Robustness of Multimodal 3D Object Detection Using Deep Learning Approach fo Autonomous Vehicles

Mémoire

Pooya Ramezani

Maîtrise en génie électrique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Robustness of Multimodal 3D Object Detection Using Deep Learning Approach for Autonomous Vehicles

Mémoire

Pooya Ramezani

Sous la direction de:

Robert Bergevin, directeur de recherche

Résumé

Dans cette thèse, nous étudions la robustesse d'un modèle multimodal de détection d'objets en 3D dans le contexte de véhicules autonomes. Les véhicules autonomes doivent détecter et localiser avec précision les piétons et les autres véhicules dans leur environnement 3D afin de conduire sur les routes en toute sécurité. La robustesse est l'un des aspects les plus importants d'un algorithme dans le problème de la perception 3D pour véhicules autonomes. C'est pourquoi, dans cette thèse, nous avons proposé une méthode pour évaluer la robustesse d'un modèle de détecteur d'objets en 3D.

À cette fin, nous avons formé un détecteur d'objets 3D multimodal représentatif sur trois ensembles de données différents et nous avons effectué des tests sur des ensembles de données qui ont été construits avec précision pour démontrer la robustesse du modèle formé dans diverses conditions météorologiques et de luminosité. Notre méthode utilise deux approches différentes pour construire les ensembles de données proposés afin d'évaluer la robustesse. Dans une approche, nous avons utilisé des images artificiellement corrompues et dans l'autre, nous avons utilisé les images réelles dans des conditions météorologiques et de luminosité extrêmes.

Afin de détecter des objets tels que des voitures et des piétons dans les scènes de circulation, le modèle multimodal s'appuie sur des images et des nuages de points 3D. Les approches multimodales pour la détection d'objets en 3D exploitent différents capteurs tels que des caméras et des détecteurs de distance pour détecter les objets d'intérêt dans l'environnement. Nous avons exploité trois ensembles de données bien connus dans le domaine de la conduite autonome, à savoir KITTI, nuScenes et Waymo.

Nous avons mené des expériences approfondies pour étudier la méthode proposée afin d'évaluer la robustesse du modèle et nous avons fourni des résultats quantitatifs et qualitatifs. Nous avons observé que la méthode que nous proposons peut mesurer efficacement la robustesse du modèle.

Abstract

In this thesis, we study the robustness of a multimodal 3D object detection model in the context of autonomous vehicles. Self-driving cars need to accurately detect and localize pedestrians and other vehicles in their 3D surrounding environment to drive on the roads safely. Robustness is one of the most critical aspects of an algorithm in the self-driving car 3D perception problem. Therefore, in this work, we proposed a method to evaluate a 3D object detector's robustness.

To this end, we have trained a representative multimodal 3D object detector on three different datasets. Afterward, we evaluated the trained model on datasets that we have proposed and made to assess the robustness of the trained models in diverse weather and lighting conditions. Our method uses two different approaches for building the proposed datasets for evaluating the robustness. In one approach, we used artificially corrupted images, and in the other one, we used the real images captured in diverse weather and lighting conditions.

To detect objects such as cars and pedestrians in the traffic scenes, the multimodal model relies on images and 3D point clouds. Multimodal approaches for 3D object detection exploit different sensors such as camera and range detectors for detecting the objects of interest in the surrounding environment. We leveraged three well-known datasets in the domain of autonomous driving consist of KITTI, nuScenes, and Waymo.

We conducted extensive experiments to investigate the proposed method for evaluating the model's robustness and provided quantitative and qualitative results. We observed that our proposed method can measure the robustness of the model effectively.

Table des matières

Liste des tableaux

Liste des figures

Dedicated to all those who lost a
loved one in a car accident.

Remerciements

This work would not have been possible without the support of my family. I could never thank them enough for their never-ending support and warm words. Also, I would like to thank my research advisor, Professor Robert Bergevin, for his comprehension and useful comments.

Introduction

Every year, car accidents are one of the main reasons for death and serious injuries worldwide. Self-driving cars may make the roads safer than they are today and decrease the number of car accidents. Hence, the efforts to improve technologies such as driver assistance and fully autonomous vehicles are valuable. However, it is necessary to solve various technical difficulties and other important issues before using autonomous vehicles commercially.

Since the cause of most traffic accidents leading to severe injuries or even death is the error of drivers [7], the use of self-driving cars can be expected to prevent accidents that arise from human error and reduce the casualties. To achieve this goal, we need to leverage accurate and robust control systems and perception algorithms in autonomous vehicles, capable of working in diverse environmental situations.

The study of autonomous vehicles is an application domain that currently attracts considerable attention from industrial and scientific communities. It is an active research area due to the promising results of utilizing perception systems based on multimodal deep learning methods for 3D object detection.

Due to the need for having high accuracy, robustness, and real-time performance for self-driving cars, developing perception systems based on multimodal deep learning approaches is particularly challenging [8]. Among several difficulties in achieving practical algorithms, evaluating the model's robustness, which quantifies the stability of the model's performance to the disturbances in input data, is one of the most challenging [9].

In modern cars, various technologies such as driver assistance are already in operation and help prevent accidents and reduce road casualties. For instance, they help drivers avoid drifting into adjacent lanes or changing the lane unsafely. These driver assistance technologies can also warn drivers while other cars are behind them, and there is a risk of an accident. Another example is an automatic braking system that activates when a car ahead brakes and slows suddenly or stops. These safety technologies help cars to find out safety risks and warn drivers.

To have fully autonomous cars, perception of the driving environment is a prerequisite. An autonomous vehicle (AV) needs an accurate and robust perception of its surroundings to operate reliably. Towards enabling autonomous driving, the perception system of an AV usually exploits machine learning algorithms. More specifically, deep learning methods are widely used to transform sensory data into semantic information.

Perception is the first component in the computational pipeline for the reliable functioning of an autonomous vehicle. Moreover, object detection is a vital function of this perception system. By leveraging 3D object detection methods and considering the third dimension, it is possible to reveal more relevant information about the object's size and location in the surrounding environment.

Accurate, robust, and real-time 3D detection of objects is crucial for autonomous vehicles. Autonomous vehicles are usually equipped with different sensors to perceive their environment and make correct decisions based on the perception of the environment. Therefore, fusing data gathered by these different sensors and extracting relevant information while considering different modalities is necessary.

3D object detection methods have become popular in recent years since they can provide a more detailed description of objects than 2D object detectors. 3D object detection aims to recover the 6 DoF (Degrees of Freedom) pose and the 3D bounding box dimensions for all objects of interest in the scene. 3D object detection, which classifies the object category and estimates oriented 3D bounding boxes of physical objects, is essential for 3D perception tasks. It serves as an essential basis of visual perception, motion prediction, and planning for autonomous driving.

The datasets for training models in a supervised learning approach are as important as the models. Moreover, the evaluation of the results is critical. Therefore, in this research, the impact of the exploited datasets and evaluation methods on the state-of-the-art multimodal model, Frustum PointNets [6], are investigated.

One of the most notable aspects of the supervised learning approach in machine learning is having a large and proper dataset for training the models. The KITTI [4], nuScenes [5], and Waymo [1] datasets are all public multimodal datasets containing 3D bounding box annotations for objects of interest such as cars and pedestrians.

KITTI dataset is one of the oldest datasets for training models for 3D object detection for self-driving cars, and its evaluation method is used widely. However, the KITTI dataset does not consist of diverse scenes, and all images are taken in the daytime while the weather is sunny. The nuScenes and Waymo datasets contain more images captured in both night and day and different weather conditions such as rainy weather.

0.1 Statement of Problem

Operating a vehicle on public roadways is a complicated task due to the number of interactions with other objects, which are usually unpredictable, including vehicles, pedestrians, cyclists, and animals. Robust perception of the surrounding environment and extracting relevant information are crucial for safe navigation on the roads.

One of the reasons that prevent self-driving cars from being used widely is that their perception systems' inability to function well in adverse weather and lighting conditions [10]. Considerable progress is needed before autonomous vehicles can operate reliably in mixed urban traffic or heavy rain and snow. In other words, without further improvements in the robustness of models, it is not possible to achieve the full potential of self-driving cars.

The principal question in the study of self-driving cars is : how a vehicle can be capable of perceiving its environment and moving between two arbitrary locations safely, with little or no human intervention. An accurate and robust perception system is a precondition for autonomous vehicles that can safely drive by themselves in complex driving environments.

Although many models and algorithms are recently proposed for 3D object detection in self-driving car scenarios, there is no appropriate method for evaluating models' robustness. Given a trained deep neural network model, the main objective of robustness verification methods is providing a robustness certificate for verifying the model's generalization capability and its ability to work in diverse conditions.

Current metrics often concentrate on examining the model's accuracy. Nevertheless, it is necessary to develop and adapt proper methods and metrics to measure the model's robustness for multimodal perception problems.

In this work, the problem of evaluating the robustness of a model in various scenarios, including diverse lighting and weather conditions, is investigated. More precisely, the question is : how can we evaluate the robustness of a trained model in a self-driving car scenario. In order to answer this question, we propose several robustness test sets. One of the proposed datasets contains images with synthetic corruptions, and the others contain images with natural corruptions due to diverse weather and lighting conditions.

