Hochschule Ravensburg-Weingarten
University of Applied Sciences

Lidar and Radar System

Task 1: Evaluation Of 3D Object Detector

Mohammed Kumail Abbas - 18743947
Gokul Gandikota - 23643918

1. Introduction

This task tests the quality of detected objects by checking how well the Lidar point cloud data fits the 3D Bounding Boxes. An evaluation method calculates the number of LiDAR points inside and outside each detected box, giving a clear and numerical result for the analysis. This even checks how reliable and accurate the detector is in real situations, such as sensor noise and distance. The results help to improve object detection methods and support better performance in multisensor fusion systems.

2. Methodology

2.1 Image Segmentation and Car Detection using YOLOv8

In this project, the YOLOv8 segmentation model was used to detect and highlight cars in images from the KITTI-360 dataset [2]. Each image was processed using the pre-trained model yolov8-segl.pt, which returned object masks and class IDs. The output was filtered to keep only the class ID 2[3, 1], which represents cars.
For every detected car, the mask was resized to fit the original image size. A random color was assigned to each mask. These colored masks were added to a blank image as well as overlaid onto the origional image for further process.

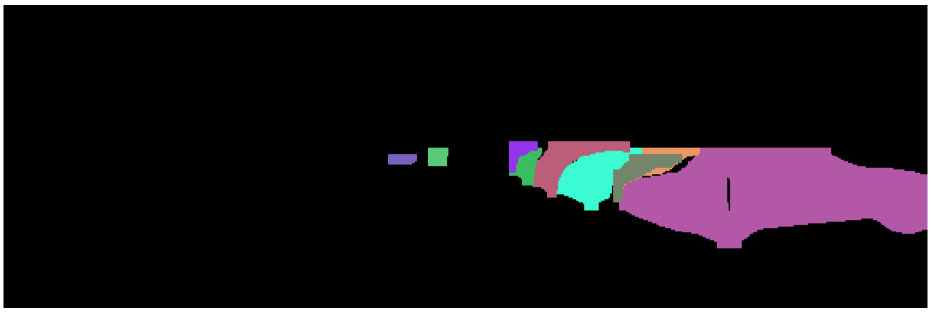

Figure 1: Masked image on Blank image

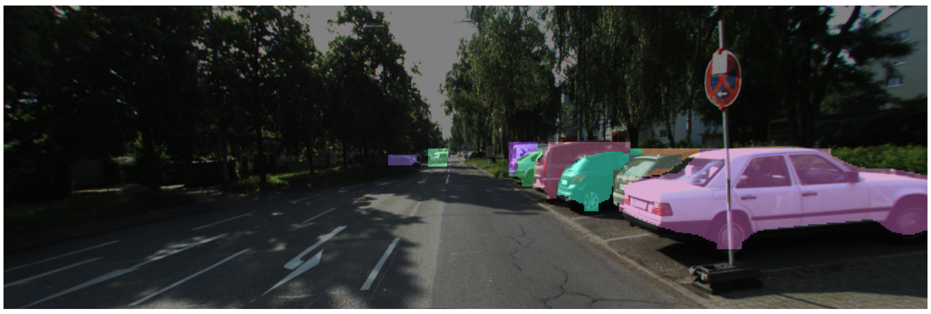

Figure 2: Masked image overlaid on original image

2.2 Projection of 3D LiDAR Point Clouds onto Segmented 2D Images

To overlay LiDAR point clouds onto segmented 2D images[4], a series of geometric transformations such as calibrated extrinsic and intrinsic parameters were used. Below is the mathematical pipeline that aligns with the implementation.

1. **Homogeneous Conversion:** Each raw LiDAR point $\mathbf{P}_{\text{LiDAR}} \in \mathbb{R}^{N\times 3}$ was extended to homogeneous coordinates:

$$\mathbf{P}_{\text{LiDAR\_h}} = \begin{bmatrix} X_L & Y_L & Z_L & 1 \end{bmatrix}^T \in \mathbb{R}^{N\times 4}$$

2. **Extrinsic Transformation (LiDAR to Camera):** The 3D LiDAR points were transformed to the camera's 3D coordinate frame using the extrinsic matrix:

$$\mathbf{P}_{\text{cam}} = \mathbf{P}_{\text{LiDAR\_h}} \cdot T_{\text{lidar\_cam}} \in \mathbb{R}^{N\times 4} \quad (T_{\text{lidar\_cam}} \in \mathbb{R}^{4\times 4})$$

   And then the spatial part were retained by excluding the last column:

$$\mathbf{P}_{\text{cam}} = \begin{bmatrix} X_C & Y_C & Z_C \end{bmatrix} \in \mathbb{R}^{N\times 3}$$

3. **Rectification:** Rectification was applied to correct distortions between the LiDAR and the camera:

$$\mathbf{P}_{\text{rect3D}} = \mathbf{P}_{\text{cam}} \cdot R_{\text{rect00}} \in \mathbb{R}^{N\times 3} \quad (R_{\text{rect}} \in \mathbb{R}^{3\times 3})$$

4. **Projection to Image Plane:** Projected the rectified 3D points onto the' 2D image space. $P_{\text{rect00}} \in \mathbb{R}^{3\times 4}$ maps these 3D homogeneous points into 2D image coordinates.

$$\mathbf{P}_{\text{rect3D\_h}} = \begin{bmatrix} \mathbf{P}_{\text{rect3D}} & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{N\times 4}$$

$$\mathbf{p}_{\text{img\_h}} = \mathbf{P}_{\text{rect3D\_h}} \cdot P_{\text{rect00}} \in \mathbb{R}^{N\times 3}$$

   Each row was of the form:

$$\mathbf{p}_{\text{img\_h}} = \begin{bmatrix} u' & v' & s \end{bmatrix}$$

   Where s is the depth information data.

