, since they emit their own signal. However, performance degrades with highly reflective surfaces and in extreme lighting environments [27], [26], [25].

When operating, a LiDAR generates a point cloud composed of scanned points, each defined by a 3D coordinate (x, y, z) relative to the sensor position and additional attributes R, such as intensity representing the strength of the returned laser signal [27]. While LiDAR can capture geometry over long distances, its effectiveness is reduced at very short or long ranges. Unlike cameras, LiDAR does not capture detailed texture information [1].

Camera

Cameras provide detailed colour and texture information but do not capture depth directly, which limits effectiveness in 3D object detection, particularly under poor lighting conditions [1]. When a scene is recorded, 3D structure is projected onto a 2D image plane using parameters defined by the camera model, resulting in the loss of spatial depth. While single images lack depth, stereo cameras, combining two or more cameras, help estimate it by capturing image pairs from slightly different viewpoints, improving detection accuracy over monocular camera setups. However, they require precise calibration and remain sensitive to lighting conditions and adverse weather [26].

Depth camera

Depth cameras estimate distance using either stereoscopic vision or ToF sensing. When paired with RGB imaging, they produce aligned colour and depth maps (RGB-D), enabling spatial awareness [1]. The depth sensor used on the robot is the Intel RealSense D435f[1], a stereo-based RGB-D camera that estimates depth by triangulating corresponding points between two infrared images. Although RGB-D cameras provide spatial information, they are limited by factors such as reduced depth resolution, sensor noise, and higher computational cost [25].

2.3.2 Data representations

After acquiring data from different modalities, it must be structured into representations suitable for detection. This section focuses mostly on point cloud representations, as image data is already organised in a regular grid and was addressed earlier. Three representation types are examined: voxel-based, pillar-based, and birds-eye view.

[1] `https://www.intelrealsense.com/depth-camera-d435f/`

Voxel-based representation

Voxels are the 3D counterpart of 2D pixels, discretising space into uniformly spaced, fixed-size units, as shown in Figure 2.2. Each voxel stores information such as occupancy and point density. This downsampling reduces computational load and enables the use of 2D convolutional neural networks for detection tasks when further processed [21].

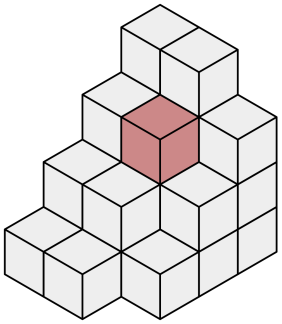


Figure 2.2: Illustration of a voxel grid. A single voxel is highlighted to show the structure [24].

Pillar-based representation

In the pillar-based representation, the point cloud is discretised into vertical pillars based on their horizontal (x, y) position. To enhance these pillars, each point is enriched with features that describe its offset from the pillar centre and local mean within the pillar. This representation enables efficient processing using standard 2D convolution operations and results in inference that is approximately 2-4 times faster than voxel-based methods. However, this scheme introduces some loss of spatial resolution [21], [12].

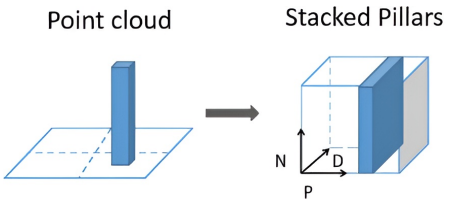


Figure 2.3: Pillar-based representation. LiDAR points are grouped into vertical pillars on a 2D grid [11].

Birds-eye view representation

Birds-eye view (BEV) is a top-down representation that projects 3D sensor data onto the horizontal plane, as illustrated in Figure 2.4. For LiDAR data, this involves discretising the point cloud using voxels or pillars and summarising features along the vertical axis. For camera data, depth must first be estimated to convert image pixels into 3D points, which are then projected onto the same ground plane grid. While this representation reduces vertical resolution, it preserves horizontal spatial structure and enables the use of efficient 2D convolutions for perception tasks [13], [26].

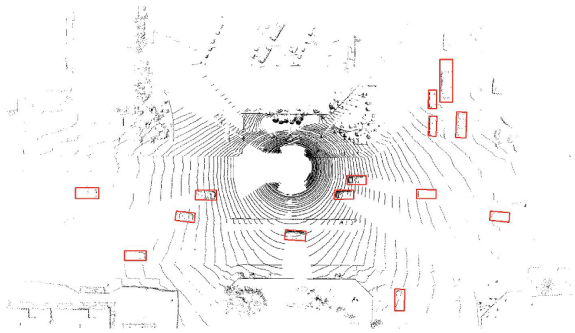


Figure 2.4: Top down BEV representation of a vehicle. LiDAR and camera data are projected onto a 2D plane [8].

2.3.3 Datasets

Several benchmark datasets have been developed for evaluating vehicle-based 3D object detection models. KITTI was among the earliest, while the Waymo Open Dataset provides greater scale and realism [10], [20], [1]. However, both lack a standardised and publicly documented schema, which complicates the generation of compatible custom datasets. The nuScenes dataset addresses this limitation by providing a complete schema and an official development kit. Figure 2.5 illustrates the metadata structure adopted throughout this thesis.

The nuScenes dataset is organised into scenes, each composed of samples recorded at 2 Hz. A sample represents either a key-frame or a sweep. Key-frames include ground truth annotations, whereas sweeps contain unlabelled intermediate frames. Each sample is linked to multiple sensor recordings, including LiDAR, camera, and radar data. Sensor recordings are stored as sample data entries, which contain timestamps, file paths, and references to the corresponding calibrated sensor parameters. Each sensor has a fixed pose relative to the ego vehicle, and its global pose is recorded for every sample. Annotations provide 3D bounding boxes, se-

mantic categories, and the number of LiDAR points contained within each box. Additional metadata captures object category definitions, scene context, and visibility information. Data collection was conducted using a vehicle equipped with six cameras, a LiDAR sensor, five radar units, and a GPS/IMU module [5].

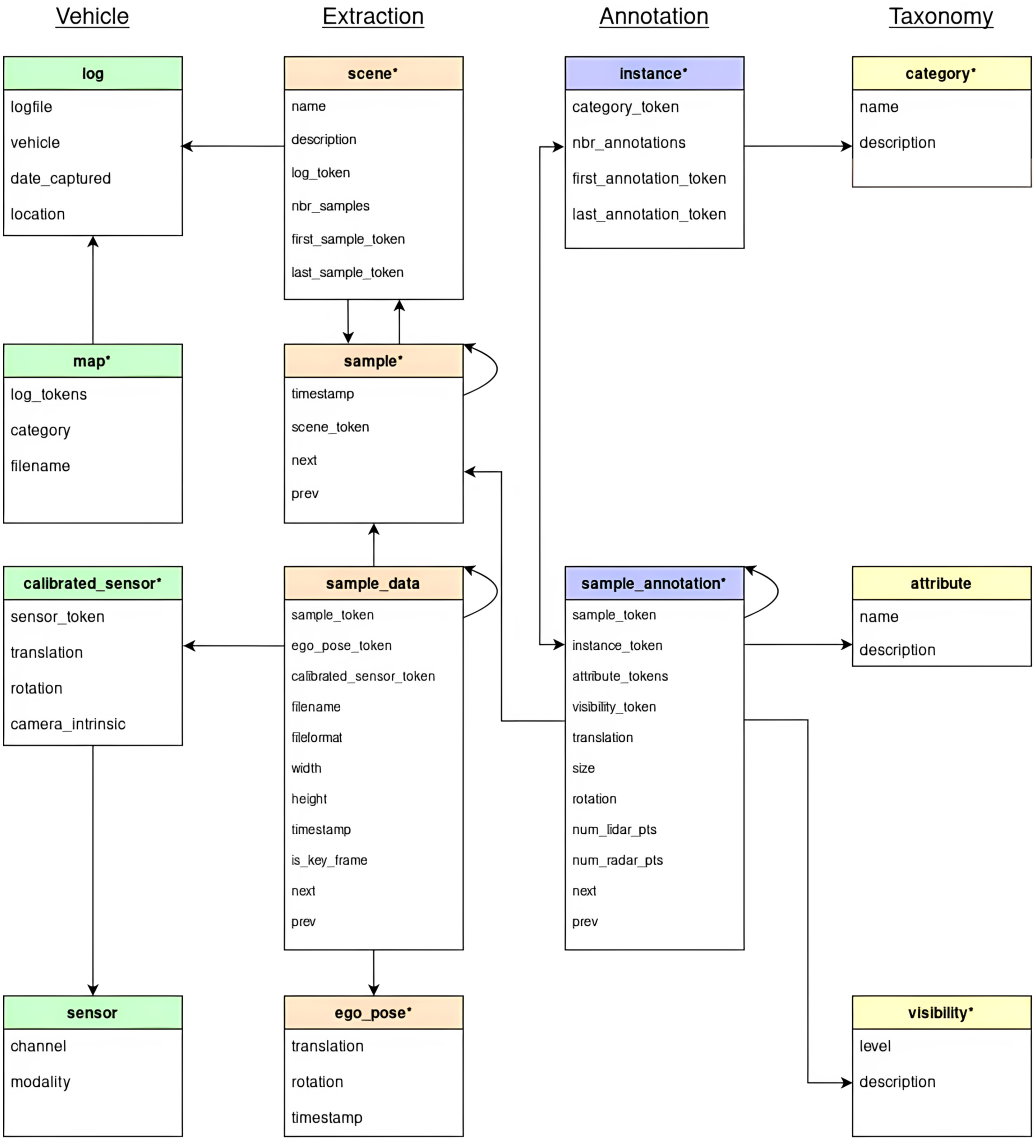


Figure 2.5: Overview of the nuScenes data schema, showing table structure and relationships [5].

2.4 Sensor fusion strategies

Recent advances in 3D object detection show that combining multiple sensor modalities, especially LiDAR and camera data, can improve both accuracy and robustness [21]. Although fusion strategies are categorised differently across studies, this section considers three main types based on the stage at which fusion takes place: early, middle, and late fusion.

2.4.1 Early fusion

Early fusion combines sensor data at the input stage, allowing the model to process a unified representation. This can improve contextual understanding by preserving raw correlations between modalities. However, the approach introduces rigidity, since changes to the architecture of a fusion model often require a complete rework. It is also sensitive to misalignments and inconsistent data formats, which complicate fusion model development [1].

2.4.2 Middle fusion

Middle fusion combines features after each modality is processed independently. This enables the model to merge higher level representations and avoids some issues associated with early fusion. Its effectiveness, however, depends on the task, network architecture, and input characteristics [1], [26].

2.4.3 Late fusion

Late fusion merges the outputs of separate models for each modality. It avoids alignment issues and allows each sensor to be processed independently. However, because the fusion occurs only at the output stage, the model is limited in learning correlations between different sensor types [1], [26].

