Imperial College London

BEng Individual Project

Imperial College London

Department of Computing

# Robustness of 3D Object Detection in Adverse Weather Conditions

*Author:*
Andrei-Tudor Spiru

*Supervisor:*
Dr. Demetriou Soteris

*Second Marker:*
Prof. Emil Lupu

June 14, 2024

**Abstract**

LiDAR (Light Detection And Ranging) is a 3D sensor that has become a crucial part in the recent developments of Autonomous Vehicles (AVs) as it provides an accurate real-time picture of the environment. Significant research has been conducted into the effect of adverse conditions on LiDARs as well as on the security vulnerabilities inherent to these sensors. Recent works have shown that it is possible to spoof LiDAR return signals in order to create fake objects. Further work has also shown that it is possible to "hide" objects through a number of different strategies.

This project combines adversarial LiDAR attacks with rainy weather conditions to evaluate and potentially improve their effectiveness. Furthermore, the methodology of these attacks could be improved to be more effective in all weather conditions.

**Acknowledgements**

I would like to express my deepest gratitude to my supervisor, Dr. Demetriou Soteris, for his unwavering support and guidance throughout this project. His insights and feedback were instrumental in navigating the difficulties of this project.

I extend my appreciation to Prof. Emil Lupu, the second marker, for his constructive criticism. His feedback helped me understand the standards and quality required for this project, motivating me to give my best.

Furthermore, I would like to thank my family and friends for their continuous encouragement and understanding during the course of this project. Their support provided me with the motivation to persevere through what was a very challenging period of my life.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

LiDARs have experienced rapid innovation in the last decade leading to the development of precise and reliable sensors. They work by emitting laser pulses and measuring the reflections thus creating a detailed map of the surrounding 3D environment. LiDAR systems have seen use in many different applications but have become especially important for Autonomous Vehicles (AVs). In this context, aspects such as security concerns and reliability in different weather condition warrant a lot of research as they may have grave real-world implications.

Previous studies have shown that LiDAR sensors can be spoofed by intercepting emitted laser pulses and returning false reflections, thereby inducing fictitious points. While the appearance of a near-front object as described in Cao et al.[8] could cause an AV to stop and disrupt traffic, an Object Removal Attack (ORA) as described in Hau et al.[18] could lead to a major accident and even fatalities.

Prior work has studied the effects of adverse weather conditions on the performance of LiDARs, identifying it as a major concern. In an article titled "Self-Driving Cars Can Handle Neither Rain nor Sleet nor Snow" the author claims that "As things stand today, the driverless car of the future can't handle more than a dusting of snow"[33].

One of the main hypothesis of this work is that rain accentuates the effects of LiDAR attacks. As overall performance diminishes, an attack that only represented a mild security risk can have a catastrophic impact. Nonetheless, even if the difference is not significant, any increase in the effectiveness of the attacks can still be very detrimental seeing as the existing methodologies already show promising results.

Having said that, there exists a significant gap in the literature when it comes to analysing the effects of adverse weather on potential attack strategies. The majority of research on LiDAR object detection utilises the KITTI[13] dataset, which does not include any rainy scenarios. Most object detection algorithms as well as attack strategies are trained and evaluated on exclusively clear weather scenarios. We consider that this oversight has significant security implications and therefore more comprehensive research is required for this aspect of LiDAR object detection and attacks.

Seeing as our work aims to evaluate and exploit the weaknesses of LiDAR sensors, we can not ignore the ethical concerns related to our findings. These are discussed in Appendix A.

## 1.2 Contributions

The main aim of this project is to combine adversarial LiDAR attacks with rainy weather conditions in order to evaluate and potentially improve them. To this end, two preliminary objectives must be fulfilled:

- **Creation of a comprehensive dataset of LiDAR measurements from comparable clear and adverse weather scenarios** This goal can be achieved by leveraging simulators such as CARLA[10] or MAVS[15].
- **Development of a detection framework** In order to evaluate and improve existing methodology, we require a pipeline for applying object detection which is robust enough to be usable for different strategies.

Having satisfied these objectives, we can then bring the following two major contributions:

- **Evaluation of existing attacks in adverse weather** Using the newly created dataset, the goal is to evaluate existing attacks in rainy conditions as well as on different sensors and using different object detectors.
- **Development of two novel attack strategies** This work aims to leverage previous findings in order to develop new attack methodologies that outperform existing strategies and are framework independent.

# Chapter 2

# Background and related work

In this section we present the research already done on the two main components of this project: LiDAR attacks and the effects of adverse weather on LiDAR sensors. Although the literature on the subject is already very comprehensive it is nonetheless disjoint. Therefore, we separately look at the two mentioned aspects with the following goals in mind:

- First of all, we need to establish some context for LiDAR object detection with an emphasis on detection algorithms and the effect of rain on these type of sensors. Therefore, we look into related work that describes the effects of adverse weather on LiDARs to ascertain if there is a real impact on their robustness. Moreover, we also want to investigate existing ways of simulating such conditions as this might become crucial in supplying data for this work.
- Secondly, we compare different types of proposed attacks and evaluate them based on factors such as feasibility and impact. This is crucial for the project as it allows us to select a particular methodology of attacks that forms the backbone of our further contributions.

Lastly, we also provide some background on optimisation strategies as they play a significant role in the development of our novel attacks.

## 2.1 Background on LiDAR sensors

This work primarily focuses on scanning LiDARs, which operate by emitting multiple lasers across specific horizontal and vertical fields of view (FoV). After the laser is reflected and received, they measure the time-of-flight (ToF) to determine the distance to the object, which is then perceived as a point cloud. These can then be passed through a Deep Neural Network (such as [21]) to obtain 3D object detection.

There are two formulas that dictate the way LiDAR sensors work and become especially useful when discussing the effects of rain on LiDARs.

- Firstly, in a ToF LiDAR, the distance to the object is calculated as:

$$R = c * (\frac{\Delta t}{2})$$

  Where $R$ is resulting distance to the object, $c$ is the speed of light and $\Delta t$ is the time of flight[1].

- Secondly, intensity is another important aspect of a LiDAR measurement. It represents the the strength of the return power of the LiDAR echo and can be calculated as such[38]:

$$P_r = P_t \Omega \rho \eta_{sys} \eta_{atm}$$

$$\Omega = \frac{\pi D_{rec}^2}{4R^2}$$

Where $P_r$ is the returned laser energy (intensity), $P_t$ is transmitted power, $\rho$ is the reflectance of the target, $\Omega$ is the scattering steradian solid angle, $\eta_{sys}$ is the efficiency value of the optics system, $\eta_{atm}$ is the atmospheric attenuation, $D_{rec}$ is the diameter of the LiDAR receiver and $R$ is the distance to the target.

Another important aspect in the discussion about LiDAR attacks is the type of hardware being used. As described in [39], LiDARs can be roughly divided into first-generation LiDARs such as the Velodyne Puck (formerly known as the Velodyne-16)[2] and the Ultra Puck (formerly Velodyne-32)[3] and next-generation LiDARs. This distinction becomes crucial as the next-generation sensors are not only constructed on a single board but also provide some additional security features.

### 2.1.1 Effects of adverse weather on LiDAR

As mentioned before, rain can have a major effect on the robustness of LiDAR sensors because the presence of rain droplets at different densities can cause two main problems: attenuation and scattering.

- **Signal attenuation** Rain droplets cause signal attenuation by absorbing and scattering the laser pulses emitted by the LiDAR. Hasirlioug et al.[17] showed that for extreme rain rates there is a significant decrease in reflection intensity. This reduction diminishes the effective range of the LiDAR and also makes it more difficult to detect objects as intensity helps differentiate vehicles that usually have high reflectivity from roads. "To date, LIDAR intensity data have proven beneficial in data registration, feature extraction, classification, surface analysis, segmentation, and object detection and recognition, to name just a few examples."[20]

- **Scattering** The presence of rain droplets causes multiple scattering effects which can lead to false return signals and induce false positives. Although this can complicate data interpretation, the effect has been found to be negligible[11]

Goodin et al.[15] tackles the problem of evaluating the effects of rain on LiDAR by first providing a simplified mathematical model and then implementing the results into a simulator. To this end, they derive two easy to use mathematical formulas:

- Firstly, in order to calculate the relative intensity returned by LiDAR:

$$P_n(z) = \frac{\rho}{z^2} e^{-0.02R^{0.6}z}$$

  Where $z$ is the target distance $\rho$ is the back scattering coefficient of the target and $R$ is rainfall rate in mm/h. By using the $z_{max}$ which is usually included in LiDAR data sheets and for a 90% diffusely reflecting surface we can calculate the minimum detectable relative power estimate:

$$P_n^{min} = \frac{0.9}{\pi z_{max}^2}$$

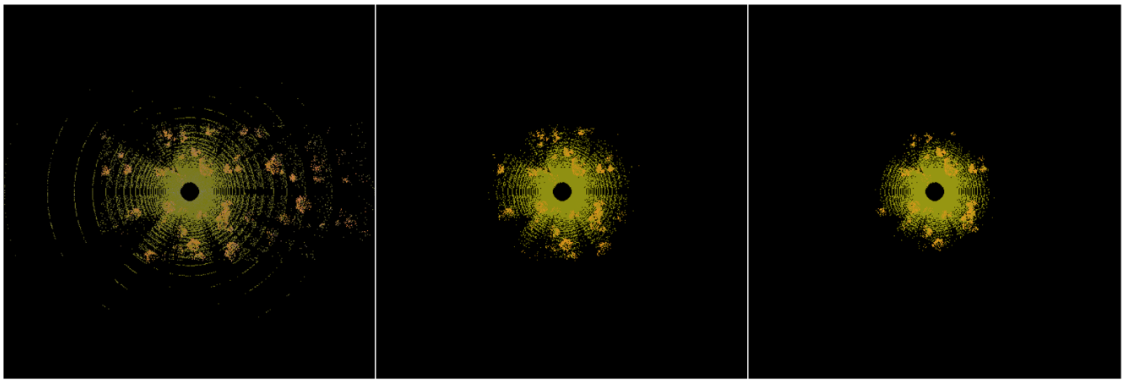
  Thus, with a few constants and only the rain rate, we can calculate the range reduction.

- Secondly, the normal distribution is used to model the noise introduced to range measurement.

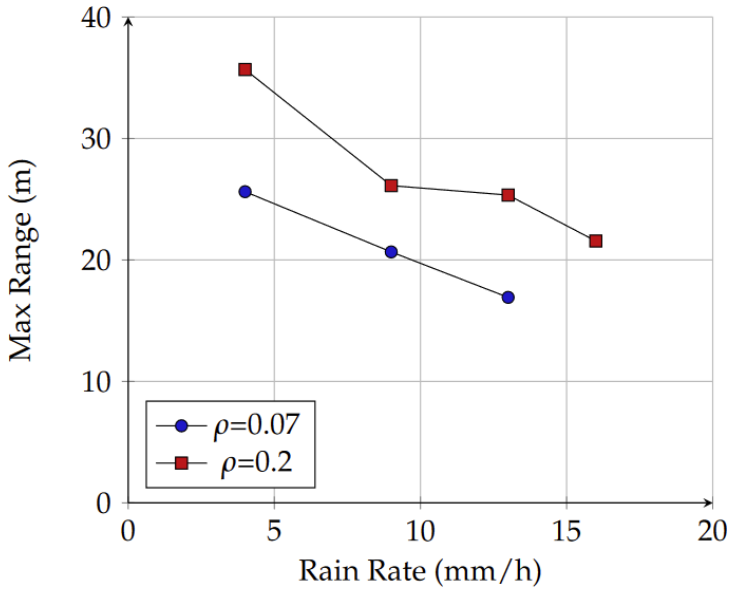
$$z' = z + \mathcal{N}(0, 0.02z(1 - e^{-R})^2)$$

  Where $z'$is the modified range. This equation has the property that the noise is zero when $R$ is zero and the variance increases by a maximum of 2%.

Goodin et al.[15] then integrates this rain model into a simulator known as Mississippi State University Autonomous Vehicle Simulator (MAVS) that accurately captures the physics of LiDAR 2.1.



(a) Visualisation of LiDAR range decrease at 0mm/h, 9mm/h, 17mm/h from [15]




(b) Decrease in max range of LiDAR as a function of rain rate from [15]

Figure 2.1: Effects of rain on LiDAR range from [15]

Yang et al.[38] in a more recent work also tackles this problem, creating a rain model for the CARLA simulator using which they then created a publicly available database. The research is mainly focused on two aspects critical to LiDAR perception in rainy conditions:

- Firstly, the effects of spray and splashes caused by the wheels of vehicles on a wet surface are thoroughly researched. This is receiving increasing attention in the context of AVs as especially in highway scenarios the spray created by a vehicle produces significant noise for LiDAR sensors. To this end, Yang et al.[38] introduces a dynamic spray model that takes into account multiple factors such as the speed of the vehicle, gravity, crosswinds and turbulent airflow at the rear of the vehicle. Furthermore, droplets annihilate when they collide with objects, get too far away from the sensor or have been suspended in air for more than 1.5s (in which case they are considered to have broken up in the air).

- Secondly, in the context of the CARLA simulator, calculating echo intensity is not feasible as it is impossible to obtain object reflectance and atmospheric attenuation. Therefore, Yang et al.[38] proposes a intensity prediction neural network that utilises a U-net[27] to estimate the object reflectance based on camera RGB data and LiDAR semantic label information. Combining this with LiDAR depth information, cloud point intensity can be generated.

The synthetic point clouds created this way are then used to augment the Waymo Dataset, showing that the simulation data closely approximates the real characteristics of the dataset.

### 2.1.2 LiDAR object detectors

3D Object detectors are a rapidly evolving area of research, having seen significant developments over the last few years. In the context of LiDARs, the process involves utilising a 3D point cloud to localise objects and determine their bounding boxes. Detection algorithms are an integral part of any work relating to LiDAR security necessitating significant context. Although they share the same goal, object detectors vary substantially in the way they approach this problem. In this section, we review three different state-of-the-art LiDAR object detectors.

#### SECOND

SECOND (Sparsely Embedded Convolutional Detection), introduced by Yan et al.[37], presents an efficient voxel-based method for 3D object detection. This approach utilizes sparse 3D convolutions to process the voxelized point cloud data, maintaining computational efficiency while achieving high accuracy.

Voxelisation involves splitting 3D space into a grid which is then used to group the raw points. This is necessary as it converts the unordered 3D point cloud into a data-structure that can then be further processed.

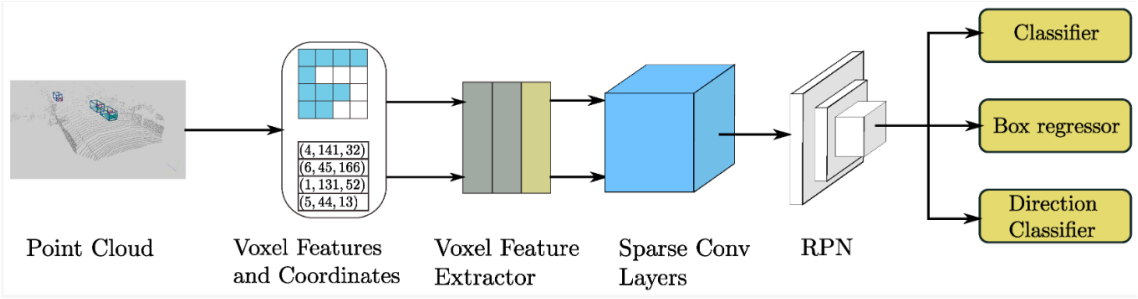



Figure 2.2: SECOND detection pipeline from [37]

One of the significant advantages of the SECOND architecture is that it utilises a sparse convolutional network to process the voxel grid, reducing computational costs. A three layer Region Proposal Network (RPN) is then used to generate the detections.

#### PointPillars

PointPillars, proposed by Lang et al.[21], introduces a novel method that organises raw point clouds into vertical columns (pillars) that can then be processed by a 2D convolutional network. This methodology allows PointPillars to leverage the computational efficiency of a 2D network, without sacrificing performance.

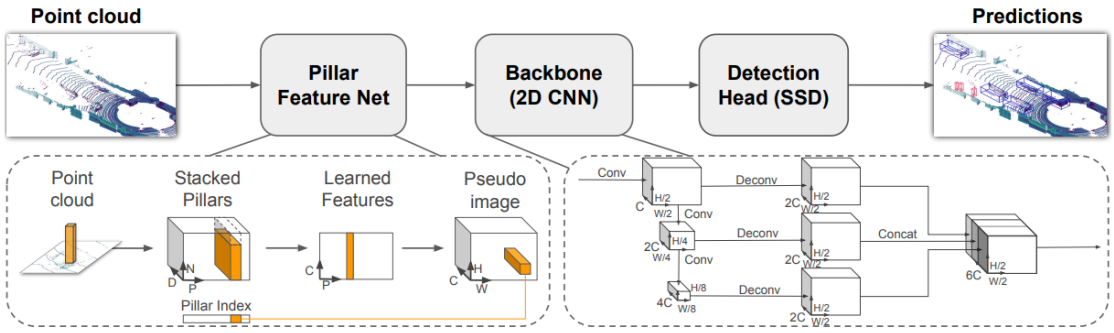



Figure 2.3: PointPillars detection pipeline from [21]

The PointPillars pipeline consists of 3 stages: "(1) A feature encoder network that converts a point cloud to a sparse pseudoimage; (2) a 2D convolutional backbone to process the pseudo-image into high-level representation; and (3) a detection head that detects and regresses 3D boxes."[21].

A significant advantage of this architecture is that PointPillars utilises end-to-end learning which allows the algorithm to utilise all the information in the point cloud and. Moreover, using pillars removes the necessity for manual adjustment of the bins in the vertical direction.

### PartA2

Part A2, introduced in Shi et al.[30], builds upon the PointRCNN framework and introduces a two-stage detection network that incorporates part-aware and part-aggregation mechanisms. The main advantage of PartA2 is that it exploits all 3D information available in the point cloud by utilising a network that can predict intra-object part locations even when they are occluded.

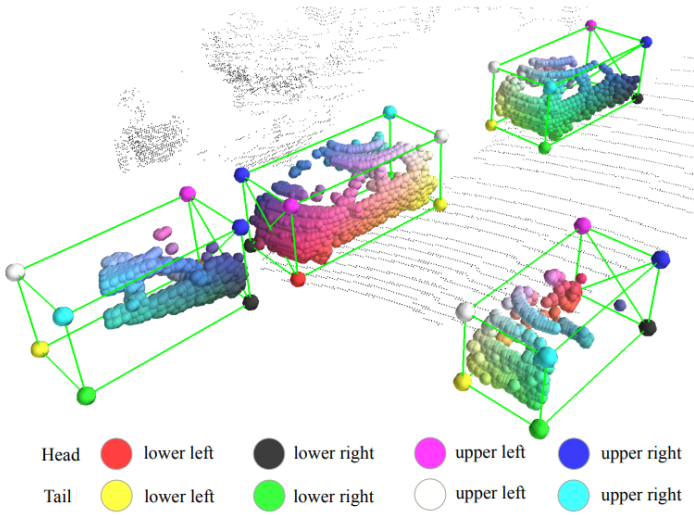



Figure 2.4: Visualisation of intra-object part detection from [30]

The overall structure of the detector's architecture can therefore be split into two stages:

- **Part Aware Stage:** This stage predicts high-quality 3D proposals and intra-object part locations using part supervisions derived from 3D ground-truth annotations
- **Part-Aggregation Stage:** This stage utilises a novel Region of Interest (RoI) aware pooling module to aggregate the results from the previous stage and then refines the bounding boxes based on the part features.

The two-stage approach improves the quality of 3D proposals and overall detection accuracy, achieving state-of-the-art results on the KITTI benchmark[13].

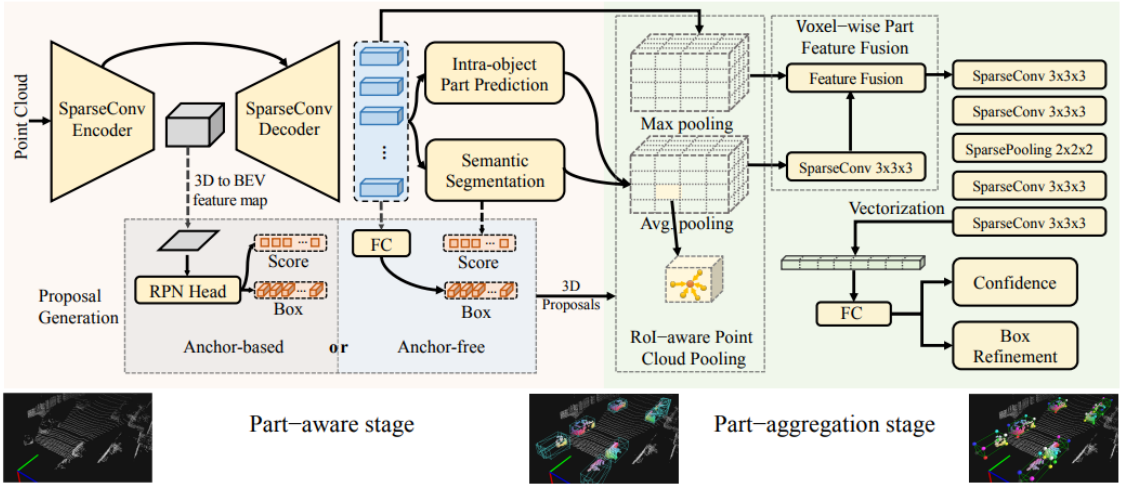



Figure 2.5: PartA2 architecture from [30]

**OpenPCDet**

OpenPCDet [35] is an open source object detection library that includes nearly all state-of-the-art detectors. The most significant advantage of this code base is that it provides a unified environment where multiple object detectors can be tested on data of the same format. Thus, by integrating models with very different architectures into the same framework, OpenPCDet facilitates efficient evaluation across multiple detectors.

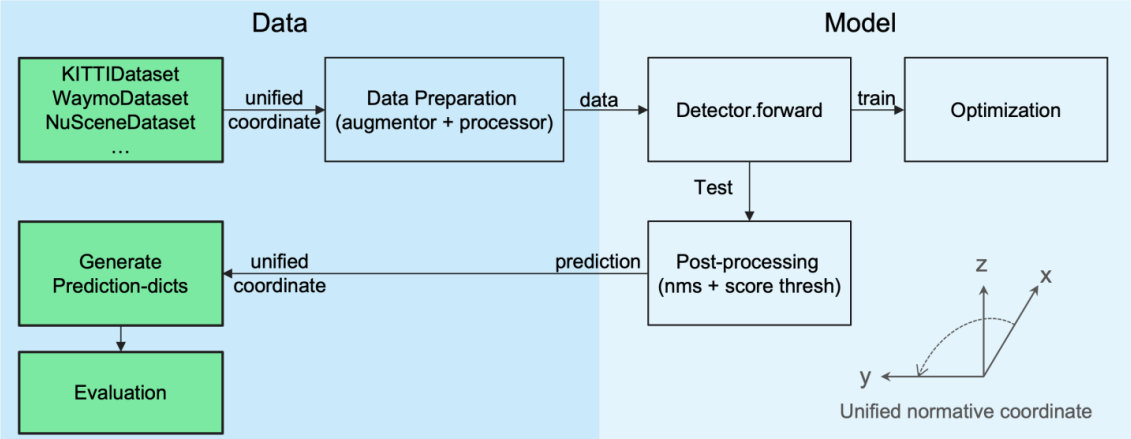



Figure 2.6: Dataset-model separation from [30]

Another crucial feature is that OpenPCDet supports multiple datasets, including KITTI[13], NuScenes[6], and Waymo Open Dataset[9], and can also be easily configured to work with custom data. This broad compatibility ensures that users can leverage a wide range of data sources for their projects. Furthermore, a "Model Zoo" is provided that includes a number of models trained on existing datasets.

## 2.2 LiDAR spoofing

Spoofing attacks for Time of Flight (ToF) LiDARs work by firing lasers back at the sensor, thereby disrupting the reflection measurements. Therefore, by doing this, an attacker can add a number of spoofed points to the sensor's point cloud, which can be used in two major ways:

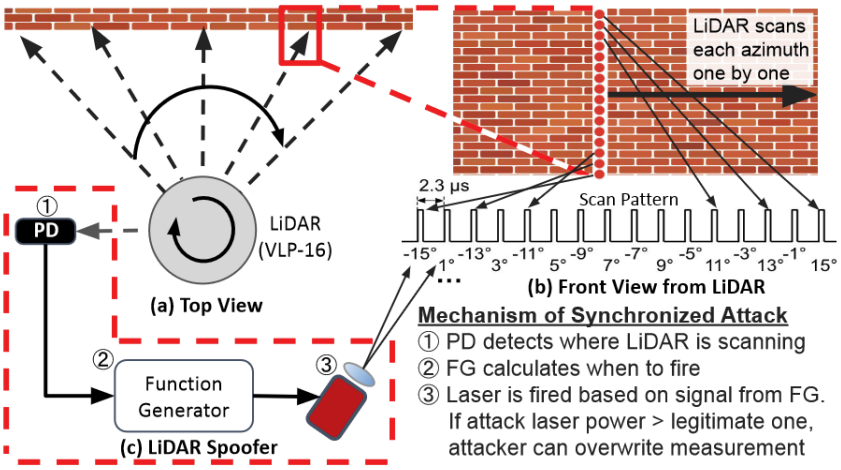
- The induced points can be used in order to mislead the detection algorithm about the existence of a fake object. This is known as object injection.