5. **Normalization to Pixel Coordinates:** The final pixel coordinates were obtained by dividing by the depth (scaling factor s):

$$u = \frac{u'}{s}, \quad v = \frac{v'}{s}, \quad \mathbf{p}_{\text{img}} = \begin{bmatrix} u & v \end{bmatrix} \in \mathbb{R}^{N \times 2}$$

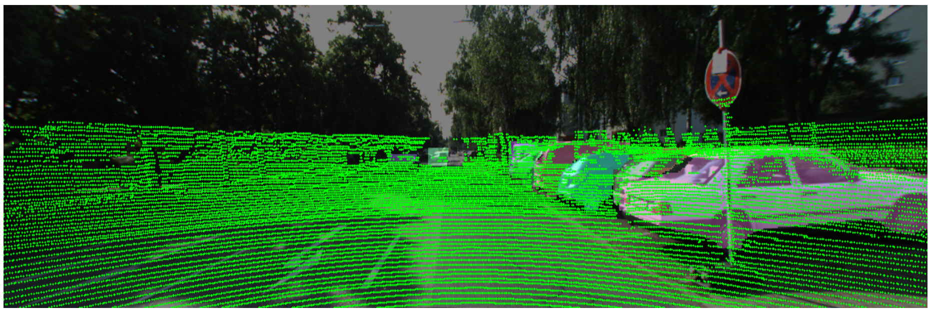
Figure 3: Lidar overlaid image on 2d image

6. **Perform the projection onto 2D masked image:** For better visualisation and filtering of the background, the 3d lidar points were overlaid onto the 2d masked image.

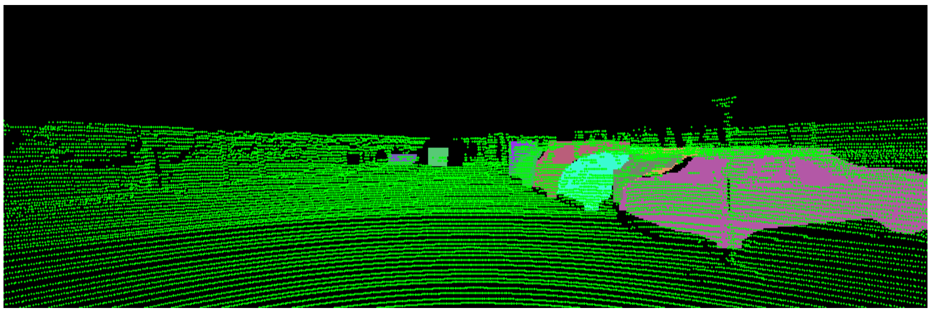
Figure 4: Lidar overlaid image on 2d segmented image

2.3 Visualization of Filtered LiDAR Points in 3D Using Open3D

In this step, each 3D LiDAR point was assigned a color by projecting it onto the corresponding image plane and sampling the pixel color at its projected location. This allowed semantic appearance from the 2D image to be transferred into the 3D space, which helped in visualizing segmented regions of cars.
Only those points whose projections fell within the image boundaries and were located in front of the camera were considered valid for color sampling. For each such valid projected point with pixel coordinates (u, v), the color at that location in the image was used. However, if the pixel intensity was too low (i.e., all RGB components below a threshold of 10), it was treated as background, and the corresponding 3D point was colored gray.
The final color assignment rule can be written as:

$$\text{color}_i = \begin{cases} \text{gray}, & \text{if img}[v, u] \approx \text{black} \\ \text{img}[v, u], & \text{otherwise} \end{cases}$$

Here, the i-th LiDAR point corresponded to the projected pixel (u, v), and its 3D representation inherited the pixel's appearance from the image. Points projected outside the image bounds or behind the camera were either skipped or assigned a default gray color.

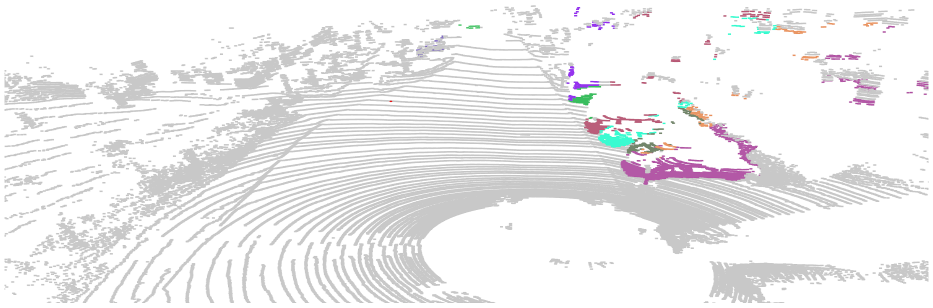

Figure 5: Visualization of Segmented Car LiDAR Points

2.4 Integration and Visualization of 3D Bounding Boxes with Filtered LiDAR Points

In this step, the colored 3D LiDAR point cloud was taken and matched with the 3D bounding boxes from the ground truth (given in JSON file). The goal was to keep only those boxes that actually contained some colored points and ignore the rest.