2.5 Detection models

This section introduces the 3D object detection models evaluated in this study: Centerpoint-Pointpillar, TransFusion-LiDAR, and SECOND as LiDAR-only methods, and BEVFusion as the fusion method. These models were selected based on reported performance in recent benchmarks and publications, as well as the availability of publicly available implementations [3], [13], [23], [28], [29]. Each relies on a different data representation, including voxel grids, point pillars, and BEV.

2.5.1 Centerpoint-Pointpillar

Centerpoint with Pointpillars encoding transforms the point cloud into a 2D pseudo-image using a pillar-based encoder. This is converted into a BEV feature map and processed by a 2D convolutional model to extract spatial features. The model predicts a heatmap indicating the likelihood of object centres and, for each detected centre, regresses object size, orientation, height, velocity, and centre offset. The top-scoring candidates are then selected and decoded into final bounding boxes [29].

2.5.2 TransFusion-LiDAR

TransFusion-LiDAR vowelises the input point cloud and extracts features using sparse 3D convolutions in the BEV frame. A heatmap is then generated to estimate the likelihood of object centres along with attributes such as size and orientation. The top-ranked candidates from this heatmap are used to initialise object queries. These queries are generated from the input data and include information about the predicted object category. A transformer-based detection module then refines these queries by allowing them to interact with the BEV features and with each other. This process produces final bounding box predictions and class scores [3].

2.5.3 SECOND

SECOND voxelises the input point cloud into a sparse 3D grid and computes geometric features for each non-empty voxel using a voxel feature encoding layer. These features are processed using sparse 3D convolutions, after which the height dimension is collapsed to obtain a 2D BEV representation. This is passed to a convolutional detection head that predicts object classes and 3D bounding boxes, including size, orientation, height, and position [28].

2.5.4 BEVFusion

BEVFusion processes LiDAR and image inputs through separate encoders and aligns them in a shared BEV frame to enable spatial fusion. LiDAR data is voxelised and processed using sparse 3D convolution, followed by vertical pooling to project the features onto the BEV frame. Camera images are encoded using a convolutional network that also predicts dense depth distributions across the image. To recover 3D spatial structure, image features are projected into 3D space by assigning each pixel to multiple depth levels, weighted according to the predicted depth distribution. These features are voxelised and pooled into the same BEV grid as the LiDAR features. Middle fusion is realised by concatenating the aligned features and applying 2D convolution over the combined representation [13].

Chapter 3

Methodology

This chapter describes the data generation and perception pipeline developed for the Avular Origin One robot. It covers the creation of a structured synthetic dataset and the evaluation of object detection models using the OpenPCDet framework. Although the task of mobile 3D object detection falls within the domain of autonomous driving research, most publicly available datasets are based on sensor configurations designed for cars. These configurations differ substantially from the sensor layout used on the Avular Origin One. As a result, a custom dataset was created that reflects operational context of the robot.

Due to limited access to the AEB WtE facility and the constraints of real-world data collection, a simulation-based approach was adopted. A virtual environment was developed to replicate the facility and generate sensor data under controlled conditions. The resulting data, referred to as the *AEB detection dataset*, is processed and structured according to the nuScenes format, enabling the use of existing 3D object detection models with minimal modification.

The full pipeline consists of several components, including simulation design, dataset construction, and model evaluation. The chapter begins by describing the hardware setup, covering both the robot and the desktop used for development and inference. It then presents the simulation environment and the data generation pipeline that converts the synthetic data into the nuScenes format.The training and evaluation datasets are then introduced, followed by a description of the inference framework accompanied by the metrics used for evaluation.

3.1 Hardware

ThThe Avular Origin One operates on electric power and provides a battery life of approximately five hours, which is sufficient to perform its daily tasks within the WtE facility [7]. The robot moves using four fixed standard wheels, and steering is achieved through differential drive by varying the relative wheel speeds. The platform is compact, measuring 655 mm in length, 578 mm in width, and 330 mm in height, including the antennas.

In terms of sensing, the robot is equipped with several components relevant to this thesis, as illustrated in Figure **??**. The first is a front-facing Intel RealSense D435f RGB-D camera, which captures RGB images with a FOV of 69 by 42 degrees and produces depth-based point clouds with an FOV of 86 by 58 degrees. The second is an Ouster 3D LiDAR sensor, mounted near the front at the top of the platform. It provides full 360 degree coverage and has a maximum range of approximately 45 metres.

Figure 3.1: The Avular Origin One, with the LiDAR sensor mounted at the front top edge and the depth camera positioned centrally at the front [2].

In addition to the LiDAR and depth camera, the robot is also equipped with six ultrasonic sensors positioned at the front and rear. However, due to inconsistent behaviour observed during preliminary testing by the Maintenance Lab at the HvA, these sensors were excluded and omitted from the data generation process.

Onboard sensor processing and robotic control are handled by embedded hardware, specifically the NVIDIA Jetson Orin NX. To support real-time inference of object detection models, an additional NVIDIA Jetson Orin Nano is mounted on top of the platform.

Finally, for simulation, data generation, and model inference, a desktop system was used alongside the robot. The system is configured with Ubuntu 22.04, an NVIDIA RTX 4070 Super GPU, a Ryzen 7 7800X3D processor, and 32 GB of RAM.

3.2 Simulation and data generation pipeline

3.2.1 Simulation environment

Simulation tools

To mitigate the risks and constraints associated with real-world data collection, a simulation-based approach was adopted for scene generation and sensor emulation. This was implemented using Gazebo Fortress, an open source physics-based simulator maintained by Open Robotics[1]. Gazebo enables the simulation of robots, environments, and sensor data. It is accurate and used in academic robotics research [9]. Gazebo was selected for this thesis based on existing infrastructure provided by the Avular Robotics team, who had already developed a compatible model of the Origin One robot [22]. This model includes all sensors present on the physical robot.

To communicate sensor data from Gazebo to the data generation pipeline, the system was integrated with Robot Operating System 2 (ROS2) Humble, the middleware used to handle all data exchange within the robotic system [14]. ROS2 is a modular framework that structures different tasks into independent processes called nodes. Each node is responsible for a specific function, such as handling sensor input, and can be easily implemented using either Python or C++. Nodes exchange data by publishing and subscribing to topics, which serve as the primary communication interface within the ROS2 ecosystem.

To enable communication between ROS2 nodes and the Gazebo simulation, a ROS2–Gazebo bridge was used. This bridge maps Gazebo topics to ROS2 topics, allowing simulated sensor data to be transferred between the two software systems. For example, as shown in Figure 3.2, the node on the left publishes LiDAR point

[1] See `https://gazebosim.org/docs/fortress` for more information.

cloud data generated in Gazebo. This data is bridged to a ROS2 topic, which is then subscribed to by the node on the right, responsible for storing the data in the nuScenes format.

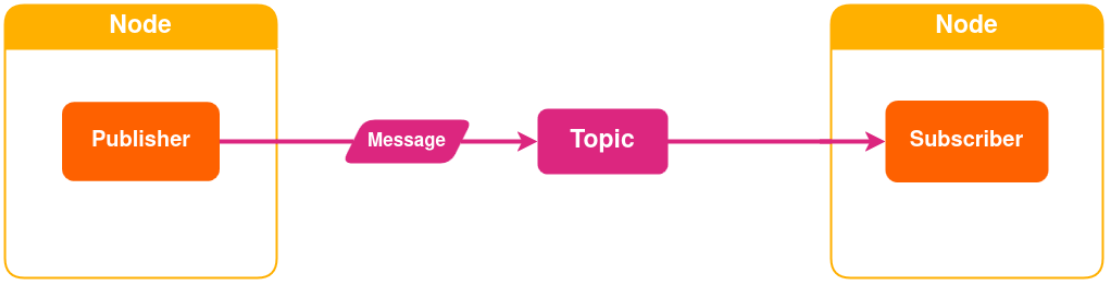


Figure 3.2: Example of two ROS2 nodes publishing and subscribing to a topic.

Each node is implemented within a ROS2 package, which contains the source code, dependencies, and configuration files. These packages are organised within a ROS2 workspace, supporting modular development and ensuring the data generation pipeline remains structured and extensible.

Environment design

The simulation environment was developed entirely in Gazebo as a simplified version of the HRC dumping hall at the AEB facility, as shown in Appendix A. It consists of grey walls and floors, with each waste chute enclosed by square walls along the right side of the hall. Raised footpaths are also included to match the real layout. The environment was simplified because most data is collected using LiDAR, where colour is not relevant. The RGB camera still observes enough colour variation to distinguish objects from the background.

The simulation environment includes the Avular Origin One and up to five truck models obtained from the official Gazebo model catalogue: a box truck and a delivery truck [17], [18]. These are not garbage trucks, since no suitable model was available, and creating one would have required excessive modelling time. Despite this limitation, both are visually adequate for the truck detection task. Each truck measures approximately 5 metres in length, 1.8 metres in width, and 2 metres in height. Although relatively small, testing showed that size did not affect detection performance. Each scene also includes up to two human models, one male and one female, selected from the same catalogue [16], [15]. All object models used in the simulation are shown in Figure 3.3.

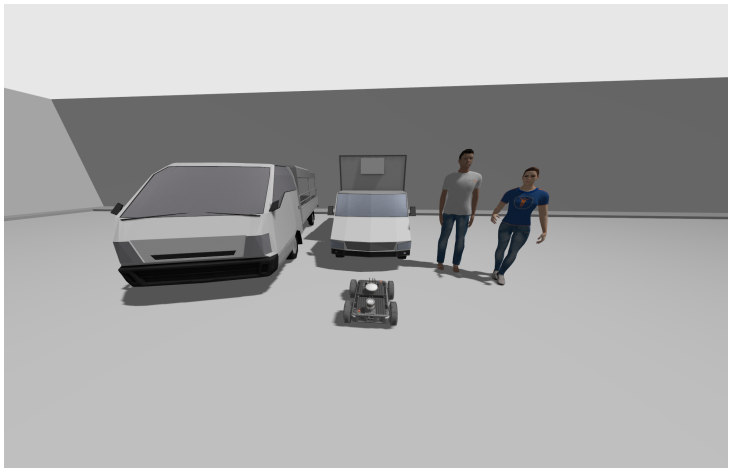


Figure 3.3: Overview of all object models used in the simulation. From left to right: delivery truck [18], box truck [17], male pedestrian [16], female pedestrian [15], and the robot model at the centre [22].

In addition to the robot, one truck is capable of movement using ROS2 commands published through Gazebo topic bridges, allowing evaluation of dynamic interactions. The robot follows predefined routes for various scenarios, including chute inspection and patrol along footpaths. Trucks and pedestrians are arranged to reflect diverse spatial relationships, with all models manually positioned at realistic locations such as near waste chutes or along walking paths as depicted in Figure 3.4.