- The induced points can also be used to remove legitimate objects. This is known as object removal.

### 2.2.1 White-box attacks

White-box (or synchronized) attacks require knowledge about the LiDAR sensor as well as about the machine learning 3D object classifier used by the victim (only for certain attack methodologies [8]). This attack mechanism is detailed in Sato et al.[28], pointing out that both injection and removal attacks use the same methodology, the only difference being whether points are moved to a target location or to an undetectable area. As shown in Figure 2.7, the process involves three steps that ensure that the attacker knows where the LiDAR is scanning and can predict it's pattern. First, the Photodiode (PD) receives the legitimate lasers and has knowledge of where the LiDAR is firing. Secondly, the Function Generator (FG) uses this information to plan where to fire the lasers based on the pattern of the LiDAR. Lastly, malicious signals are sent out. As is discussed in Section 2.2.1 this is problematic when considering the capabilities of next-gen LiDARs.

Figure 2.7: Illustration of synchronised attacks on Velodyne-16 from [28]




### Injection attacks

There have been numerous works describing synchronised injection attacks, all building on previous research and bringing sizeable improvements. However, as was outlined before, all of them follow roughly the same methodology and have the same strengths and weaknesses. As such, we only focus on a limited number of strategies as that is enough to perform a sufficient analysis.

The findings in Shin et al.[31] represent a major breakthrough and are the foundation of all further white-box injection attacks, as among other contributions the work proved it possible to spoof up to 10 fake points at different distances, even closer to the spoofer location[8]. Shin et al.[31] also identifies a few of the problems with this type of attack:

- **Aiming problems:** As we are focusing on LiDARs in AV scenarios, tracking the sensor itself can prove to be quite difficult. To mitigate this, the author suggests the attacker mounts the tool on another vehicle and follows the victim, therefore reducing the relative speed to zero. This solution has become the standard modus operandi for subsequent works on LiDAR attacks.

- **Limited number of induced points:** As previously described, this work demonstrates the capability to inject only ten fake points in a 2°wide range. Although insignificant at first glance, at 55m distance, this corresponds to an object 1.9m wide. In spite of the fact that this can be considered dangerous at highway speeds, it is relatively harmless when it comes to low speed city scenarios. Therefore, the author suggested increasing the number of fake points as a crucial improvement to this attack.

Cao et al.[8] built upon the findings in Shin et al.[31] and brought some major improvements. First of all, by modifying the experimental setup they managed to increase the number of spoofed points to a theoretical maximum of 100, although it is stated that 60 is a more realistic estimate. This was achieved with a similar design to the previous work but using improved electronics and calibration (Figure 2.8).

Secondly, Cao et al.[8] finds that blindly firing the allocated amount of fake points is not impactful after the machine-learning object detection step, hence requiring a more sophisticated approach. This underlines one of the major limitations of white-box attacks, requiring knowledge about the perception model used by the victim. In this case, the perception pipeline used by Baidu Apollo was utilised, which consists of 3 steps 2.9:

- **Step 1: Pre processing** In this step, the raw LiDAR data in the form of a four dimensional vector is transformed into an absolute coordinate system. Afterwards, the Region of Interest (RoI) is selected, disregarding irrelevant parts of the 3D map. Lastly, a feature is generated that represents the input to the machine learning model

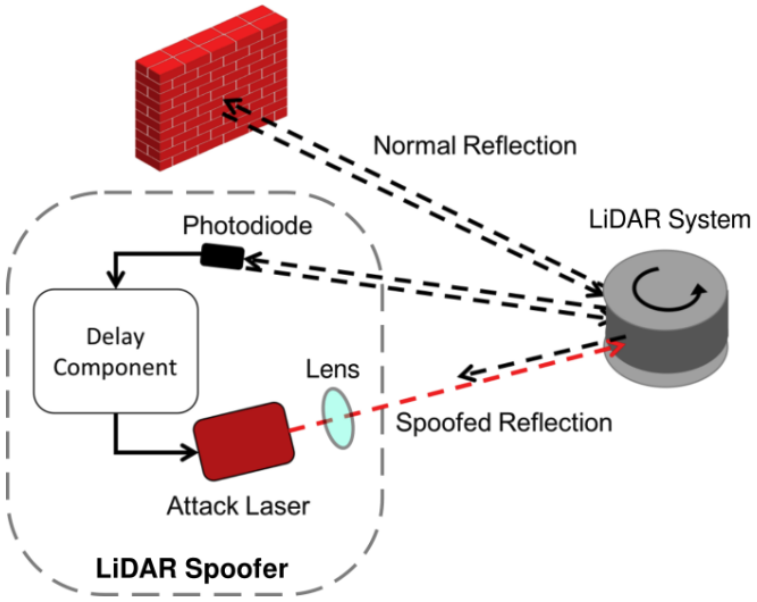



Figure 2.8: Illustration of spoofing attack setup from[8]

- **Step2: Machine learning model** For this part, a Deep Neural Network is used that takes in a feature matrix and as output produces a set of cells each being labeled with the probability of being part of an object.
- **Step2: Post processing** Lastly, cells are clustered and using results from the previous step bounding boxes are formed representing objects. The tracking component associates boxes in sequential frames, thus being able to track the movement of an object over time.

This entire process can prove to be very hard to circumvent for a simple approach and therefore Cao et al.[8] proposes an adversarial machine learning method called Adv-LiDAR. This combines input perturbation together with a novel sampling approach to achieve significantly better results than previous works. However, it still has significant drawbacks that were discussed before, mainly real world concerns such as accurately aiming at the sensor in a road environment and also the dependency on the understanding of the perception pipeline. Although the methodology could be translated to different object detection algorithms, that would not only require significant further research but also would not be of much help in a real situation unless the attacker has white-box access.

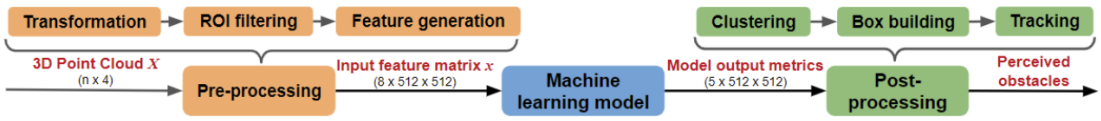



Figure 2.9: Data processing pipeline for Baidu Apollo from [8]

### Removal attacks

In general, removal attacks have a major advantage over injection attacks when it comes to their real world implication. While an injection attack might slow down traffic or might cause an AV to suddenly brake and even injure passengers, removal attacks can lead to very serious accidents and even fatalities. In a scenario such as highway traffic, "removing" a vehicle in front of an AV could cause a series of subsequent collisions and a pile-up involving many victims.

As discussed before, when it comes to synchronised attacks, removal and injection methods are quite similar. Cao et al.[7] in a more recent work that explores this type of LiDAR attack, brings significant improvements compared to the existing literature. Firstly, the number of spoofed points

was increased from 100 to up to 4000. This greatly improved the overall effect of the attack, allowing it to target a much larger area and also makes it possible for a roadside attacker to remove cloud points from in front of an AV (Figure 2.10).

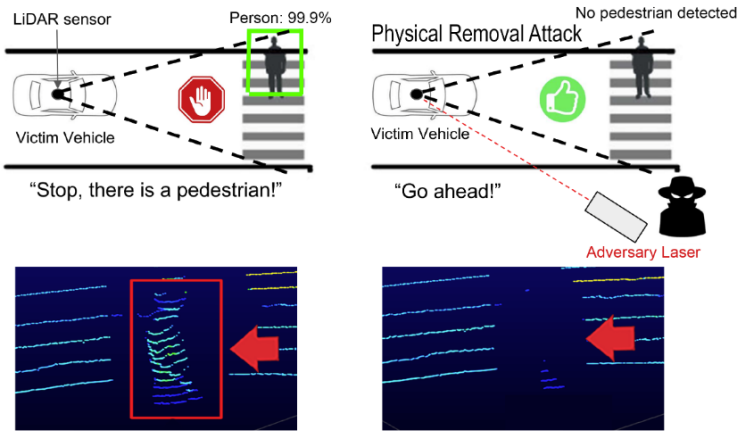



Figure 2.10: Overview of roadside object removal attack from [7]

The main idea behind the attacks in [7] is to exploit a vulnerability in the LiDAR sensor's operational range. Normal commercial 3D LiDARs have a minimum horizontal range under which detection of echoes near the sensor is not guaranteed. In short, LiDARs can still receive echoes under this range but the results might be inaccurate. Therefore, manufacturers implement a filter that discards these points. Cao et al. [7] takes advantage of this by injecting fake points below the LiDAR's Minimum Operational Threshold (MOT) in front of the objects that should be concealed 2.11.

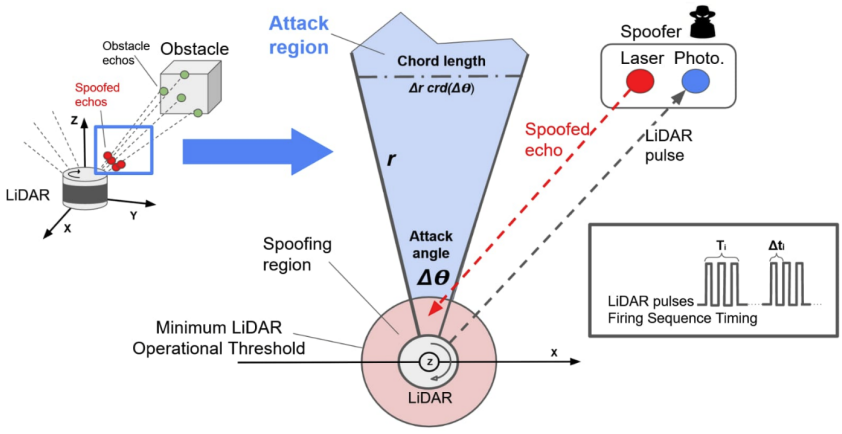



Figure 2.11: Illustration of synchronised removal attack from [7]

Overall, this type of attack presents a lot of advantages compared to the presented injection attacks:

- Firstly, it is more potent owing to the increased amount of ficticious points that can be induced. Although this change could be translated to the injection attack as well, it still presents other further advantages.
- Secondly, because the attack happens at the LiDAR firmware level and not in the perception pipeline it is considerably more versatile. In a real scenario it is significantly easier to differentiate types of LiDAR than to know what perception model is in use.

- Lastly, the attack is overall simpler and thus easier to recreate and improve on.

#### Problems with white-box attacks

Sato et al.[28] conducted an excellent analysis of existing attacks, underlining unproven assumptions and general problems faced by white-box attacks as well as proposes an alternative black-box attack which we discuss in Section 2.2.2. The first major problem is that the assumed Chosen Pattern Injection (CPI) capability is only feasible on the VLP-16. Furthermore, new security features such as timing randomisation of pulses mean that **synchronised attacks can no longer be applied to next-gen LiDAR**. The study also shows that pulse fingerprinting which is another new feature may not be as effective against adversarial attacks as it was mainly designed to prevent interference between multiple sensors.

Taking all of this into consideration, although this is a significant setback to the future prospects of synchronised attacks, we must consider that first-gen LiDARs are already in widespread use and are very expensive to replace. Thus, we can still assume that such attacks will be a valid threat for years to come.

### 2.2.2 Black-box attacks

Black-box attacks differ from white-box attacks in that they do not require any knowledge about the victim LiDAR sensor or 3D-classifier. This approach typically represents a compromise between robustness, effectiveness, and ease of use.

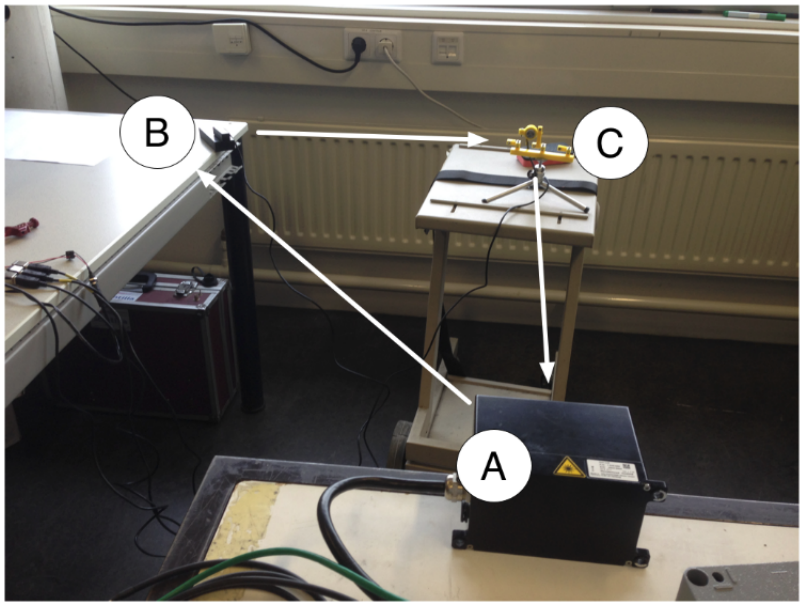



Figure 2.12: Setup used in Petit et al.[25]

#### Relay attacks

Relay attacks as presented in Petit et al.[25] consist of relaying the original signal from another position to create fake echoes. The setup for the attack is a bit more complicated, requiring two transreceivers (B and C in Figure 2.12)

In Figure 2.12, both transreceivers are positioned one meter away from each other, this however not being a requirement. Because LiDAR signals reflect, a direct line of sight is not necessary to perform a relay attack. Considering the setup, the author suggests that "a relay attack is most likely to happen from the roadside, where the attacker would receive LiDAR signals from vehicles and relay them to another vehicle located at a different location."

Petit et al. [25] leverages this methodology in order to achieve an object injection attack with

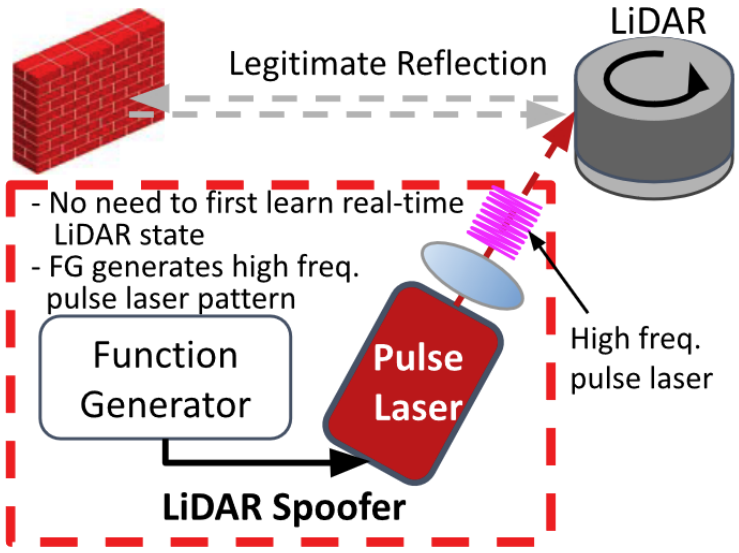



Figure 2.13: Overview of HFR attacks from [28]

reasonable success. The point injection is successful in injecting up to 200 points [28] but is limited by a few drawbacks. The main limitation is that this setup only allows for point injection at medium to large distances, for example it being possible to inject a copy of a wall at approximately 40 meters. This is important in the context of an injection attack as at low speeds creating a fake object at 40 meters distance may have a limited effect. Furthermore, the attack has a limited range of up to 100 meters which is significant considering the roadside component of the setup. To this end, the author suggests that multiple photodetectors might be necessary therefore increasing the cost.

#### High-Frequency-Removal (HFR) attacks

High-Frequency-Removal attacks are presented in Sato et al.[28] as an alternative to white-box attacks that do not function on next-gen LiDARs. This type of attack works by firing a high frequency pulse laser at the victim. Crucially, the frequency used by the attacker has to be bigger that the one used by the victim LiDAR. Another important aspect to consider is that because this is not a synchronised attack, the legitimate points are to random positions. Nonetheless, this attack was proven to be extremely effective, being able to remove more than 5000 points in a 10 $m^2$ area.

HFR attacks present a multitude of advantages such as effectiveness, robustness and ease of use. However, this type of attack also has a few drawbacks:

- Firstly, although the area that of attack can be controlled, it is not possible to control where the real point clouds are moved by this strategy of attack
- Secondly, this is in essence a "brute-force" approach meaning that just as white-box attacks have been nullified by next-gen LiDARs, so could this approach by just raising the frequency at which the LiDAR operates. This leads to an "arms race" scenario between attackers and LiDAR manufacturers.
- Lastly, because of it's design this methodology of attack could be detected by the victim that could then take counter measures. Shin et al.[31] suggested that in such cases the AV could perhaps abandon sensor output in the attacked area and try to safely move to the roadside. Even though this would stop the AV, this outcome would still be preferable to an accident.

### 2.2.3 Object Removal attacks (ORA)

ORAs[18] are a very promising new methodology of LiDAR attack that aims to exploit the fact that LiDARs deployed on AVs operate in the Strongest Return Mode, meaning that for each ray direction only the strongest intensity echo is recorded. In practice, this means that if an attacker

can inject fake points directly behind a legitimate object, they can effectively make that object disappear, leaving only a scattering of random distant points. This can be achieved using the following relatively simple algorithm:

**Algorithm 1** Obtaining Attack Trace (ORA-Random)

**Input:** list(*obj_pts_coords*) ▷ Target Object's Point Cloud
*candidate_pts_coords* =[]
*attack_trace_pts_coords* = []
**for each** `pt` **in** *obj_pts_coords* **do**
  **if** `pt` **within** `spoofing_horizontal_angle` **then**
    *candidate_pts_coords* ← `pt`
  **end if**
**end for**
*attack_pts* ← random(*candidate_pts_coords*, $\mathcal{A}_{budget}$)
*attack_trace_pts_coords* ← (*pts* in *obj_pts_coords* ∧ not in *attack_pts*)
**for each** `pt` **in** *attack_pt* **do**
  *attack_pt_coords* ← dist_increment_along_ray(`pt`)
  *attack_trace_pts_coords*.append(*attack_pt_coords*)
**end for**
**return** *attack_trace_pts_coords*

Figure 2.14: Pseudocode for ORA algorithm from Hau et al.[18]

The algorithm also assumes that as the attacker is in close proximity to the victim, they can sense the environment and, using a transformation matrix that can be computed relatively easily from the distance to the victim, can estimate what the victim perceives. In theory, it begins by picking points from the target's point cloud at random within the point injection budget (in [18] this is assumed to be 200). Afterwards the selected points are shifted in the direction of the ray further away from the sensor and added back to the legitimate points. Thus, using this simple algorithm, we can model the effects of this type of attack to evaluate it's effectiveness. In the proposed setup, the author claims to reduce the target's recall for pedestrians and cyclists to under 25%.

For the purposes of this work ORA brings a multitude of advantages:

- Firstly, ORAs do not need any access to the victim's 3D classifier and thus they are way more flexible for real world applications.
- Secondly, the ORA model is not dependent on the injection methodology. Although ORAs presented in Hau et al.[18] are based off of the 200 fictitious point budget from [8] and utilise a synchronised approach (see Section 2.2.1), they could also be implemented under some assumptions on the position of the spoofer as a relay black-box attack (Section 2.2.2).
- Lastly, as the author themselves states, the algorithm is in a preliminary stage and therefore is easy to recreate, the methodology allowing for a lot of potential improvement.

## 2.3 Optimisation strategies

Lastly, we need to establish some context for the different optimisation strategies we can leverage in creating our novel attack strategies. The scope of this section is relatively limited; its goal is to present the main ideas and mathematical background of these approaches and discuss the common challenges and choices associated with them.

### 2.3.1 Genetic Algorithms (GAs)

Genetic algorithms (GAs) are a powerful optimisation technique inspired by the principles of natural selection and genetics. GAs are a class of algorithms tailored for complex optimisation and

search problems, which makes them an ideal candidate in the context of optimising the point selection in an ORA scenario. The GA consists of a population of individuals that are to be evolved, a fitness function to determine how efficient the solutions are and genetic operations consisting of selection, crossover, and mutation.

Holland et al.[19], first published in 1975, laid the mathematical groundwork for GAs and introduced the "building block hypothesis". It posits that small sub-solutions (schemata) can be sampled and combined to form effective higher-order solutions. As presented in the book, at the foundational level a genetic algorithm consists of the following five stages:

- **Initialisation** A population of potential solutions to the proposed problem (individuals) is created. The initial population is usually generated randomly but can also be predetermined in certain situations.
- **Selection** Individuals are chosen to reproduce based on their fitness, determined by a fitness function. Better individuals are more likely to be selected.
- **Crossover** Selected individuals are paired and their genotypes are combined to produce offspring. Goldberg (1989)[14] introduced different crossover strategies such as single-point, multi-point and uniform crossover.
- **Mutation** Random changes are introduced to some individuals in order to maintain population diversity.
- **Replacement** The new individuals replace the old individuals and the process is repeated until a certain termination condition is met. A multitude of strategies can be used to this end, the process being ended when a certain number of generations was reached, a certain amount of time has passed, the rate of improvement has reached a plateau, or a combination of all.

This structure creates a very flexible framework that can be applied to many different problems with minimal changes. As Darrell Whitley put it in his 1994 tutorial: "Usually there are only two main components of most genetic algorithms that are problem dependent: the problem encoding and the evaluation function."[36] Seeing as determining the fitness function is normally trivial as it is the objective of the algorithm, most difficulty arises from choosing between selection and crossover strategies as well as creating an encoding for the genotype. In most cases, it is represented by a bitstring, but can also be a real-valued vector.

In his tutorial[36], Whitley synthesizes the argument introduced in Holland et al.[19] that explains how a genetic algorithm "can result in complex and robust search by implicitly sampling hyperplane partitions of a search space." In this context, hyperplanes represent subsets of the search space defined by schemata. Take the given example of a problem encoded with 3 bits that can either be "0", "1", or "*" which represents a wildcard. The possible search space can be illustrated as a cube with each combination at one of it's corners (Figure 2.15). We can observe that the "back" plane of the cube contains all solutions that start with "1" and can thus be encoded as "1**". Schemata are therefore defined as any string containing a "*" and correspond to a hyperplane. Figure 2.15 also illustrates how this concept can be extended to a further dimension: all strings in the inner cube start with "1", while all strings in the outer cube start with "0". What makes GAs work in practice is that the algorithm samples multiple hyperplanes in parallel due to the diverse representation in the population. Selection increases the proportion of fit individuals hence promoting the spread of their corresponding hyperplanes that through crossover are then combined. Therefore, schemata can increase or decrease their representation in the population based on their "estimated" fitness.

GAs are an extremely versatile and flexible type of machine learning algorithm, being applied in various different fields. Although they present numerous advantages, GAs also come with some drawbacks:

- In the case where the fitness function is complex or computationally expensive, the repeated evaluation for each individual makes it necessary to limit the number of generations in order to achieve results in reasonable time. This might however limit the efficiency of the resulting solution.

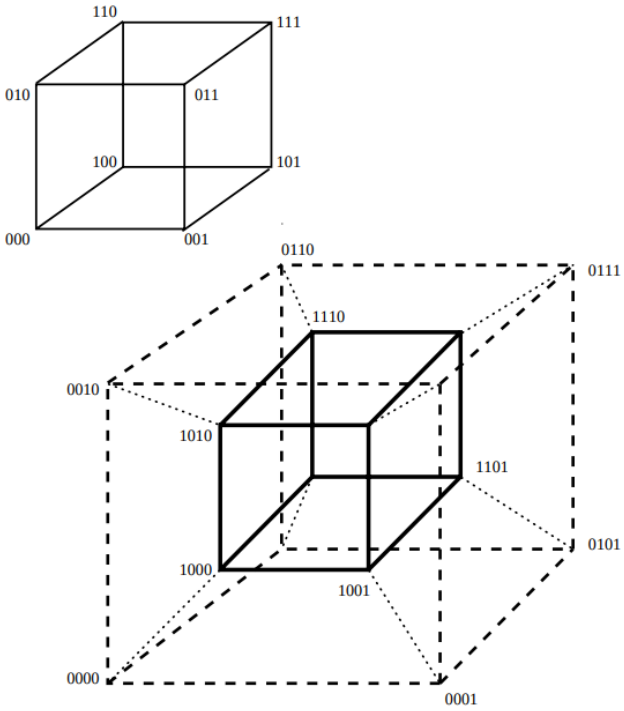



Figure 2.15: 3D and 4D representation of hyperplanes from[36]

- Just as many other machine learning algorithms, GAs are prone to converge on local optima as well as overfit to the training data.
- The performance of GAs can be highly sensitive to parameter settings, such as population size, mutation rate, and crossover rate.