1. Read and Prepared Point Cloud

The 3D point cloud file (`.ply`) was loaded using Open3D. If any point had no color, a default light gray color was assigned using:

```
light_gray_rgb = [200/255, 200/255, 200/255]
```

2. Loaded the Bounding Boxes

The 3D bounding boxes were read from the JSON file. Each box had 8 corner points in the camera coordinate system. These were converted to the LiDAR coordinate system using the transformation matrix ($T_{\text{cam\_to\_lidar}}$).

3. Created Box Edges for Visualization

For each box, dense edge points were generated using the 8 corners. `numpy.linspace` was used to create 100 points per edge, which helped display the box properly in 3D.

4. Attached All Boxes to the Point Cloud

At this stage, all boxes were added to the point cloud regardless of whether they contained colored points or not.

5. Finding Which Color Belonged to Which Box

For each unique color in the LiDAR point cloud, the following steps were carried out:

- For each color, the number of its points that lay inside each box was counted.
- To check if a point was inside a box, its coordinates were compared with the box bounds:

$$\text{inside} = (\text{point} \geq \text{box\_min}) \wedge (\text{point} \leq \text{box\_max}) \tag{1}$$

- Here,
  - **box_min** = minimum value of the 8 corners of the box in x, y, z directions.
  - **box_max** = maximum value of the 8 corners of the box in x, y, z directions.

- These were calculated using:

```
box_min = np.min(corners, axis=0)
box_max = np.max(corners, axis=0)
```

- Afterward, the box with the maximum number of color points inside was selected.
- If two boxes had the same number of maximum points (tie), that color was skipped.

6. Assigned Each Color to Only One Box

Each color was assigned to only one box that had the highest number of points. Even if some of its points were inside other boxes, they were ignored. This ensured each color was fixed to one car (box).

7. Removed Unused Boxes

After matching, boxes that did not get any color assigned, were considered not useful and were removed from the final point cloud.

8. Saved the Final Point Cloud

The final point cloud was saved containing:

- All LiDAR points with their color.
- Only those 3D boxes which were matched to at least one color.

Outcome: The combined visualization of segmented car points and their corresponding 3D bounding boxes provides a comprehensive spatial context for validation and further processing.