To make the dataset of synthetic corrupted images, we exploited 15 different corrupting methods, each with five diversities. For the datasets of images with diverse weather conditions, we leveraged images captured in rainy weather. To test the robustness of the model for diverse lighting conditions, we collected images captured during the nighttime.

0.2 Scientific Objectives

Generally, in many research areas, utilizing challenging datasets as a benchmark is widespread since it is possible to exploit them as a standard comparison between different models and set objectives for solutions [11]. Therefore, we investigate two different approaches to cope with benchmarking for evaluating the robustness in this work.

In the first approach, we consider a dataset and then divide it into training and test subsets. We induce different artificial corruptions with various severities on the data of the test subset. After training the model on the training subset, we evaluate the trained model on the original test subset and the subsets we made by inducing corruptions. By comparing the results, we can evaluate the robustness of the model.

The advantage of this method is the ability to exploit it on an arbitrary dataset. Moreover, it can effectively measure a trained model's robustness against some noises and digital corruptions such as JPEG compression. However, this method can not accurately assess the model's robustness in real-world conditions such as diverse weather or lighting conditions. Hence, we investigate a second method to examine the model's robustness with real data to solve this problem.

To this end, we leverage datasets that consist of captured data in rainy weather and also at night. We divide them into three sub-datasets corresponding to clear, rainy, and night conditions. Then we first train and evaluate the model on the clear sub-dataset. Afterward, to estimate the robustness, we evaluate the trained model on rainy and night sub-datasets.

In this thesis, the robustness and accuracy of a state-of-the-art model in an autonomous vehicle scenario are investigated and discussed. The proposed approaches are useful for autonomous driving and robotic perception. In general, achieving accurate and robust algorithms for perceiving the environment is a significant and fundamental problem.

0.3 Outline of Following Chapters

Chapter 1 presents the necessary and relevant theoretical concepts for this research. Moreover, at the end of the first chapter, a brief review of related works in 3D object detection for autonomous vehicles is provided. In chapter 2, the methodology, proposed datasets, and evaluation metrics are presented. Chapter 3 presents extensive quantitative evaluations and rich qualitative results for understanding the proposed method's strengths and limitations. At the end, the conclusion is provided.

Chapitre 1

Literature Review and Theoretical Background

1.1 Object Detection

When we look at an image, one question that arises could be : "what objects are there in the image ?". This question is one of the fundamental questions that algorithms in computer vision try to answer. This question is categorized as image classification. When we find out the answer to this question, another important question that can be asked is, "where are the objects in the image ?". Object detection in computer vision tries to answer these two questions for images.

Although, in many cases, recognizing and detecting objects in an image are easy tasks for humans and can be done accurately and rapidly, these are significantly complicated and challenging tasks for computers. Due to the definition of the object is a philosophical question that philosophers asked it through history [12], in some cases, the answer to the question "what are the objects in the image ?" can be different for different observers. However, this problem is beyond the scope of this research. In this thesis, we suppose that the definition of the objects of interest includes cars, pedestrians, and other traffic participants are determined.

In fact, in supervised learning, we accept the correctness and accuracy of annotators' judgments and opinions and the verification processes during labeling the datasets. Therefore, we expect that the model trained on the part of a dataset that we call the training subset can classify and localize the objects of interest in another part of the dataset called the test subset. Furthermore, the model does not use the test subset to update the model's parameters during training.

For several decades, object detection has been an active area of research in the computer vision community [13, 14]. Object detection is one of the most primary yet challenging problems in computer vision. An object detector aims to determine whether there are any instances of objects from given categories in an image and if it exists, finding the spatial location and extent of each object instance by returning appropriate bounding boxes that delimit the entire object of interest. The object categories for detecting could be any object of interest in the images, such as cars, pedestrians, and cyclists in the self-driving car scenarios. A wide range of applications uses object detectors. For example, the object detection algorithms are already used in autonomous vehicles, vision for robots, human-machine interactions, augmented reality, and intelligent video surveillance.

Object detectors are essential parts of the perception systems [7]. Several works have tried to solve the object detection problem by proposing different methods, including 2D object detection and 3D object detection. The majority of the proposed methods are 2D detectors. However, 2D detection methods do not provide information about the depth and heading angle of objects, while these are necessary for many tasks. For example, path planning and collision avoidance are crucial tasks for self-driving cars, requiring depth and heading angle information of detected objects. On the other hand, the 3D object detection methods give information about detected objects' depth and heading angle in a 3D space. However, 3D object detectors' accuracy is usually lower than the accuracy of 2D object detectors.

1.1.1 2D Object Detection

The input of a 2D object detector is an image, and its output is 2D bounding boxes with a class label for all objects of interest in the given image. The 2D bounding box is usually a rectangle aligned with the image coordinate system's axis, containing all parts of the associated object of interest while its size is minimum.

For the 2D object detection task, there are two approaches, namely two-stage and one-stage approaches [13]. Two-stage detection frameworks, which are also called object proposal approach, involve a pre-processing step for producing object proposals. One-stage detection frameworks, or regression-based frameworks, are based on a single proposal method. This approach does not separate the process of generating the detection proposal step from classifying proposals as background or category-specific objects and post-processing detection to improve their fitness to objects.

It is possible to present a 2D bounding box with four parameters. They could be the coordinates of two opposite corners of the bounding box, $(x_{min}, y_{min}, x_{max}, y_{max})$, or the height, width and center coordinates, $(x_{center}, y_{center}, height, width)$. An example of 2D bounding boxes is illustrated in figure 1.1.

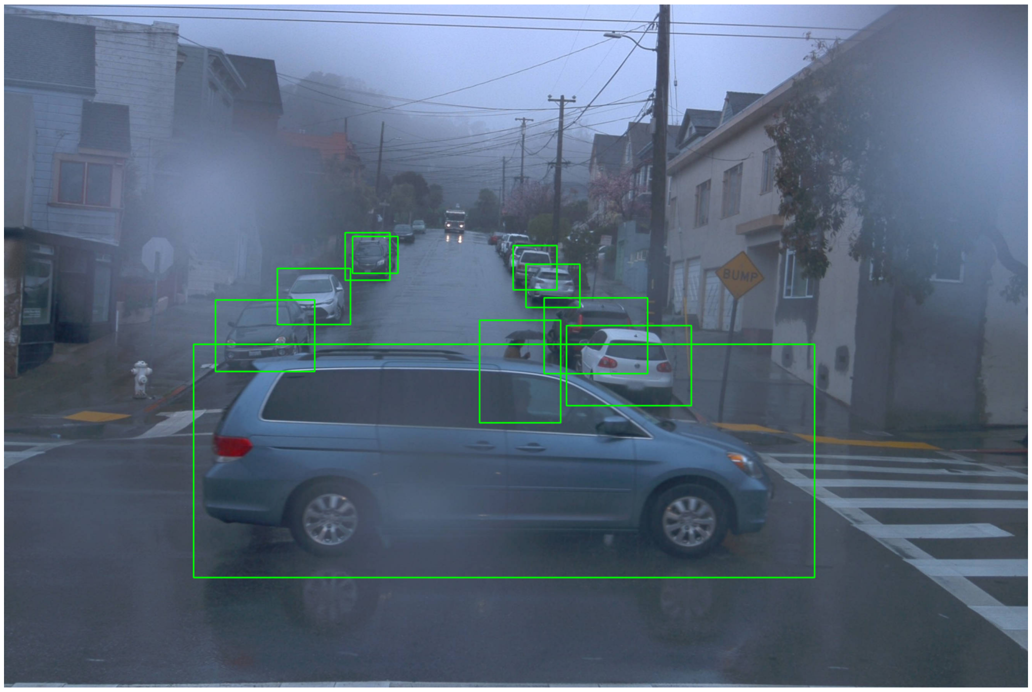

FIGURE 1.1 – Illustration of the ground truth 2D bounding boxes for a sample from the rainy subset of Waymo dataset [1].

Two-stage 2D Object Detectors

The two-stage object detector framework first generates a set of proposal bounding boxes that possibly contain objects by using region proposal methods such as selective search [15]. It then passes the detected object proposals to the CNN (Convolutional Neural Network) classifiers to determine whether they are backgrounds or a specific object class.

Proposal-based object detectors such as R-CNN (Regions with CNN features) [16], extract several potential object candidates called regions of interest (ROI) or region proposals (RP) from an image in their first step. Then, these candidates are verified, classified, and refined in terms of classification scores and locations.

By introducing the spatial pyramid pooling layer (SPPNet) in [17], the speed of the original R-CNN was improved because of allowing the classification layers to reuse features computed over feature maps generated at different image resolutions.

Afterward, by exploiting the advantages of R-CNN and SPPNet, the Fast R-CNN [18] is proposed, which speeds up the network's convolution processing time by sharing computation. The Fast R-CNN's design gives the possibility to use a multi-task loss function and minimize the loss related to prediction scores and bounding box regression simultaneously in an end-to-end fashion. Fast R-CNN leverages the selective search method [15] to produce object region proposals. In the next step, the detection stage uses these proposals and the entire image.

In this step, a CNN backbone extracts the feature map of the image. The model produces a feature vector by projecting the region proposals into the extracted feature map and using the pooling function. Afterward, two fully-connected branches use this feature vector to predict the offsets for proposed bounding boxes and their confidences. Fast R-CNN uses only one CNN forward pass during training and inferring, leading to a remarkable speed improvement.