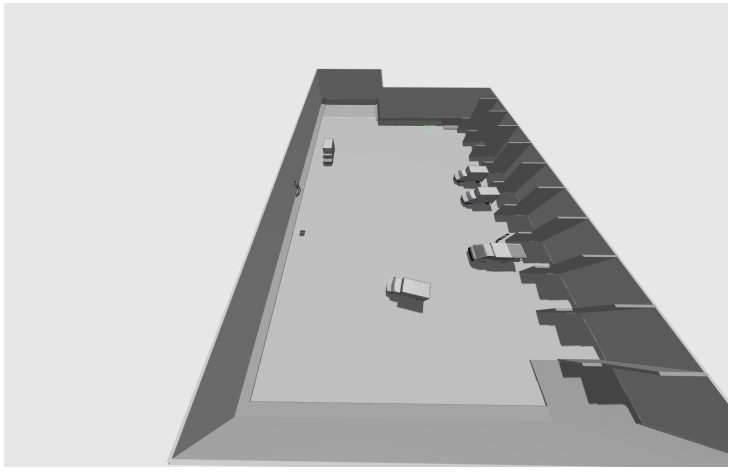


Figure 3.4: Top view of the simulated HRC environment. The robot is positioned in the middle left on a patrol route, with trucks and pedestrians placed at realistic locations.

3.2.2 Dataset generation

To enable the creation of the *AEB detection dataset* using the simulation environment, a dedicated data generation pipeline was developed. The pipeline integrates with the Gazebo simulator via ROS2, captures sensor data and object annotations, and stores all metadata in the nuScenes format, as described in Section 2.3.3. Its implementation is divided into two ROS2 packages written in Python. The main package handles data collection, processing, and metadata generation, while a supporting package defines the custom message types used by the ROS2 topics published by nodes in the main package.

The remainder of this section describes the architecture of the data generation pipeline and the process used to generate a scene.

Architecture

The primary package contains seven ROS2 nodes responsible for acquiring, processing, subscribing and publishing data from various sources, including LiDAR, RGB-D cameras, and object poses. A central controller node, referred to as the main node, manages the initialisation and termination of the pipeline. It executes each node in a separate thread using the ROS2 multithreading module. This ensures that data from Gazebo is captured and published in parallel for consistent data generation.

The dataset generation pipeline is composed of six dedicated ROS2 nodes, each responsible for handling a specific type of data acquisition or transformation. Together, these nodes enable structured collection of multimodal data from the Gazebo simulation and support conversion into the nuScenes format.

Nodes

The LiDAR processor node subscribes to raw point cloud data generated by the simulated LiDAR sensor within Gazebo and publishes a processed version to a designated ROS2 topic. This node performs filtering to remove invalid data points and constructs a message comprising the filtered points, along with associated intensity values, timestamps, and index identifiers, attributes included to maintain compatibility with the model framework. The processed message is published at the maximum rate supported by the system.

The depth points processor node receives point cloud data from the simulated RGB-D camera and transforms it into the LiDAR sensor frame using a static transform between the two sensors. This transformation ensures spatial alignment of the depth data with the LiDAR frame. The point cloud is then downsampled to reduce computational load, and invalid points are removed. The processed data, including intensity, index information, and timestamp is published to a custom ROS2 topic at the highest possible rate.

The camera image processor node handles RGB images captured by the front-facing camera in the simulation. Each image is converted to JPEG format, compressed, and packaged into a message that includes the image data, its dimensions, timestamp, and encoding type. This message is then published to a custom ROS2 topic with minimal delay

The ego pose processor node subscribes to the robot pose topic provided by the simulation, retrieving the 3D position in the global coordinate frame along with orientation expressed as a quaternion, as required by the nuScenes format. The position, defined relative to the centre of the robot, is transformed to the base link frame, which serves as the reference for all other sensor offsets. A message containing the finalized position, orientation, and timestamp is then published to a ROS2 topic at the maximum supported frequency.

The object annotation processor node 3D positions and orientation quaternions for both trucks and pedestrians from the simulation environment. Object sizes are predefined as fixed rectangular bounding boxes and assigned to each corresponding entity extracted from the simulation. The node adjusts the vertical position of each bounding box to correspond with the object centre and annotates each entity with attributes. This information, together with a timestamp, is compiled into a message and published to a ROS2 topic at a fixed frequency of 20 Hz, due to constraints with the Gazebo object poses topic, which could not be correctly bridged to ROS2.

The final and most important node is the metadata writer node, which manages the scene-level construction of the *AEB detection dataset*. It subscribes to all relevant ROS2 topics produced by the processor nodes and converts the collected data into the nuScenes dataset format. Once all messages from these topics are received without error, the node processes a frame at a time. The LiDAR point cloud, as well as a fused version combining LiDAR and depth point clouds, is saved as binary files in their respective sample directories. Camera images are also saved to the corresponding image directory.

Each frame initiates the creation of structured metadata. To ensure consistency across time, the node tracks corresponding object instances throughout the scene and maintains temporal continuity by linking consecutive sample entries. As part of this process, sensor calibration parameters are registered statically for each sensor, and written in the nuScenes format. Scene-level details, including the scene name, description, and total number of frames, are also recorded. All other metadata files, as outlined in Section 2.3.3, are generated accordingly. To maintain performance when samples are added, metadata is written to disk only after the full scene has been processed. Additionally, the node supports automatic extension of the dataset by reinitializing the main node, allowing new scenes to be incorporated without manual file handling or post-processing.

Scene generation

The scene generation pipeline, outlined in Figure 3.5, is initiated from the ROS2 command line using the main node. The user specifies the scene type, number of frames to record, and a short description. For example:

```
ros2 run nodes nodes_main --scene_type t --max_frames 100
--scene_desc "Example description."
```

The scene type flag sets the category as either training (t) or validation (v). Test scenes are not supported because the nuScenes test split excludes annotations, but testing can still be performed on validation scenes within a separate test dataset.

Once launched, the metadata writer node processes frames continuously, achieving approximately eight frames per second (FPS) on a the desktop computer explained in Section 3.1. In contrast, systems without a GPU often experience significant bottlenecks. These limitations can delay message handling and lead to dropped messages from the processor nodes, which in turn reduces the reliability and consistency of the data generation pipeline.

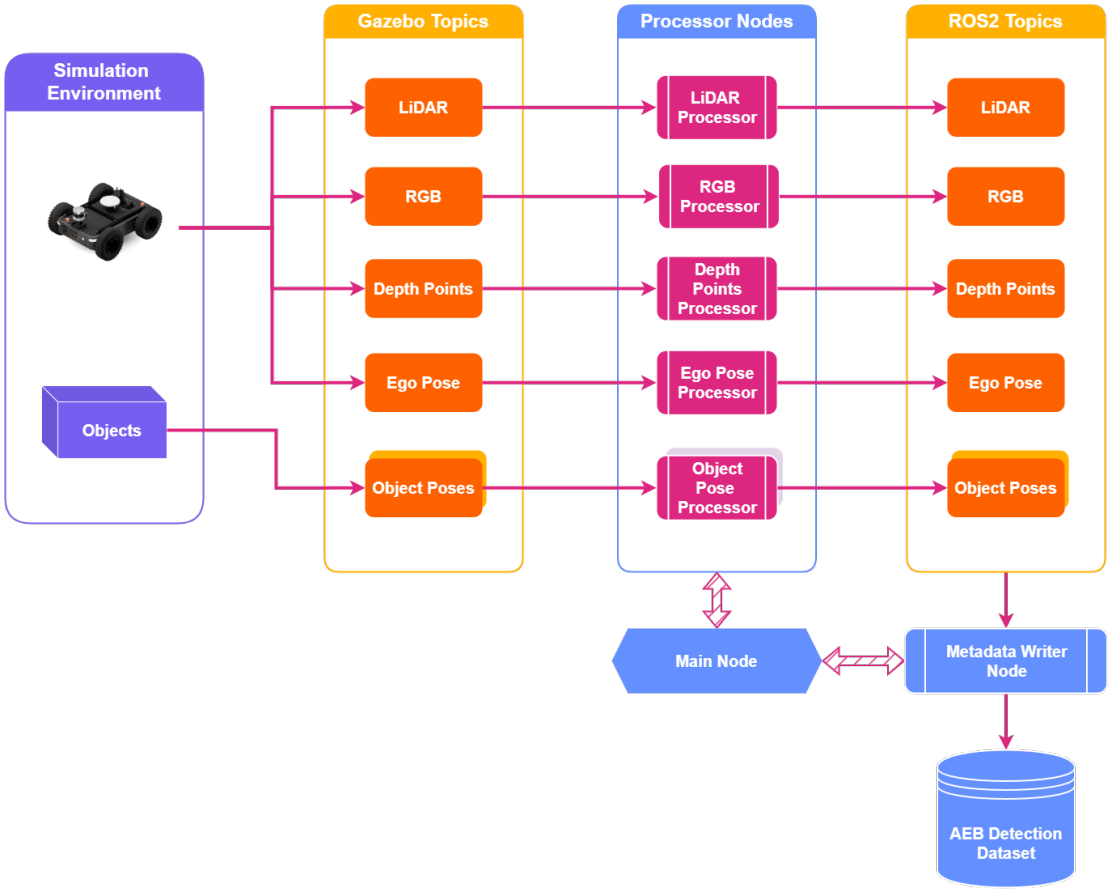


Figure 3.5: Overview of the data generation pipeline used to construct the *AEB detection dataset*. The main node launches all processor nodes and the metadata writer node. The processor nodes collect and publish sensor and pose data, while the metadata writer node subscribes to their topics and converts the data into the nuScenes format.

3.3 Datasets

3.3.1 Training set

The *AEB detection dataset* was created using the data generation pipeline. Two training dataset versions were generated: a standard version containing approximately 2,000 annotated samples across 20 scenes, and an extended version with 4,000 samples across 40 scenes. Each scene contains 100 samples. The structure of both versions is provided in Appendix B.

Training set configurations

To evaluate model performance, four training set variants were defined by combining two types of variation: point cloud filtering and the use of early fusion. In the filtered variants, ground points were removed using a height threshold, and points within a cuboidal region around the robot were excluded. The non-filtered variants retain the complete raw point cloud. For early fusion, standard variants contain only LiDAR data, while combined variants include a fused point cloud composed of LiDAR and depth camera data. This early level fusion, as described in Section 2.4, was used to assess whether incorporating front-facing depth points improves detection. The four resulting variants are denoted $nf{:}s$, $nf{:}c$, $f{:}s$, and $f{:}c$, where nf and f refer to non-filtered and filtered, and s and c refer to standard and combined input types.

Each of the four data variants were used in two training strategies: custom-training and transfer learning. In the custom-training strategy, models were initialised and trained solely on the *AEB detection dataset*, which consisted of 2,000 annotated samples distributed across sixteen training scenes and four validation scenes. Although a larger version of the dataset containing 4,000 samples was available, it was not used. Preliminary experiments indicated that the 2,000-sample version already provided sufficient performance for comparative evaluation. Training with the larger set was expected to increase computational time without offering clear advantages, while also risking overfitting to the simulation environment.