### 2.3.2 Bayesian optimisation

Bayesian optimisation is a powerful strategy for optimising expensive to compute functions which is especially useful in situations where traditional methods are impractical due to their high cost[5]. As stated in Shahriari et al.[29], "Bayesian optimisation is a sequential model-based approach to solving problem 2.1"

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{2.1}$$

Bayesian optimisation combines concepts from statistics with machine learning in order to determine an optima for a black-box function. The core idea is to create a probabilistic model of the function and use our prior beliefs to refine the model by observing more of its results.

**Algorithm 1** Bayesian optimization

1: **for** $n = 1, 2, \ldots$ **do**
2: select new $\mathbf{x}_{n+1}$ by optimizing acquisition function $\alpha$

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$$

3: query objective function to obtain $y_{n+1}$
4: augment data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
5: update statistical model
6: **end for**

Figure 2.16: Pseudocode bayesian optimisation algorithm from[29]

In practice, the process consists of the following steps:

- **Surrogate model creation** A surrogate model, typically a Gaussian Process (GS) is created to approximate the objective function. "The GP is defined by the property that any finite set of $N$ points $\{x_n \in \mathcal{X}\}_{n=1}^{N}$ induces a multivariate Gaussian distribution on $R^N$."[32] A GP is specified by a mean function $m(\mathbf{x})$ and a covariance function (or kernel) $k(\mathbf{x}, \mathbf{x}')$. The choice of the kernel function is crucial as it encodes assumptions about the function's smoothness and structure[26]
- **Acquisition function** An acquisition function $\alpha$ is used to determine where the objective function is to be evaluated next. In Snoek et al.[32] a few different acquisition functions are described: **Probability of Improvement (PI)** which focuses on maximising the improvement over the current best solution, **Expected Improvement (EI)** which maximises expected improvement and **Upper Confidence Bound (UCB)** which follows the "idea of exploiting lower confidence bounds (upper, when considering maximization) to construct acquisition functions that minimize regret over the course of their optimisation."[32]
- **Function evaluation** The objective function is evaluated at the point that maximises the acquisition .
- **Model update** The surrogate is updated with the new information that was gained.

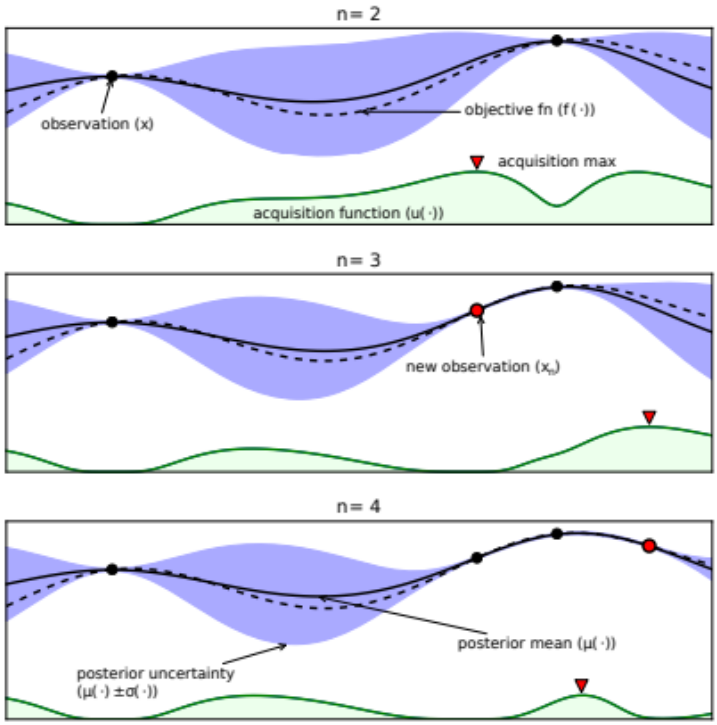



Figure 2.17: Representation of three iterations of Bayesian optimisation from[29]. We can observe that the acquisition function is high where there is a lot of uncertainty(exploration) or where it predicts a high objective result (exploitation)

# Chapter 3

# GORA & BORA: Novel 3D Object Removal Attacks

## 3.1 Choice of methodology

As was presented in Chapter 2, numerous LiDAR attack methodologies have been proven effective in real-life scenarios. Broadly, they can be divided by their level of access to the victim's sensor into white-box and black-box attacks and by their goal into injection and removal attacks.

Section 2.2.1 discusses the problems associated with a white-box approach, making them less feasible on next-gen sensors. However, considering first-gen LiDARs are in wide use and expensive to replace, this type of attack will still represent a valid threat in the future.

Although injection attacks (Section 2.2.1) pose a significant threat by inducing a near-front vehicle and causing an AV to abruptly stop, they are less dangerous than removal attacks, which can potentially cause collisions.

Considering all these factors, the ORA methodology provides the most flexibility and threat, being a removal attack that is not dependent on the injection method. As discussed in Section 2.2.3, even though the approach is based on "white-box" access, the methodology does not require any knowledge about the victim's 3D classifier and also could be implemented as a relay attack. Furthermore, the preliminary aspect of the findings also allows room for improvements. Thus, this work mostly focuses on recreating ORA and bringing improvements at a theoretical level, ignoring aspects related to injection method.

## 3.2 Threat model

Based on the premise of our improvements, we adopt a similar threat model to the one used in Hau et al.[18]. We assume an adversary $A$ can spoof the target's LiDAR return signals by deploying a device within the line of sight of the sensor. By modifying the return signal, adversary $A$ can modify the sensor's 3D measurements, having the capacity to inject up to 200 points. Although Cao et al.[7] showed that it is possible to inject up to 4000 fictitious points, we mostly keep the original budget for the sake of comparison. Additionally, $A$ can visually identify the model of the sensor the target employs and $A$ can spoof the resulting measurements to make objects appear closer or further away from the target vehicle than they actually are. Being in close proximity, $A$ can sense the environment and detect nearby objects as well as use simple transformations to change the 3D coordinates of the scene to reflect the target's point of view.

However, in addition to the threat model presented in Hau et al.[18], for our Genetic Object Removal Attack (GORA) as presented in Section 3.6.1 and our Bayesian Object Removal Attack (BORA) as presented in Section 3.6.2, we assume $A$ has knowledge of what object detection algorithm the target is utilising.

## 3.3 Overview

The final goal outlined in Section 1.2 involves improving on the existing ORA functionality. The most effective and straightforward way of achieving this is by refining the point selection algorithm to perform better than it does randomly.

A crucial aspect of the strategies presented in this chapter is that they are not dependent on the detection framework used, thus being easily integrated into our framework presented in Chapter 4 as well as any other.

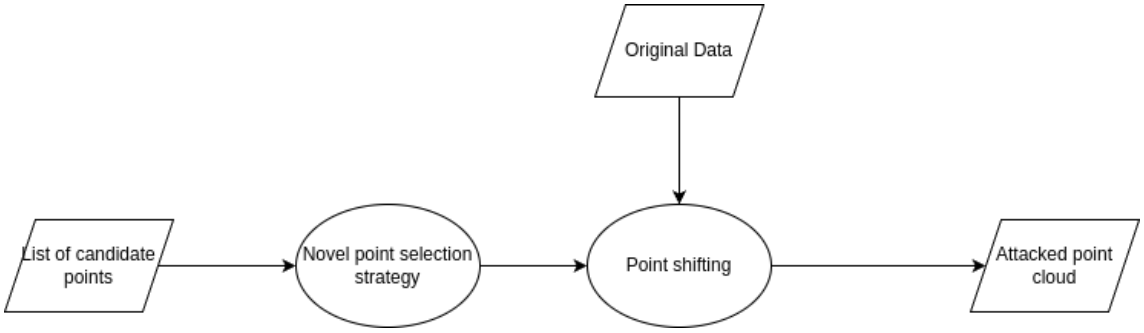



Figure 3.1: Pipeline for novel strategies

Because we are only concerned with the selection module, our improvements are self contained and can be adapted to different datasets and frameworks with ease.

In this context, it is useful to abstract the problem in order to make it easier to reason about. At its core, the problem involves selecting *budget* points out of a total of $n$ points in order to minimise a black-box function. In the examples presented in Section 5.7, two objectives are used:

- **Minimising the Intersection over Union (IoU)** This is the metric used in [18] and most related works. As the name suggests, given two bounding boxes it is calculated by determining the volume of the intersection and dividing it by the volume of the union. The goal is to ensure that the predicted bounding box's position is as far away as possible from the ground truth.

- **Minimising confidence** This alternative approach aims to completely obscure the targeted vehicle. Each bounding box identified by an object detector has a score that represents the confidence the algorithm has in the prediction. As presented in Lee et al.[22], most object detectors use a single hyperparameter as a threshold for confidence values, usually set to 0.5. Lee et al.[22] proposes an adaptive thresholding system that improves the overall performance of object detectors.

## 3.4 Distance heuristic

A straightforward idea for improving the point selection algorithm would be to select the closest points to the victim vehicle (under the budget). The intuition behind this approach is that the closest points should be the most accurate thus altering them causing the most disruption, which ought to be even more impactful under adverse weather conditions.

An interesting facet to this method is the direction in which the selected points are shifted. Shifting the points to be further away from the ego vehicle should compact the predicted bounding box and reduce detection IoU. On the other hand, shifting the points in the opposite direction has the possibility of reducing the detection IoU even further but goes against the idea of a removal attack, having a real effect similar to an injection attack as the vehicle appears closer than it actually is.

## 3.5 Intensity heuristic

Another area for improvement is utilising the intensities of the candidate points when performing the selection. As discussed in Section 2.1.1, intensity reduction is one of the main factors that limits the performance of LiDARs in adverse weather conditions as it plays an essential role in detection. Therefore, shifting the points with the highest intensity should significantly affect the performance of 3D object detectors.

Although this approach is promising in theory, in practice having access to the intensity readings from the victim's LiDAR is unrealistic. As presented in Hau et al.[18], the coordinates of points from the attacked LiDAR's point of view can be inferred using a simple translation matrix. However, the intensity values from the attacker and target perspective differ and cannot be simply deduced. Thus, although the attacker's point intensities can be used as an approximation, it is not feasible to have access to precise values. For these reasons, this heuristic is mostly used as a reference for our evaluation.

## 3.6 Machine learning approaches

Using machine learning to improve the selection of points in ORA is promising seeing as the problem involves large and complex amounts of data that are difficult to understand and exploit by conventional means. This kind of approach does however have some drawbacks, mainly the reliance on the quality and amount of data as well as the amount of computational resources required.

As the field of machine learning is extremely varied and constantly evolving it is crucial to narrow down possible approaches for our specific problem. Firstly, it is important to realise that because our problem is open-ended and it is not possible to provide labels for our data, an unsupervised learning approach is necessary. Another factor to consider is that our objective function is not differentiable and has to be treated like a black-box due to its nature.

Yet another concern is that each different scenario can have a variable number of candidate points out of which we have to make the selection. This entails some additional implementation complexity but more importantly it has the potential to make it more difficult to extract meaningful features out of the data.

One approach considered is leveraging recent advances in Reinforcement Learning (RL), specifically integrating transformers into the policy network of the RL setup. Although this solution would fit our requirements for an unsupervised approach utilising a non-differentiable objective function, it would nonetheless not exploit the main strengths of RL, its main use case being modelling the actions of an agent that interacts with an environment usually in a real time scenario.

### 3.6.1 Genetic Object Removal Attacks (GORA)

Genetic algorithms (GAs) as presented in Section 2.3.1 are a type of machine learning algorithm inspired by natural selection that perfectly fit the requirements of our problem. Owing to their robustness and flexibility, GAs can easily adapt to changes in the search space or objective function. A GA approach also presents some drawbacks, the most significant of which being the fact that it is extremely computationally expensive, especially in our scenario where we are dealing with a large amount of cases each of which consisting of hundreds of thousands of point coordinates. Combining this with the parameter sensitivity inherent to GAs and the limited time and computational resources available, it is particularly hard to extract the full potential of this approach.

As was outlined in Section 2.3.1, the main challenges when designing a GA are problem encoding and the evaluation function:

- **Problem encoding** Returning to our initial problem abstraction of selecting *budget* points out of a total of $n$, the natural way of representing our genotype would be as a list of length *budget* containing the indexes of the selected points. However, this does not address the fact that the number of candidate points $n$ can vary from scenario to scenario, potentially

leading to selected indexes being out of bounds. Limiting the values of the indexes selected or using padding would cause significant bias towards lower values and severely affect the performance on large entries as the algorithm learns that lower indexes are a safer overall option with guaranteed impact.

In order to avoid this, the solution is to initialise the genotype with the maximum size of one of the entries in the dataset and then scale the indexes down as appropriate. The idea is that by scaling down instead of padding, bias should be greatly reduced, and features inherent to the data should translate better between scenarios.

**Algorithm 1** scale_indices

**Input:** $individual$, $data\_length$, $max\_length$
**Output:** Scaled indices as a list
**function** SCALE_INDICES($individual$, $data\_length$, $max\_length$)
    $scale\_factor \leftarrow \frac{data\_length}{max\_length}$
    $scaled\_indices \leftarrow \emptyset$
    **for each** $idx$ **in** $individual$ **do**
        $proposed\_index \leftarrow \text{int}(idx \times scale\_factor)$
        **while** $proposed\_index$ **in** $scaled\_indices$ **and** $\text{len}(scaled\_indices) < data\_length$ **do**
            $proposed\_index \leftarrow (proposed\_index + 1) \mod data\_length$
        **end while**
        $scaled\_indices$.add($proposed\_index$)
    **end for**
    **return** list($scaled\_indices$)
**end function**

It is also crucial to ensure all indexes are unique as not utilising the entire budget can have a drastic effect on performance.

- **Evaluation function** In our case, the evaluation function is dependent on the evaluation metric used (IoU or confidence) as well as the detection framework used for detection. Since this is one of the only parts of the algorithm that interacts with functionality outside the point selection module, it requires minimal adjustment based on external factors. Therefore, for the sake of simplicity, the evaluation function for an individual is the mean of the chosen metric across all scenarios.

  This also applies in the case where we train our genetic algorithm on multiple object detection algorithms at the same time, the result being the mean of the combined scores for all scenarios. The weights of the results can be easily changed in specific situations where performance on a particular object detection algorithm is preferred over another, perhaps due to their prevalence.

One of the most significant pitfalls of GAs is premature convergence, which happens when the diversity of a population diminishes, thus causing the results to remain on a local optima. Therefore, most of our specific design choices are made with the goal of preserving diversity in the population seeing as our search space is very large.

Distributed Evolutionary Algorithms in Python (DEAP)[12] is an evolutionary computational framework that is highly flexible and customizable, allowing for great ease of use and rapid prototyping. The most important advantage of using DEAP is that it has a very modular design, allowing users to modify and extend various aspects of the evolutionary process. This capability can be leveraged in the context of this work by customising the essential steps (Section 2.3.1) of a GA for our needs:

- **Initialisation** Based on the defined population size and maximum size of one of the cases, each individual is initialised randomly.
- **Selection** There exist a multitude of selection strategies, most representing a compromise between convergence speed and diversity preservation. Considering our problem has a very

large search space that likely contains a lot of local optima, maintaining diversity in our population is of the upmost concern. Thus, the approach used is tournament selection with a small tournament size. Three individuals are picked at random and the best one is selected to reproduce, this being repeated until the desired number of individuals has been reached.

- **Mutation and Crossover** The mutation function takes the mutation rate as an argument and for each index inside an individual attempts to change it to a random new index while maintaining their uniqueness so as not to use less points than the budget allows. Uniform crossover is used for the main reason that the ordering of the selected indexes has no impact on the fitness of the individual. This crossover strategy also serves to enhance the diversity within the population along with the usage of the *varAnd* variation strategy from DEAP which allows both mutation and crossover to potentially be applied on each individual.

- **Replacement** Instead of just replacing the old population with their offspring, our approach also utilises fitness sharing and elitism in order to promote diversity among the population. Fitness sharing involves penalising solutions that are too similar to each-other based on a specific metric. In our case, the Hamming distance is used, individuals are penalised based on the number of others they share at least 75% of their selected indices with.

$$f'(i) = f(i) \times \left( \sum_{j \in \text{Population}} \delta(\text{distance}(i, j) \leq \text{threshold}) \right)^{\alpha}$$

  Where $\delta$ is the indicator function that is 1 if the condition inside is true and 0 otherwise and $\alpha$ is set to 0.05 from empirical results. Elitism is a strategy used in GAs where a certain number of the best individuals from the current generation are guaranteed to be carried over to the next generation without undergoing crossover or mutation. Utilising elitism has the objective of stabilising the population to counteract the randomness induced by fitness sharing.

Another important aspect of the GA is the utilisation of K-fold cross-validation in the training process. This works by dividing the dataset into $k$ equal sized folds, $k$ - *1* of which are used for training at each iteration and the last being used for evaluation. Therefore, each fold will be evaluated on once and trained on several times, giving a more comprehensive idea of the algorithm's performance. The most crucial advantage of this approach is that it efficiently uses even limited amounts of data while still providing an unbiased evaluation result. Thus, although this has a significant computational costs, it ensures that our GA can perform efficiently on datasets of all sizes.

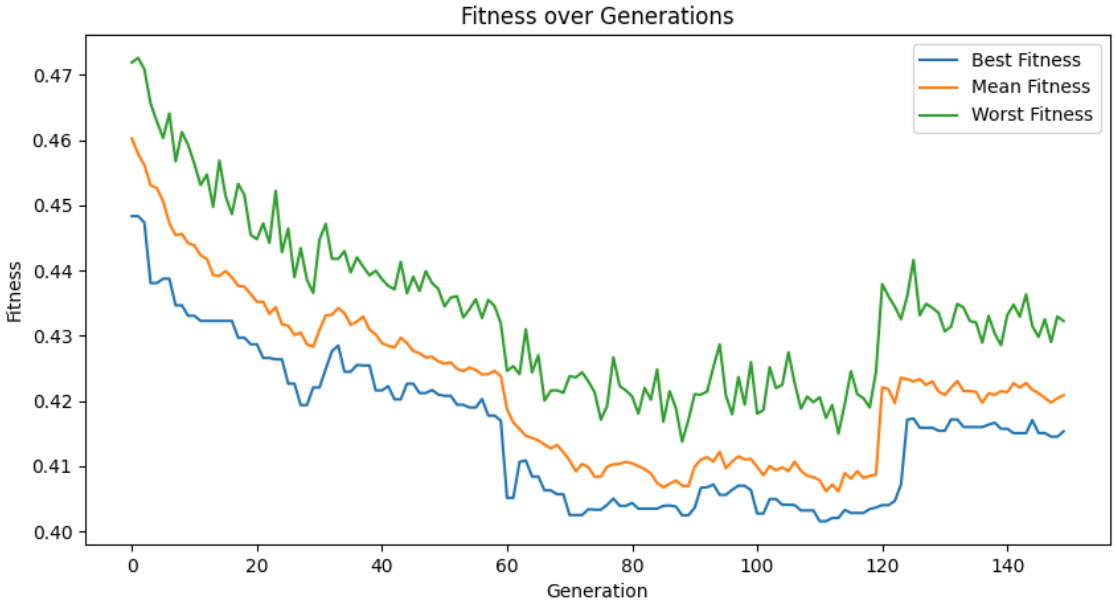



Figure 3.2: Training fitness over generations for the HDL-64E sensor across 5 folds of 30 generations each

In Figure 3.2 we can observe the training finesses decreasing inside each fold of 30 generations, while having sharp changes between folds. This is caused by the fact that data inside each fold is likely to have quite different characteristics, being more or less difficult to optimise.

### 3.6.2 Bayesian Object removal Attacks (BORA)

Bayesian optimisation (BO) as previously described in Section 2.3.2 is a strategy for global optimization of black-box functions that are expensive to evaluate, making it a perfect candidate approach for our problem.

The main challenges of designing a BO algorithm are the similar to the ones in designing a GA, namely problem encoding and evaluation function. Two different approaches for implementing Bayesian ORA were attempted:

- The first strategy involves trying to optimise the chosen metric (IoU or confidence) for each scenario separately. This approach exploits specific weaknesses in each dataset case, allowing for improved performance on a case-by-case basis. Another significant advantage is that this strategy allows for a high degree of parallelisation given the necessary computational resources. As we will further explore in Section 5.6, the execution duration of this algorithm does not allow it to be used in a real-time scenario using our threat model presented in Section 3.2. We do still consider that exploring this approach is beneficial as it could be employed in offline scenarios or perhaps just used as a performance reference for other strategies.

- The second approach is similar to the GA as it utilises cross-validation while trying to minimise the average metric across all training cases. For this approach, our search space for the BO algorithm is similarly defined to the individuals from GORA by using the maximum size of one of the cases, indices then being scaled down. The results would also be used in a similar manner to the GA ones, where a predefined list of best indexes chosen through training is applied to unseen data. Compared to the previous approach, this strategy conforms to our threat-model and is a possible option for a real scenario.

In order to streamline the implementation of the BO algorithm, the *scikit-optimise* library was utilised because it offers significant built-in functionality while still being flexible enough to be customised for our specific problem. Therefore, we can develop a modular implementation that only requires minimal changes between the two approaches outlined above or for changing the target metric of the optimisation. This also allows our implementation to be more easily adapted for different detection frameworks, most of the effort required for the implementation being invested into loading and processing the possible candidate points as well as designing the evaluation function. One of the advantages of using BO over a GA is that a BO algorithm has less hyperparameters and thus requires less manual adjustment. For our implementation, there are two main parameters:

- **Number of iterations** This parameter dictates the optimisation's budget, the algorithm being allowed to evaluate the black-box function a given number of times. In this regard, this approach suffers from a similar problem to our GA strategy, namely that due to the limited available computational resources it is not possible to extract the all the available performance.

- **Acquisition function** This choice has great effect on both the performance and efficiency of the BO algorithm. In our approach, the Expected Improvement (EI) acquisition function is utilised as it balances exploration and exploitation as well as the fact that it is computationally efficient compared to more complex function, allowing the usage of as many iterations as possible.

Overall, the the main advantage of this strategy is that it is relatively less complex than GAs, requiring less parameter adjustments. Although they have similar implementations, the two BO approaches serve very different use-cases, allowing for a comprehensive performance analysis. Furthermore, this optimisation strategy presents significant potential for further improvements especially given significant computational resources.

# Chapter 4

# Design and implementation of detection framework

## 4.1 Core concept

The main idea behind this project is to evaluate and subsequently exploit the limitations of LiDARs under rainy conditions, as presented in Section 2.1.1. The hypothesis posits that as overall performance diminishes, the real-world impact and consequences of an attack increases. In a scenario where a sensor already struggles to function correctly, even an unoptimised attack can have grave consequences.

In order to verify this hypothesis, as was outlined in Section 1.2, we would need to first select or create a comprehensive dataset for adverse weather conditions and replicate existing attack methodologies

## 4.2 LiDAR dataset generation

Generating a LiDAR dataset for adverse weather conditions is a critical step in this project. The dataset must accurately reflect the challenges faced by LiDAR sensors in rainy environments. Therefore, there are three main considerations when it comes to the dataset:

- **Size:** This is the most obvious factor, as it helps mitigate the effects of outlier scenarios on the results. Additionally, for a machine learning algorithm, having a sufficiently large dataset is essential to achieve desirable outcomes.
- **Weather conditions:** Representing a range of weather conditions in the dataset is crucial. To ensure the dataset is comprehensive, it must include a wide range of adverse weather scenarios such as rain with varying intensities, from light($<$ 2.5mm/h), moderate(2.6 - 7.5 mm/h) and heavy ($>$7.5 mm/h)[4]
- **Accuracy:** This is particularly important for simulated data. The dataset must accurately depict real-world driving scenarios to ensure reliability.

### 4.2.1 Comparison of real and simulated data

Various labeled LiDAR datasets already exist online and are extensively used in research. Perhaps the best known is the KITTI dataset[13] which was used for evaluation and validation in Hau et al.[18]. The dataset was collected in the Karlsruhe region of Germany using a setup incorporating a variety of sensors, including an HDL-64E sensor.

One significant advantage of the KITTI dataset is its precise labeling, as a team of annotators was hired to manually label the readings. Moreover, its widespread use in literature allows for easy comparison of results


Figure 4.1: KITTI autonomous driving platform from[13]

However, the fundamental problem with the KITTI dataset is that it only includes clear weather scenarios, making it unsuitable for our work.

Another popular alternative is the Waymo dataset[9]which was created by Google. When compared to the KITTI dataset, its main advantage is the diversity in both geography and conditions. Data was collected from 6 major cities in the US, encompassing different environments and weather conditions.

Although the Waymo dataset is large and also includes adverse weather, it has a significant drawback: it is challenging to directly compare clear and rainy scenarios because they might be collected in different locations, at different times of day, with varying vehicle types around. Thus, even though we could infer some general characteristics about the effects of adverse weather on LiDAR by combining the results we get over the whole dataset, it would be extremely hard to isolate rain as the sole factor. Additionally, there are other considerations, such as the potential for skewed statistics due to the disproportionate number of clear weather data points compared to rainy ones. All of these factors would diminish the impact of our findings, as any conclusions drawn could be unknowingly biased.