However, Fast R-CNN still needs an external method to obtain region proposals. This requirement prevents the application of end-to-end training and causes efficiency degradation of the model. Faster R-CNN [19] model tries to solve this problem by introducing a Region Proposal Network (RPN) that shares the entire image feature map with the second stage detection network. Faster R-CNN extracts a global feature map of the image by utilizing a CNN at the first step. Next, both the RPN and the Fast R-CNN detection stage use the same global feature map. This method enables end-to-end training of the Faster R-CNN. The architecture of Faster R-CNN is depicted in figure 1.2.

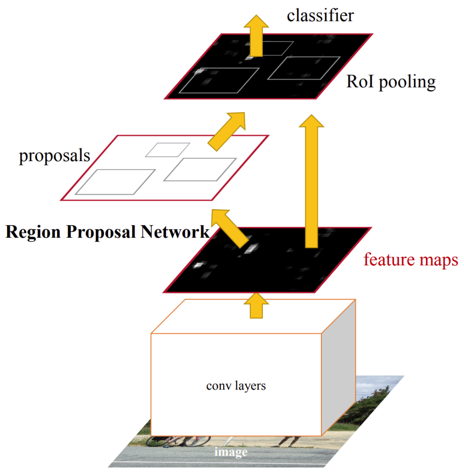

FIGURE 1.2 – This figure illustrates the Faster R-CNN pipeline. Initial layers are convolutional layers, which share the feature map with the RPN, that generates region proposals to be fed into the classifiers and the Fast RCNN module (figure reproduced from [19]).

One-stage 2D Object Detectors

The one-stage object detectors are regression-based object detection methods formulated as a regression problem with spatially separated bounding boxes and associated class probabilities. This method uses the extracted features directly to regress bounding boxes and obtain classification scores by a unified CNN model. One-stage 2D object detectors such as different versions of YOLO (You Only Look Once) [20, 21, 22], SSD (Single Shot Detector) [23, 24] and RetinaNet [25] are simpler in comparison with object proposal-based methods because they do not require proposal generation.

One of the well-known one-stage object detectors is YOLO. It divides the input image into sparse grids. Then, by leveraging the extracted features by convolutional layers, it proposes potential bounding boxes for each grid cell along with their confidence scores. Compared with the RPN-based approach, YOLO exploits the entire image's feature map to predict each bounding box. The confidence scores indicate the model's reliance on the predicted bounding box's size and the confidence that the box contains an object of interest. If there is no object in a grid cell, the model assumes a zero confidence score.

The one-stage object detectors are faster and easier to optimize. However, their accuracy is typically lower than two-stage object detectors [8]. In general, two-stage object detectors like Faster R-CNN [19] and Mask R-CNN [26] can achieve better performance for detection accuracy due to the region proposal generation and refinement methods. The drawback of two-stage methods in comparison with one-stage approaches is higher inference time.

1.1.2 3D Object Detection

A 3D object detector aims to locate, classify, and estimate oriented bounding boxes in the 3D space (figure 1.3). In the context of 3D object detection, each object is usually associated with a 3D bounding box $(c_x, c_y, c_z, l, w, h, \theta)$, where c_x, c_y, c_z represent the center coordinates, l, w, h are the length, width, height, and θ denotes the heading angle of the bounding box.

Compared to 2D detection, 3D detection is more challenging since the object's depth in the 3D environment needs to be estimated. A 3D object detection framework can take different sensors data as input and outputs a 3D bounding box with a class label for all objects of interest in the sensors' field of view (FOV). 3D object detectors are essential parts of the visual perception system of self-driving cars. Modern autonomous driving cars are usually equipped with multiple sensors.

Based on the exploited sensors for estimating the 3D bounding boxes of objects of interest in the environment, the 3D object detector methods can be generally broken down into various subcategories such as image based, point cloud based and multimodal based detectors.

In the image based approaches, the monocular or stereo images of the surrounding environment captured by RGB cameras are exploited to detect the objects of interest and their locations in the images. The detection results are 3D bounding boxes in the camera coordinate, and by leveraging the calibration of cameras, the coordinate of 3D bounding boxes in real-world coordinates can be calculated.

Several models and techniques were proposed to detect the objects of interest in 3D space by exploiting LiDAR point clouds. For instance, VoxelNet [28] divides the point clouds into voxels and then leverages point-wise features extracted directly on a 3D voxel grid. Another approach to cope with unstructured point clouds is projecting them onto the 2D plane. This technique makes it possible to deal with point clouds as 2D images where 2D convolutions are applicable.

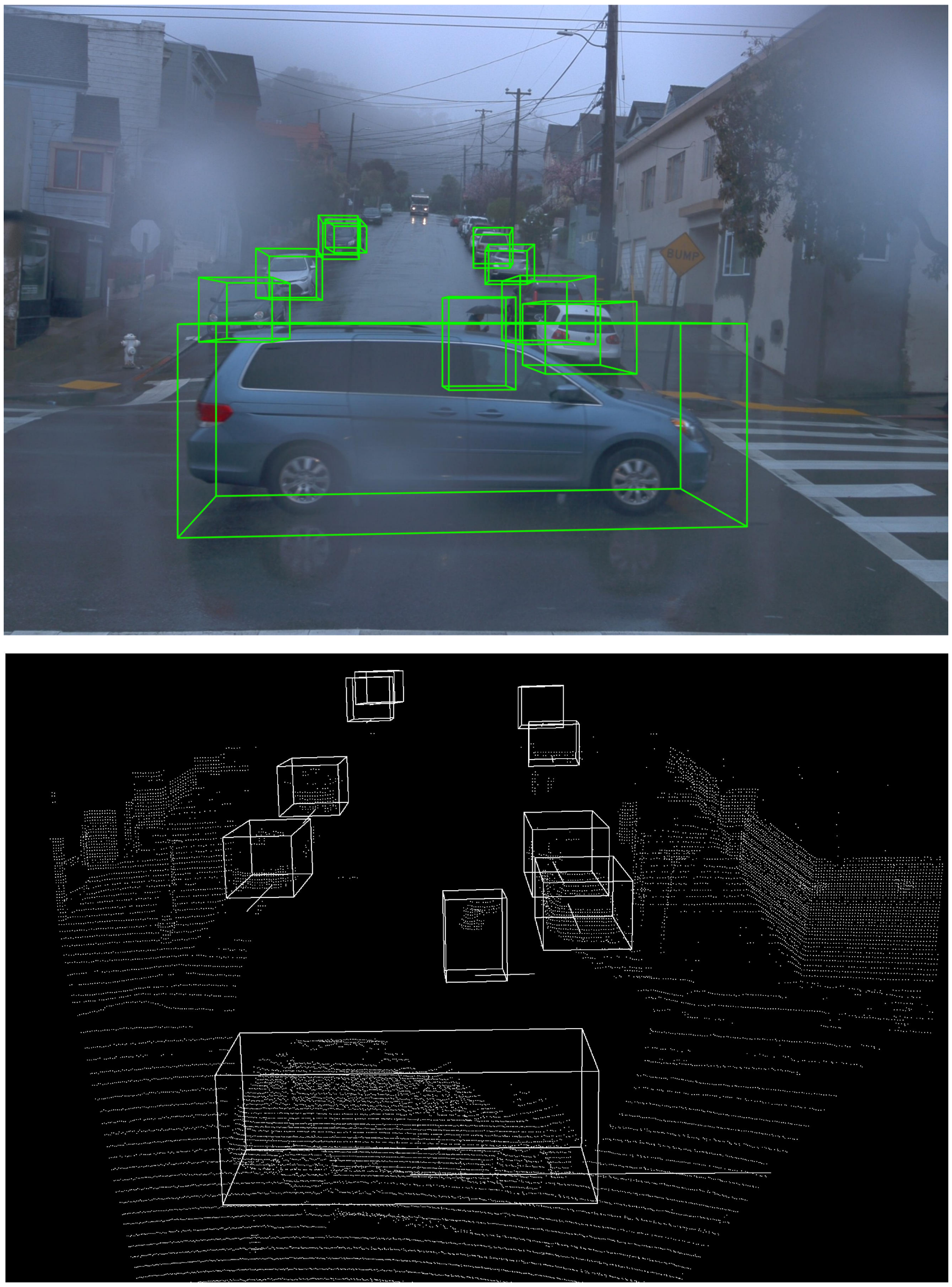

FIGURE 1.3 – An image and its corresponding point cloud with projected ground truth 3D bounding boxes from rainy subset of Waymo [1] dataset. The point cloud illustration is created using Mayavi [27].

For example, PIXOR [29] is a one-stage object detector that efficiently exploits the 2D Bird's Eye View (BEV) representation of the point clouds of the scene. It is also possible to use the unstructured point clouds directly. For instance, RoarNet [30] processes 3D point clouds directly. Furthermore, multimodal based algorithms such as AVOD [31] and PointFusion [32], leverage both images and point clouds in a sensor fusion manner to improve the performance.