In the transfer learning strategy, models pre-trained on the nuScenes dataset were further trained on the *AEB detection dataset*. This approach allows the models to retain general features learned from diverse urban driving scenarios while adapting to the specific characteristics of the AEB facility environment. A third strategy, used as a baseline, employs the pre-trained model without any fine-tuning. This configuration does not rely on any of the four dataset variants and is applied without modification.

In total, this results in nine configurations per model: eight derived from the two training strategies across the four dataset variants, and one representing the pre-trained baseline. These configurations are denoted using a naming scheme such as $ct_{f:c}$ for the custom-trained, filtered, and combined model or $tl_{nf:s}$ for the transfer-learned, non-filtered, and standard model. The pre-trained baseline is referred to as pt.

3.3.2 Test set

To evaluate the different training set configurations, three distinct test sets were created, each designed on different operational context in which the robot is expected to operate. This division also enables targeted analysis of detection performance at varying distances, which would be harder to isolate in a single, aggregated test set. All test sets were generated using the same simulation environment as the training set, but with new object placements, orientations, and different instances of object models. They contain 300 samples each spanning 3 scenes, with the aggregated set containing 900 samples. Three trucks were replaced with larger variants, only one of which was seen during training, resulting in four delivery trucks and one box truck. Pedestrians were replaced with smaller female models not present in the training set.

Evaluation scenarios

The first test set, patrol, contains scenes in which the robot follows its standard route alongside the footpaths within the facility. The main focus in this set is on detecting trucks and pedestrians at longer distances. Trucks are typically parked at waste chutes or positioned along designated drive paths. Pedestrians are placed either beside parked trucks at the chutes or walking along the footpaths, resulting in long-range and mid-range detection cases.

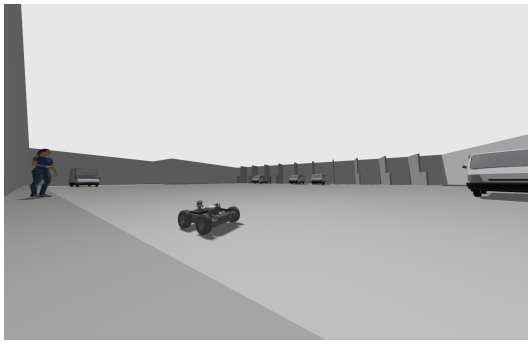


Figure 3.6: Example from the patrol test set.

The second set, inspect, consists of scenes where the robot approaches or leaves a chute location as part of an inspection task. Trucks are again positioned at the chutes or driving in the facility, but object distances are generally shorter than in the patrol set. Increased occlusion is introduced by trucks at farther chute locations. Pedestrian placement follows the same logic as in the patrol scenes.

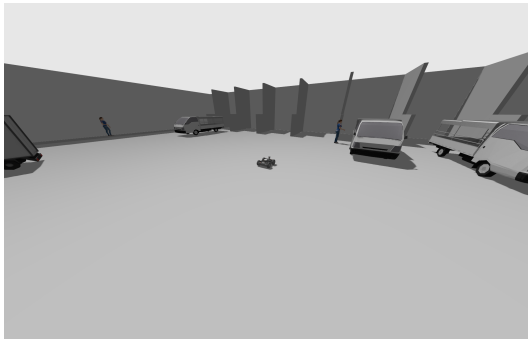


Figure 3.7: Example from the inspect test set.

The final set, danger, focuses on close-range interactions between the robot and nearby objects. Trucks and pedestrians are positioned within close proximity, often simulating near-collision scenarios where the robot passes or is passed by a truck or person at minimal clearance.

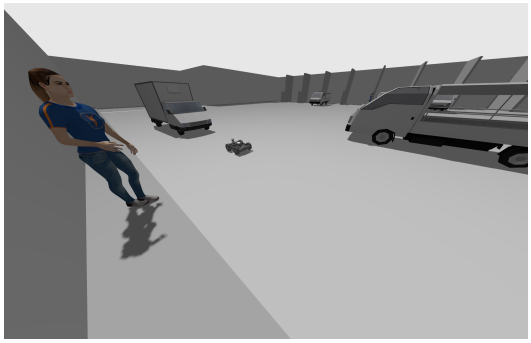


Figure 3.8: Example from the danger test set.

3.4 Inference framework

To support training on the AEB detection dataset and evaluation on the test sets (patrol, inspect, and danger), OpenPCDet was adopted as the primary framework [23]. Developed and maintained by the OpenMMLab community, OpenPCDet is an open-source library built on PyTorch [19] and is focused on 3D object detection using LiDAR data. It provides a modular and extensible framework for training and evaluating a range of state-of-the-art models. While broader libraries such as MMDetection3D [6] support a wider selection of models, OpenPCDet was preferred due to its simpler structure, ease of modification, and availability of relevant pre-trained models.

To aid model development and evaluation, OpenPCDet includes a model zoo with pre-trained weights for various architectures, alongside support for standard datasets such as KITTI, nuScenes, and Waymo. The framework operates through configuration files that define both dataset and model specifications. To enable compatibility with the *AEB detection dataset*, it was necessary to adapt the OpenPCDet data loading pipeline, ensuring it could correctly interpret the custom format and structure.

3.4.1 Dataset integration

OpenPCDet uses a structured pipeline to load dataset samples, generate training inputs, and execute model training or evaluation. Prior to training, point cloud segments that fall within annotated bounding boxes are extracted and stored as individual files, which are subsequently consolidated into a single aggregated file. Additional operations, such as class balancing, are also applied during this stage. Once setup is complete, the training and testing scripts can be executed to train the models or evaluate their performance.

However, this default procedure assumes compatibility with the official nuScenes dataset. Since the custom AEB dataset differs in both class set and sensor configuration, the OpenPCDet pipeline required modification. The original data loader was hardcoded for the standard nuScenes format and expected all eight object classes, which was reduced to two classes: trucks and pedestrians. Similarly, while nuScenes includes six cameras placed around their vehicle, the Avular Origin One uses only a single front-facing camera. The data loader was therefore adapted to support this reduced input and eliminate assumptions tied to the full nuScenes sensor layout. All functionality related to sweeps was also disabled, as the AEB dataset did not include them. Finally, the nuScenes dataset configuration file was appended to correctly register the to ensure compatibility within the framework.

3.4.2 Model configurations

The selected models from the OpenPCDet framework include BEVFusion for middle fusion, as well as TransFusion-LiDAR, Centerpoint-Pointpillar, and SECOND for LiDAR-only detection. These models are described in more detail in Section 2.5. To ensure compatibility with the *AEB detection dataset*, the configuration files were adapted to support two object classes. Most hyperparameters were left unchanged, as the models performed reliably with their default settings. However, several implementation components and assumptions were tied to the original nuScenes dataset and required modification prior to training.

To reduce memory usage and accelerate training, Automatic Mixed Precision (AMP) was enabled for the models that support it, namely Centerpoint-Pointpillar and SECOND. Batch size and the number of data loading workers were selected to fit within the constraints of the training hardware. Finally, training was limited to a maximum of ten epochs, with early stopping enabled to halt training once validation loss no longer improved.

3.4.3 Evaluation metrics

To evaluate detection performance on the test sets, the nuScenes metrics are used. These are computed using the official nuScenes devkit, which integrates directly with the *AEB detection dataset* format. The evaluation focuses on the accuracy of 3D bounding box predictions for trucks and pedestrians. The primary metric is distance average precision (dAP), which is computed using a centre-distance matching approach in the horizontal plane. Each predicted bounding box is matched to the nearest ground truth box within a fixed centre-distance threshold, using a greedy one-to-one matching strategy. Average precision is calculated at thresholds of 0.5, 1, 2, and 4 metres. The final dAP is the mean over these thresholds, computed separately for each object class [5].

In addition to dAP, true positive (TP) metrics are used to quantify the quality of matched predictions, where scores closer to zero indicate better performance. These metrics are computed using a fixed matching threshold of 2 metres. The first TP metric is the average translation error (ATE), which measures the 2D Euclidean distance between the centres of predicted and ground truth boxes on the horizontal plane, expressed in metres. The second TP metric is the average scale error (ASE), defined as $1 - \text{IoU}$ (intersection over union), where IoU calculates the amount of overlap between the predicted and ground truth boxes after aligning their centres and orientations. This metric captures the mismatch in object scale. The third is the average orientation error (AOE), which measures the smallest yaw

angle difference between matched predictions and ground truth boxes, expressed in radians.

Lastly, to assess the runtime performance of each model configuration, the average inference time per sample in milliseconds (ms) is recorded during testing. This value is then converted to FPS using the formula $FPS = 1000/ms$.

Chapter 4

Results

This chapter presents the results of the object detection experiments across the patrol, inspect, and danger scenarios, using the nine training configurations defined in Section 3.3.1. Inference was conducted using the final checkpoint from each model, with a batch size of one to simulate real-time conditions, and the resulting predictions were visualised using the Open3D library.

Quantitative results are presented first, followed by a qualitative analysis highlighting correct detections and common failures.

4.1 Quantitative results

The quantitative results are structured in three parts. First, the detection performance of each model is evaluated individually. Secondly, the performance across different training set configurations is compared. Finally, the Centerpoint-Pointpillar model is examined in more detail to assess its performance across the three test scenarios.

4.1.1 Per-model comparison

Inference speeds differ between fusion and LiDAR-only models, with a disparity in performance. Figure 4.1 shows Centerpoint-Pointpillar, a LiDAR-only model, achieving the highest speed at about 84 FPS, while the fusion-based BEVFusion records the lowest speed at around 22 FPS.

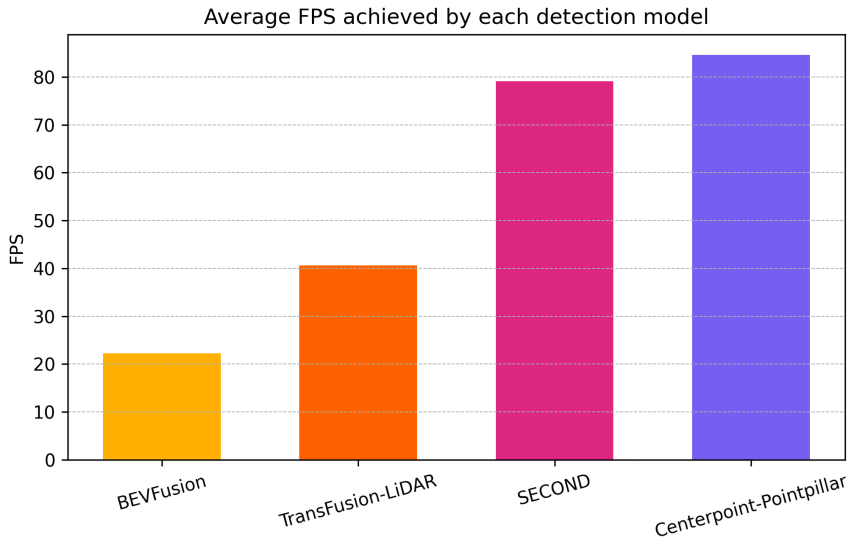


Figure 4.1: Model-wise comparison of inference speed averaged over all scenarios and training set configurations.