In contrast, using simulated data provides several advantages, the most important of which is flexibility in creating scenarios. By leveraging the capabilities of a simulator, we can recreate the exact same positions, vehicles and environments in both clear and rainy conditions. Thus, a direct comparison can be drawn by isolating rain as the only differentiating factor. Moreover, multiple models of LiDAR sensors can be employed in the same scenario, allowing for an even more comprehensive analysis of their performance

Nevertheless, using simulated data also has disadvantages, the most notable of which being a decrease in accuracy. Although, as presented in Section 2.1.1, Goodin et al.[15] and Yang et al.[38] show promising results for their rain models, there is still a domain gap between real and simulated data. Because of computational concerns, certain aspects have to be approximated and other less impactful ones completely disregarded. This discrepancy is be further investigated in Section 5.2

Another consideration when it comes to using simulated data is the amount of time and effort necessary for its creation. Simulators tend to be difficult to set-up and learn to use due to their complex structure and extensive functionality. On top of that, time is also required to create the scenarios necessary in gathering data. Although this process can be somewhat automated, in or-

der to guarantee the quality of the dataset, a significant amount of manual intervention is required.

Taking everything into account, although real data provides better accuracy and ease of use, for the specific context of investigating the robustness of LiDAR in adverse weather, using artificial data allows for more flexibility in isolating the relevant factors to our study.

### 4.2.2 Choice of simulator and rain model

The CARLA simulator[10] is an open-source simulator developed from the ground up to support training and validation of autonomous driving systems. CARLA provides a flexible and highly detailed simulation environment that enables researchers and developers to create and test autonomous driving algorithms under a wide range of conditions.

One of the main strengths of CARLA is that it has a very large number of assets that allow for extensive customisation. The simulator provides a variety of maps, vehicles, pedestrian types and allows for in-depth control of their behaviour. Another significant advantage is that owing to it's popularity, the simulator is well-maintained and there are ample resources online for learning its functionality.

The primary drawback of using CARLA is that although it provides some support for simulating adverse weather conditions, the changes are mostly visual and do not affect LiDAR sensors. As discussed in section 2.1.1, Yang et al.[38] presents a rain model for the CARLA simulator that utilises an intensity prediction U-net, showing promising results overall. However, the code-base for the model has not been maintained since the publication of the paper and is poorly documented, thus making it almost impossible to set-up and use.

In contrast, the MAVS simulator provides an integrated rain model that Goodin et al.[15] shows accurately represents real scenarios, making it a very attractive choice for this project. However, the simulator is not open-source (although it is free for academic use) and thus there are fewer learning resources online, making it harder to set-up and use. Furthermore, the most significant downside of MAVS is that it only includes a very limited number of assets such as maps and vehicles.

Considering all these factors, MAVS is the best viable solution for simulating adverse weather scenarios for LiDARs. Nevertheless, this choice means that the findings presented in this work focus on vehicle detection because the limitations of MAVS make it very difficult, if not impossible, to add pedestrians and cyclists. We consider this to be a worthwhile trade-off, seeing as vehicle detection is the most common case in AV scenarios and the methodology presented should seamlessly translate to both pedestrian and cyclist detection.

## 4.3 Detection pipeline

In order to complete the second objective outlined in Section1.2, we first need to develop a method for applying different object detectors to our simulated data. The MAVS simulator outputs labelled data in the Point Cloud Data (PCD) format[24].

```
VERSION 0.7
FIELDS x y z intensity label
SIZE 4 4 4 4 4
TYPE F F F F F
COUNT 1 1 1 1 1
WIDTH 19682
HEIGHT 1
VIEWPOINT 11.0945 7.48734 2.72633 0.999989 6.75839e-06 0.00105548 -0.00454081
POINTS 19682
DATA ascii
-10.1183 8.84573e-07 -2.7112 0.000676396 4
-11.5841 1.01271e-06 -2.67439 0.000202572 4
-13.7644 1.20332e-06 -2.67553 7.20241e-05 4
-16.9507 1.48188e-06 -2.68473 5.33551e-05 4
-10.1196 -0.0353235 -2.71155 0.000789129 4
-11.5663 -0.0403734 -2.6703 0.000159164 4
-13.7681 -0.0480592 -2.67627 0.000266353 4
-16.9396 -0.0591297 -2.68299 0.000125384 4
-10.1063 -0.0705559 -2.70803 0.000698943 4
-11.5799 -0.0808442 -2.67351 6.27009e-05 4
```

Figure 4.2: Beginning of a PCD file. Each point entry has the format *(x, y, z, intensity, label)*

This format presents a significant number of advantages, being fast, very flexible as it can encode different data types and includes information about the structure and size of the data in the header. It is important to also notice that the viewpoint from which the reading was recorded is specified as a translation *(tx, ty, tz)* and a quarterion *(qx, qy, qz)*.

However, this format differs from the one used by OpenPCDet[35] which is used as the backbone of our detection pipeline due to it's flexibility and robustness, integrating multiple object detectors. OpenPCDet uses NumPy or binary files of the format *(NPoints, 4)*, each entry being of the type *(x, y, z, intensity)*. To this end, the pypcd library[23] was used to convert the files to the right format.

The converted data can then be used with the *demo.py* provided by OpenPCDet to perform detection with a model trained on the KITTI dataset.

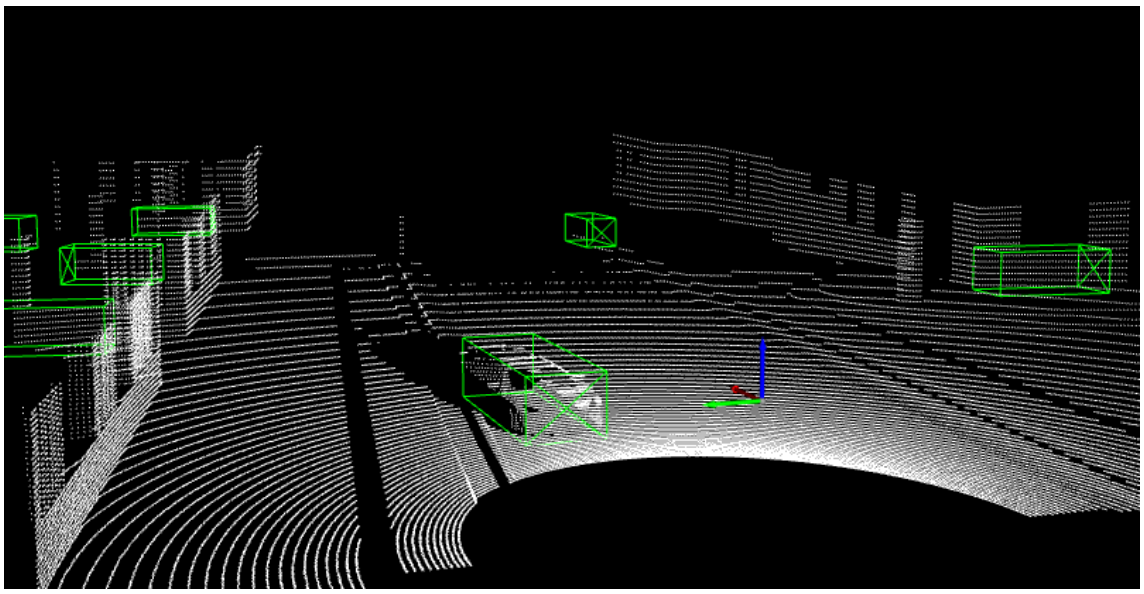


Figure 4.3: Detection example using PointPillars from an HDL-64E sensor

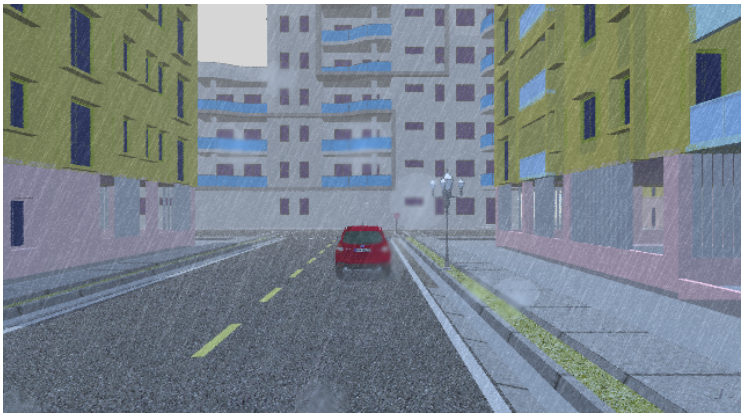
For each input file, the detector outputs a number of predictions, each consisting of the following:

- A bounding box of the format *(x, y, z, dx, dy, dz, heading)* where *x, y, z* are the coordinates for the center of the box, *dx, dy, dz* are extents along each of the axis and the heading ranges from 0 to $2 * \pi$.
- A score that represents the confidence the detector has in the prediction. In Figure 4.3 multiple erroneous bounding boxes can be observed which is explained by the fact that all of them have very low confidence scores and would not affect a real scenario.
- A label, classifying the prediction as either a vehicle, as traffic sign, a cyclist or a pedestrian.

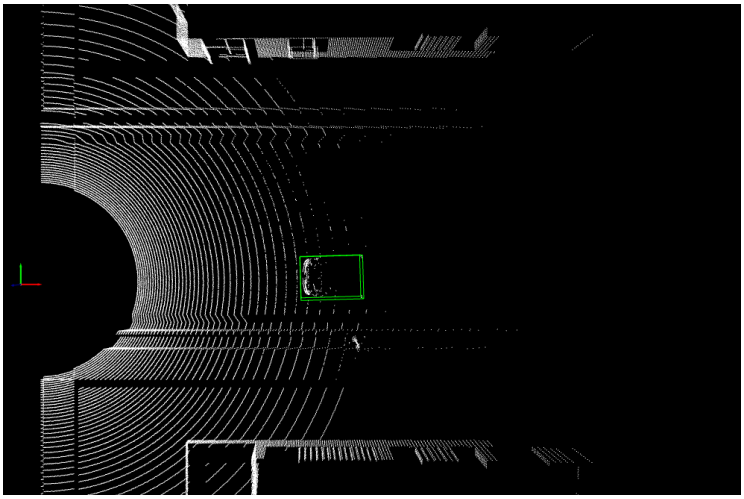
In the context of 3D object detection, the metric usually used to quantify the quality of a prediction is the Intersection over Union (IoU). As the name suggests, given two bounding boxes it is calculated by determining the volume of the intersection of the boxes and dividing it by the volume of the union. In our case, in order to calculate the IoU for the prediction we first need to identify the ground-truth bounding box of the vehicle in our scenario.

As mentioned before, the KITTI dataset ground-truth bounding boxes are annotated manually. Although we could do the same for our synthetic dataset, this would not only take significant effort but also make it notably harder to extend the dataset in the future. Thus, even if the quality of the bounding boxes is lower, an automated approach to generating the ground truth is preferable. To this end, the Open3D library[41] provides functionality for creating oriented bounding boxes from a point cloud.

The resulting bounding boxes are in a different format than the one mentioned previously, being defined by the coordinates of the corners. In order to convert the bounding boxes to the format used by OpenPCDet, the center coordinates and extent can be calculated from the corners and the heading can be determined using the rotation matrix of the box. The main disadvantage of the obtained bounding boxes is that they are limited by the quality of the original data, specifically in the case of a sparse point cloud with a limited number of vehicle points, the result might not accurately reflect the truth.



(a) Ego vehicle perspective



(b) Top-down view of LiDAR results and ground-truth box without adjustments

Figure 4.4: Driving scenario in 25 mm/h rain. Because of the adverse conditions and angle, only part of the vehicle is represented in the LiDAR data, leading to a ground truth bounding box that is too small

To increase the accuracy of our ground-truth bounding boxes, the dimensions of the vehicles are augmented to a fixed size, based on averages. Although the same exact positions of the vehicle could be set manually based of the models used in the simulator, this would once again make it harder to extend our dataset in the future.

The augmented ground-truth bounding boxes can be used together with the predictions to determine the IoU. To this end, the IoU function used for the PillarNet detector[16] is used (Appendix B.1). Calculating the volume of the intersection is done by first reducing the problem to two dimension along the $x$ and $y$ axis. After the dimensions of the intersection rectangle is computed, its area is multiplied by the extent of the intersection along the $z$ axis. The volume of the union can then be determined by subtracting the volume of the intersection from the sum of the volumes of the starting bounding boxes. Because of its design, the function allows for all of these calculations to happen in parallel for multiple bounding boxes thus covering all predictions in a scenario. These values are used for our analysis is Section 5.

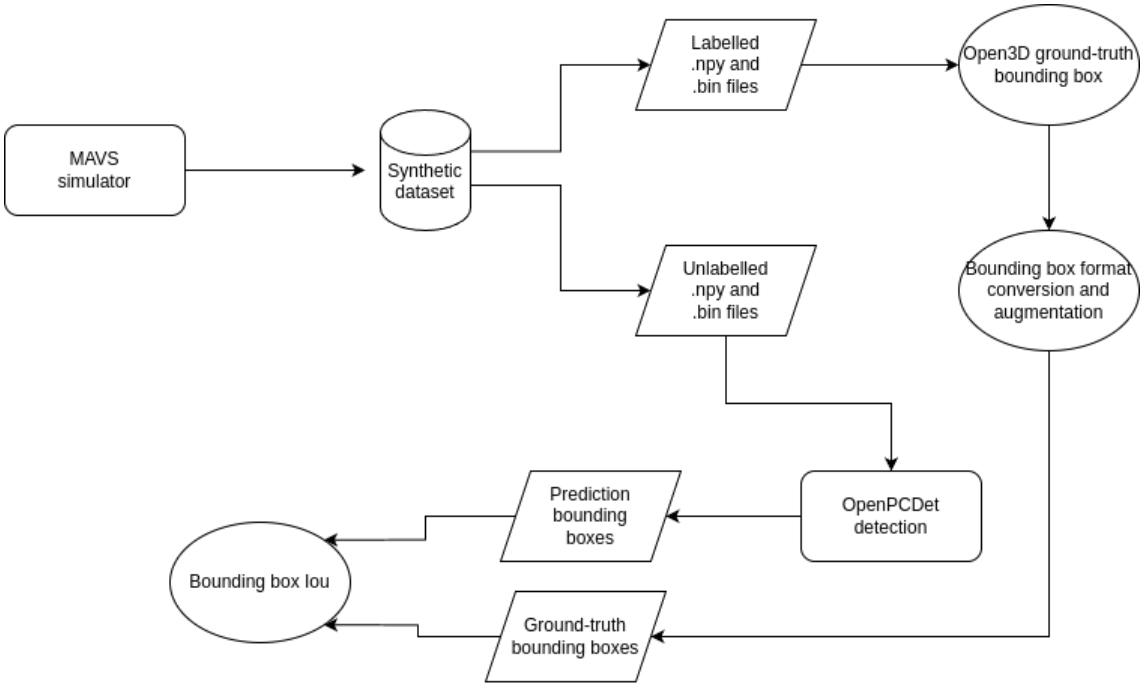



Figure 4.5: Detection pipeline

## 4.4 Replication of ORA

In order to replicate ORA, we can build upon our already existing detection pipeline. Considering the ORA pseudocode in Section 2.14, we need to determine the list of possible candidate coordinate points and then shift them up to 2 meters. Although we could directly select candidate points from the ground-truth bounding boxes, this would not be applicable in a real scenario. Thus, similar to the original paper, candidate points are chosen from the predicted bounding box, necessitating a two stage approach. In the first detection, the bounding box that is the most accurate (best IoU) is chosen and the points inside of it are selected as candidates. This is done using the *points_in_boxes_cpu* function from the *roiaware_pool3d_utils* module of OpenPCDet because it provides excellent efficiency seeing as it is compiled in C++ instead of Python and uses bitwise operations.

**Algorithm 2** ray_shifting

**Input:** $point\_to\_be\_shifted$, $shifting\_distance$
**Output:** Shifted point as a tensor
**function** RAY_SHIFTING($point\_to\_be\_shifted$, $shifting\_distance$)
    $delt\_x \leftarrow point\_to\_be\_shifted[0] - origin[0]$
    $delt\_y \leftarrow point\_to\_be\_shifted[1] - origin[1]$
    $delt\_z \leftarrow point\_to\_be\_shifted[2] - origin[2]$
    $(az, el, r) \leftarrow \text{cart2sph}(delt\_x, delt\_y, delt\_z)$
    $shifted\_r \leftarrow r + shifting\_distance$
    $(shifted\_delt\_x, shifted\_delt\_y, shifted\_delt\_z) \leftarrow \text{sph2cart}(az, el, shifted\_r)$
    $shifted\_x \leftarrow shifted\_delt\_x + origin[0]$
    $shifted\_y \leftarrow shifted\_delt\_y + origin[1]$
    $shifted\_z \leftarrow shifted\_delt\_z + origin[2]$
    $shifted\_intensity \leftarrow point\_to\_be\_shifted[3]$
    $shifted\_point \leftarrow \text{torch.tensor}([shifted\_x, shifted\_y, shifted\_z, shifted\_intensity])$
    **return** $shifted\_point$
**end function**

Candidate points are then chosen randomly (under a budget) to be shifted. Shifting is done by converting the point's Cartesian coordinates to Spherical coordinates, adjusting the radius to reflect the change and then converting the coordinates back to the Cartesian coordinate system.

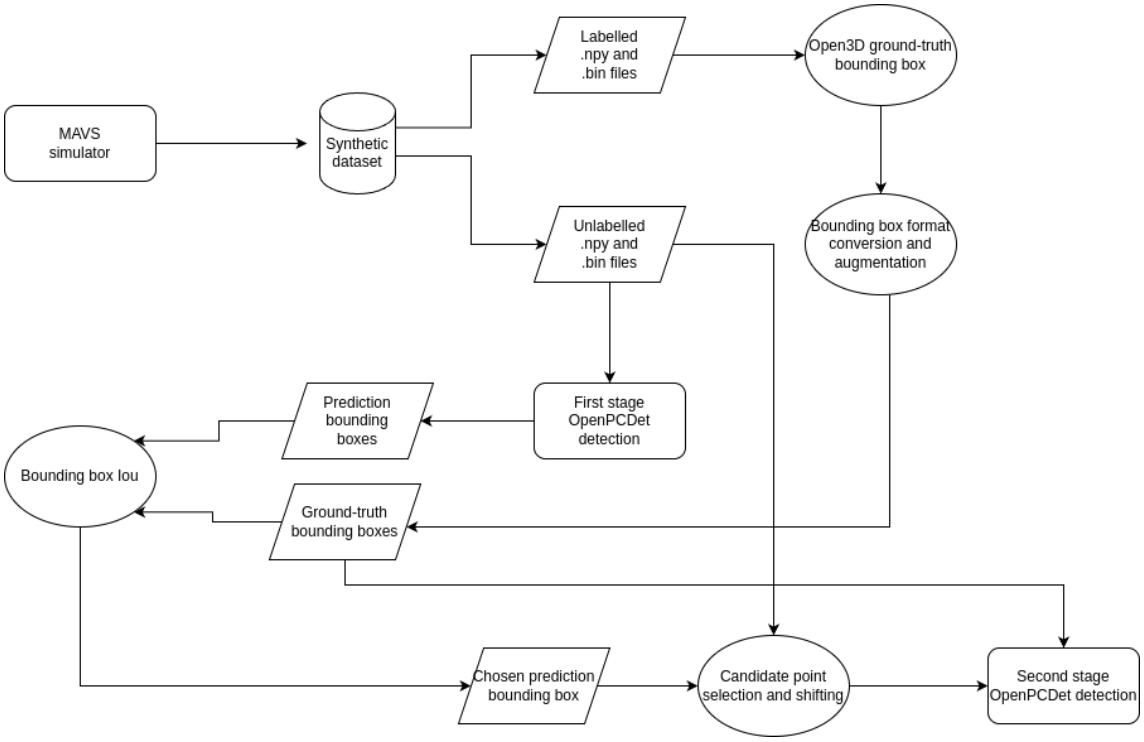



Figure 4.6: ORA pipeline

# Chapter 5

# Evaluation

In this section we evaluate the success of our work by answering the research questions below. Where necessary, we draw comparisons to the results presented in related works.

- **RQ1** - *How well does our simulated dataset reflect the findings in Goodin et al.[15]?*
- **RQ2** - *How does the base detection rate of our pipeline compare to related works? Can we establish reasonable thresholds for all sensors and object detectors?*
- **RQ3** - *How do the results of our recreated Random ORA compare to the original ones from Hau et al.[18]?*
- **RQ4** - *How does rain impact the effectiveness of Random ORA?*
- **RQ5** - *How well do our novel algorithms perform?*
- **RQ6** - *Which target metric yields the best result for our approaches?*

## 5.1 Experimental setup

As was discussed in Chapter 4, he MAVS simulator is used for creating the dataset these experiments are run on. All test scenarios are categorised by these factors:

- **Rain rate** As briefly outlined in section 4.2, we utilise 5 different weather settings: clear, light (for 2.5 mm/h rain), moderate (for 5 mm/h rain), heavy (for 10 mm/h rain) and extreme for (25 mm/h rain). Although some related works have investigated even heavier rain than our extreme case, 25 mm/h is the simulator's limit
- **Distance** The ego and target vehicle are placed at varying distances, ranging from 5 to 25 meters. This interval was chosen based on the results from a breaking distance calculator[34] which utilises the formula for stopping distance proposed by the American Association of State Highway and Transportation Officials. Even when assuming an instant reaction time which is unfeasible in practice, at 20 mph which is the lowest speed limit in the UK in school zones and dry conditions, the breaking distance would be 5 meters. Thus, investigating distances shorter than that is not required. At 30 mph which is the speed limit for built-up areas and in wet conditions the stopping distance would be 25 meters, thus justifying the interval.

  It is also important to consider the fact that the number of cases are distributed evenly among 4 different distance buckets: 5-10 meters, 10-15 meters, 25-20 meters and 20-25 meters.
- **Sensor** Two different sensors were used to create the dataset: the Velodyne HDL-64E and Velodyne VLP-16. They were chosen due to the fact that they are some of the most widely used and available sensors as well as the fact that the KITTI dataset (utilised for evaluation in Hau et al.[18]) was created using a Velodyne HDL-64E sensor.

For each combination of sensor and distance 40 different scenarios were created at varying distances, combining for a total of 400 distinct cases. Each of them consists of a labeled ".pcd" file as well as reference and annotated pictures used for manual inspection of results. Although a larger dataset would have improved the quality of our results, considering the resources available and the computationally expensive nature of our approaches, training times would become unreasonably large. Furthermore, a significant amount of manual intervention is required when creating such a dataset in order to maintain quality standards.

(a) Reference image

(b) Annotated image

As presented in Section 2.1.2, we use three different object detection algorithms: PointPillars, SECOND and Part_A2_free (which is the Part_A2 version without predefined anchors). These were selected because they all have state-of-the-art results as well as being extremely popular. In addition to this, each of them is meant to represent a different category of object detector, PointPillar being pillar based, SECOND being voxel based and Part_A2_free being point based.

All experiments were done on a Dell XPS 17 laptop with an Intel I7-11800H, 32 GB of RAM and a Nvidia RTX 3060, running Ubuntu.

## 5.2 Research question 1

*How well does our simulated dataset reflect the findings in Goodin et al.[15]?*

Answering this question is essential for establishing the validity of our work since our synthetic data needs to represent reality as accurately as possible. The first metric we compare is the total number of points detected in a scan, depending on rain rate, as this provides a solid overall understanding of the effects of adverse weather.