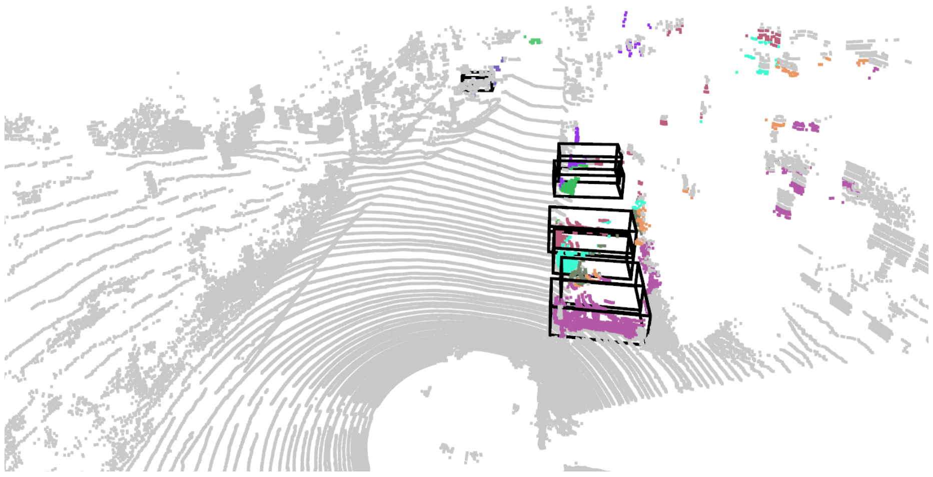

Figure 6: Visualization of Segmented Car LiDAR Points with 3D Bounding Boxes

3. Results and Analysis

3.1 Outcomes

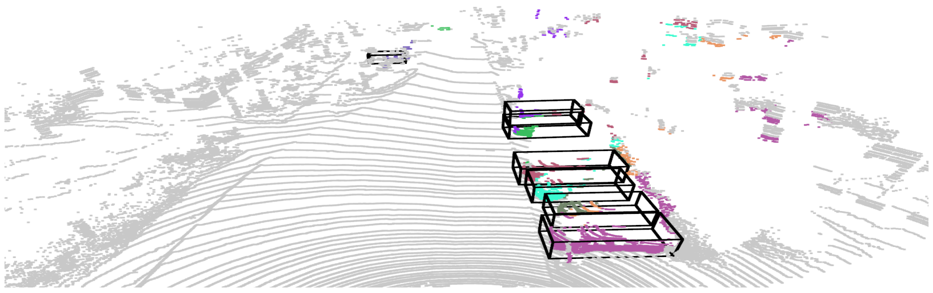

Figure 7: Image-1

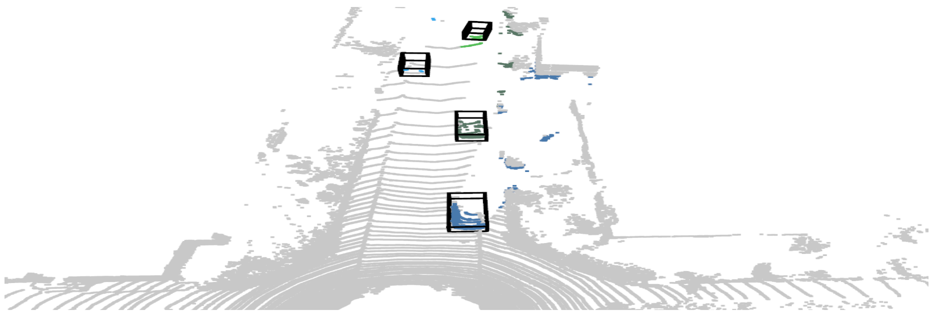

Figure 8: Image-2

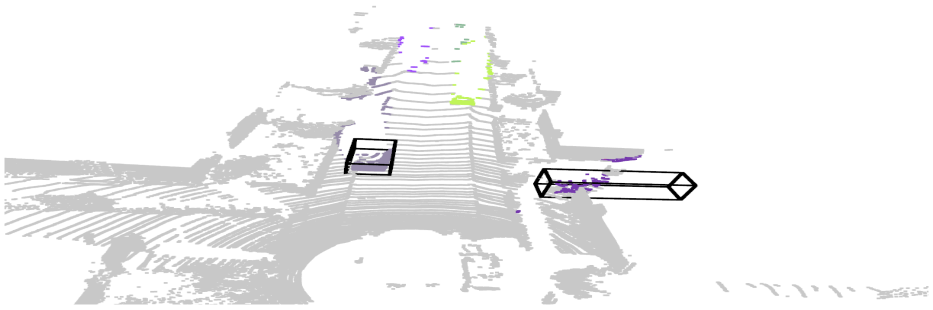

Figure 9: Image-3

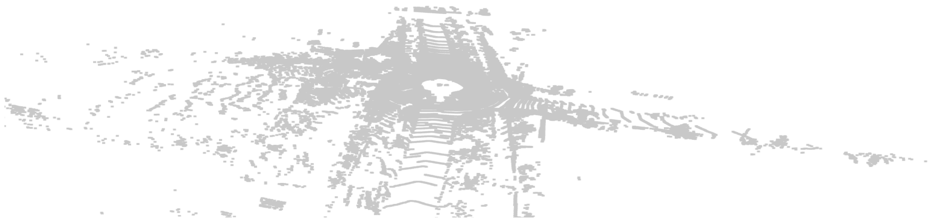

Figure 10: Image-4

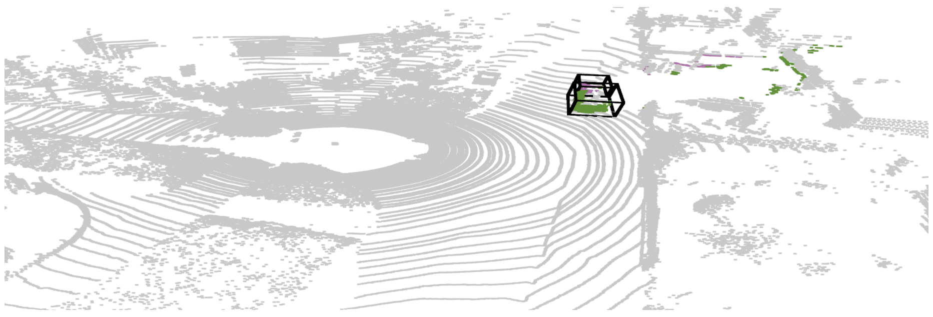
Figure 11: Image-5

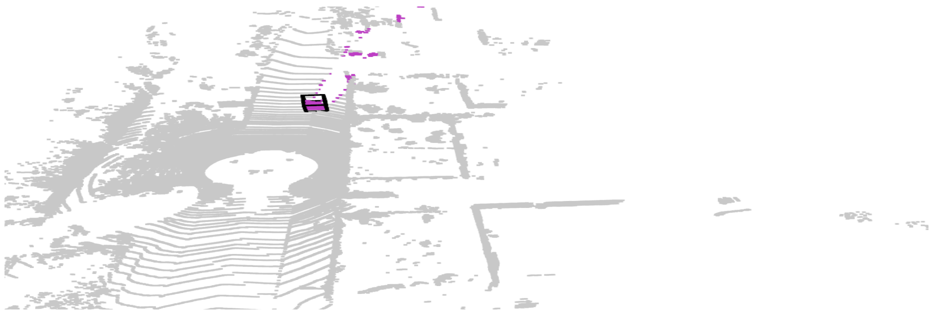
Figure 12: Image-6

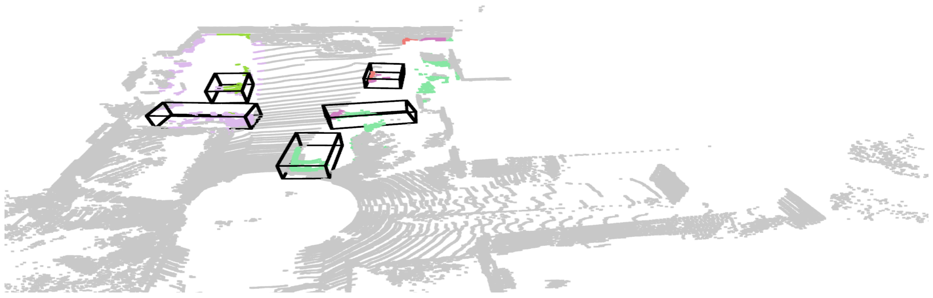
Figure 13: Image-7

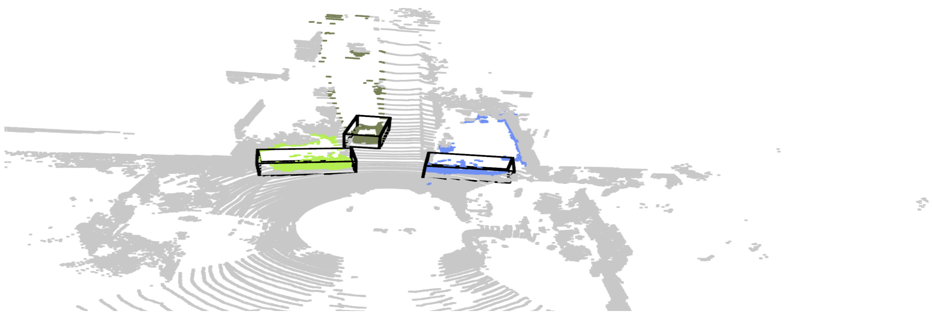
Figure 14: Image-8

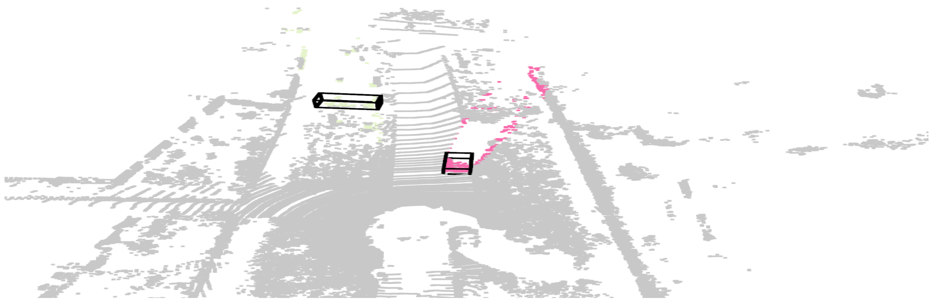