Next, Figure 4.2 compares the best ATE scores for each model across the patrol, inspect, and danger scenarios. The inspect scenario yields the lowest ATE values, while the patrol scenario generally produces higher errors. BEVFusion achieves the lowest ATE overall, whereas SECOND records the highest.

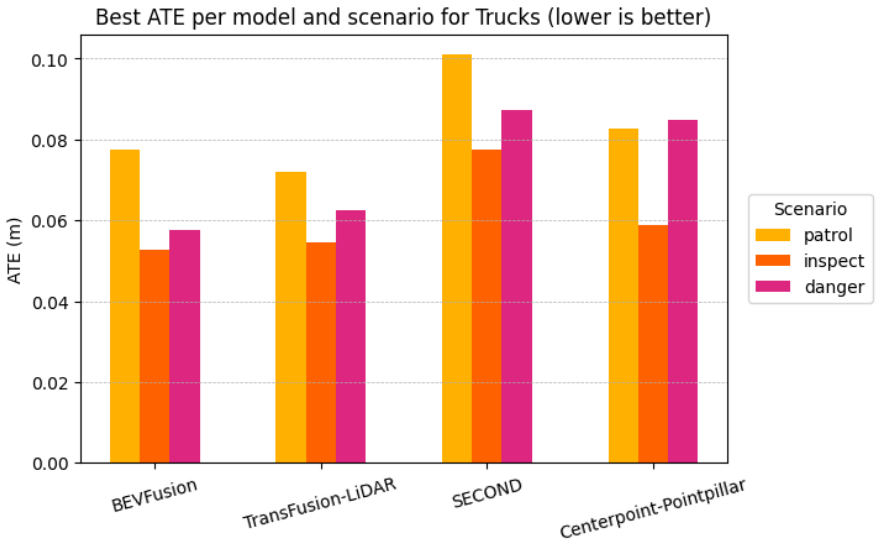


Figure 4.2: ATE comparison across patrol, inspect, and danger scenarios for each model, averaged over training configurations.

Detailed detection metrics for each model and configuration are provided in Appendix D. For truck objects, BEVFusion attains the lowest dAP of 0.979 among the compared models (Table D.1). SECOND achieves dAP scores above 0.98 in several configurations but exhibits consistently higher orientation errors, with AOE values exceeding 1.8 radians (Table D.2). The Centerpoint-Pointpillar model also achieves dAP values above 0.98, with ATE, ASE, and AOE scores comparable to BEVFusion (Table D.4). The TransFusion-LiDAR model delivers detection performance similar to Centerpoint-Pointpillar, with dAP scores reaching 0.981, and achieves the best ATE, ASE, and AOE scores (Table D.3). While TP metrics differ between models, their overall detection performance remains comparable.

For pedestrian detection, BEVFusion and TransFusion-LiDAR achieve the highest dAP scores, exceeding 0.94 in their best configurations. All models, however, show worse TP metrics for pedestrians compared to trucks, particularly in orientation accuracy, with AOE values typically exceeding 1.6 radians (Tables D.5 and D.7). Centerpoint-Pointpillar records dAP values approximately 0.06 lower than BEVFusion and TransFusion-LiDAR, but its TP metrics are similar (Table D.8). SECOND performs similarly to TransFusion-LiDAR in terms of dAP but does not improve on TP metrics (Table D.6).

4.1.2 Training set configuration performance

The following tables summarise average detection performance for all models, grouped by training set configuration. In both the truck (Table 4.1) and pedestrian (Table 4.2) results, transfer learning variants consistently rank highest across all metrics, while custom-trained models yield lower dAP scores and poorer TP metric values. Pre-trained models generally report zero dAP, making their TP metrics inapplicable for comparison. An exception is the pre-trained Centerpoint-Pointpillar variant, which does produce non-zero predictions but still underperforms, with a dAP of approximately 0.5.

When comparing filtered to non-filtered and early fusion to standard variants, detection performance differences are relatively minor. For trucks, non-filtered variants generally achieve higher dAP and lower TP errors than their filtered counterparts. Early fusion yields a slightly higher dAP than the standard variant but results in worse TP metrics. A similar pattern appears in pedestrian detection, where the non-filtered standard variant achieves the best TP metrics, with only slightly lower dAP values.

Table 4.1: Average detection metrics for truck objects across all models, grouped by training set configuration and sorted by dAP.

Configuration	dAP	ATE	ASE	AOE
$tl_{nf:c}$	**0.981**	0.081	0.046	0.502
$tl_{nf:s}$	0.978	**0.077**	**0.036**	**0.485**
$tl_{f:c}$	0.977	0.080	0.045	0.492
$tl_{f:s}$	0.970	0.078	0.051	0.503
$ct_{nf:s}$	0.970	0.096	0.096	0.557
$ct_{nf:c}$	0.968	0.093	0.051	0.540
$ct_{f:s}$	0.950	0.097	0.067	0.527
$ct_{f:c}$	0.931	0.103	0.121	0.537
pt	0.000	1.075	0.961	0.926

Table 4.2: Average detection metrics for pedestrian objects across all models, grouped by training set configuration and sorted by dAP.

Configuration	dAP	ATE	ASE	AOE
$tl_{f:c}$	**0.917**	0.085	0.114	1.805
$tl_{nf:s}$	0.910	**0.081**	0.112	**1.692**
$tl_{nf:c}$	0.899	0.085	0.122	1.899
$tl_{f:s}$	0.890	0.083	0.117	1.771
$ct_{nf:s}$	0.860	0.098	0.155	1.884
$ct_{f:s}$	0.838	0.105	0.119	1.860
$ct_{nf:c}$	0.816	0.105	**0.111**	1.887
$ct_{f:c}$	0.789	0.112	0.188	1.845
pt	0.000	1.000	1.000	1.000

4.1.3 Centerpoint-Pointpillar scenario-based results

Centerpoint-Pointpillar was selected for closer analysis due to its superior average FPS and competitive performance in truck classification, making it suited for examining scenario-specific differences.

In the patrol scenario, the highest dAP reaches 0.990 with the best configuration, though TP metrics are comparatively weaker (Table 4.3). The medium-range inspect scenario yields the best TP scores, indicating more accurate scale and orientation estimates than in the other scenarios (Table 4.4). The danger scenario records the lowest dAP, with a maximum of 0.974. Notably, one custom-trained model achieves the third-highest dAP in this scenario but shows significantly higher orientation error than all other top configurations (Table 4.5).

Table 4.3: Top 3 Centerpoint-Pointpillar training set configurations for truck detection in the patrol scenario

Configuration	dAP	ATE	ASE	AOE
$tl_{f:s}$	**0.990**	0.085	0.045	0.067
$tl_{f:c}$	0.987	0.084	**0.044**	0.067
$tl_{nf:c}$	0.987	**0.083**	0.049	**0.044**

Table 4.4: Top 3 Centerpoint-Pointpillar training set configurations for truck detection in the inspect scenario.

Configuration	dAP	ATE	ASE	AOE
$tl_{nf:c}$	**0.986**	0.069	**0.038**	**0.025**
$tl_{f:c}$	0.978	0.070	0.050	0.043
$tl_{f:s}$	0.978	**0.059**	0.062	0.029

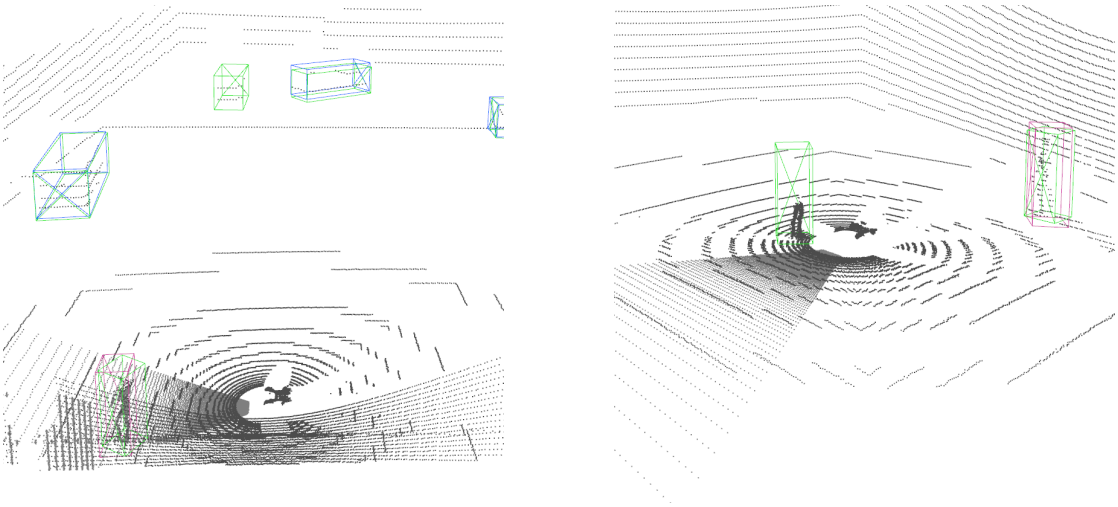
Table 4.5: Top 3 Centerpoint-Pointpillar training set configurations for truck detection in the danger scenario.

Configuration	dAP	ATE	ASE	AOE
$tl_{nf:s}$	**0.974**	**0.085**	**0.043**	**0.026**
$tl_{nf:c}$	0.973	0.096	0.046	0.047
$ct_{nf:s}$	0.966	0.110	0.057	0.246

4.2 Qualitative analysis

The following figures illustrate common detection patterns observed in the three test scenarios. All visualisations are based on LiDAR point clouds, with predicted and ground truth boxes overlaid. Ground truth boxes are shown in green, truck predictions in blue, and pedestrian predictions in purple.

Two common failure cases related to object distance are shown in Figure 4.3. In the patrol scene (a), a distant truck is missed entirely by the model. Although it appears in the point cloud, the sparse representation at that range reduces its detectability. This behaviour was consistent across all models. In contrast, the danger scene (b) shows a failure at close range, where a nearby pedestrian is not detected despite its legs being clearly visible in the point cloud.