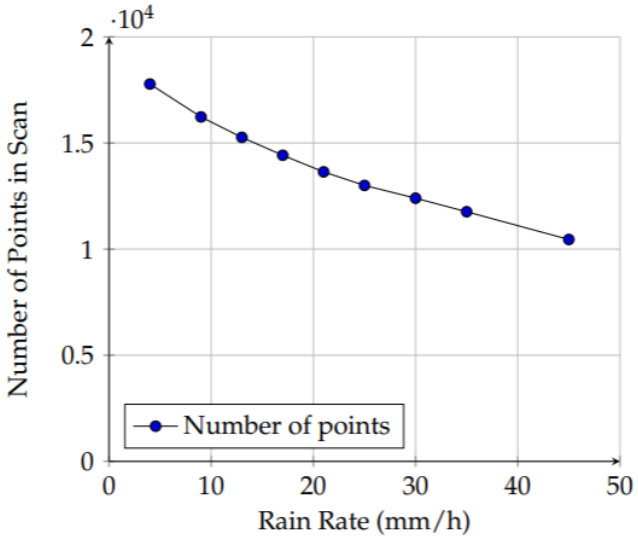



Figure 5.2: Total number of points per rain rate from Goodin et al.[15], using the HDL-64E sensor

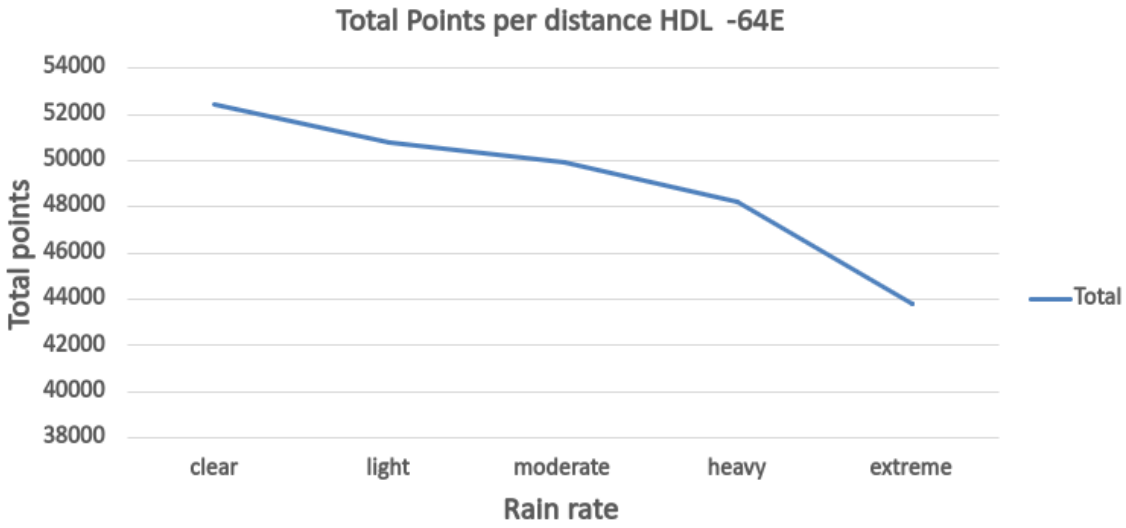



Figure 5.3: Total points for different rain rates: clear - 0 mm/h, light - 2.5 mm/h, moderate - 5 mm/h, heavy - 10mm/h, extreme - 25 mm/h

An important first observation is that our rain rates are not numerically equidistant, thus resulting in the overall different aspect of the results. Another clear difference is that the values for the number of points differ notably, this being caused by differences in scenario and distance, and therefore non-consequential. We are mostly interested in the proportional decrease in total points caused by the rain, for example between the clear case and the extreme one. In our experiment, this accounts for a 17% decrease compared to approximately 25% decrease for the results from Goodin et al.[15].

The reduction in points can also be observed when specifically analysing only detected vehicle points.

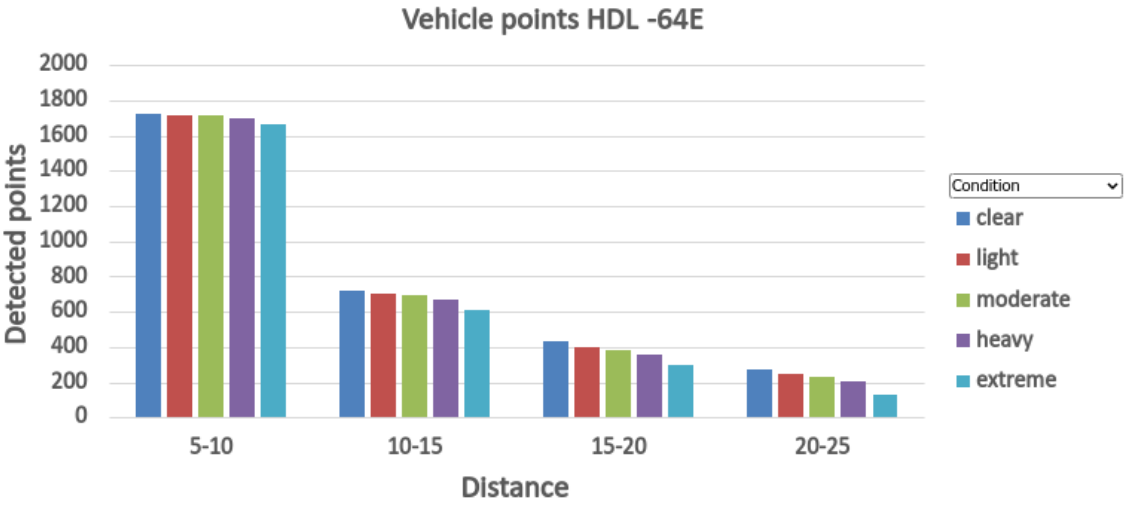



Figure 5.4: Number of vehicle points per distance for different conditions

We can notice that although rain does have an effect on the number of detected vehicle points, when it comes to this particular metric, distance to the target is the most important factor. Further analysing this result, based on Figure 5.5, the decrease in vehicle points is not caused by the target vehicle falling outside of the maximum range of the sensor, as even in extreme conditions the range is 27.3 meters.

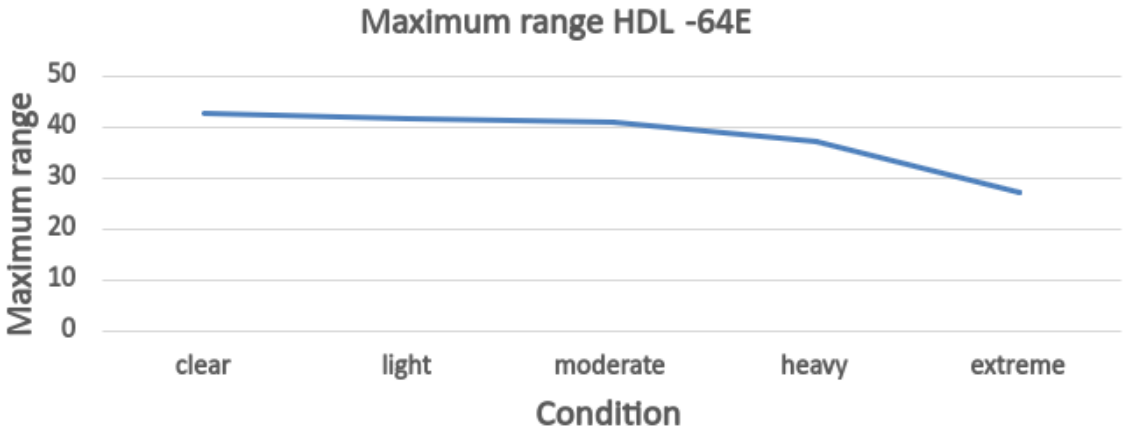



Figure 5.5: Max range for HDL-64E in different conditions

Thus, it is safe to assume that the sharp decrease in the number of vehicle points even at $10-15$ meters is caused by the effect of rain on the point intensities, causing a number of them to fall below the minimum threshold.

The next important metric we analyse is the decrease in intensity caused by adverse weather. To this end, we utilise the following formulas from Gooding et al.[15]:

$$\delta = (P - P_0)/P_0 \tag{5.1}$$

$$\delta = e^{-2aR^b z} - 1 \tag{5.2}$$

Where $P_0$ is the intensity reflected in the absence of rain, $P$ is the intensity in the rain, $z$ is the distance to the target, $R$ is the rain rate and $a, b$ are constants defined as $0.001, 0.6$. Thus, using the intensity values for our clean scenarios, we can utilise the formula to calculate the predicted intensity for each rainy case, which we can then compare to the recorded values.

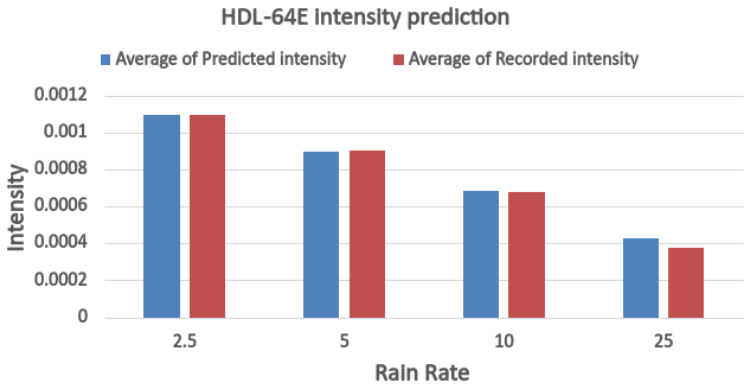



Figure 5.6: Predicted and measured intensity for HDL-64E sensor at differing rain rates, calculated using Equations 5.1, 5.2

The predicted results for intensities compare almost perfectly to the recorded ones, especially considering the fact that the formula has to be averaged across all vehicle points because it is impossible to individually match each pair of points from two different readings. Therefore, averaged intensity and distance values were used, still producing a very accurate prediction.

Furthermore, the results for the VLP-16 sensor also closely match our predictions, indicating that the adverse weather intensities are calculated using the same formula and the same values for coefficients $a, b$.

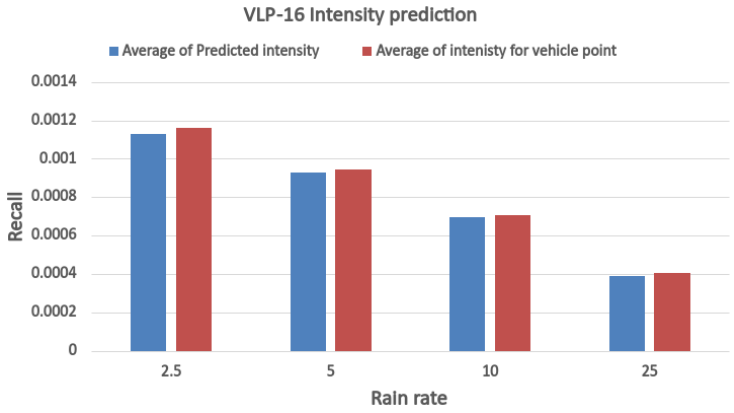



Figure 5.7: Predicted and measured intensity for VLP-16 sensor at differing rain rates, calculated using Equations 5.1, 5.2

Although not significant, the results for the VLP-16 sensor have more of a discrepancy between the predicted and recorded values, caused by the the scarcity of points in the point cloud, making our averaged formulas more prone to noise.

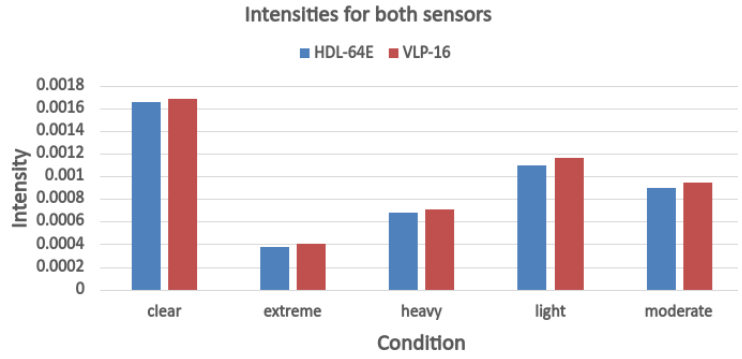



Figure 5.8: Measured intensity for HDL-64E and VLP-16 sensors in different conditions

Further comparing the sensors directly, we one again see that they closely reflect each-other.

### 5.2.1 Discussion

To answer our proposed question, our results for the HDL-64E and VLP-16 sensors accurately represent the findings in Goodin et al.[15]. This conclusion is crucial because it supports the validity of all further evaluation results. Nonetheless, comparing the number of total and vehicle points between our two sensors (Figure 5.4, 5.3, Appendix B.8, B.7) we can observe that the HDL-64E is overall the superior sensor, performing better in all metrics. Thus, although both sensors are utilised in our evaluation, the HDL-64E is given the most attention.

## 5.3 Research question 2

*How does the base detection rate of our pipeline compare to related works? Can we establish reasonable thresholds for all sensors and object detectors?*

Answering this question is essential as it serves to contextualise all further results in this chapter and establishes a performance baseline which can be compared to later. Having chosen to replicate ORA methodology, we use the results in Haut et al.[18] to asses our own findings. For the second part of the question, we intend to establish realistic thresholds for our chosen metric specific to each possible setup. For **RQ2, RQ3, RQ4, RQ5** we use IoU as our chosen metric seeing as it is the one mostly used in related works, including Hau et al.[18].

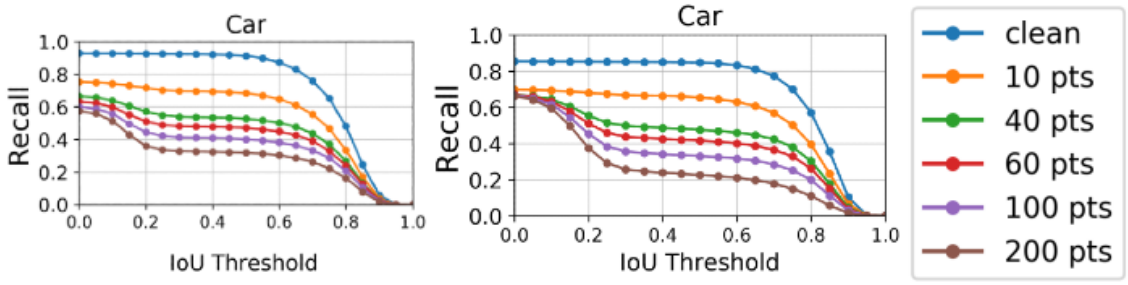



Figure 5.9: Random ORA results using Point-RCNN (left) and Point-GNN (right) from Hau et al.[18] At this stage, we are only concerned with the "clean" results depicted in blue, representing the performance of the detectors on unperturbed data.

These results were compiled using the KITTI dataset that only includes clear weather data and utilises a Velodyne HDL-64E sensor. Thus, for the most direct comparison, we recreate this exact setup using our synthetic dataset. Furthermore, as results only include scenarios where the target is between 5 and 20 meters from the ego vehicle, we only utilise cases that fall within that range.

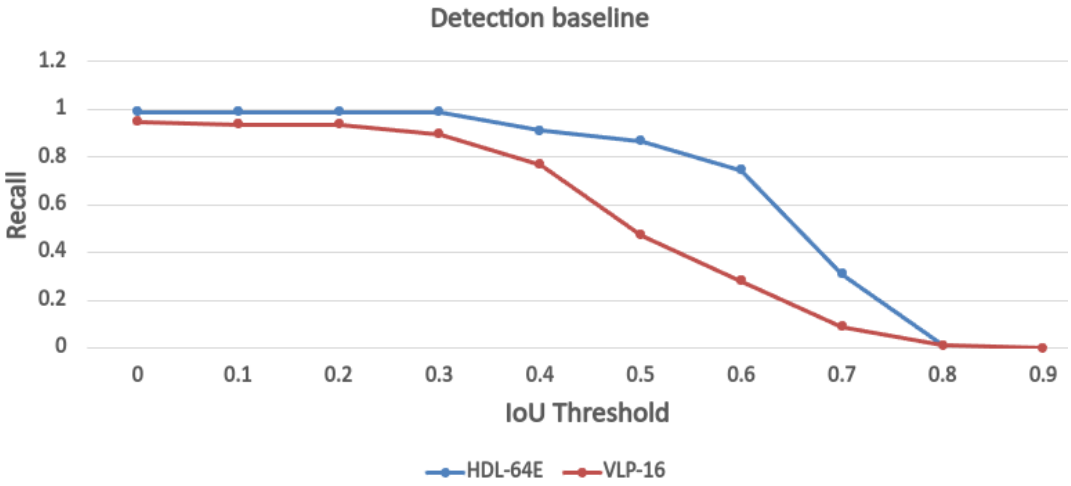
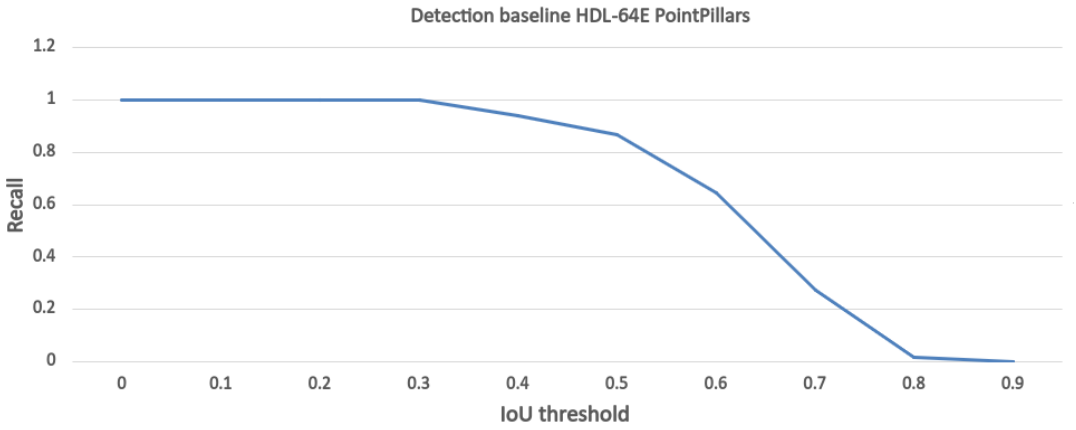



Figure 5.10: Results for HDL-64E and VLP-16 sensors in clear scenarios, averaging the performance across all 3 detectors used

By first comparing the results between our two sensors, it is easy to observe that as expected the HDL-64E is overall the superior sensor, having better recall for all IoU thresholds.

As mentioned before, we mostly focus on the direct comparison between our results on the HDL-64E and those presented in [18]. The first thing to notice is that for IoU values $< 0.3$, our results have a perfect recall of 1 while both detectors from [18] completely miss some of the cases. This might either be caused by the simulated data being generally more consistent or by the synthetic scenarios we have chosen being more forgiving. However, the performance of our detectors starts to decrease earlier but at a slower rate, the results from the KITTI dataset staying relatively level for thresholds between 0 and 0.6. By manually inspecting case by case, we have determined that this difference is most likely caused by the quality of the generated ground-truth boxes that although relatively accurate, do not reflect reality as well as the manually annotated KITTI labels. Nonetheless, it is important to consider that for an IoU of 0.5 the values between the two datasets are still very similar, making this value a candidate for our chosen threshold.

Having looked at the comparison it is also crucial to establish the IoU thresholds for all our different setups. Even though we continue to investigate the recall at all possible IoU values, we require realistic values that fit the clear weather scenarios so that we can better understand the effects of potential attacks and adverse weather conditions. For example, we first look at the combination of the HDL-64E sensor and PointPillars detector.




For this setup, a reasonable value for the threshold would be 0.5, with a recall value of 0.866. Choosing a lower threshold would lead to many false positives while choosing a higher value would deteriorate the detector's performance excessively. Furthermore, at this threshold, the results from Hau et al.[18] have similar recall values allowing for easier comparison for future research questions.

Using similar reasoning and using the graphs Appendix B.2, B.3, B.4, B.5, B.6, we can choose the following IoU thresholds:

- **HDL-64E** PointPillars - 0.5, SECOND - 0.5, Part_A2 - 0.5
- **VLP-16** PointPillars - 0.4, SECOND - 0.4, Part_A2 - 0.4

Although there are minor differences between the object detectors at the mentioned values, choosing a single threshold simplifies the evaluation process while still maintaining realistic performance.

### 5.3.1 Discussion

Considering the large differences between the used datasets and overall setups, our findings compare well to the ones in Hau et al.[18], in particular for the HDL-64E sensor. As previously mentioned, one of the main factors for the disparity between the results is the quality of the ground-truth bounding boxes. Significant effort has been allocated to refining the ground-truth bounding box creation and augmentation systems, reaching optimal results considering the restrictions. For some cases there are simply not enough vehicle points to determine an accurate approximated bounding box or the box is misaligned because of the angle from which the target vehicle is seen. This is particularly prevalent in cases where the target is perceived from one of the corners, where the Principal Component Analysis (PCA) used in the Open3D library to create oriented bounding boxes leads to misaligned boxes.

As for the thresholds, we consider the values chosen to be reasonable, especially for the HDL-64E sensor where they would also be good fits for the results in Hau et al.[18]. As the VLP-16 sensor is a cheaper and simpler alternative, performance is lower thus requiring smaller thresholds. Nonetheless, precisely because of the reasons mentioned, the VLP-16 is widely used therefore being an important research target.

## 5.4 Research question 3

*How do the results of our recreated Random ORA compare to the original ones from Hau et al.[18]?*

In order to answer this question, we analyse the results of Random ORA for different budgets from our work and Hau et al.[18] using the two methods primarily used in the cited paper:

- Firstly, we look at the impact of different budgets attacks on recall with regards to IoU threshold. The graphs from Figure 5.9 are the reference point to which we compare our own results.

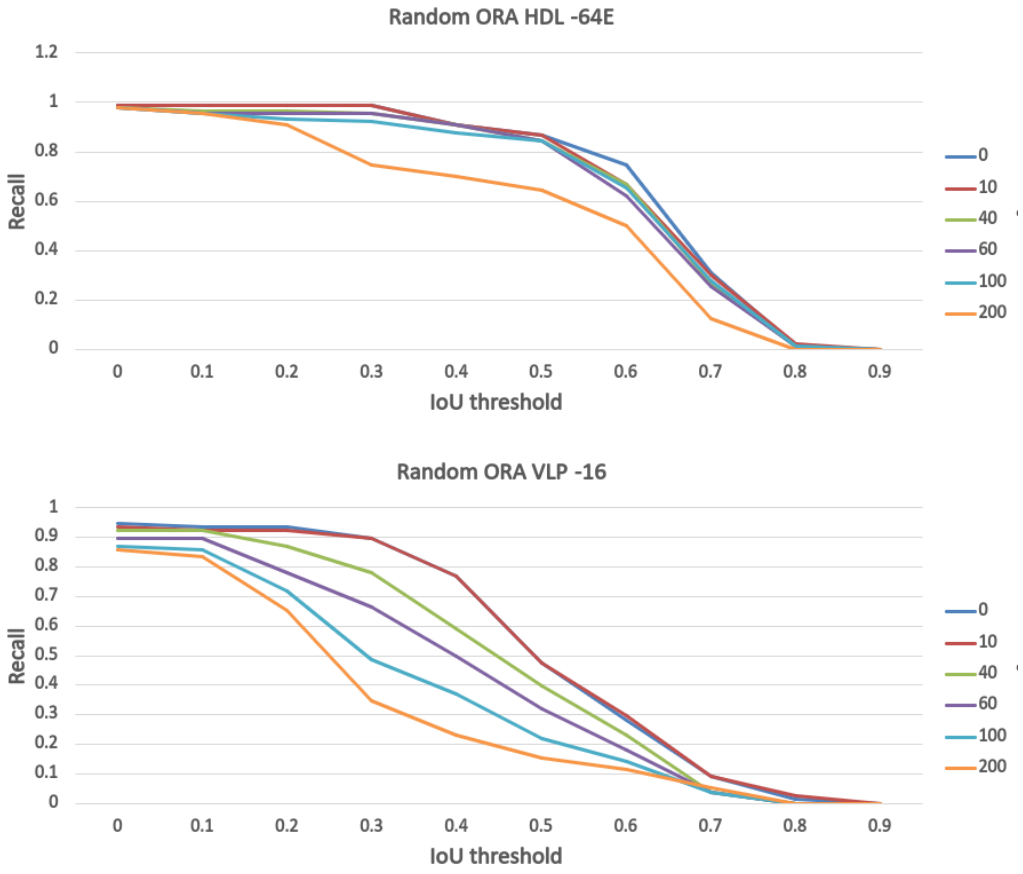



Figure 5.11: Results for HDL-64E (top) and VLP-16 (bottom) sensors in clear scenarios, averaging the performance across all 3 detectors used at all budgets

We can observe that the impact of Random ORA on on the HDL-64E sensor is greatly reduced compared to previous findings. Furthermore, as the HDL-64E sensor outputs a very large amount of 3D points, only the biggest budget random ORA has a significant impact on the detection performance. On the other hand, the attacks have more of an impact even for lower budgets on the VLP-16 sensor, the results being closer to the ones in Hau et al.[18]. For our chosen IoU threshold values, we observed a decrease in recall of 0.22 between the clean and 200 budget scenarios on the HDL-64E sensor and a decrease of 0.544 for the VLP-16 sensor, which is close to the values reported in [18].

- Secondly, we analyse the impact of different budget attacks on recall with regards to distance (for a fixed IoU threshold). This analysis provides further insights into the real-world performance and implication of Random ORA.

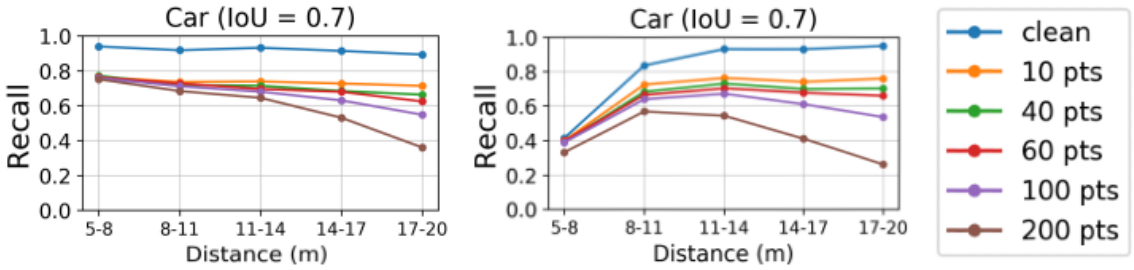



Figure 5.12: Recall metric by distance using Point-RCNN (left) and Point-GNN (right) from Hau et al.[18]

It is important to observe that in general performance on the unperturbed data remains quite

stagnant even at larger distances, with the notable exception of the Point-GNN detector which struggles at detecting close vehicles. In contrast, the effectiveness of the attack increases with distance, this probably being caused by the decreased number of vehicle points in the point cloud.