1.2 Multimodal Deep Learning

Deep learning is a machine learning approach that leverages multiple network layers to extract features progressively. The concept of learning based artificial neural networks was first introduced in the 1940s. It attracted attention significantly throughout the 1970s and 1980s, and basic methods such as backpropagation and automatic differentiation are developed. However, throughout the 1990s and 2000s, research in artificial neural networks has slowed down due to various reasons [33]. Nevertheless, in recent years, research in artificial neural networks has gained attention again due to the rapid growth of computational power and data, combined with advances in network architecture and training strategies. Notably, deep learning-based methods have demonstrated great potentials.

The development of algorithms and methods that consider multiple aspects of perception in the multimodal platform is necessary to perceive the environment with human-like capabilities. The modality refers to the way that something happens or is experienced. It is usually related to sensory modalities. For example, brightness and colors can be captured by RGB cameras, while range sensors such as LiDAR or RADAR can measure the distances to the objects in the environment.

On the other hand, by exploiting data from different modalities, one is more likely to capture the correspondences between modalities and achieve a comprehensive understanding of the environment [34]. Deep learning provides the possibility of obtaining information about a scene by extracting the features of data from different modalities [35].

Moreover, deep learning techniques have emerged as a powerful strategy for learning feature representations directly from different types of data and have led to remarkable breakthroughs in the field of generic object detection [13]. Sensor fusion leverages multiple types of sensors with complementary characteristics to enhance 3D perception.

The systems that only use mono-camera for 3D object detection tasks can not provide reliable results because of the lack of depth information. Although stereo cameras can provide 3D geometry, they have problems in diverse weather and lighting conditions. Furthermore, high occlusion and textureless environments can limit their performance [33]. On the other hand, range sensors can provide accurate 3D geometry, which does not depend on lighting conditions but is limited by low resolution, low refresh rates, and high cost. Many recent works exploited these two complementary sensors to cope with this challenge and demonstrated remarkable performance improvements.

There are three main approaches for integrating different sensing modalities in deep learning

[8]. It is possible to fuse data from two modalities in the early, middle, or late stages (figure 1.4). Each of these methods has its advantages and drawbacks. For instance, architectures with late and middle fusion are more flexible than models with early fusion, while early fusion needs less memory and computation resources.

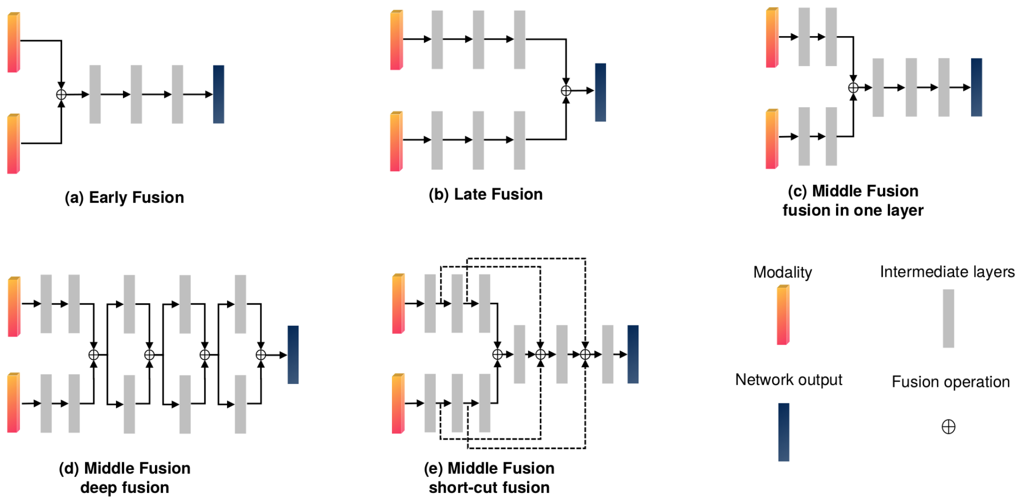
(a) Early Fusion
(b) Late Fusion
(c) Middle Fusion fusion in one layer
(d) Middle Fusion deep fusion
(e) Middle Fusion short-cut fusion
Modality
Intermediate layers
Network output
Fusion operation

FIGURE 1.4 – Several possibilities for fusing two modalities in different stages in deep learning models. Fusion operation could be the concatenation of features or addition operations (figure reproduced from [8]).

1.3 Autonomous Vehicles

Decades after the first cars' production, the automotive industry has witnessed developments in various fields. More safety and comfort for the occupants, higher power, more speed, and more efficient fuel consumption are among the advantages of today's cars over their predecessors. Despite all these advances, as yet, a human driver's need to operate a car is inevitable.

Recently, extensive efforts have been made to build driverless cars. Today, with the development of computer algorithms for information processing and increasing processors' speed, besides developments in sensors, hopes have been created to enable vehicles to recognize their surrounding environment. The first step to achieve this goal is to have a system that can understand its surroundings to take appropriate action in various situations based on its perception.

Autonomous vehicles are complicated robotic systems that operate in unpredictable environments. A fully automated vehicle could provide new mobility options for older people and those with disabilities. Autonomous vehicles are usually equipped with a wide range of onboard sensors. By leveraging the data from different sensors, self-driving cars can perceive their environment and make decisions in different conditions. The steering, acceleration, and braking are the main output commands of an autonomous vehicle's control system.

Various installed sensors on the car gather environmental information. An efficient algorithm must then be used to process the collected information to extract the relevant information and allow the car to perceive its surrounding. The quality and quantity of information acquired by an autonomous vehicle directly relate to the utilized sensors' type and performance.

The prototypes of fully autonomous vehicles already exist. However, they are currently in the development and testing stage. Despite the progress, many technical improvements are necessary before a practical self-driving car can operate without human intervention under all diverse conditions [2]. Reliable self-driving car technologies have a long way to go before becoming widely available. Figure 1.5 demonstrates an example of fully autonomous vehicles.

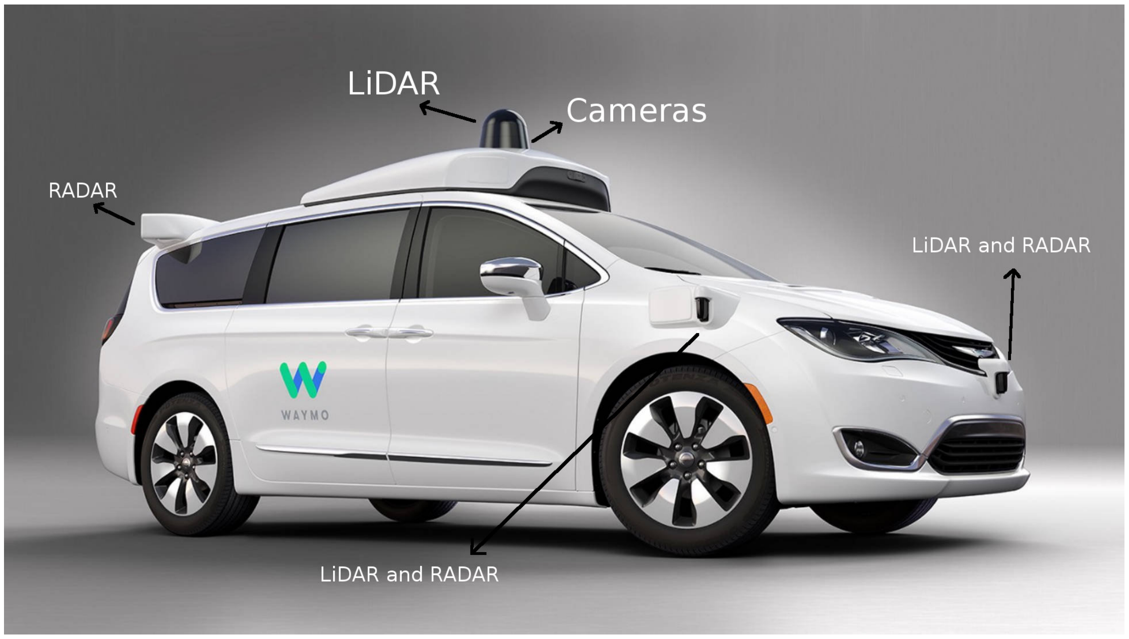

Figure 1.5 – Waymo is one of the pioneering companies in the autonomous vehicle industry that practically launched transporting passengers in limited regions by self-driving cars. The image illustrates one of Waymo's fully autonomous cars [2].

1.3.1 Sensing Modalities

For empowering self-driving vehicles to perceive the surrounding world, sensors play an essential role [35]. It is possible to improve self-driving cars' safety and reliability by leveraging more sensors to obtain a precise perception of the environment.

Each type of sensor has its benefits and limitations. Therefore, cooperation between multiple sensors installed on an autonomous vehicle can directly improve its safety. For instance, cameras can accurately capture the objects' textures and color distributions while they can not work correctly at night. On the other hand, range sensors can work at night perfectly. However, their output is sparse.

Various kinds of cameras, LiDARs, and RADARs are typically adopted sensors in autonomous vehicles [35]. Notably, cameras and LiDARs (Light Detection and Ranging) are the most popular sensors for multimodal perception in autonomous vehicles because these two types of sensors are currently the most informative sensors [33]. Table 1.1 presents the usual sensors for perception in self-driving cars. In various environmental conditions, the reliability and performance of different sensors can change significantly. Hence, the fusion of information gathered by a multi-sensor setup can improve the performance of perception tasks. Moreover, the calibration and alignment of sensors can significantly impact the accuracy and robustness of multimodal perception systems.