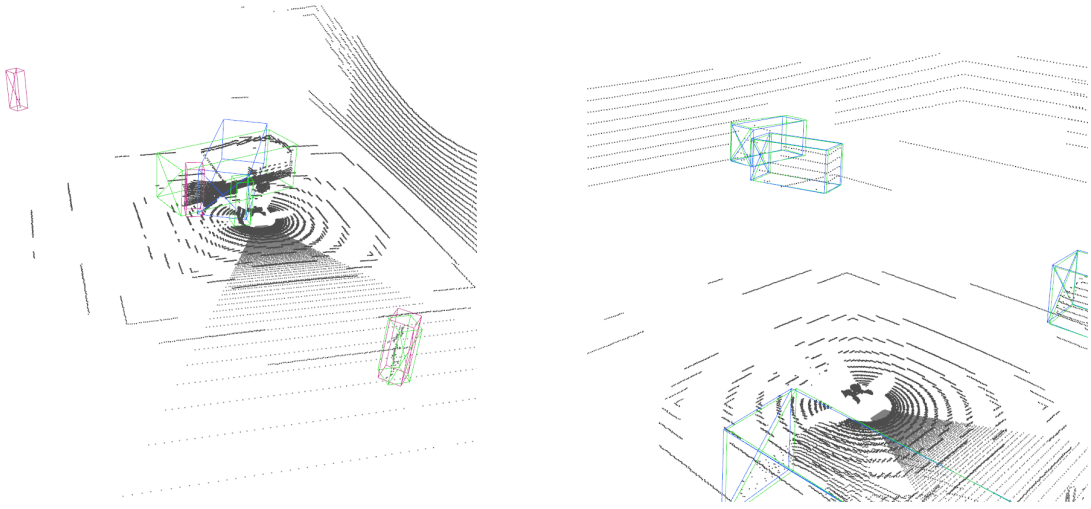


(a) patrol test set

(b) danger test set

Figure 4.3: Examples of distance-related detection failures. (a) shows a missed detection of a distant truck, while (b) shows a missed detection of a nearby pedestrian. These issues occurred consistently across all models in similar conditions.

Scenes with spatial clutter and occlusion often challenge detection reliability, as illustrated in Figure 4.4. In (a), a dense and dynamic scene from the danger test set produces several false positives around a truck and nearby pedestrians, with multiple pedestrian bounding boxes predicted in regions where no pedestrians are present. In contrast, (b) shows a correct detection in the inspect scenario, where a partially occluded truck is still localised accurately. While mild occlusion does not significantly affect performance, spatial clutter frequently leads to false positives.



(a) danger test set

(b) inspect test set

Figure 4.4: Contrasting model behaviour in complex scenes. (a) shows a failure case from the danger test set, where both a truck and multiple pedestrians are incorrectly predicted. (b) shows a success case from the inspect test set, where a partially occluded truck is correctly detected.

Close-range interactions were included in the danger scenario to assess detection performance under high-risk conditions. Figure 4.5 shows an example in which a truck passes in close proximity to the robot and is detected in all three sequential frames. This behaviour was observed consistently across all models.

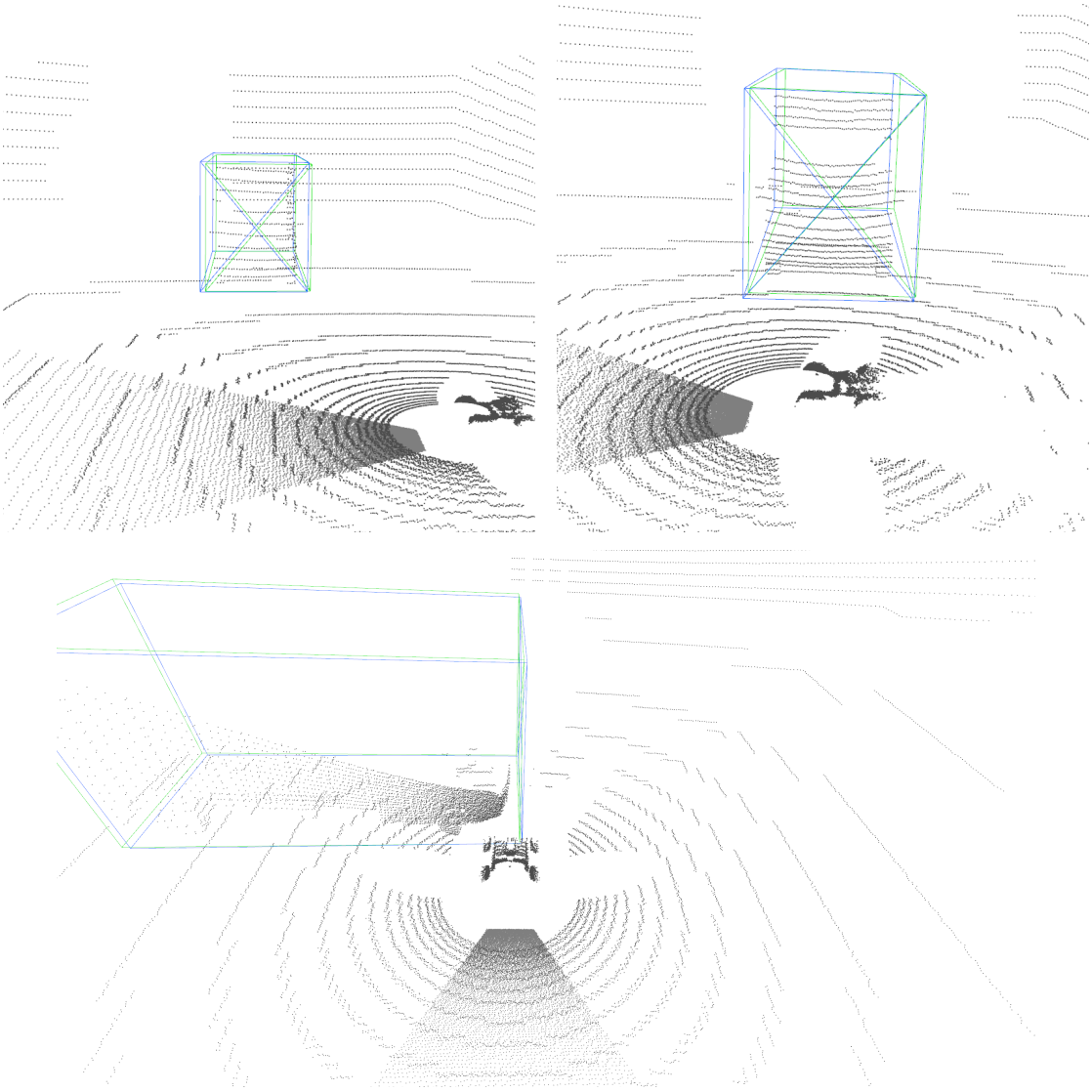


Figure 4.5: Three sequential frames from the danger test set showing the robot near a passing truck. The model correctly detects the truck in each frame despite close proximity.

Chapter 5

Evaluation

The results show clear differences in inference speed between fusion-based and LiDAR-only models. BEVFusion records the lowest average FPS, indicating that its computational demands make it unsuitable for deployment on the Avular Origin One. This reduced performance is attributable to the additional overhead introduced by its fusion architecture. In contrast, Centerpoint-Pointpillar achieves the highest FPS among the evaluated models. While all models convert raw point clouds into BEV representations, Centerpoint-Pointpillar applies a more efficient pillar-based encoding scheme, as outlined in Section 2.5 and further discussed in Section 2.3.2. This design choice yields a runtime advantage over voxel-based methods such as SECOND, with a measured difference of approximately 6 FPS. TransFusion-LiDAR is approximately half as fast as the other LiDAR-only models, likely due to the computational overhead introduced by its transformer-based architecture.

Detection accuracy was evaluated using ATE across all scenarios. BEVFusion and TransFusion-LiDAR consistently generate bounding boxes closest to ground truth, performing reliably across different object distances. The patrol scenario records a slightly higher ATE, by approximately 0.02 metres on all models, suggesting a weak correlation between distance and localisation accuracy. In the danger scenario, performance declines across all models, with Centerpoint-Pointpillar exhibiting the largest reduction in accuracy. The model demonstrates increased difficulty when operating in conditions close to objects.

Class-wise performance exhibits limited variation in truck detection, with dAP, ATE, and ASE scores remaining comparable across all models. An exception is observed in SECOND, which shows a substantially higher AOE, indicating frequent orientation errors. The mean angular error of 103° points to systematic confusion

between front and rear orientation, likely caused by averaging between 0° and 180°. In contrast, pedestrian detection yields consistently lower performance across all metrics, particularly in AOE. This can be attributed to the smaller bounding box dimensions, which reduce introduce ambiguity in estimated orientation. Despite incorporating the front-facing camera, BEVFusion shows no measurable improvement in detection performance for either class, likely due to the limited visibility provided by the narrow FOV.

Nine different training strategies were compared. Transfer learning consistently outperformed the other strategies across all metrics. This is likely due to broader scene understanding encoded in the pre-trained weights, whereas the custom-training strategy relied solely on the *AEB detection dataset*. During transfer learning, the pre-trained weights were adapted to the specific domain of the Origin One, improving generalisation to the test data. In contrast, the baseline strategy resulted in poor detection performance. It had been trained using a higher LiDAR mounting position, which led to poor alignment with the test set point clouds and resulted in inaccurate bounding box predictions. The custom-training strategy performed slightly worse than transfer learning, suggesting that a larger dataset would be required to achieve comparable generalisation.

Additional dataset variants involving early fusion and point cloud filtering had limited impact on detection performance. Early fusion resulted in a slight increase in dAP but also led to degraded TP metrics, while filtering reduced truck detection accuracy. These effects were not sufficient to justify the added processing steps. The limited contribution of the front-facing camera likely explains the weak influence of early fusion. Since early fusion was applied only during data pre-processing, it had no measurable impact on inference speed. Filtering likewise produced no observable improvements in detection performance.

Examining Center-point-Pointpillar in more detail, the model performs best in the long-range patrol scenario, where dAP is highest. However, its TP localisation errors are higher than in the mid-range inspect scenario. This indicates that while the model can detect distant objects, its localisation accuracy decreases at range. Performance in the danger scenario is consistently lower across metrics such as dAP and ATE, suggesting difficulty with close-range or high object density conditions.

Qualitative observations confirm these limitations. Nearby pedestrians are frequently missed, indicating insufficient training data for extreme close-range interactions. Trucks and pedestrians located farther from the robot also tend to be

missed, likely due to point sparsity at long range. These detection failures appear linked to the distribution of training data and may be addressed through targeted collection of under-represented cases. In contrast, occlusion does not consistently degrade performance. Sparse but distinctive point patterns still enable successful prediction. Cluttered scenes with high point density, however, can result in unstable or incorrect detections. Figure 4.5 shows that close proximity to large objects such as trucks does not inherently cause detection failure. The presence of accurate predictions in such cases indicates that the models can still detect critical near-miss situations relevant to safe navigation in complex environments.

Chapter 6

Conclusion

This chapter revisits the main research questions and summarises the outcomes of the evaluation. It also reflects on key limitations and identifies opportunities for future work.

6.1 Summary and conclusion

This thesis explored the feasibility of real time 3D object detection on the Avular Origin One using LiDAR, and assessed how detection performance changes when fusing it with a front-facing camera, all within a simulation environment.