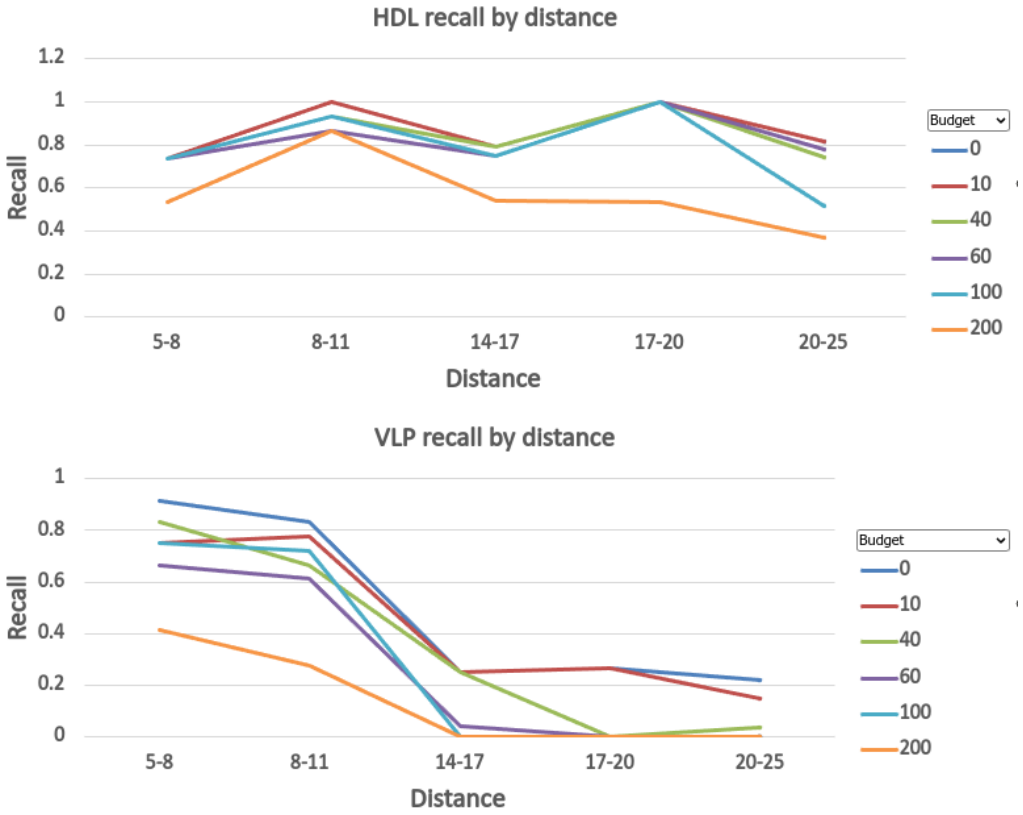



Figure 5.13: Recall by distance for HDL-64E (top) and VLP-16 (bottom) sensors in clear scenarios, averaging the performance across all 3 detectors used at all budgets. The IoU thresholds discussed in the previous section were used (0.5 for the HDL-64E and 0.4 for the VLP-16)

First looking at the HDL-64E results, we can observe similar trends to those in the cited work, namely that the performance on unperturbed data is relatively similar for all distances, while the attacks become more effective with distance. The fact that our clean performance is not completely the same across all distances can be attributed to the overall much smaller size of our dataset, being more prone to noise caused by outlying cases. The effect of the attacks on the VLP-16 sensor is even more pronounced, especially for distances greater than 14 meters, recall dropping to 0.

### 5.4.1 Discussion

The first takeaway from the findings in this section is that less complex sensors such as the VLP-16 are extremely vulnerable to ORAs, especially at medium to large distances. This is a very important security vulnerability considering that these are the distances most commonly encountered in driving scenarios.

Overall, our results do not seem to accurately compare to the ones presented in Hau et al.[18], specifically for the HDL-64E sensor when not taking distance into account. The data presented can however be somewhat misleading, the main reason for the discrepancy being attributed to the object detectors used.

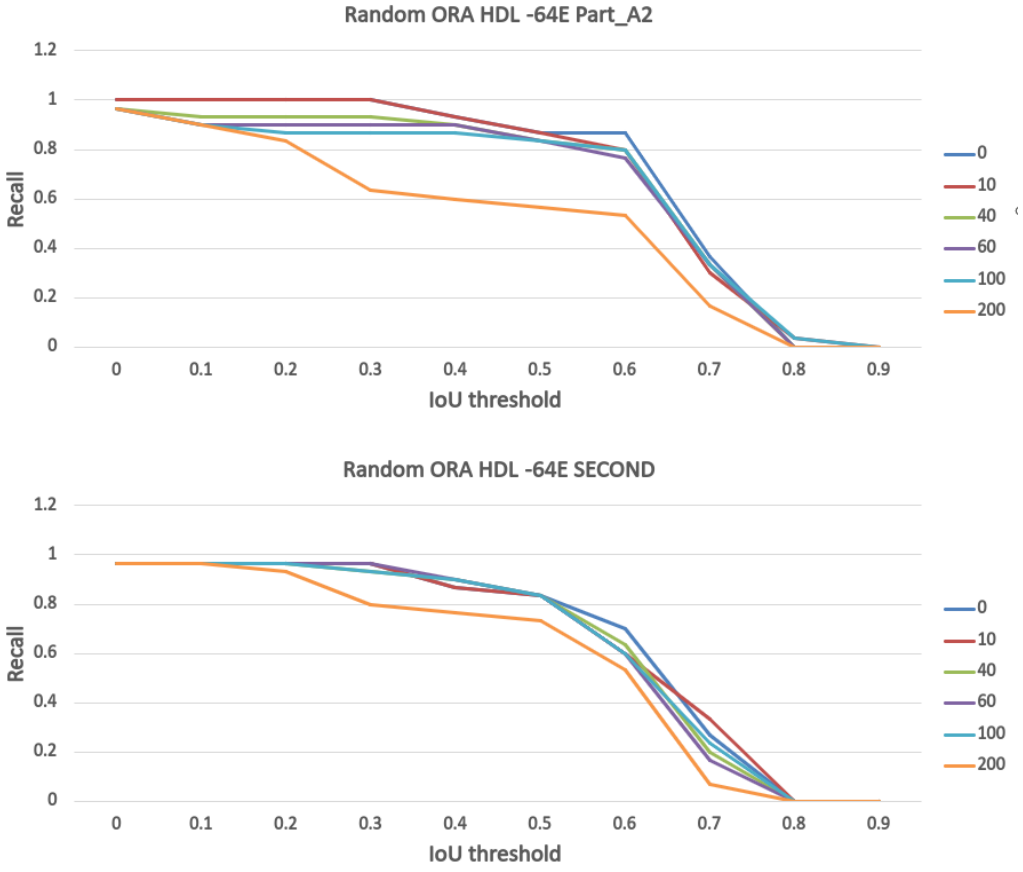



Figure 5.14: Results for Part_A2_free(top) and SECOND(bottom) detectors in clear scenarios

We can observe that while the SECOND detector is only marginally affected by the attacks, Part_A2 is influenced the most out of all detectorss. This becomes even more relevant when considering that Part_A2 is an improvement of the Point_RCNN detector used in Hau et al.[18].

Taking all of this into account as well as the fact that the recall values by distance compare favourably, we can assume that the inconsistency can be justified by the use of newer, more efficient object detection algorithms as well as the previously presented differences in the baseline performance. Thus, as the methodology is simple, we presume that our recreation is valid and can be used as reference for our future evaluation.

## 5.5 Research question 4

*How does rain impact the effectiveness of Random ORA?*

Answering this research question constitutes one of the main objectives of this work, thus necessitating significant attention. Having already analysed the effects of adverse weather on LiDAR and the performance of Random ORA in clear scenarios, we can now investigate the attack's performance in rain. For this section, we continue using IoU as our metric in order to have a direct comparison to previous findings.

We first analyse the results for the HDL-64E sensor across all detectors from Figure 5.15. Comparing the extremes, we can observe a 19% decrease in performance for the maximum budget between clear and extreme rain conditions, while the change for the unperturbed data is only 13%. Therefore, we can already observe that adverse conditions have a disproportional effect on the performance of the attacked point clouds. Moreover, the recall for the unperturbed data only drops significantly for extreme conditions, while the attacks with the biggest budgets are sensitive

to even a lesser amount of rain.

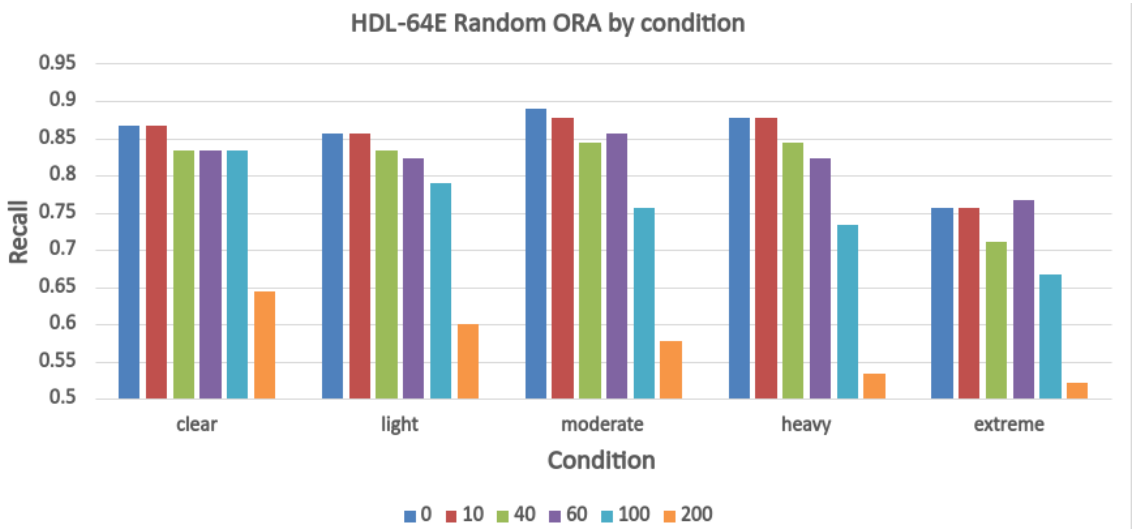



Figure 5.15: Recall for IoU = 0.5, on the HDL-64E sensor using different budgets of attack. Averaged for all detectors

A recall value of just 0.5 for the perturbed data in extreme rain raises significant safety concerns, having the potential for severe real life consequences. Similar results to the ones discussed can also be observed for the VLP-16 sensor (Appendix B.9).

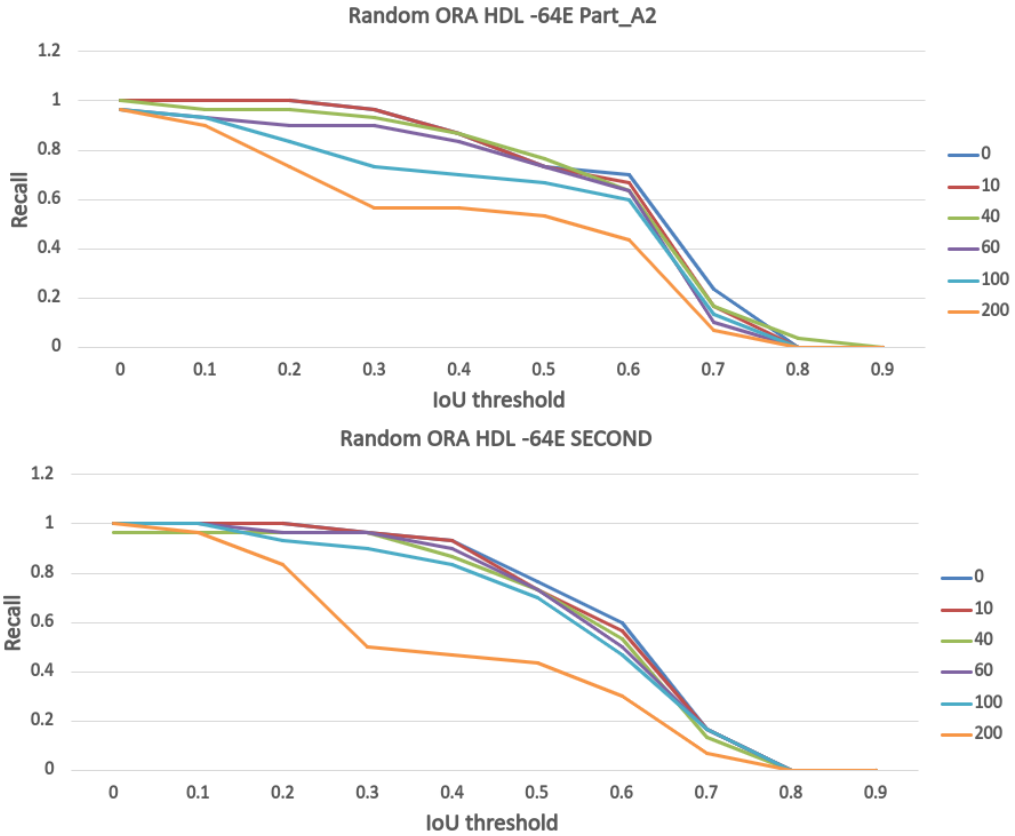



Figure 5.16: Results for Part_A2_free(top) and SECOND(bottom) detectors in extreme weather conditions

Another crucial aspect to consider is how the changes in condition affect each particular detection algorithm. Analysing Figure 5.14 from the previous section we observed that in clear conditions the SECOND detector is much more resilient to ORAs than Part_A2. However, for extreme rain (Figure 5.16) this trend seems to be reversed, Part_A2 experiencing a proportional overall decrease in performance, while for the SECOND detector the attacks become significantly more efficient compared to the clean scenario. The same particularities in the performance of the SECOND detector can also be observed for the VLP-16 sensor (Appendix B.10), further indicating that this behaviour is caused by the detection algorithm.

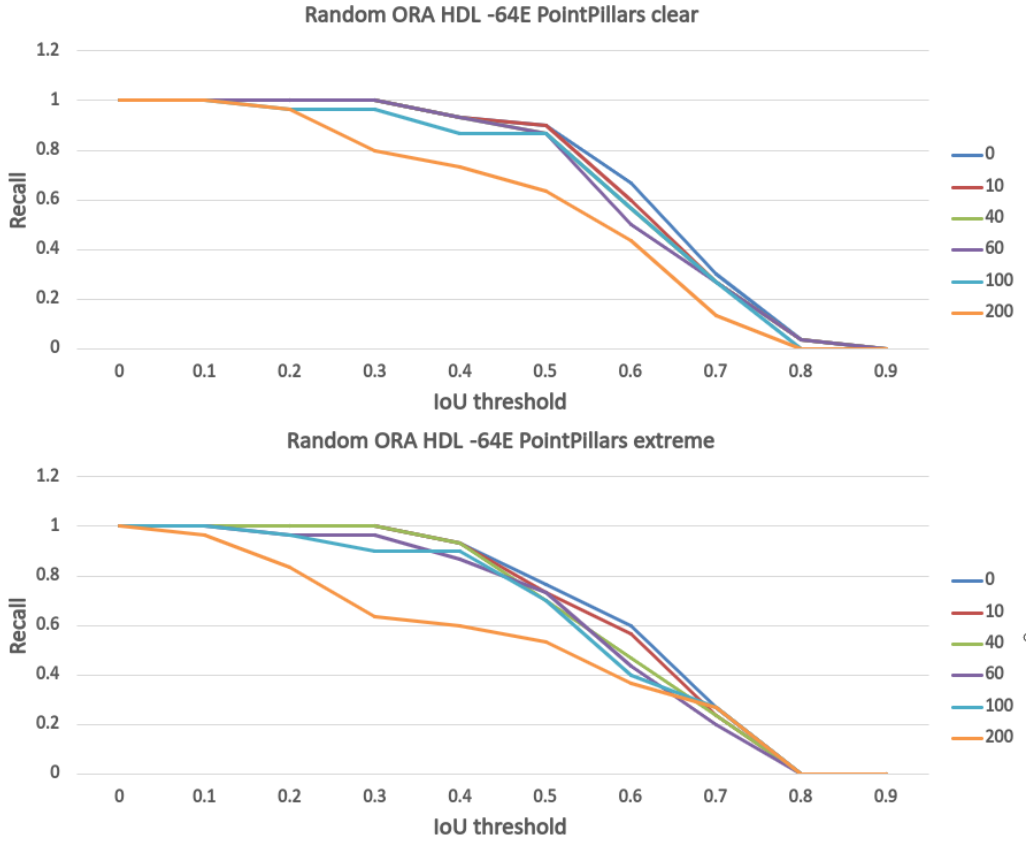



Figure 5.17: Results for PointPillars using an HDL-64E sensor in clear (top) and extreme (bottom) conditions

In most aspects, the performance of the PointPillars detector seems to fall somewhere in between Part_A2 and SECOND, being affected by the attacks in a significant manner for both clear and extreme conditions, while also exhibiting a further vulnerability to attacks in the extreme case.

### 5.5.1 Discussion

Overall, it is clear that rain has a major impact on the performance of Random ORA, making the strategy even more potent, which validates our initial hypothesis. Furthermore, it is crucial to consider recall values for attacked point clouds were already low in clear conditions, hence the fact that the decrease in performance is proportionally larger than for unperturbed cases means that these attacks have the potential to be catastrophic in a real scenario.

Another important takeaway from these results is that there are significant differences in the behaviour of object detectors in adverse weather conditions. Most research into this field uses the KITTI dataset for training and evaluation, completely disregarding performance in adverse weather. Judging by our findings, we consider that there should be more emphasis on ensuring object detectors are well suited to rainy conditions.

## 5.6 Research question 5

*How well do our novel algorithms perform?*

In order to answer this question, we analyse the results of each our approaches from Chapter 3, while mainly utilising IoU as our target metric in order to have a straightforward comparison with the findings from previous questions.

To begin with, we look at the two proposed heuristics, involving distance and intensity:

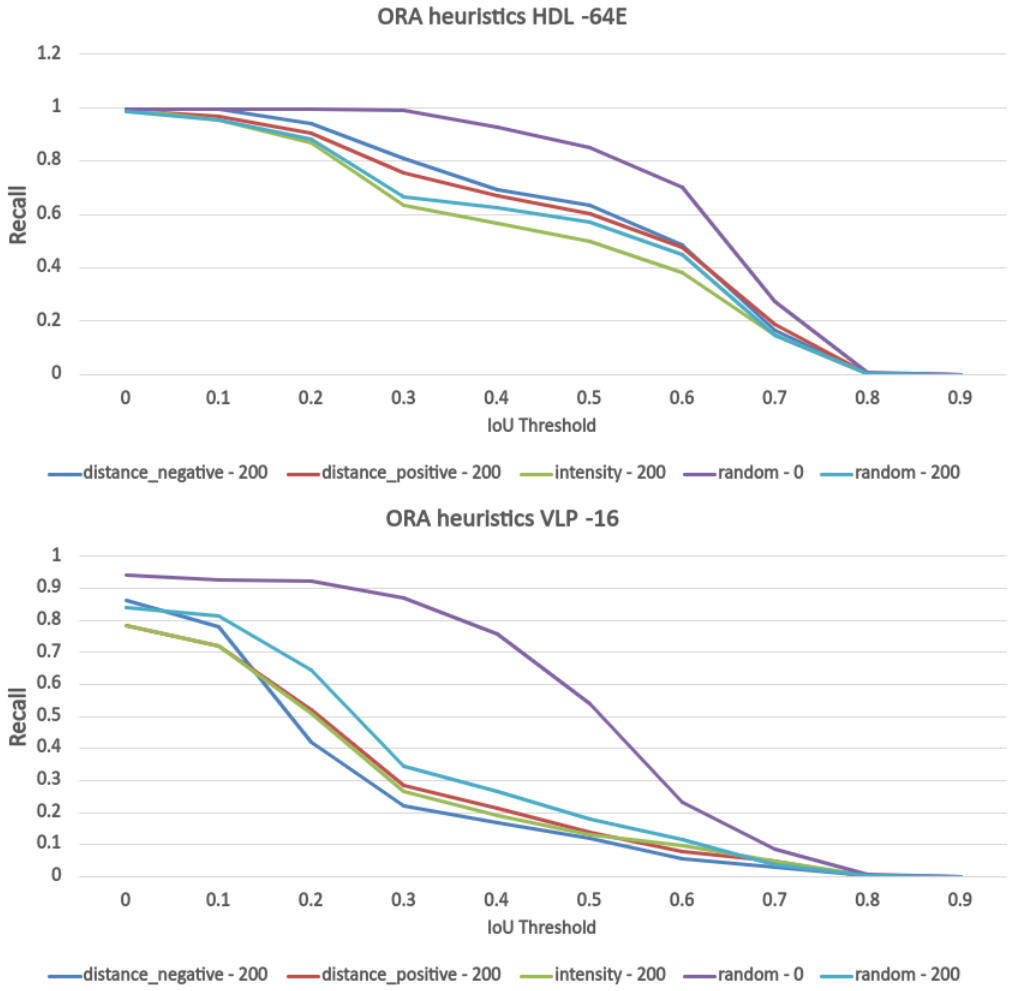



Figure 5.18: Results for distance and intensity heuristics for HDL-64E (top) and VLP-16(bottom), using all detectors and in all weather conditions. All heuristics were applied with a budget of 200, the "random-0" being the unperturbed case only included as a reference point. For the distance heuristics, positive and negative refer to the direction the points were shifted (positive - further away, negative - closer)

The reason for the inclusion of these strategies, especially considering that the intensity heuristic is not feasible in a real scenario, is to illustrate that although the selection process for Random ORA is simplistic, because of the very complex problem and data used it is still difficult to improve on. Compared to the performance on the unperturbed data, especially on the HDL-64E, the improvement that the heuristics make is limited.

We can also observe that the heuristics perform very differently on the two sensors. On the VLP-16, all of the approaches have a bigger impact than Random ORA, shifting the closest points to be even closer to the ego vehicle having the best results. However, the contrary is true for the HDL-64E, the "distance_negative" heuristic having the worst performance, even worse than ran-

dom selection. For this sensor, only the intensity heuristic manages to provide a small improvement.

We next analyse the results of both versions of BORA. As mentioned in Section 3.2, from now on our results only utilise one object detection algorithm at a time. Our main focus is the PointPillars detector as it is the least computationally expensive as well as the fact that in most aspect it's performance falls between the SECOND and Part_A2 algorithms. We start with our cross validation strategy which attempts to find underlying features in the data across all cases, not requiring any further training for unseen data.