Figure 15: Image-9

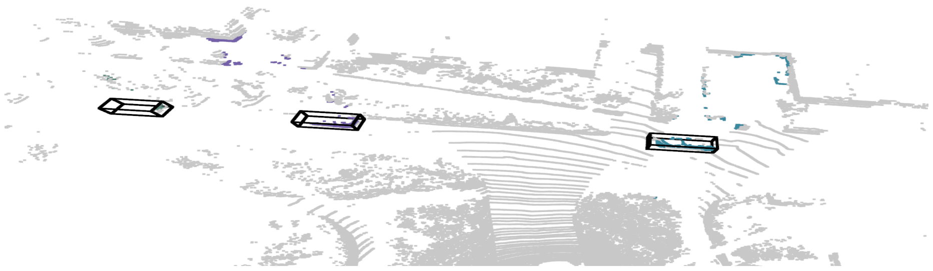
Figure 16: Image-10

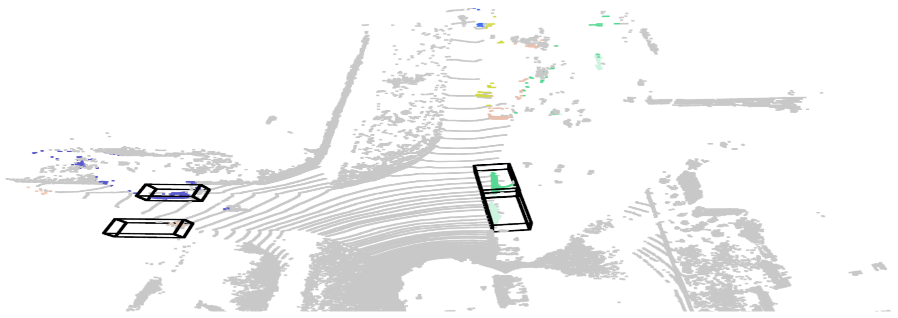
Figure 17: Image-11

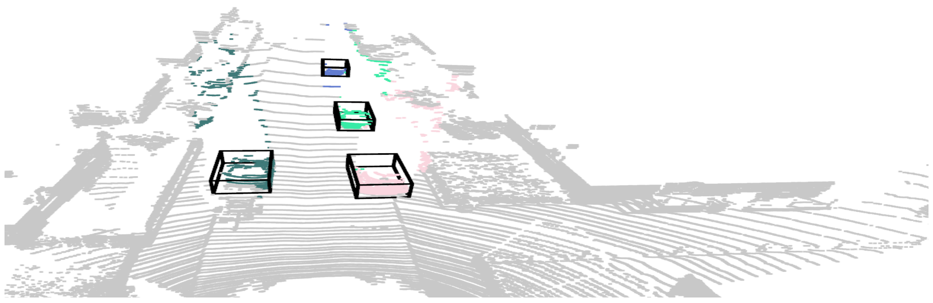
Figure 18: Image-12

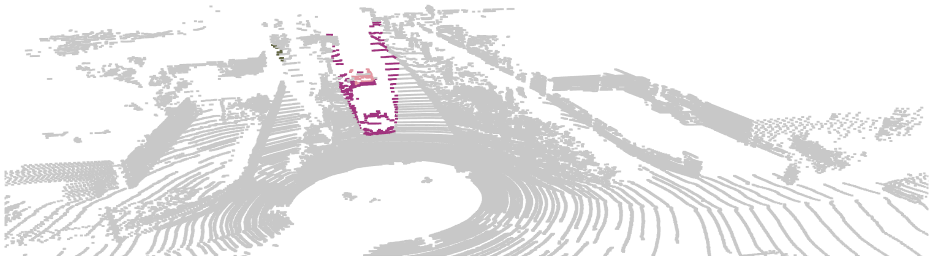
Figure 19: Image-13

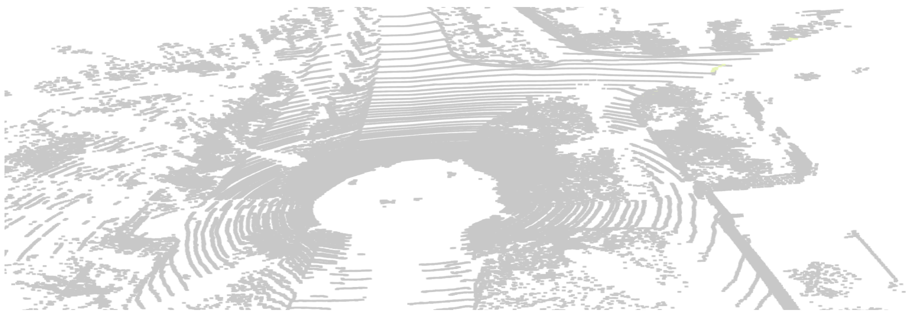
Figure 20: Image-14

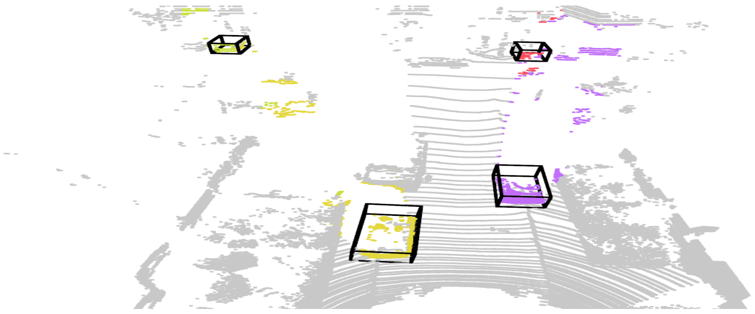
Figure 21: Image-15

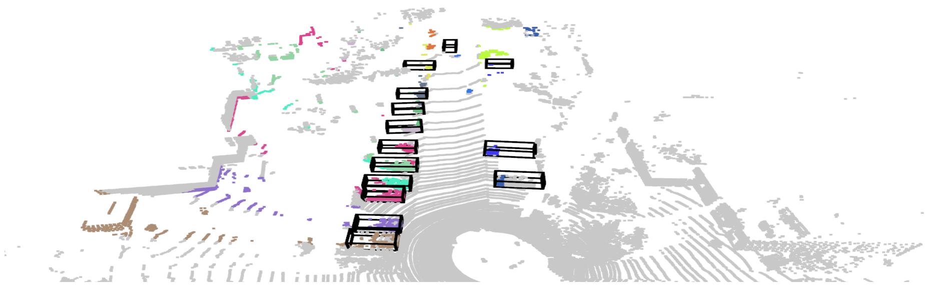
Figure 22: Image-16

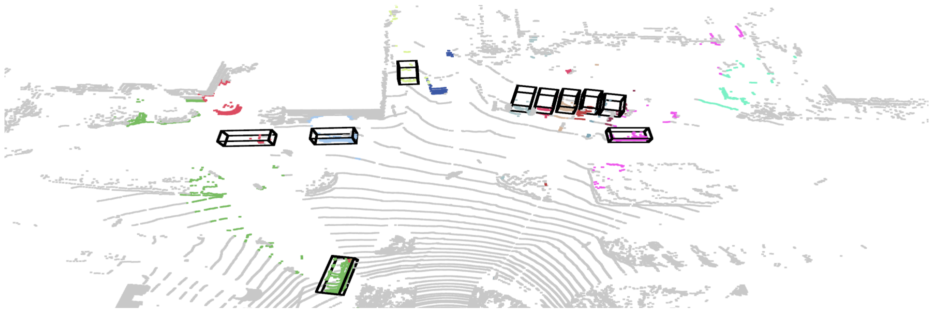
Figure 23: Image-17

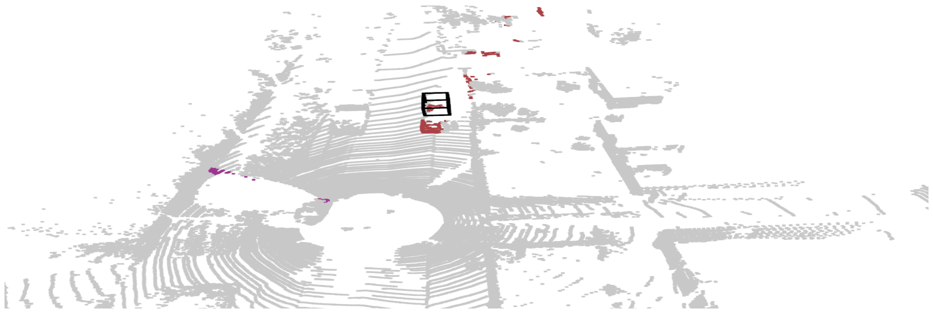
Figure 24: Image-18

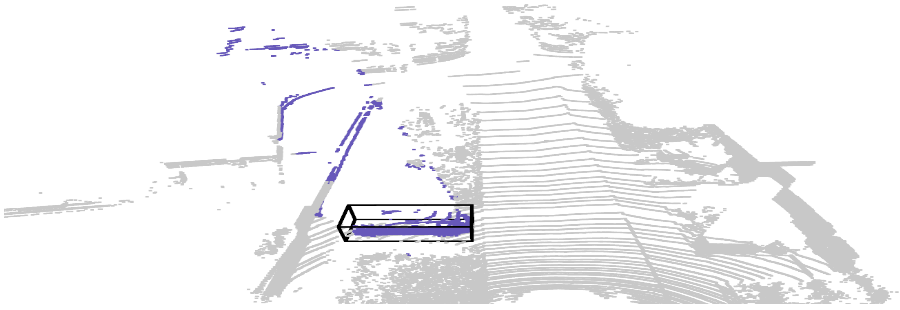
Figure 25: Image-19