Sensor Type	Advantages	Disadvantages	Max Working Distance
Camera	Capturing the contour, texture, and color distribution accurately, Low price	Vulnerable to diverse lighting and weather conditions, Heavy calculation burden	250m (depending on the lens)
LiDAR	Applicable for all lighting conditions, High range resolution, High angle resolution	High price, Sparse output	200m
RADAR	Long working distance, Applicable for all weather conditions	Unapplicable for static objects, Generating false detection easily	5m-200m
Ultrasonic	Inexpensive	Low resolution, Inapplicable for high speed	2m

Table 1.1 – Summary of different sensors for object detection in autonomous vehicles. Among them, LiDAR and camera are more popular for perception tasks while RADAR and Ultrasonic sensor are less common [35].

Camera

The camera is the primary choice for companies and researchers in the domain of the self-driving car. Most main autonomous driving industry players have the camera as their primary sensors in their vehicle sensor suite. The camera's advantage is that it can accurately capture the contour, texture, and color distribution information. These abilities help to detect different objects under various environment conditions precisely.

The role of the camera in the perception pipeline of autonomous vehicles is like the eyes of humans. The captured images by cameras can help the self-driving cars detect different traffic participants such as other cars and pedestrians, and predict their behaviors and track their movements. Moreover, it is also possible to exploit these images to detect and localize road markings, signals, and signs to empower the car to safe and lawful self-driving.

Images captured by the camera convert the 3D information into the 2D one. It is necessary to establish the relationship between the pixels and the real-world to obtain the target's position from the image. For this purpose, we utilize camera calibration.

Visual cameras are passive sensors that mean they do not actively emit any signal for measurements. Hence they do not interfere with other systems [36]. Among various sensors for autonomous vehicles, cameras are the most informative sensors [35] that provide detailed texture information of a vehicle's surroundings.

However, cameras have certain disadvantages. They are sensitive to lighting and weather conditions. One must consider that the image can become unreliable in some cases, such as the sudden change of light or even the near inability to perceive the surrounding environment at night. Moreover, the cameras cannot directly provide depth information for 3D perception tasks [8]. Extensive researches have been done to enhance monocular camera-based depth perception. Nevertheless, self-driving cars need to use modalities for which diverse lighting and weather conditions can not significantly affect them [36].

Light Detection And Ranging (LiDAR) Scanners

The LiDAR is the most commonly used laser rangefinder type of sensor for self-driving cars. Many autonomous vehicle manufacturers and researchers adopt LiDAR as one of their primary sensors for the 3D object detection task, and it plays an increasingly significant role in self-driving cars.

A LiDAR operates based on measuring the duration between emitting laser beams and detecting their reflections by objects and using this time-of-flight to calculate objects' distance to the sensor. Depending on the number of emitter-detector pairs, also called channels, there are several kinds of LiDARs on the market. For instance, there are LiDARs with 32, 64, or 128 channels. Although a LiDAR with more channels is more expensive than a LiDAR with few channels, it can acquire more points and provide denser point clouds of the surrounding environment.

The illumination condition of the environment does not influence the functioning of LiDAR, and practically it can provide accurate measurement of the distance of obstacles during both day and night. Moreover, diverse weather conditions such as rain and snow have less impact on LiDAR operation than visual cameras. However, in rainy or foggy weather, the high amount of water in the atmosphere and on the surface of objects has negative impacts on the atmospheric transmission of the laser beam and target reflectance. Hence, due to environmental water impact on atmospheric extinction and target surface reflectance [37], diverse weather conditions slightly decrease the LiDAR performance.

1.4 Point Cloud

Point clouds are geometric point sets that are collections of points in the Euclidean space. From a mathematical point of view, a point cloud is a set of 3D points $\{P_i | i = 1, ..., n\}$, where each point P_i is a vector of its (x, y, z) coordinates. It can also include additional dimensions for representing other local or global features such as normal or intensity of received laser beam by LiDAR.

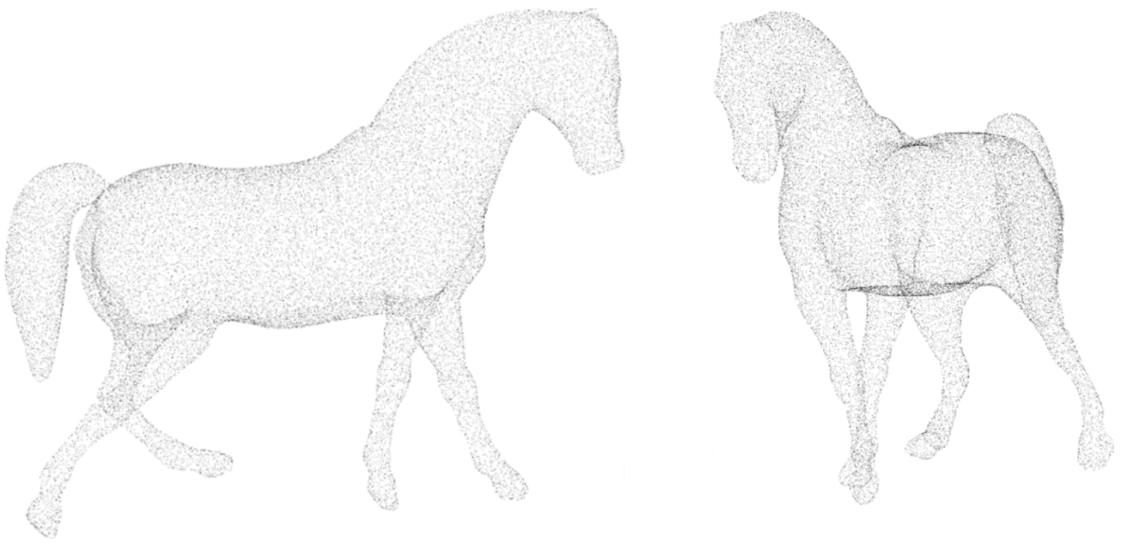

FIGURE 1.6 – An example of point cloud representation of a 3D horse model from two different viewpoints (3D model from SHREC database [3]).

While pixel arrays in images or voxel arrays in volumetric grids have structured arrangement, a point cloud is a group of points without a specific order, therefore invariant to its members' permutations [8]. The points are from a vector space with a distance metric. The distance from origin for a point (x, y, z) is $\sqrt{x^2 + y^2 + z^2}$. Figure 1.6 illustrates an example of a point cloud of a horse from two different viewpoints.

For autonomous driving applications, many 3D object detection algorithms exploit point clouds as their input data [36]. Specifically, there are 3D object detectors with architectures capable of extracting features directly from point clouds.

1.4.1 Point cloud Processing and Representation

There are three main methods for processing and representing the point clouds [8]. 2D view representation, volumetric representation, and point representation are standard approaches for addressing the point cloud processing and representation problem.

The volumetric representation method is based on discretizing the 3D space into 3D voxels and assigning the points to the voxels. In the point representation approach, the model directly learns over 3D point clouds in continuous vector space without voxelization. PointNet [38] and its improved version PointNet++ [39] extract all individual point features and then by utilizing the max-pooling technique, they extract the features of several points. 2D view representation approaches represent the point cloud as a collection of projected 2D image views, to which 2D convolutional layers can be applied.

In volumetric representation, first, we divide the point cloud into a 3D grid. A simple approach for partitioning considers equal size voxels that make it possible to exploit standard 3D convolution on the voxels. Nevertheless, by increasing the number of voxels to achieve higher resolution voxel space, computation and memory costs increase remarkably. Moreover, the number of voxels that do not contain any point grows significantly.

It is possible to use methods that leverage tree-like structures to obtain high-resolution voxel space without increasing computational costs and the number of empty voxels [33]. In this approach, the number of voxels in different regions of point cloud depends on the density of points in that region. Therefore, regions with lower point densities have lower resolutions and require less computation and memory.

Before leveraging the 2D convolution and popular CNN architectures for processing the point clouds, it is necessary to project the point cloud to a particular 2D view plane. This technique helps to have an image-like feature map that is suitable to use by CNNs. The most famous views in self-driving car applications are the top-down view or the Bird's Eye View (BEV) [33]. The advantages of BEV include minimizing perspective occlusions. Moreover, raw information about objects' orientation does not change.

Point representation based methods use raw point cloud. In [38], a novel deep neural network, PointNet, is proposed that directly consumes point clouds. The proposed network provides a unified approach to various 3D recognition tasks, including object classification, part segmentation, and semantic segmentation. The proposed model learns global point cloud features that can be used for object classification. The PointNet architecture consumes directly unstructured point clouds and uses max-pooling layers in the middle to extract global features.

The basic idea of PointNet, which allows the processing of unordered data, is to employ symmetric functions such as max-pooling. This method is invariant to rigid transformation and permutation. PointNet is highly robust to small perturbation of input points such as corruption because of outliers or missing data [38]. In fact, PointNet learns a spatial encoding of each point and then accumulates all individual point features to a global point cloud feature map.

1.5 Robustness

Achieving a high level of robustness is an important goal for machine learning and computer vision models. Moreover, having robust models and algorithms is essential for creating deep learning systems capable of being deployed in safety-critical applications. For instance, self-driving cars need to be able to cope with widely varying outdoor conditions such as fog, frost, snow, or sand storms.