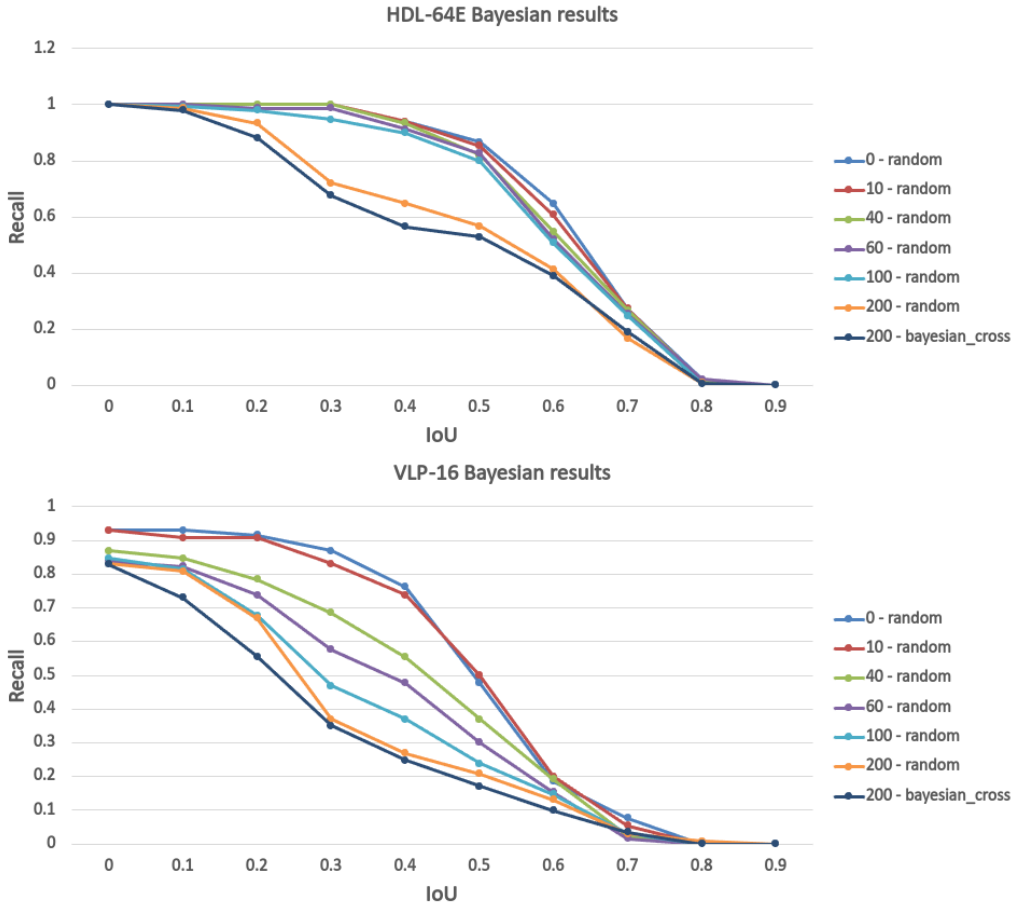



Figure 5.19: Recall for cross-validation BORA for HDL-64E (top) and VLP-16 (bottom), using PointPillars in all conditions

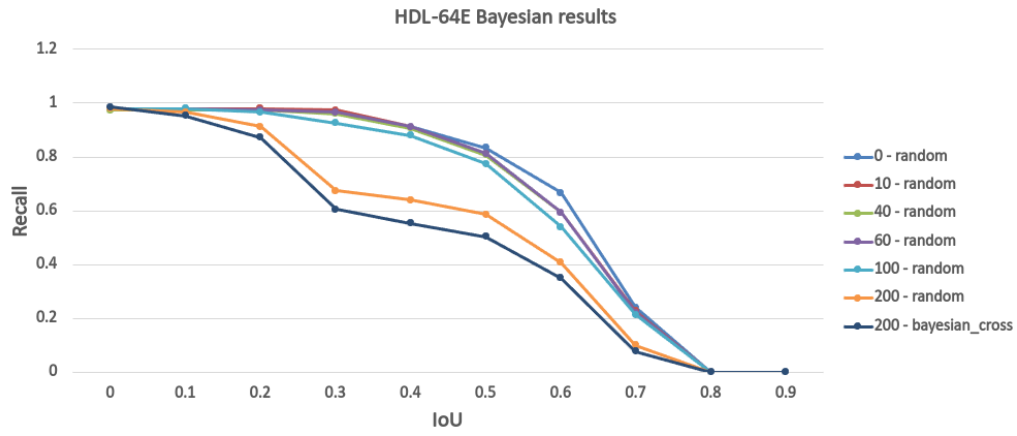



Figure 5.20: Recall for cross-validation BORA on the HDL-64E sensor, utilising Part_A2

For each sensor, the algorithm was run in 5 folds, each being able to evaluate the mean IoU across all train cases 100 times, after which it was evaluated on the validation set. On the VLP-16 sensor, we can observe a decrease in performance for low IoU values which can be relevant in the case an adaptive thresholding approach is used where this sensor would likely have a low threshold. However, especially on the HDL-64E, the improvement is only marginal. Similar results can be observed for the Part_A2 detector in Figure 5.24. Taking into account the fact that this method requires no further calculations if applied on unseen data making it faster than Random ORA, even the fact that it marginally outperforms Random ORA at all thresholds would make this approach worth considering.

As mentioned in Section 3.6.2, the second BORA approach attempts to individually maximise performance for each case, being allowed to evaluate the objective function 100 times. Although this strategy does not conform to our threat model, it is nonetheless presented as a reference for the hypothetical optimal results and could perhaps also be leveraged in an offline attack.

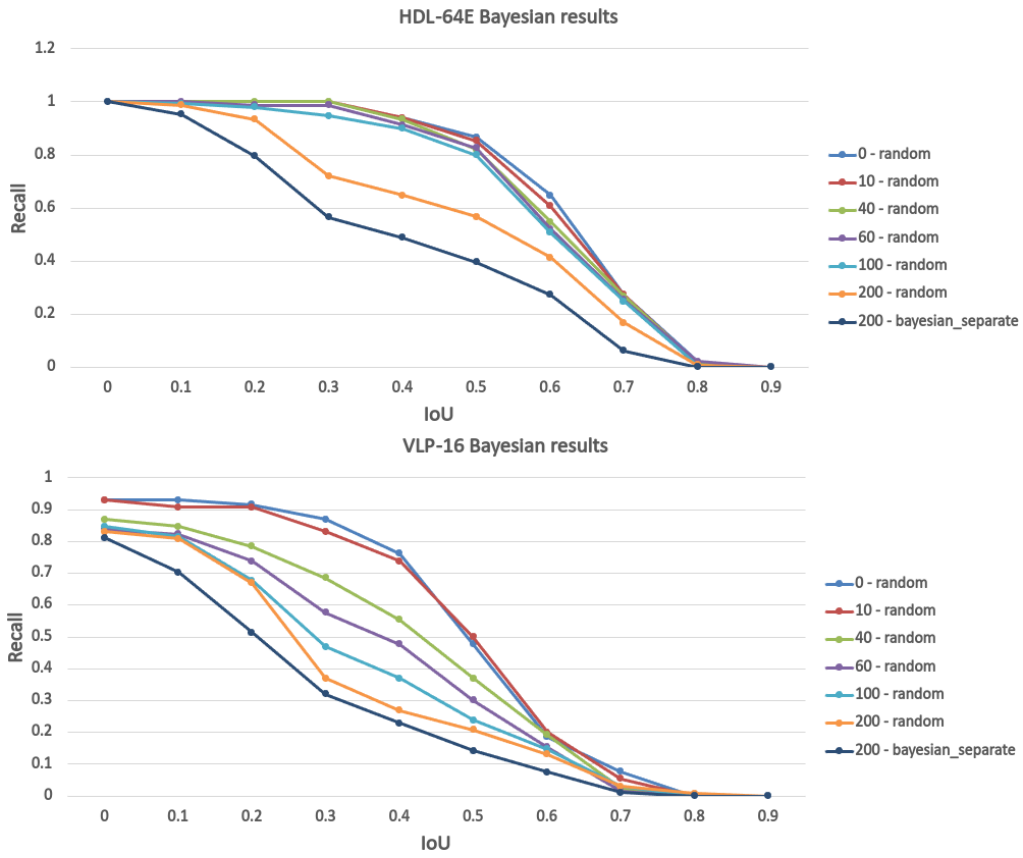



Figure 5.21: Recall for separate BORA for HDL-64E (top) and VLP-16 (bottom), using PointPillars in all conditions

We can observe that for the HDL-64E sensor, there is a significant improvement when compared to Random ORA. However, on the VLP-16 sensor, the difference is not as large, suggesting that due to the sensor's reduced overall capabilities, a random ORA attack is powerful enough to extract most available performance.

Next, we analyse the results for our GORA approach, which is the most flexible and best performing out of our novel strategies. Similarly to our first BORA approach, it is evaluated using cross-validation ensuring that our results are not biased by our training set. As this approach is meant to generalise effortlessly, it can be applied to unseen data without any additional computation, making it faster and more efficient than Random ORA. For each sensor, the algorithm was run in 5 folds of 30 generations each, using a population of 50 individuals.

When compared to previous presented approaches, GORA has similar performance while dis-

regarding the drawbacks. Furthermore, we can observe that GORA performs particularly well for our IoU thresholds chosen in Section 5.2 (0.5 for HDL-64E and 0.4 for VLP-16), making its performance more meaningful in a real scenario.

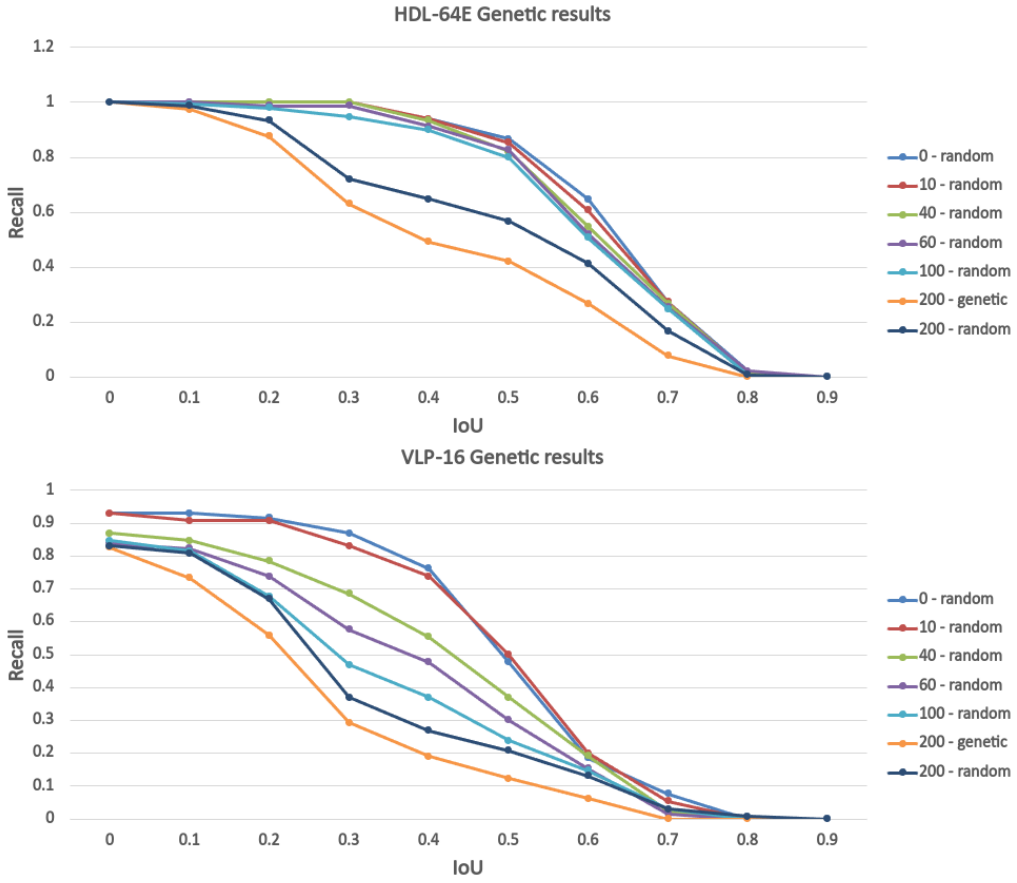



Figure 5.22: Recall for GORA for HDL-64E (top) and VLP-16 (bottom), using PointPillars in all conditions

In particular for the VLP-16 sensor, at our chosen threshold of 0.4, the difference between GORA and Random ORA using a budget of 200 points is similar to the one between the latter and Random ORA using a budget of 100 points.

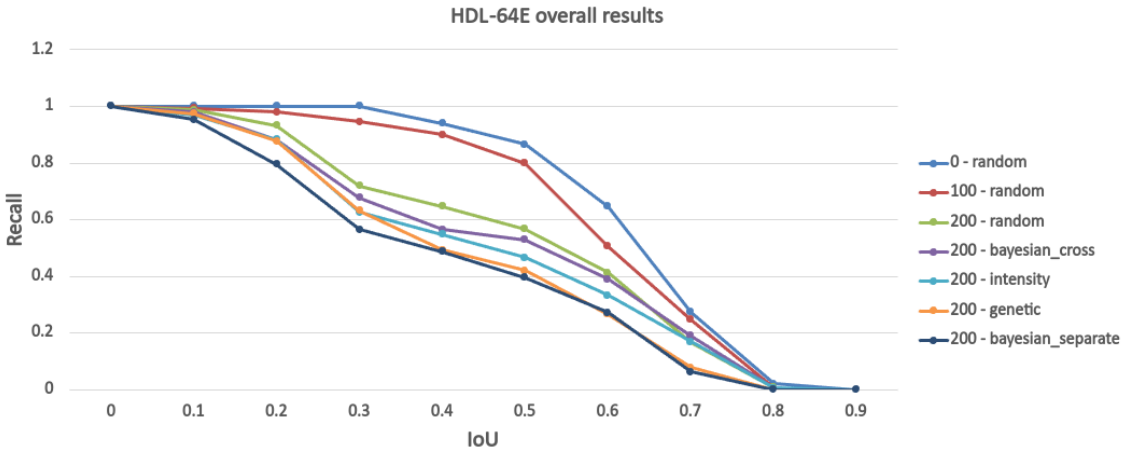



Figure 5.23: Recall for all strategies on the HDL-64E sensor using PointPillars

Comparing GORA and BORA with the the best heuristic for each sensor, we can observe that for the HDL-64E, GORA outperforms the infeasible intensity heuristic and has very similar performance to separate case BORA. For our chosen threshold of 0.5, recall would fall from 0.86 for the unperturbed data to 0.56 for Random ORA with 200 budget, 0.42 for GORA and 0.39 for separate case BORA.

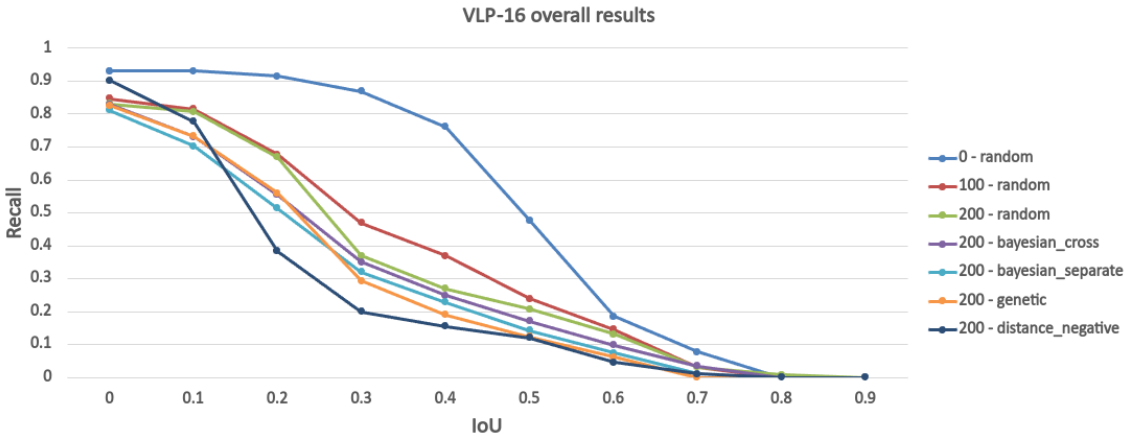



Figure 5.24: Recall for all strategies on the VLP-16 sensor using PointPillars

Analysing the results for the VLP-16 sensor, it is very interesting to observe that GORA outperforms separate case BORA, but the negative distance heuristic has the best performance of all. This comparison is not completely fair, seeing as both BORA and GORA operate with a fixed positive shifting distance, thus highlighting a potential area of improvement for our algorithms, specifically for the VLP-16 sensor. At our chosen IoU of 0.4, recall would fall from 0.76 for the unperturbed data to 0.26 for Random ORA with 200 budget, 0.18 for GORA, 0.22 for separate case BORA and 0.15 for the distance heuristic.

### 5.6.1 Discussion

Overall, we can conclude that our novel approaches outperform random ORA across all cases, showing significant improvement at the relevant IoU thresholds. In a real scenario, GORA is the most robust option for both sensors, being efficient while not sacrificing any performance. It is also crucial to take into account the fact that the GORA and BORA results presented were obtained using very limited computational resources, having the potential to achieve even better figures. Our findings also highlight the prospect of attempting to also optimise the shifting distance in our novel approaches, specifically for the VLP-16 sensor.

Most results were only presented for the PointPillars algorithm because it has the most representative performance while requiring less training time. It is important to consider that some sensor and detector combinations require up to a few days to complete the training process.

## 5.7 Research question 6

*Which target metric yields the best result for our approaches?*

In this section we utilise confidence as our target metric and compare the results to the ones from the previous research question. In order to answer this question, we mainly focus on the VLP-16 sensor due to the fact that the HDL-64E sensor has little potential for an impactful attack owing to its high confidence scores. Analysing Figure 5.26, we can observe that for the unperturbed data recall values are very high, as well as the fact that attacks don't have a significant impact. Even if we were to select a high confidence threshold of 0.7, recall for the attacked data would still not fall below 0.6. This is vastly inferior to the recall values obtained when utilising IoU as our target metric.

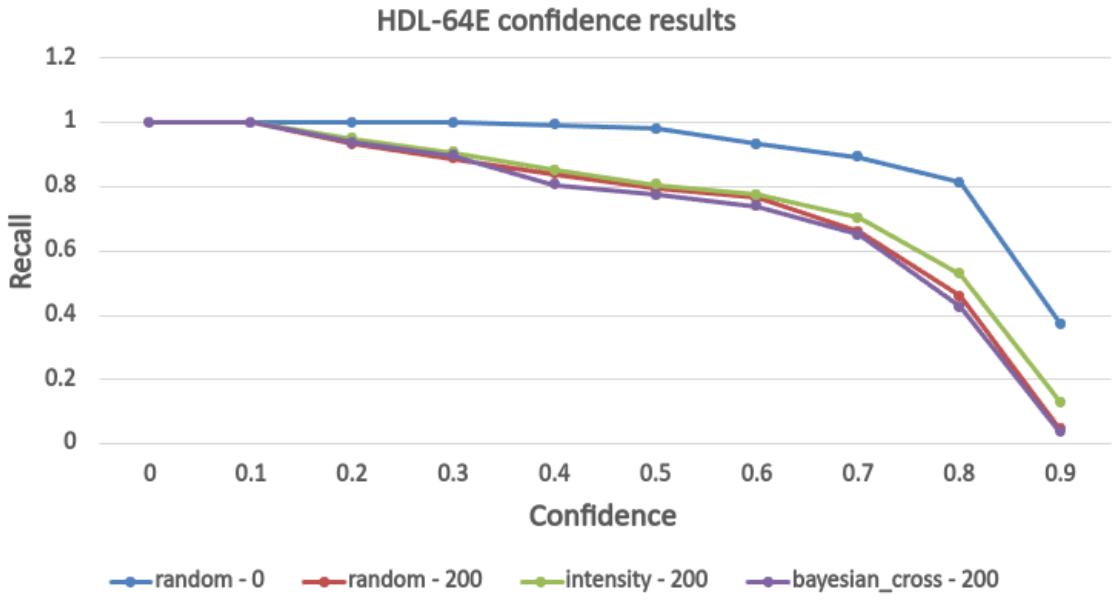



Figure 5.25: Recall for HDL-64E sensor using PointPillars

Being a less robust sensor, the VLP-16 presents more of an opportunity for this strategy of attack.

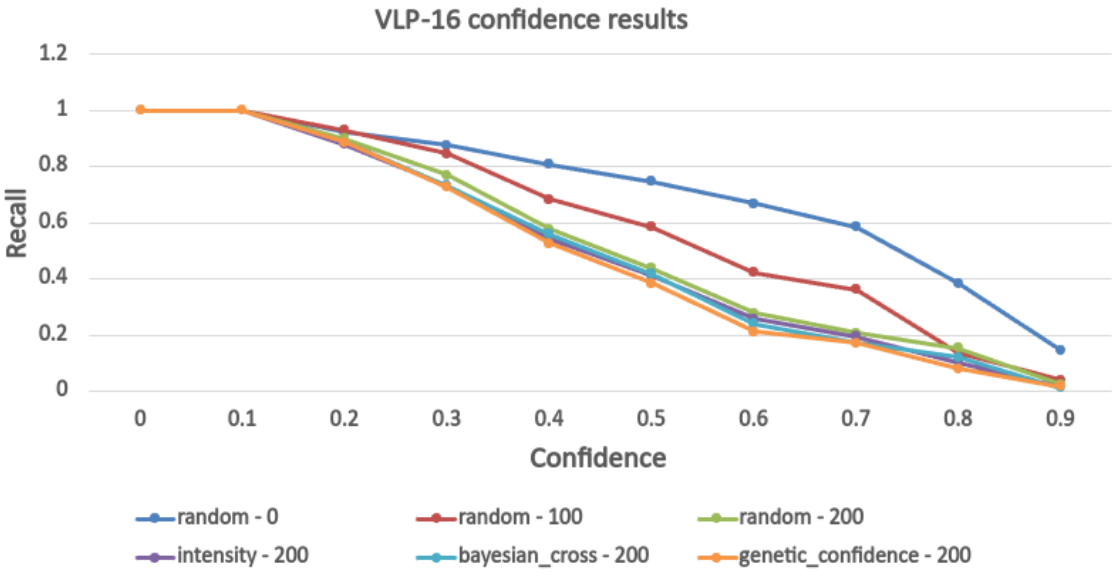



Figure 5.26: Recall for VLP-16 sensor using PointPillars

Sadly, we can observe that although there is a significant difference in confidence between the unperturbed data and Random ORA, our approaches fail to significantly improve its performance. This indicates that attack strategy might not have a major impact on the confidence of object detectors, being more influenced by the budget of the attack. At a reasonable threshold of 0.4, the recall values are: 0.8 for unperturbed data; 0.57 for 200 budget Random ORA; 0.56 for cross-validation BORA; 0.52 for GORA.

### 5.7.1 Discussion

To answer our research question, IoU is clearly the superior target metric when it comes to ORAs. This does not however mean that using confidence as the target metric has no merit. Considering all the presented results, minimising confidence for an object removal attack can still have a significant

impact on a less complex sensor such as the VLP-16, although the results do not compare favourably to the ones obtained in the previous sections. Nonetheless, the investigation into the effectiveness of a confidence approach for ORAs has proven that there is potential to this strategy, which could be exploited in the future using a specialised methodology or a larger attack budget.

# Chapter 6

# Conclusion and future work

In this work, we successfully achieved all our objectives. We created a synthetic dataset and developed a detection pipeline, as presented in Chapter 4. Additionally, we validated the dataset and pipeline, and comprehensively evaluated existing attacks under various weather conditions using different sensors and detectors in Chapter 5. Furthermore, we have also introduced our own state-of-the-art novel approaches, GORA and BORA, as described in Chapter 3.

Although due to its nature the project involves considerable amounts of research and evaluation, a significant number of technical challenges also had to be overcome. The field of LiDAR object detection is relatively recent, experiencing rapid development while still not being very widespread. What this means in practice is that most tools such as simulators and object detectors are complex to set-up and use, sometimes having limited documentation and assuming the user is very proficient. A notable amount of effort was necessary just to establish the basis of our detection pipeline and utilise the MAVS simulator at a basic level. Additionally, managing different formats of data and bounding boxes proved to be difficult, especially considering our decision to automatically generate ground-truth bounding boxes. Sadly, although this approach justifies its use because of the flexibility provided, it has proven to be very challenging to obtain quality ground-truth bounding boxes and impossible to get them to the level of manually annotated boxes, leading to discrepancies in our comparison with previous works.

Another significant challenge that had to be overcome was improving the performance of GORA and BORA. Although they demonstrated potential from the start, initial results were considerably worse than the final ones. All design and implementation choices presented in Chapter 3 were difficult to make, requiring significant experimentation and analysis. This need for adjustments was compounded by the main limitation we faced in developing this work, namely the limited computational resources available and the time required to train our approaches. Seeing as a full training process which could take up to three days is necessary to extract meaningful insights about the effects of a potential change in the algorithm, it became extremely burdensome to rapidly iterate on our methodologies. Furthermore, an improvement method later discovered would invariably lead to the need to scrap all previous results and retrain them, creating very difficult decisions. Our approach for evaluating these novel strategies was also overambitious, aiming to evaluate each of our three approaches, using two different target metrics, on two sensors and on all three object detectors. Considering each of the combinations could take days to train, this was unrealistic, leading to our evaluation section only focusing on some of the mentioned factors. Nonetheless, we consider that through answering all of our research question we have showed the validity of our findings, proving the merit of the novel strategies.

Another important topic to discuss is the recent publication of You et al.[40], which has won the best paper award at the SafeThings workshop 2024. This work also tackles the problem of improving the ORA methodology, proposing a novel attack strategy that involves restricting the spoofing area and achieves a 15% improvement in performance with a 50% decrease in spoofing area. Sadly, seeing as the paper was published in May 2024, it could not be considered in the process of developing this work, even though it presents an interesting approach to a similar problem to the one we faced. Although a direct comparison of results is not possible because of the fact

that You et al.[40] utilises the KITTI dataset which only contains clear cases and the fact that our threat model is considerably more restrictive by requiring knowledge of the object detector, we can boast comparable figures, GORA achieving a 25% increase in performance at our chosen IoU threshold over Random ORA on the HDL-64E sensor.

## 6.1 Future work

In this section we discuss possible additions to our project we would like to make in the future. Most of these depend on the assumption that in the future we will have access to more computational resources, allowing for faster training on complex scenarios.

- **Evaluation on all possible setups:** To start with, given more time and resources, the evaluation could be made more comprehensive by showcasing results for all possible sensor, metric and detection algorithm combinations.
- **Multiple detectors GORA and BORA:** This was attempted over the course of the project but proved not to be feasible due to the amount of time required for training. Such an approach is not difficult to implement and could potentially allow us to remove the restriction in our threat model that requires the attacker to have knowledge of the target's object detector. Although training on multiple detection algorithms at the same time is not guaranteed to have the desired results, it would nonetheless be important to explore.
- **Optimisation of shifting distance for GORA and BORA:** As was shown in Chapter 5, shifting distance can play a major role in the performance of ORA. We would like to further investigate this by integrating shifting distance into the training process of both GORA and BORA, allowing the algorithms to pick any distance between -2 and 2 meters for each selected point.
- **Further investigation into the use of confidence as the target metric:** In the last research question we have proven that trying to minimise detection confidence instead of IoU might have potential that we did not manage to exploit. In the future, we would like to further explore this approach, by adapting our existing methodologies or introducing a novel one.
- **Integration and comparison with You et al.[40]:** Lastly, the findings in You et al.[40] represent a unique approach to the problem of improving ORA, which could be comprehensively compared to the results of our strategies. Furthermore, the work could be an important source of inspiration, perhaps in the future leading to some form of integration with our own findings.