To enable this, a modular data generation pipeline was developed to extract synthetic sensor data from Gazebo and convert it into the *AEB detection dataset* in nuScenes format, ensuring compatibility with the OpenPCDet inference framework. Three LiDAR-only models and one fusion model were tested: Centerpoint-Pointpillar, SECOND, TransFusion-LiDAR, and BEVFusion. Each model was evaluated under three scenarios using varied training set configurations.

Evaluation showed that LiDAR-only detection produced accurate results in the simulated environment. Centerpoint-Pointpillar achieved the highest FPS with competitive detection performance, making it the most practical option for real-time use on AMRs. Fusion using the BEVFusion model did not lead to measurable improvements, due to the limited contribution of the front-facing camera and its restricted FOV. Moreover, it incurred significant computational cost due to its complex architecture. Detection failures across all models were primarily caused by insufficient close-range data and point sparsity at long distances, while occlusion had limited impact. Nonetheless, detection remained reliable in several near-miss cases despite increased scene complexity.

Of the training strategies evaluated, transfer learning consistently improved detection performance across all models. In contrast, the pre-trained baseline strategy underperformed due to mismatches in sensor placement, while the custom-training strategy was constrained by reduced generalisation resulting from the smaller dataset. Additional variants such as early fusion and point cloud filtering had minimal effect. Early fusion did not affect inference speed but remained limited by the narrow FOV of the front-facing camera, while filtering had no observable impact on model output.

Ultimately, LiDAR proved to be a suitable sensing modality for 3D object detection in the simulated environment, while camera–LiDAR fusion did not yield measurable improvements under the tested conditions. Among the evaluated models, Centerpoint-Pointpillar trained using transfer learning offered the best trade-off between detection accuracy and inference speed. It achieved the highest average FPS and produced fewer true positive errors than voxel-based models. Fusion using the BEVFusion model incurred significantly lower inference speed due to its architectural complexity, and early fusion had no measurable effect. These results suggest that fusion approaches, in the tested configuration, are unsuitable for real time deployment on the Avular Origin One.

6.2 Discussion

While LiDAR-based 3D detection proved feasible in the simulation environment, several assumptions and simplifications limit the generalisability of the findings. Most notably, the simulation does not accurately reflect the complexity of the HRC environment. The sensor data is less noisy and more uniform, while environmental variation is limited. Consequently, models trained on this data may not generalise to the real-world domain, and detection performance is expected to degrade outside simulation, although the extent of this degradation remains unknown.

In addition, the simulated objects do not represent real garbage trucks. Their geometry and scale may differ, which could affect the accuracy of bounding box predictions when models are deployed in the HRC. The use of synthetic annotations based on assumed ground truth poses also introduces limitations, as they presume perfect object localisation, which is not attainable in a real deployment. This reliance on known positions applies to both the robot and nearby objects. While the simulation provides exact pose information, real deployments rely on SLAM, which is less precise. This mismatch may introduce annotation errors that reduce detection performance.

Finally, the evaluation was conducted on desktop hardware, which offers substantially more computational capacity than the Jetson Orin Nano used on the Avular Origin One. As a result, the reported inference speeds may not be achievable during real deployment. This discrepancy limits the ability to fully answer the research question, which concerns the feasibility of real-time detection on platforms with constrained resources. Although the evaluation identifies promising model candidates, whether they can be deployed on the target hardware remains unresolved.

6.3 Future work

To address the identified limitations, a step should involve evaluating how models trained in simulation perform on the real robot. This would provide insight into the degree of domain shift and help quantify how well the simulated training data transfers to real-world conditions. Based on these findings, future work should focus on collecting a real-world dataset in the HRC using the robot platform. This would ensure that the training data captures realistic sensor noise, object geometry, and scene complexity. In particular, including representative examples of garbage trucks is needed for improving the accuracy of bounding box predictions.

Since annotations in simulation were generated automatically from known object poses, creating a real-world dataset will require an annotation pipeline. This could involve manual 3D labelling tools, semi-automated methods based on weak supervision, or tracking-based approaches using SLAM. If quality annotation proves too challenging in the real environment, an alternative path would be to expand the simulation environment to better reflect the details of the HRC, thereby producing more representative synthetic data.

Another key challenge is localisation. In simulation, perfect pose information was assumed, which is impossible in a real-world scenario. Alternative localisation strategies such as SLAM or marker systems should be investigated to estimate the position of the robot and the spatial alignment of nearby objects.
Finally, deploying models on embedded hardware will require significant optimisation. Techniques such as TensorRT conversion, model pruning, and quantisation should be explored to reduce computational load while maintaining detection performance. If inference speed can be improved to an acceptable level, fusion approaches may become viable. In that case, future work could also explore expanding the camera setup to increase the total FOV, potentially making fusion more effective than in the current single-camera configuration.

Bibliography

[1] Simegnew Yihunie Alaba, Ali C. Gurbuz, and John E. Ball. "Emerging Trends in Autonomous Vehicle Perception: Multimodal Fusion for 3D Object Detection". In: *World Electric Vehicle Journal* 15.1 (2024). ISSN: 2032-6653. DOI: 10.3390/wevj15010020. URL: https://www.mdpi.com/2032-6653/15/1/20.

[2] Avular Robotics. *The Origin One autonomous mobile robot platform.* https://www.avular.com/robots. Accessed 2025-06-09.

[3] Xuyang Bai et al. "TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 1080–1089. DOI: 10.1109/CVPR52688.2022.00116.

[4] Andrea Bonci et al. "Human-Robot Perception in Industrial Environments: A Survey". In: *Sensors* 21.5 (2021). ISSN: 1424-8220. DOI: 10.3390/s21051571. URL: https://www.mdpi.com/1424-8220/21/5/1571.

[5] Holger Caesar et al. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[6] MMDetection3D Contributors. *MMDetection3D: OpenMMLab next-generation platform for general 3D object detection.* https://github.com/open-mmlab/mmdetection3d. 2020.

[7] Evelien van Driel et al. *Verbetering van veiligheid en traceerbaarheid in de storthal van AEB Amsterdam: Simulatie van een Autonomous Guided Vehicle*. Student report, Hogeschool van Amsterdam. 2025.

[8] Moritz Drobnitzky et al. "Survey and systematization of 3D object detection models and methods". In: *The Visual Computer* 40.3 (2024), pp. 1867–1913. ISSN: 1432-2315. DOI: 10.1007/s00371-023-02891-1. URL: https://doi.org/10.1007/s00371-023-02891-1.

[9] Andrew Farley, Jie Wang, and Joshua A. Marshall. "How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion". In: *Simulation Modelling Practice and Theory* 120 (2022), p. 102629. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2022.102629. URL: https://www.sciencedirect.com/science/article/pii/S1569190X22001046.

[10] A Geiger et al. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237. DOI: 10.1177/0278364913491297. eprint: https://doi.org/10.1177/0278364913491297. URL: https://doi.org/10.1177/0278364913491297.

[11] Shangfeng Huang et al. "Multi-layer Pointpillars: Multi-layer Feature Abstraction for Object Detection from Point Cloud". In: *Pattern Recognition and Computer Vision*. Ed. by Yuxin Peng et al. Cham: Springer International Publishing, 2020, pp. 626–637. ISBN: 978-3-030-60633-6.

[12] Alex H. Lang et al. "PointPillars: Fast Encoders for Object Detection From Point Clouds". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12689–12697. DOI: 10.1109/CVPR.2019.01298.

[13] Tingting Liang et al. "BEVFusion: A Simple and Robust LiDAR-Camera Fusion Framework". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 10421–10434. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/43d2b7fbee8431f7cef0d0afed51c691-Paper-Conference.pdf.

[14] Steven Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[15] OpenRobotics. *Casual female*. Open Robotics. Sept. 2023. URL: https://fuel.gazebosim.org/1.0/OpenRobotics/models/Casual%20female.

[16] OpenRobotics. *Standing person*. Open Robotics. Sept. 2023. URL: https://fuel.gazebosim.org/1.0/OpenRobotics/models/Standing%20person.

[17] OpenRobotics. *TruckBox*. Open Robotics. Sept. 2023. URL: https://fuel.gazebosim.org/1.0/OpenRobotics/models/TruckBox.

[18] OpenRobotics. *TruckDelivery*. Open Robotics. Sept. 2023. URL: https://fuel.gazebosim.org/1.0/OpenRobotics/models/TruckDelivery.

[19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f701Paper.pdf.

[20] Pei Sun et al. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2443–2451. DOI: 10.1109/CVPR42600.2020.00252.

[21] Yingjuan Tang et al. "Multi-modality 3D object detection in autonomous driving: A review". In: *Neurocomputing* 553 (2023), p. 126587. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.126587. URL: https://www.sciencedirect.com/science/article/pii/S0925231223007105.

[22] Avular Robotics Development Team. *Avular Origin simulation: The simulation environment for the Avular Origin One, using Gazebo Fortress and ROS2 Humble*. https://github.com/avular-robotics/avular_origin_simulation. 2024.

[23] OpenPCDet Development Team. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. https://github.com/open-mmlab/OpenPCDet. 2020.

[24] Vossman and M. W. Toews. *A series of voxels in a stack with a single voxel shaded*. Licensed under CC BY-SA 2.5. 2006. URL: https://commons.wikimedia.org/wiki/File:Voxels.svg.

[25] Yangfan Wang et al. "Recent advances in 3D object detection based on RGB-D: A survey". In: *Displays* 70 (2021), p. 102077. ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2021.102077. URL: https://www.sciencedirect.com/science/article/pii/S0141938221000846.

[26] Yingjie Wang et al. "Multi-Modal 3D Object Detection in Autonomous Driving: A Survey". In: *International Journal of Computer Vision* 131.8 (2023), pp. 2122–2152. DOI: 10.1007/s11263-023-01784-z. URL: https://doi.org/10.1007/s11263-023-01784-z.