In comparison with existing computer vision systems, the human vision system is significantly more robust [40]. Unlike current classifiers and object detectors implemented by deep learning methods, humans' vision system can not be misled by small changes in images. Besides, many forms of corruption, such as changing contrast, adding noise, manipulating the brightness, and other combinations of these corruptions, can not confuse the human vision system. Noticeably, humans can handle even abstract changes in the structure and style of images and their contents.

In recent years, models based on the deep neural network have achieved high accuracy on many classification and object detection tasks in 2D and 3D spaces. However, measuring the uncertainty of their predictions remains a challenging problem [41]. In many applications such as self-driving cars or medicine, finding methods for predicting a model's robustness could be useful. These robustness estimation methods can help determine the generalization capability of the model.

Deep learning has recently profoundly influenced various machine learning tasks, such as object detection, speech recognition, and natural language processing. Nevertheless, the shortage of methods to assess the model's robustness to perturbation of input is one of the most critical limitations of deep neural networks [13], which currently restricts their utilization in real-world applications.

1.6 Related Work

In [42], an approach for monocular 3D object detection from a single RGB image is proposed. As opposed to other methods in the literature, this approach does not take additional information as input like depth obtained from LiDAR or other depth estimators. The authors introduced a monocular 3D object detection network, MonoDIS, to estimate the objects' size and orientation at different scales. A two-stage architecture is used, which consists of a one-stage 2D detector (first stage) with an additional 3D detection head (second stage) constructed on top of features pooled from the detected 2D bounding boxes.

In [43], a method for 3D object detection and pose estimation from a single image is proposed. The proposed approach produces 3D bounding boxes by combining the estimation of stable 3D object properties, which is regressed by using a deep convolutional neural network with geometric constraints provided by a 2D object bounding box. The constraint that the 3D bounding box fits into the 2D detection window requires that each side of the 2D bounding box be touched by the projection of at least one of the 3D box corners.

In [44], a stereo R-CNN based 3D object detection method in autonomous driving scenarios is proposed. This method extends Faster R-CNN for stereo inputs to simultaneously detect and associate objects in the left and right images. Extra branches are added after stereo Region Proposal Network (RPN) to predict sparse keypoints, viewpoints, and object dimensions, which are combined with 2D left-right boxes to calculate a coarse 3D object bounding box and then recover the accurate 3D bounding box by a region-based photometric alignment using left and right ROIs.

In [28], the authors leveraged the point set feature learning and the RPN (region proposal network) for 3D detection. They also introduced a method that converts the point cloud into a dense tensor structure. To reach these goals, they introduced VoxelNet, which directly operates on the raw point cloud and produces the 3D detection results using a single end-to-end trainable network. Specifically, VoxelNet divides a point cloud into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding (VFE) layer.

In [45], an approach for 3D object detection by utilizing both LiDAR and image data is introduced. In the proposed method, the idea of utilizing multimodal information is used to perform region-based feature fusion. The proposed Multi-View 3D object detection network (MV3D) fuses the 2D projections with the camera image to bring additional information. In fact, the MV3D takes multimodal data as input and predicts the full 3D extent of objects in 3D space.

In [46], a two-stage 3D object detection framework that exploits the advantages of voxel-based and point-based methods is proposed. The first stage is a proposal generation network that uses raw point clouds as input to generate accurate proposals, while the second stage is for box prediction.

In [32], PointFusion, an early fusion model for 3D box estimation, is proposed, which directly learns to combine image and depth information optimally. The proposed model takes full advantage of combining the heterogeneous image and 3D point cloud data sources without introducing any data processing biases for 3D object box regression. The proposed model performs 3D bounding box regression from a 2D image and a corresponding 3D point cloud typically produced by LiDAR sensors.

In [47], Semantify-NN, a model-agnostic and generic robustness verification approach against semantic perturbations for neural networks is proposed. It is possible to insert the proposed semantic perturbation layers (SP-layers) to any given model's input layer. Semantify-NN can support robustness verification against a wide range of semantic perturbations. The proposed SP-layers allow us to explicitly define the dimensionality of perturbations and put explicit dependence between the manner and the effect of the semantic perturbation on different pixels of the image. However, the approach is proposed for classification tasks with RGB images input, and it is not described how it is possible to use it for regression tasks such as 3D object detection.

In [40], to evaluate the robustness of classifiers, ImageNet-C and ImageNet-P are introduced that are obtained by corrupting the ImageNet [48] test set with classical corruptions, such as blur, different types of noise and compression, and simulated weather corruptions such as snow and fog. These datasets include corrupted images by a total of 15 noise, blur, weather, and digital corruption types, each appearing at five severity levels or intensities.

In [10], an approach to evaluate 2D object detection models' performance when images are synthetically corrupted is introduced. It is shown that a range of usual object detection models experiences a drastic performance loss on corrupted images. Similar to [40], the authors proposed 15 corruptions on five severity levels to evaluate the impact of different corruption types on the 2D object detection models. Moreover, the authors focused on the less extreme but far more common problem of perceptible image distortions like blurry images, noise, or simulated natural distortions like snow and fog.

Chapitre 2

Method

The multimodal deep learning approach has achieved impressive 3D object detection results for self-driving cars where the training and testing distributions match. However, in the real-world, the conditions in which the models are deployed can often differ significantly from the model's training conditions. It is necessary to understand the impacts that dataset shifts have on the performance of these models. This problem has gained considerable attention, and several research projects [10, 49, 40] have shown unexpectedly high sensitivity of object detectors to various weather and lighting conditions.

In this work, we develop two different datasets to measure the robustness of a model. The first dataset contains artificially corrupted images with 15 different corruption types, each with five severity levels. The other dataset is a set of images consisting of scenes with diverse weather and lighting condition, including images in the night and rainy weather.

2.1 Datasets

Large volumes of high-quality training data are necessary to achieve successful machine learning models. Datasets are crucial for researchers and developers because algorithms and models for self-driving cars have to be trained and tested before the vehicle can go on the road. Particularly, big data has had a remarkable impact on the success of deep learning in computer vision. Recent works suggest that there is significant potential to increase object detection performance by utilizing even larger datasets [50, 36]. A self-driving car needs to be prepared to drive in different situations and confront rare scenarios on the roads. Hence, proper datasets for both training and testing of a self-driving car should consist of diverse scenes.

Supervised learning is the approach that most deep multimodal 3D object detection algorithms are exploiting [8]. Small datasets are insufficient to train high-capacity models that require to capture the further complexity caused by considering a third dimension in the estimation problem [51]. Hence, for training multimodal deep neural networks, large multimodal datasets with various driving conditions, object labels, and sensors are necessary. In other words, leveraging large multimodal datasets can remarkably enhance the accuracy and robustness of multimodal 3D object detectors.

The obtained results in [49] suggest that, for models with sufficient capacity, it is possible to improve the evaluation results by increasing the amount of training data without additional changes. Moreover, using large supplemental datasets for extra pre-training can enhance the robustness remarkably.

Although several datasets exist for multimodal deep learning, their size is relatively smaller than datasets for images [8]. Furthermore, for autonomous vehicles, the variation and scale of the environments that the data are acquired are not diverse [1], and the datasets are usually recorded in limited driving scenarios, weather conditions, and sensor setups. Moreover, they are usually imbalanced datasets with more cars than pedestrians and cyclists in the scenes.

Throughout the history of object recognition research, datasets have played an important role [13]. Researchers exploited datasets for measuring and comparing the performance of competing algorithms. Furthermore, benchmarks can encourage researchers to address complex and challenging problems. Table 2.1 provides an overview of well-known datasets for autonomous vehicle scenario that we leverage in this work.

Recently, to train the models for self-driving car applications, many new datasets are provided by companies and researchers. For instance, the CADC dataset [52] aims to promote research to improve self-driving in adverse weather conditions. This is the first public dataset to focus on real-world driving data in snowy weather conditions. Another example is the ApolloScape dataset [53] that contains large and rich labeling including semantic point cloud for each site, stereo, pixel-wise semantic labeling, lane-mark labeling, instance segmentation, 3D car instance, and highly accurate location for every frame in various driving videos from multiple sites, cities and day times.

We have chosen the KITTI, nuScenes, and Waymo datasets for creating our robustness test datasets. The main motivation behind this choice is the fact that these datasets are well-known in the self-driving car domain and they are leveraged widely by researchers to evaluate the models. The proposed benchmarks including accepted methods for evaluation of the results that help to increase the comparability of the test results on these datasets.

2.1.1 KITTI Dataset

Although the KITTI dataset [54] is one of the most widely used datasets for autonomous driving research, the diversity of its recording conditions is relatively low, and it is obtained from a narrow domain [8]. The scenes are collected by driving through a mid-sized German city - Karlsruhe - in clear weather during the day.

	KITTI	**nuScenes**	**Waymo**
Scenes	22	1000	1150
Hours	1.5	5.5	6.4
3D Boxes	80K	1.4M	12M
Lidars	1	1	5
Cameras	4	6	5
AVg. Points/Frame	120K	34K	177K
Maps	No	Yes	No
Visited Area (km^2)	-	5	76

Table 2.1 – An overview and comparison of the KITTI, nuScenes, and Waymo datasets that we have used in our work [1].