# Appendix A

# Ethical issues

Considering the nature of the project, it is very important to consider the ethical implications of this work.

Firstly, regarding the dataset, seeing as exclusively synthetic data is used, there are no concerns obscuring any kind of personal data. This ensures that privacy issues, often a significant concern in data-driven research, are entirely mitigated.

Secondly, in this project we aim to explore the capabilities of ORAs in rainy conditions and further improve the methodology. Because of the life-threatening effect such an attack can have, we consider that this research is crucial for understanding the security risk this poses. Furthermore, an attack that exploits vulnerabilities in LiDAR systems, especially under adverse weather conditions, can lead to severe accidents, including collisions with other vehicles, pedestrians, and infrastructure. The ethical justifications for this research lies in its potential to further the understanding of the vulnerabilities inherent to LiDAR systems, with the goal of facilitating the development of specialised countermeasures.

# Appendix B

# Additional figures

```python
def bbox3d_overlaps_diou(pred_boxes, gt_boxes):
    """
    Calculate the intersection over union (IOU) for 3D bounding boxes.

    Args:
        pred_boxes (torch.Tensor): Predicted bounding boxes (N, 7).
        gt_boxes (torch.Tensor): Ground truth bounding boxes (N, 7).

    Returns:
        torch.Tensor: Maximum IOU values.
    """
    assert pred_boxes.shape[0] == gt_boxes.shape[0]

    qcorners = center_to_corner2d(pred_boxes[:, :2], pred_boxes[:, 3:5])
    gcorners = center_to_corner2d(gt_boxes[:, :2], gt_boxes[:, 3:5])

    inter_max_xy = torch.minimum(qcorners[:, 2], gcorners[:, 2])
    inter_min_xy = torch.maximum(qcorners[:, 0], gcorners[:, 0])

    volume_pred_boxes = pred_boxes[:, 3] * pred_boxes[:, 4] * pred_boxes[:, 5]
    volume_gt_boxes = gt_boxes[:, 3] * gt_boxes[:, 4] * gt_boxes[:, 5]

    inter_h = torch.minimum(pred_boxes[:, 2] + 0.5 * pred_boxes[:, 5], gt_boxes[:, 2] + 0.5 * gt_boxes[:, 5]) - \
              torch.maximum(pred_boxes[:, 2] - 0.5 * pred_boxes[:, 5], gt_boxes[:, 2] - 0.5 * gt_boxes[:, 5])
    inter_h = torch.clamp(inter_h, min=0)

    inter = torch.clamp((inter_max_xy - inter_min_xy), min=0)
    volume_inter = inter[:, 0] * inter[:, 1] * inter_h
    volume_union = volume_gt_boxes + volume_pred_boxes - volume_inter

    return torch.max(volume_inter / volume_union)
```

Figure B.1: IoU function

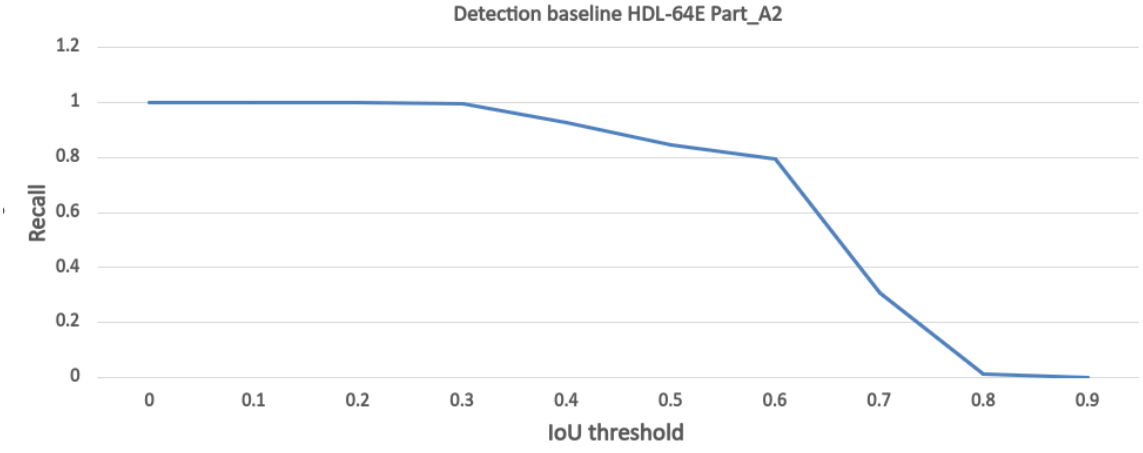



Figure B.2

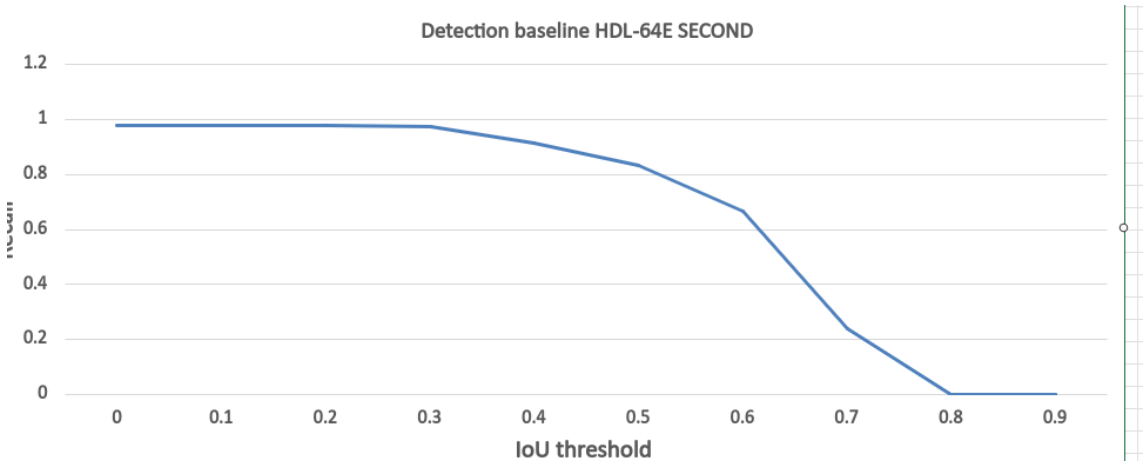



Figure B.3

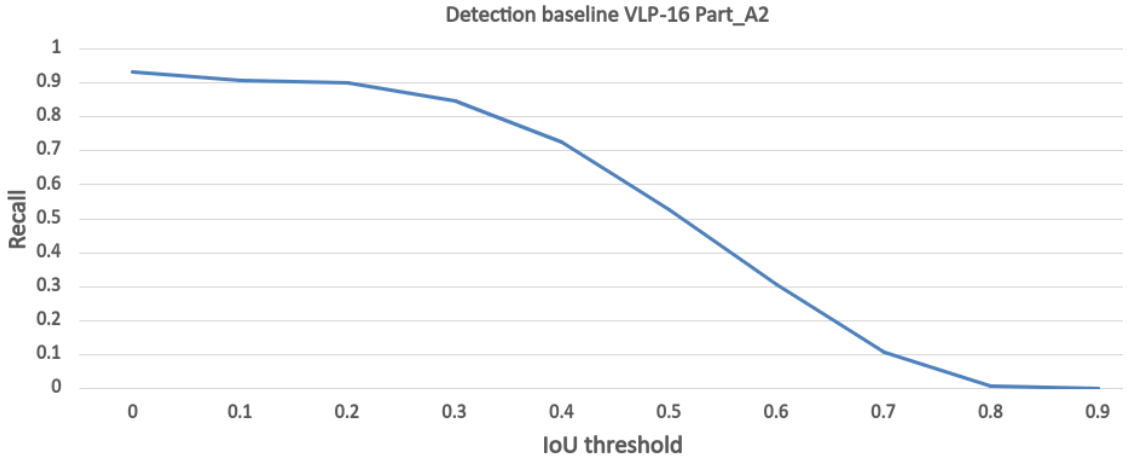



Figure B.4

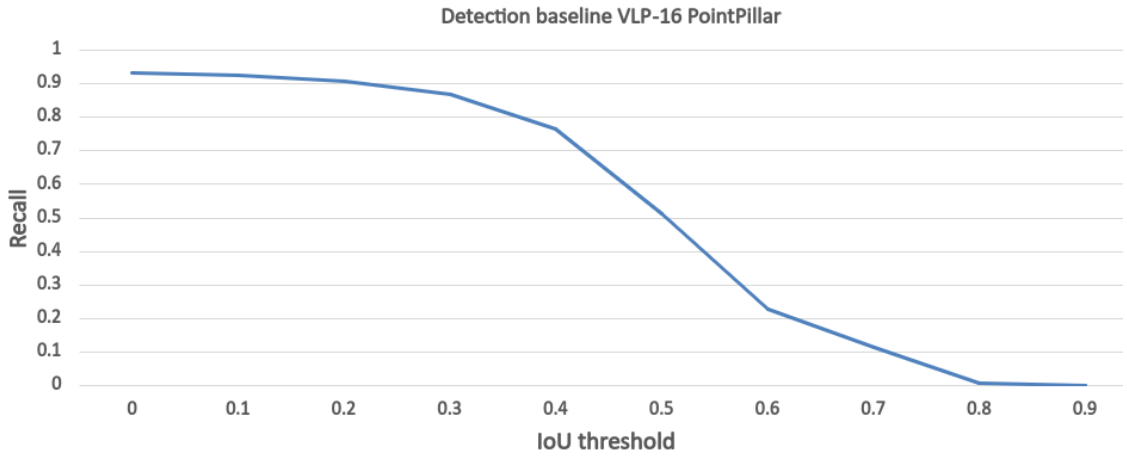



Figure B.5

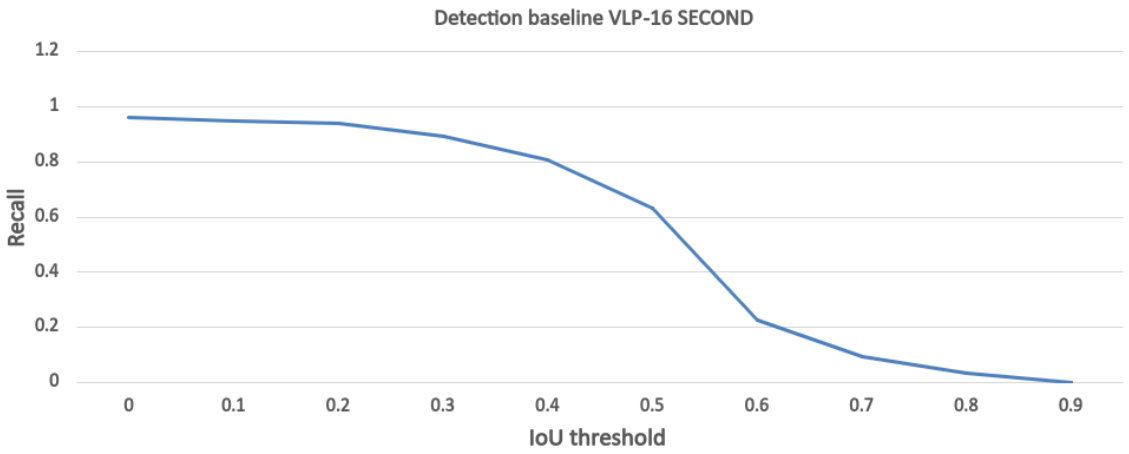



Figure B.6

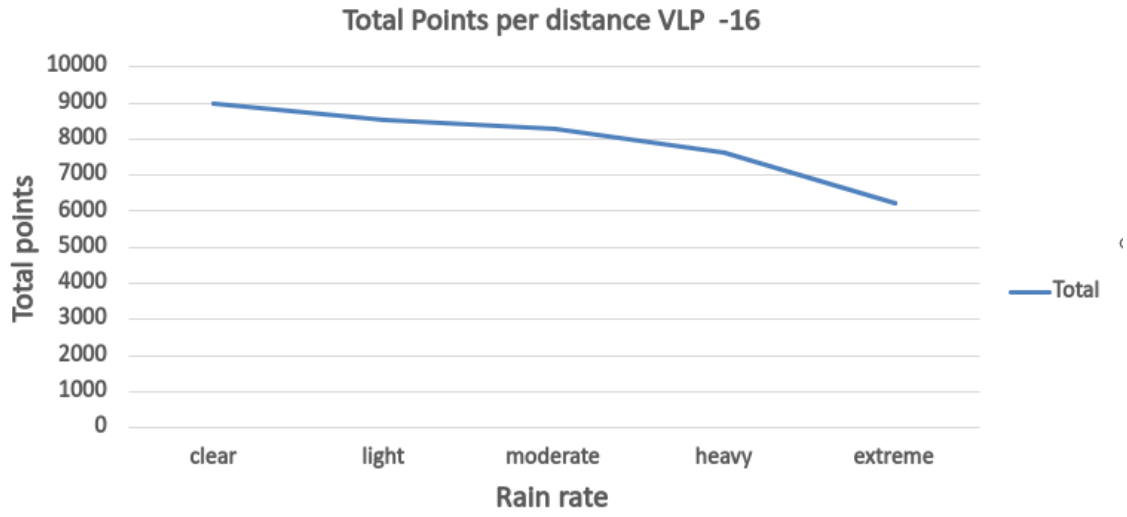



Figure B.7

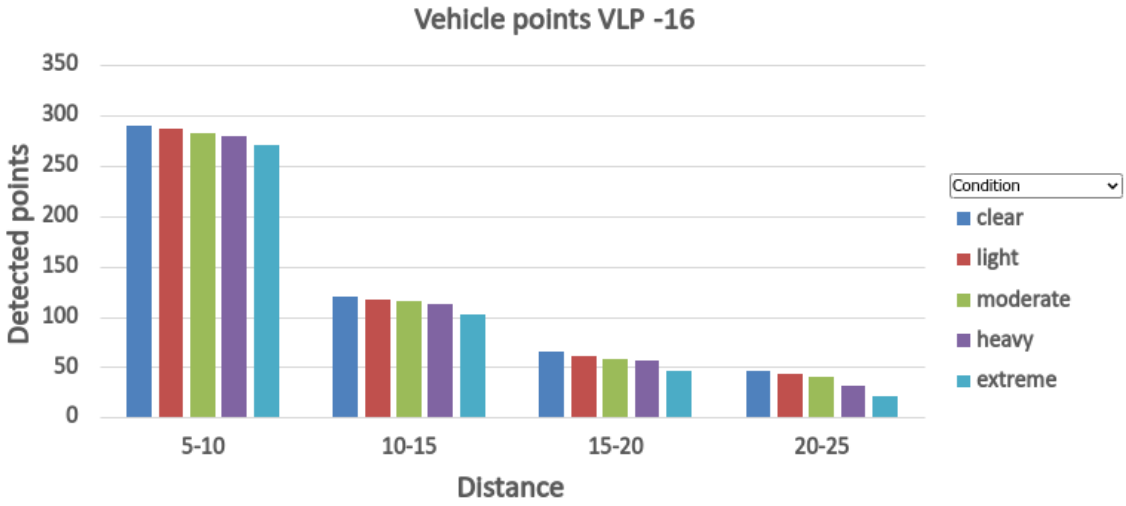



Figure B.8

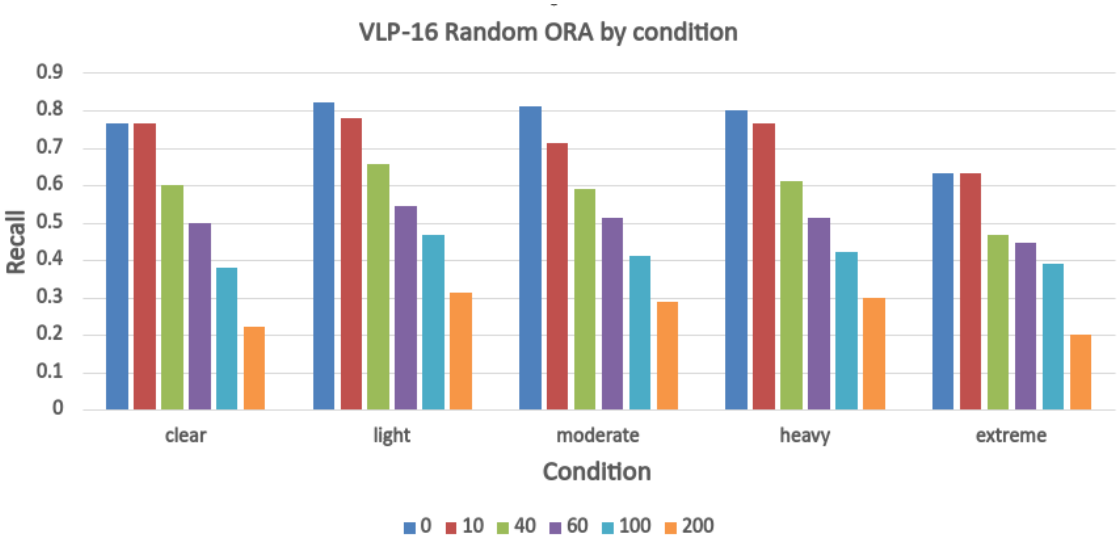



Figure B.9

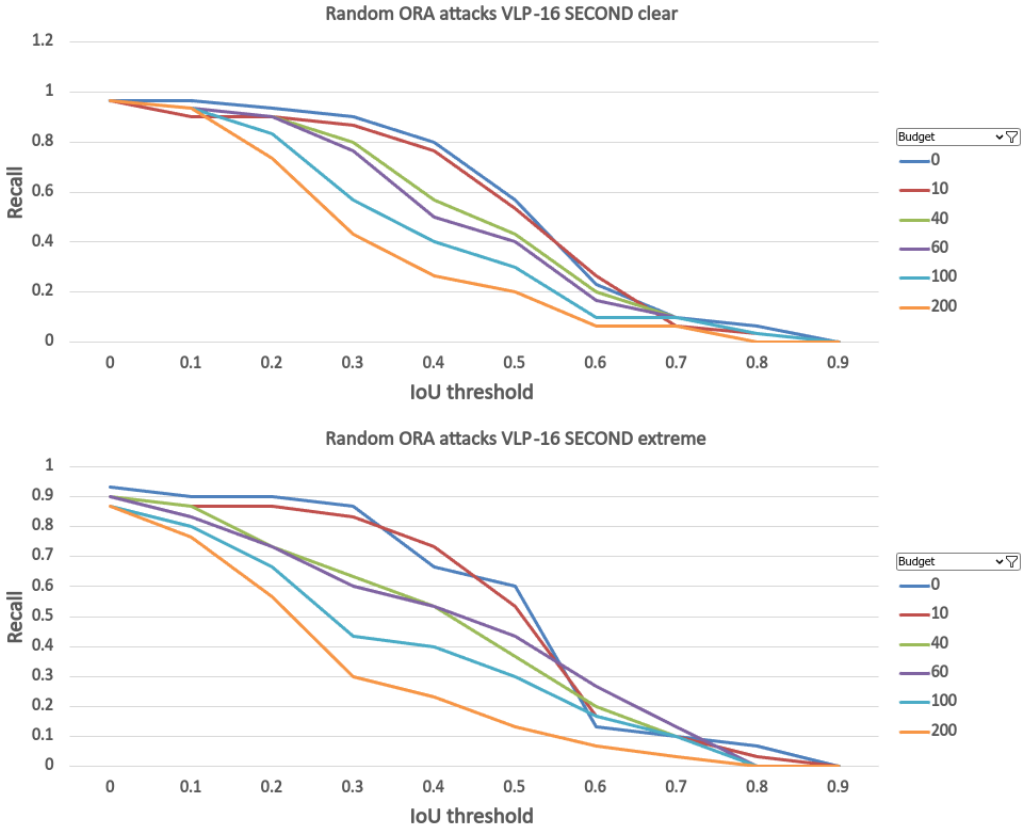



Figure B.10

# Bibliography


[1] Understanding the calculations, limitations and opportunities for the improvement of lidar range for adas and autonomous driving applications. Technical report, LeddarTech.

[2] Velodyne puck data-sheet.

[3] Velodyne ultra puck data-sheet.

[4] Jan Barani. Rain rate intensity classification, 2020. Last accessed: May 30, 2024.

[5] Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[7] Yulong Cao, S. Hrushikesh Bhupathiraju, Pirouz Naghavi, Takeshi Sugawara, Z. Morley Mao, and Sara Rampazzi. You can't see me: Physical removal attacks on lidar-based autonomous vehicles driving frameworks. *USENIX Security Symposium arXiv:2210.09482*, 2023.

[8] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. *arXiv preprint arXiv:1907.06826*, 2019.

[9] Kan Chen, Runzhou Ge, Hang Qiu, Rami Al-Rfou, Charles R. Qi, Xuanyu Zhou, Zoey Yang, Scott Ettinger, Pei Sun, Zhaoqi Leng, Mustafa Baniodeh, Ivan Bogun, Weiyue Wang, Mingxing Tan, and Dragomir Anguelov. Womd-lidar: Raw sensor dataset benchmark for motion forecasting. 2023. Available at Waymo Open Dataset.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *1st Conferece on Robot Learning (CoRL) arXiv:1711.03938*, 2017.

[11] Thomas Fersch, Alexander Buhmann, Alexander Koelpin, and Robert Weigel. The influence of rain on small aperture lidar sensors. In *2016 German Microwave Conference (GeMiC)*, pages 84–87, 2016.

[12] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Professional, 1989.

[15] C. Goodin, D. Carruth, M. Doude, and C Hudson. Predicting the influence of rain on lidar in adas. *Electronics, 8(1), 89;*, 2019.

[16] Chao Ma Guangsheng Shi, Ruifeng Li. Pillarnet: Real-time and high-performance pillar-based 3d object detection. *ECCV*, 2022.

[17] Selma Hasirlioglu, Andry Rakotonirainy, and Felix Behrendt. Adverse weather effects on lidar performance. *Journal of Transportation Technologies*, 6(2):115–125, 2016.

[18] Zhongyuan Hau, Kenneth T. Co, Soteris Demetriou, and Emil C. Lupu. Object removal attacks on lidar-based 3d object detectors. *arXiv preprint arXiv:2102.03722*, 2021.

[19] John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[20] Alireza G. Kashani, Michael J. Olsen, Christopher E. Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[21] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds, 2019.

[22] Eunho Lee, Minwoo Jung, and Ayoung Kim. Toward robust lidar based 3d object detection via density-aware adaptive thresholding. *arXiv preprint arXiv:2404.13852*, 2024.

[23] Daniel Maturana. pypcd: Pcl pcd fileformat i/o in python, 2021. Accessed: 2024-06-02.

[24] Point Cloud Library (PCL). The pcd (point cloud data) file format. https://pointclouds.org/documentation/tutorials/pcd_file_format.html. Accessed: 2024-06-02.

[25] Jonathan Petit, Bas Stottelaar, and Michael Feiri. Remote attacks on automated vehicles sensors : Experiments on camera and lidar. 2015.

[26] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 2006.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015 18th International Conference, Munich, Germany, arXiv:1505.04597*, 2015.

[28] Takami Sato, Yuki Hayakawa, Ryo Suzuki, Yohsuke Shiiki, Kentaro Yoshioka, and Qi Alfred Chen†. Revisiting lidar spoofing attack capabilities against object detection: Improvements, measurement, and new attack. *arXiv preprint arXiv:2102.03722*, 2023.

[29] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[30] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *arXiv preprint arXiv:1907.03670*, 2019.

[31] Hocheol Shin, Dohyun Kim, Yujin Kwon, , and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2017.

[32] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[33] Kyle Stock. Self-driving cars can handle neither rain nor sleet nor snow. *Bloomberg Businessweek Technology*, 2018.

[34] Bogna Szyk. Stopping distance calculator. https://www.omnicalculator.com/physics/stopping-distance. Accessed on Jun 06, 2024.

[35] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[36] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

[37] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018.

[38] Donglin Yang, Zhenfeng Liu, Wentao Jiang, Guohang Yan, Xing Gao, Botian Shi, Si Liu, and Xinyu Cai. Realistic rainy weather simulation for lidars in carla simulator. *arXiv rpeprint arXiv:2312.12772*, 2023.

[39] Kentaro Yoshioka. A tutorial and review of automobile direct tof lidar socs: Evolution of next-generation lidars. *IEICE Transactions on Electronics*, E105.C, 10 2022.

[40] Chengzeng You, Zhongyuan Hau, Binbin Xu, and Soteris Demetriou. Adversarial 3d virtual patches using integrated gradients, 2024. IEEE/ACM Workshop on the Internet of Safe Things, May 23rd, 2024.

[41] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.