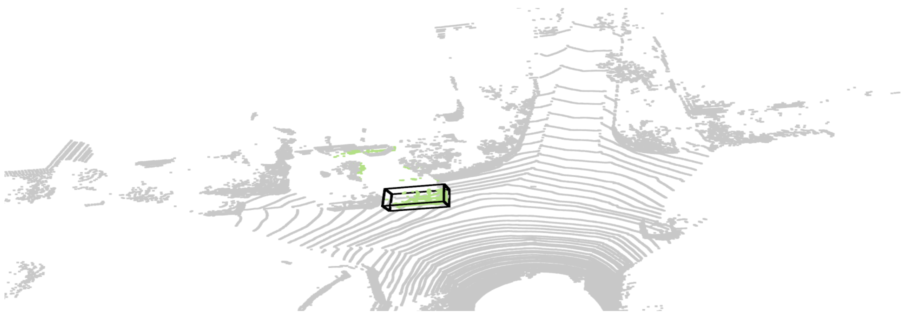
Figure 26: Image-20

4. Evaluation

4.1 Number of Points Inside and Outside

The following bar graph represents the average number of LiDAR points inside and outside of each car in an image across 20 images.

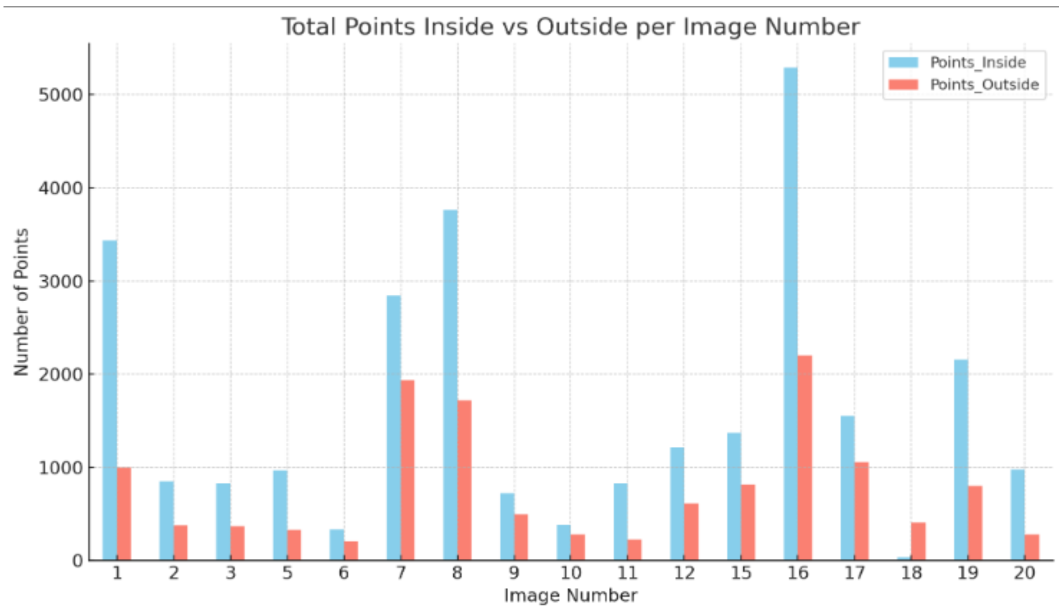

Figure 27: Graph showing average inside and outside points per image

4.2 Importance of Absolute vs. Relative Point Counts

This section aimed to understand whether the absolute number of LiDAR points inside bounding boxes or the relative ratio (inside vs. total points) was more meaningful for evaluating car detection from 3D LiDAR data.