The 3D object detection benchmark of KITTI includes 7481 training images and 7518 test images, and the corresponding point clouds, consisting of a total of 80K labeled objects. Only the ground truth for training is provided, and the ground truth for the test set is not available to the public. Hence, we used the training set in this work and following [45, 55], we divided it into two almost equal and fixed subsets for training and testing the model in all experiences. In figure 2.1 two samples from the KITTI dataset contain objects from Pedestrian, Car, and Cyclist classes with their corresponding ground truth 2D bounding boxes are illustrated. The KITTI dataset acquired point clouds with a 64-beam LiDAR sensor.

2.1.2 nuScenes Dataset

In the context of autonomous vehicles, one of the largest datasets with ground truth labels is the nuScenes dataset [5], with overall 1000 different scenes. Each scene is a 20 seconds record for different sensors installed on a car. They are scenes captured in two cities in different lighting and weather conditions. the sensors include six cameras that captured panoramic views of the car's surrounding. It also contains LiDAR and RADAR sensors. It includes nearly 1.4 million frames.

The nuScenes dataset is a publicly available dataset that leverages a 32-beam LiDAR with 20Hz sweep capture frequency, a front camera with 1600×900 resolution, and a horizontal FOV (Field Of View) of 70°. The nuScenes dataset contains 390,000 LiDAR sweeps.

The nuScenes dataset has limitations, such as leveraging a low-quality LiDAR sensor with 32-channel that acquires 34K points per frame. Moreover, the dataset covers $5km^2$ effective area that limits its geographical coverage diversity. However, the nuScenes dataset includes additional data about the roads and sidewalks for all scenes by providing top-down semantic maps of the related areas. In figure 2.2 various samples of nuScenes dataset are illustrated.

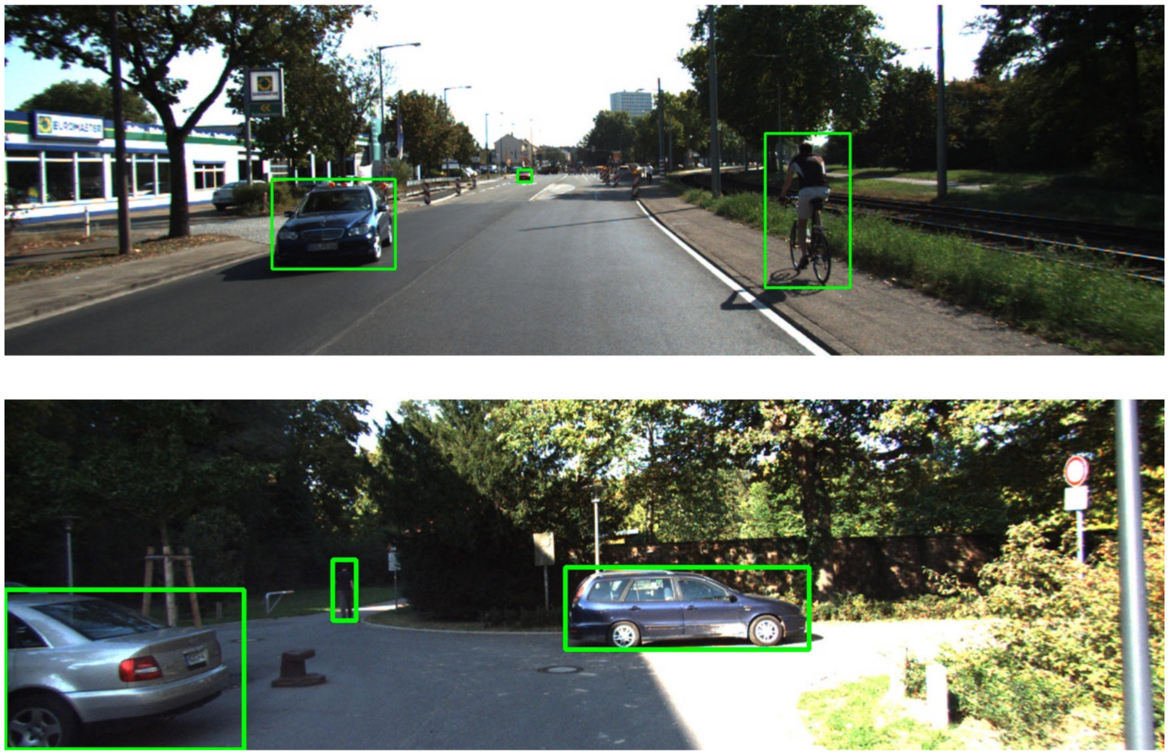

Figure 2.1 – Visualization of the ground truth 2D bounding boxes for two sample images from the KITTI dataset [4]. Green bounding boxes correspond to the car, pedestrian, and cyclist object classes.

2.1.3 Waymo Dataset

Waymo dataset [1] is a new large-scale dataset for training and evaluating models in the context of autonomous vehicles. The Waymo dataset includes 1150 scenes with 20 seconds duration. Waymo dataset is a diverse dataset consisting of well synchronized and calibrated high-quality LiDAR and camera data captured across diverse geographical areas at different times.

The Waymo dataset contains numerous manually annotated 3D bounding boxes for the LiDAR data and 2D bounding boxes for the camera images. The Waymo dataset contains around 12 million LiDAR box annotations. Figure 2.3 illustrates various samples from Waymo dataset in diverse weather and lighting conditions.

2.1.4 Robustness Evaluation Datasets

We adopted various datasets to evaluate the trained model's robustness. We derived these datasets from three well-known and widely used datasets for multimodal 3D object detection for self-driving cars, including KITTI, nuScenes, and Waymo datasets.

Following [40, 10], we built the KITTI-C dataset that contains images from the KITTI dataset

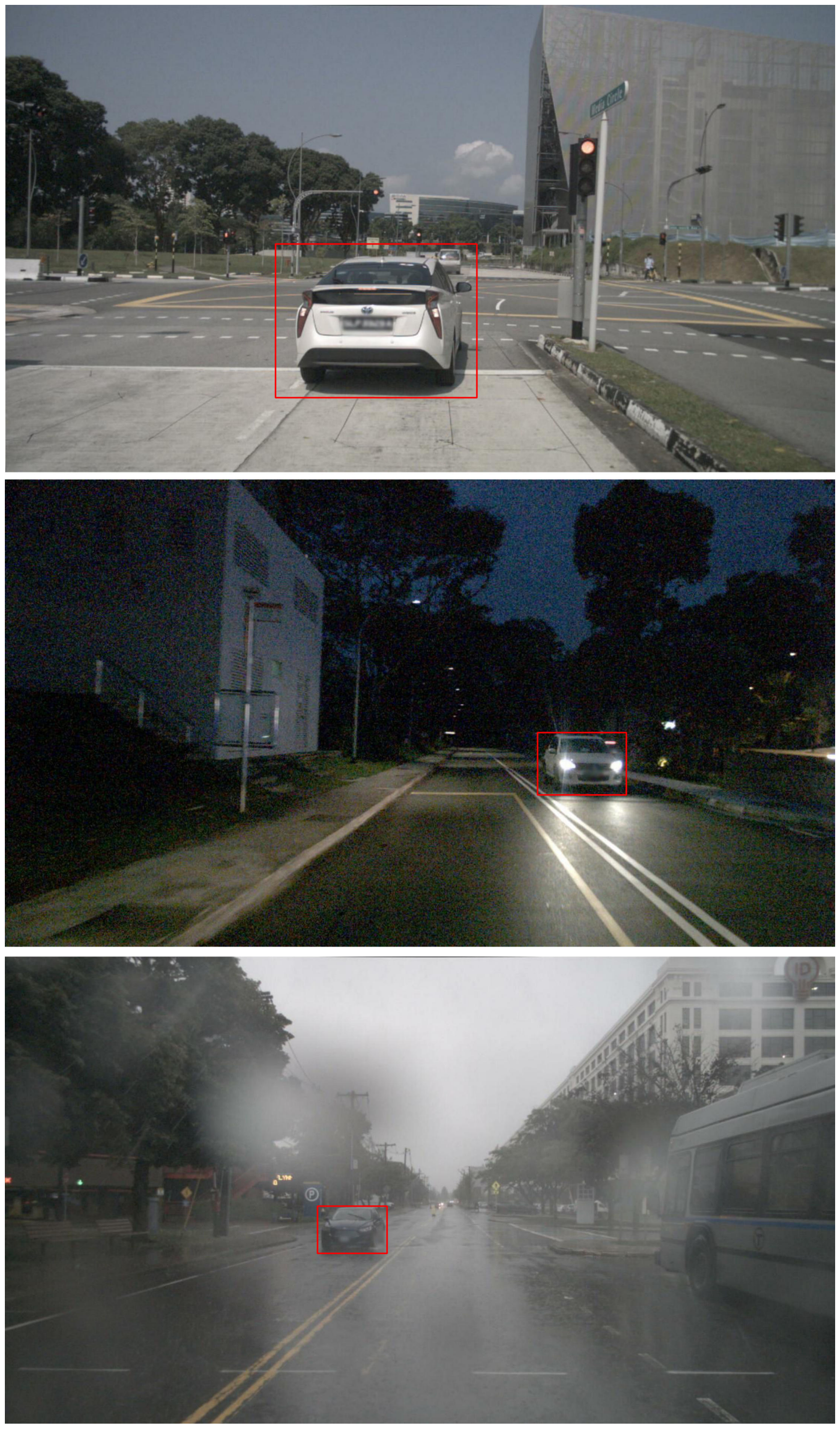

FIGURE 2.2 – Visualization of the ground truth 2D bounding boxes for three sample images in clear, night, and rain from the nuScenes dataset [5]. Red bounding boxes correspond to the car object class ground truth.