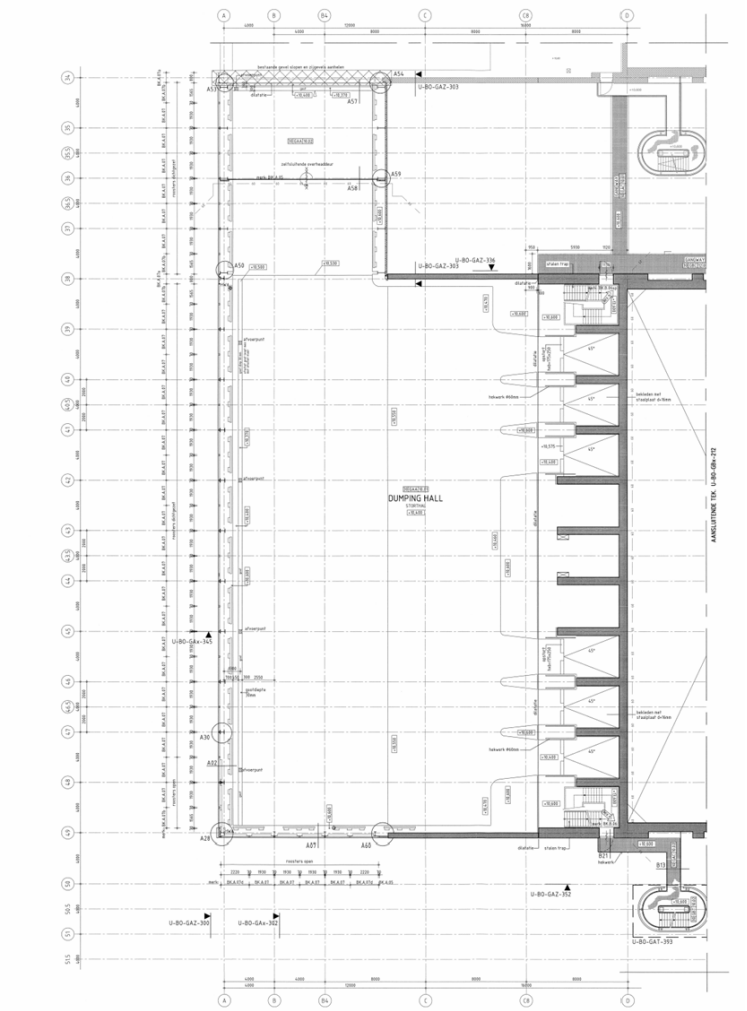
[27] Yutian Wu et al. "Deep 3D Object Detection Networks Using LiDAR Data: A Review". In: *IEEE Sensors Journal* 21.2 (2021), pp. 1152–1171. DOI: 10.1109/JSEN.2020.3020626.

[28] Yan Yan, Yuxing Mao, and Bo Li. "SECOND: Sparsely Embedded Convolutional Detection". In: *Sensors* 18.10 (2018). ISSN: 1424-8220. DOI: 10.3390/s18103337. URL: https://www.mdpi.com/1424-8220/18/10/3337.

[29] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. "Center-based 3D Object Detection and Tracking". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11779–11788. DOI: 10.1109/CVPR46437.2021.01161.

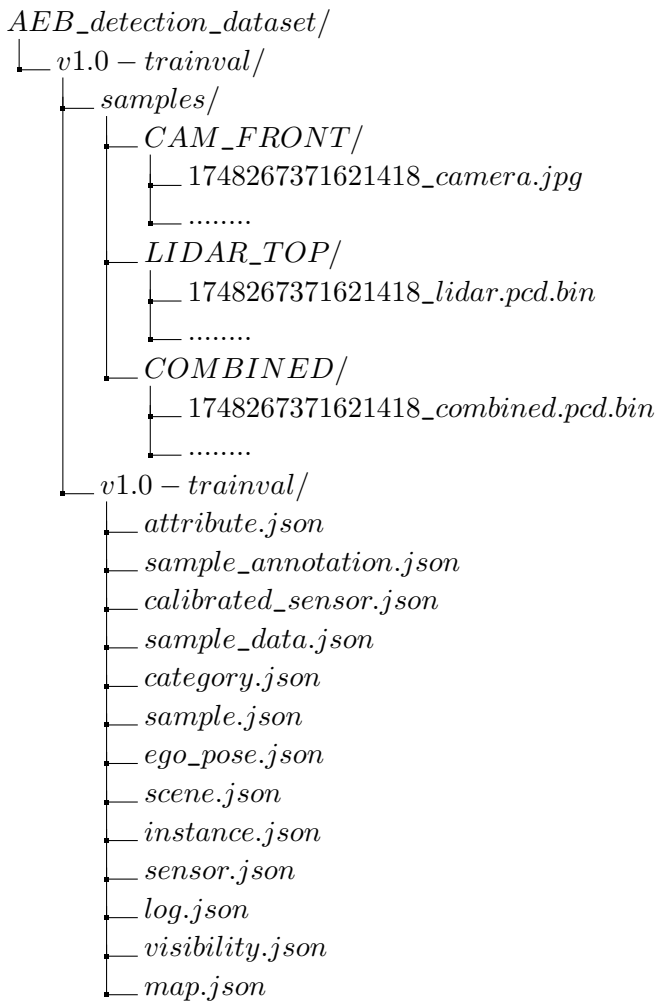
Appendix A

HRC blueprint



Appendix B

Dataset directory tree



Appendix C

Source code

https://github.com/Nordin-26/3D-object-detection-bsc-thesis.gitGitHub repository data generation pipeline.

Appendix D

Performance by configuration

D.1 Truck detection performance

Table D.1: Detection metrics for Truck objects using BEVFusion, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{nf:c}$	**0.979**	0.069	0.068	0.027	22.46
$tl_{nf:s}$	0.978	**0.065**	**0.032**	**0.024**	22.25
$tl_{f:c}$	0.978	0.069	0.044	0.034	22.29
$tl_{f:s}$	0.976	0.068	0.047	0.033	22.06
$ct_{nf:c}$	0.974	0.078	0.066	0.060	22.31
$ct_{nf:s}$	0.972	0.113	0.181	0.102	22.35
$ct_{f:c}$	0.949	0.086	0.313	0.064	22.29
$ct_{f:s}$	0.943	0.079	0.071	0.065	22.23
pt	0.000	1.075	0.961	0.926	22.13

Table D.2: Detection metrics for Truck objects using SECOND, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{f:c}$	**0.982**	0.100	0.044	**1.805**	79.88
$tl_{nf:c}$	**0.982**	0.106	0.039	1.821	79.16
$tl_{f:s}$	0.980	0.092	0.041	1.833	79.35
$tl_{nf:s}$	0.975	**0.089**	0.042	1.873	79.88
$ct_{nf:s}$	0.967	0.103	**0.021**	1.904	78.87
$ct_{nf:c}$	0.958	0.114	0.027	1.824	79.24
$ct_{f:s}$	0.942	0.123	0.027	1.824	78.25
$ct_{f:c}$	0.919	0.146	0.026	1.824	78.97
pt	0.000	1.000	1.000	1.000	78.66

Table D.3: Detection metrics for Truck objects using TransFusion-LiDAR, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{nf:c}$	**0.981**	**0.065**	0.032	0.020	40.88
$tl_{nf:s}$	0.979	0.070	**0.029**	**0.017**	40.16
$tl_{f:c}$	0.978	0.068	0.042	0.030	40.76
$ct_{nf:s}$	0.973	0.073	0.127	0.039	40.85
$ct_{nf:c}$	0.965	0.079	0.050	0.043	40.86
$tl_{f:s}$	0.963	0.069	0.057	0.024	40.55
$ct_{f:s}$	0.954	0.080	0.116	0.042	40.32
$ct_{f:c}$	0.931	0.081	0.070	0.044	40.44
pt	0.000	0.842	0.878	0.814	40.65

Table D.4: Detection metrics for Truck objects using Centerpoint-Pointpillar, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{nf:c}$	**0.982**	0.083	0.044	0.038	85.32
$tl_{nf:s}$	0.979	**0.082**	**0.039**	**0.025**	84.84
$ct_{nf:c}$	0.974	0.100	0.062	0.254	85.16
$tl_{f:c}$	0.969	0.083	0.052	0.055	85.13
$ct_{nf:s}$	0.967	0.096	0.055	0.182	84.89
$ct_{f:s}$	0.960	0.106	0.055	0.178	84.86
$tl_{f:s}$	0.960	0.083	0.058	0.061	84.51
$ct_{f:c}$	0.926	0.101	0.073	0.214	82.41
pt	0.537	0.217	0.279	2.108	22.46

D.2 Pedestrian detection performance

Table D.5: Detection metrics for Pedestrian objects using BEVFusion, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{nf:s}$	**0.949**	**0.076**	**0.110**	**1.616**	22.46
$tl_{f:s}$	0.948	0.077	0.114	1.650	22.35
$tl_{f:c}$	0.923	0.081	0.117	1.788	22.31
$tl_{nf:c}$	0.920	0.083	0.150	1.911	22.29
$ct_{nf:s}$	0.914	0.123	0.281	1.783	22.29
$ct_{f:s}$	0.839	0.112	0.142	1.844	22.25
$ct_{nf:c}$	0.826	0.101	0.124	2.074	22.23
$ct_{f:c}$	0.814	0.117	0.433	1.812	22.13
pt	0.000	1.000	1.000	1.000	22.06

Table D.6: Detection metrics for Pedestrian objects using SECOND, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{f:s}$	**0.925**	**0.085**	0.097	**1.750**	79.88
$tl_{f:c}$	0.918	0.090	0.110	1.880	79.56
$tl_{nf:s}$	0.869	0.088	0.104	**1.750**	79.35
$tl_{nf:c}$	0.866	0.087	0.107	1.877	79.16
$ct_{f:s}$	0.848	0.105	0.099	1.932	78.87
$ct_{nf:s}$	0.829	0.090	0.099	1.980	78.25
$ct_{nf:c}$	0.757	0.124	0.100	1.840	79.24
$ct_{f:c}$	0.684	0.123	**0.096**	1.882	78.97
pt	0.000	1.000	1.000	1.000	78.66

Table D.7: Detection metrics for Pedestrian objects using TransFusion-LiDAR, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{f:c}$	**0.942**	0.085	0.105	1.841	40.88
$tl_{nf:c}$	0.933	.081	0.116	1.995	40.16
$tl_{nf:s}$	0.930	**0.079**	0.117	**1.724**	40.76
$tl_{f:s}$	0.906	0.085	**0.101**	1.809	40.55
$ct_{nf:s}$	0.901	0.089	0.128	1.781	40.85
$ct_{f:s}$	0.896	0.104	0.135	1.835	40.32
$ct_{nf:c}$	0.876	0.096	0.117	1.938	40.86
$ct_{f:c}$	0.863	0.099	0.118	1.737	40.44
pt	0.000	1.000	1.000	1.000	1.000

Table D.8: Detection metrics for Pedestrian objects using Centerpoint-Pointpillar, sorted by dAP

Configuration	**dAP**	**ATE**	**ASE**	**AOE**	**FPS**
$tl_{nf:s}$	**0.890**	**0.079**	0.115	**1.680**	85.32
$tl_{f:c}$	0.884	0.083	0.124	1.712	85.13
$tl_{nf:c}$	0.875	0.090	0.115	1.714	84.89
$ct_{nf:c}$	0.805	0.099	0.104	1.695	84.86
$ct_{f:c}$	0.796	0.109	0.104	1.948	84.86
$ct_{nf:s}$	0.796	0.090	0.112	1.991	84.51
$tl_{f:s}$	0.780	0.085	0.145	1.728	84.56
$ct_{f:s}$	0.769	0.099	**0.103**	1.827	82.41
pt	0.006	1.240	0.971	1.428	82.41