As observed in the graph, some frames like Frame 16 and Frame 8 had very high absolute counts inside bounding boxes, which might have indicated strong detection performance. However, frames like Frame 6 and Frame 8 also showed high numbers but appeared less consistent in terms of distribution.

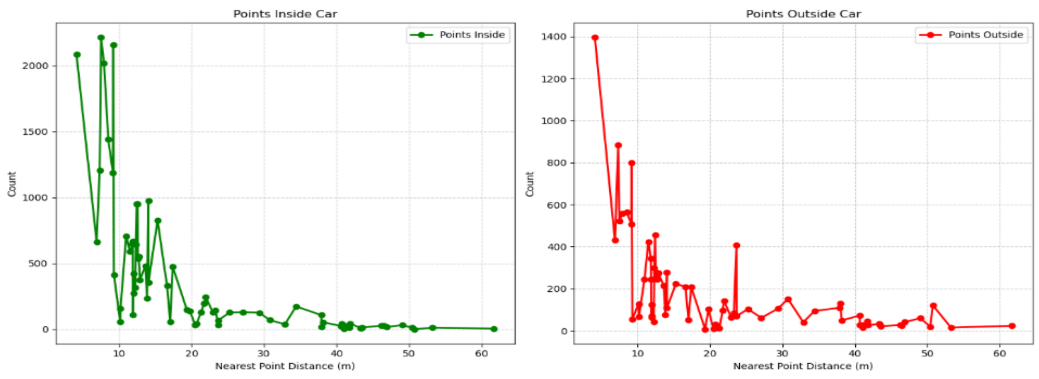

Figure 28: Graph showing inside and outside points per image

Even though the absolute number of points inside the bounding boxes had a strong impact on determining the car detection, the points outside the boxes also showed similar behavior depending on the distance. When the cars were closer to the LiDAR sensor, the points inside and outside appeared to be more equally spread.

The absolute count of points inside a box showed how well the segmented area matched with the bounding box. On the other hand, the relative proportion between inside and outside points gave a better understanding in situations where both values were high. For example, frames like Frame 16 with mostly inside points suggested high confidence, but frames like Frame 7 or 17 with high values on both sides needed closer inspection using ratios.
After the the analysis, the following relative ratio formula was used:

$$\text{Relative Ratio} = \frac{\text{Points Inside}}{\text{Points Inside} + \text{Points Outside}}$$

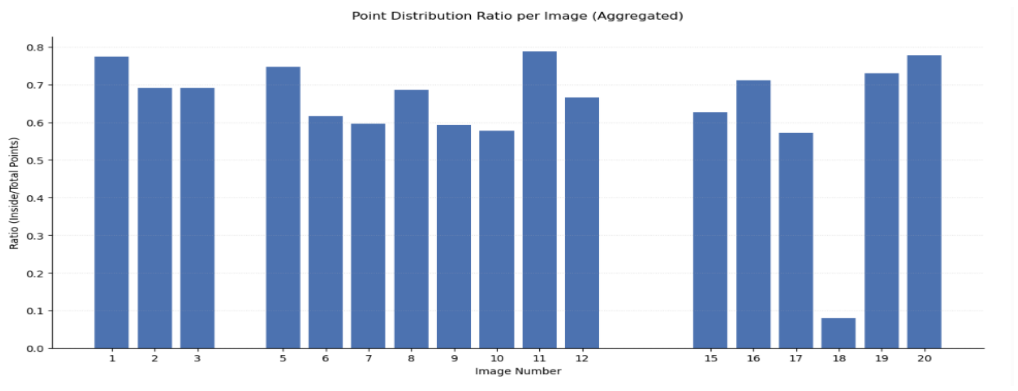

Figure 29: Graph showing relative point ratios per image

4.3 Which Detected Object Can Be Considered a Correct Detection?

As seen in the previous figure, the relative point ratio was more stable across different images, mostly staying between 0.6 and 0.8[5], even when the total number of points varied a lot.
This showed that bounding boxes captured relevant points well, regardless of the total number of points, which could change due to distance, occlusion, or LiDAR quality.
By using a threshold on the relative ratio, it was assumed that any ratio above 0.5 indicated a true positive detection of the car.

Anomaly Detections

The table below shows cars that had anomaly detections:

Table 1: LiDAR Points and Car Segmentation Analysis

Image_Num	Car_RGB	Points_Inside	Points_Outside	Nearest_Point_Distance	Relative_Points_Ratio	Sum	TP or FN
2	55-170-235	4	16	41.07	0.20	20.00	FN
2	82-188-85	13	15	53.25	0.46	28.00	FN
10	93-134-123	11	32	43.33	0.26	43.00	FN
15	200-223-57	32	60	49.04	0.35	92.00	FN
16	185-250-58	1	121	50.74	0.01	122.00	FN
16	104-193-131	38	40	32.87	0.49	78.00	FN
16	222-232-88	13	18	50.40	0.42	31.00	FN
16	222-118-61	6	23	61.65	0.21	29.00	FN
17	219-243-148	18	41	46.90	0.31	59.00	FN
17	220-74-96	17	131	37.95	0.11	148.00	FN
17	168-194-197	25	72	40.63	0.26	97.00	FN
17	138-51-82	9	27	43.41	0.25	36.00	FN
17	124-163-179	14	21	43.55	0.40	35.00	FN
17	184-62-62	15	44	41.70	0.25	59.00	FN
18	171-65-69	35	407	23.65	0.08	442.00	FN

Considering the distance of each detection, only three cars were within 40 meters but still got classified as false negatives.

These errors could be due to following reason listed below:
1. Multiple Cars Close to Each Other
In cases where cars were lined up or very close to each other (like in **Image 16**), it became harder to correctly assign LiDAR points. Some points from one car might have ended up inside the bounding box of another, especially when cars visually overlapped in the image.
2. Surface Materials of Cars
In Image 18, many LiDAR points were found far outside the car. This could have happened because of the materials used on the car's surface. Cars often have glass windows, shiny metals, and plastics that reflect LiDAR beams in unusual ways. Some materials, like glass, may cause ghost points (wrong depth), missing points (no reflection), or reflections from nearby objects. These effects could create points that appear wrongly outside the actual car shape.

References

[1] Dr. Professor Stefan Elser. Dataset: *KITTI 360 Selection*. Provided through the Moodle course platform at Hochschule Ravensburg-Weingarten University, 2024. Available at: `https://elearning.rwu.de/course/view.php?id=2344`

[2] YOLO Documentation. Available at: `https://docs.ultralytics.com/models/yolov8/#supported-tasks-and-modes`

[3] Dataset Class Usage for Car Detection. Available at: `https://docs.ultralytics.com/datasets/detect/coco/#applications`

[4] Projection of 3D LiDAR Point Clouds onto Segmented 2D Images: `https://www.secs.oakland.edu/~li4/research/student/JuanLi2015.pdf?utm_source=chatgpt.com`

[5] Vehicle localization by Lidar point corelation improved by change detection . Available at: `https://www.ikg.uni-hannover.de/fileadmin/ikg/Forschung/publications/Schlichting_Brenner_ISPRS_Prague.pdf`