by applying 15 various corruptions with five levels of severity for each one. The applied corrup-

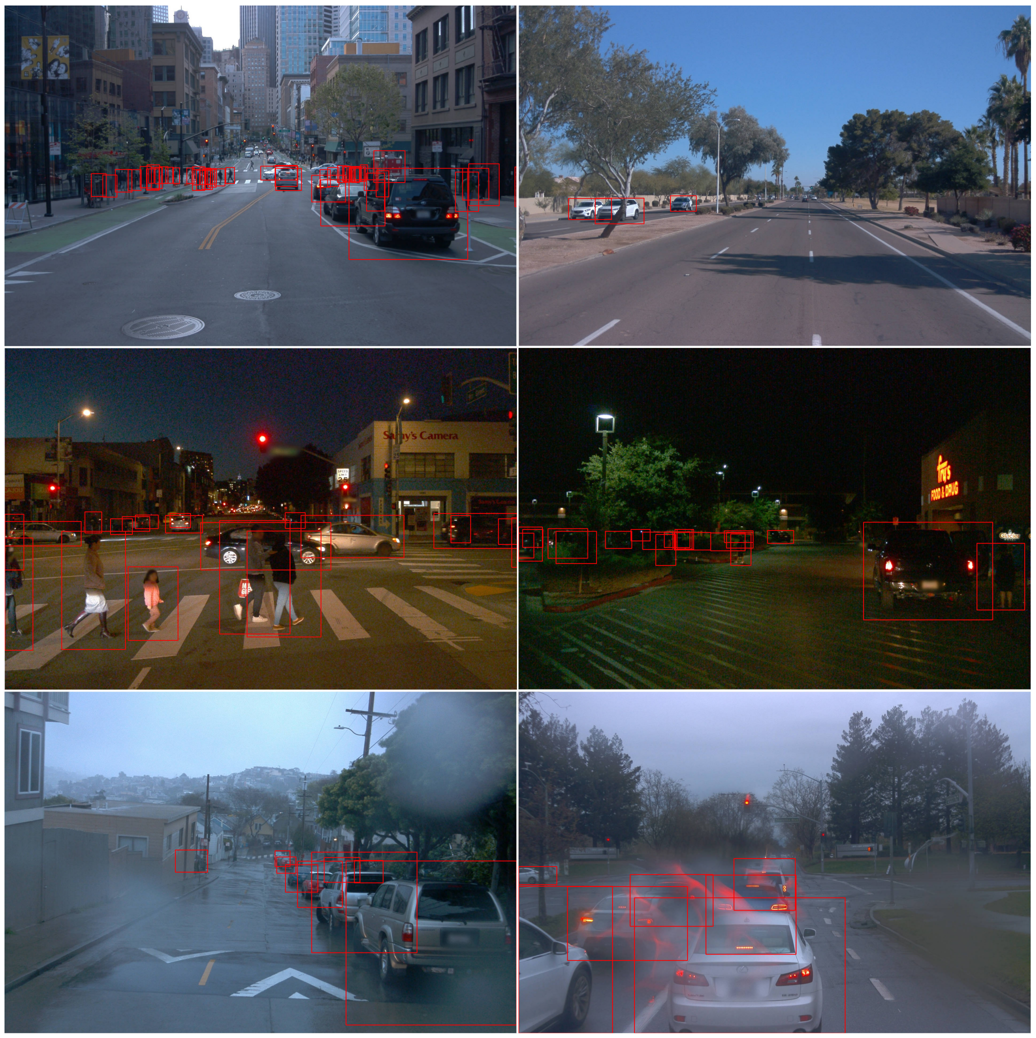

FIGURE 2.3 – Visualization of six sample images with their ground truth 2D bounding boxes in diverse weather and lighting conditions from the Waymo dataset [1].

tions and perturbations include zoom blur, brightness, fog, glass blur, motion blur, impulse noise, defocus blur, pixelate, frost, snow, elastic transform, JPEG compression, shot noise, Gaussian noise, and contrast (figure 2.5).

Some of the proposed artificial corruptions try to simulate the effects of diverse weather conditions on the images. These corruptions include snow, fog, and frost. Figure 2.4 demonstrates them for all five severity levels in the KITTI-C dataset.

Despite the benefits of leveraging artificial corruptions for assessing the model's robustness against some kinds of distortions, including noises, digital and blur corruptions, and using them to augment the training data, the fact is, an autonomous vehicle in practice needs to cope with naturally diverse weather conditions. Therefore, we need to estimate the robustness of the model against natural and realistic images.

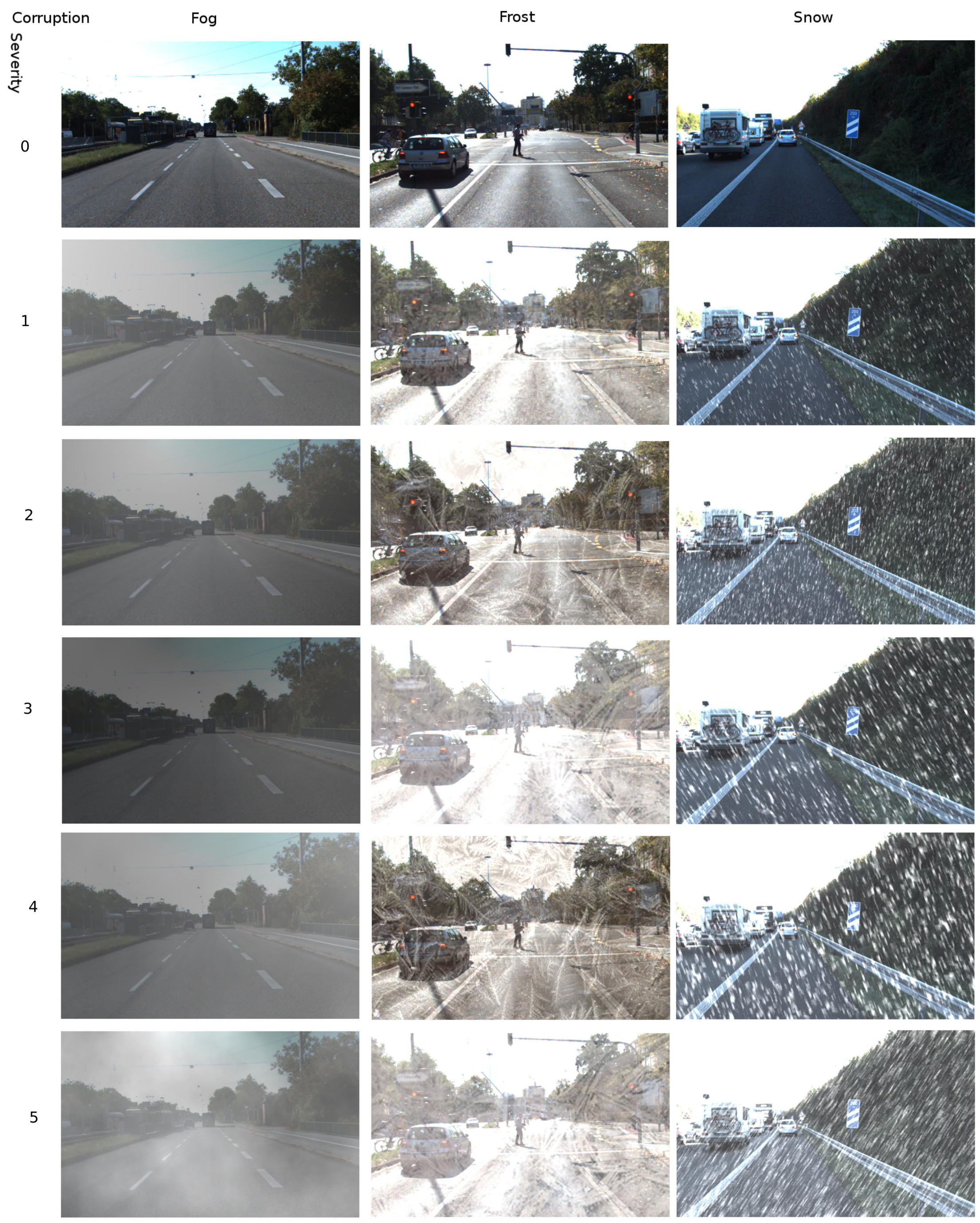

Figure 2.4 – Three corruptions for simulating natural phenomena including snow, fog, and frost for the arbitrary images from KITTI-C for all five severity levels. We cropped the images to be able to illustrate all cases in limited space.

To provide robustness evaluation against natural corruptions, by using descriptions provided during the annotation of scenes in nuScenes, we have divided scenes into three categories : nuScenes-Night, nuScenes-Rainy, and nuScenes-Clear, as illustrated in figure 2.6. We exploited

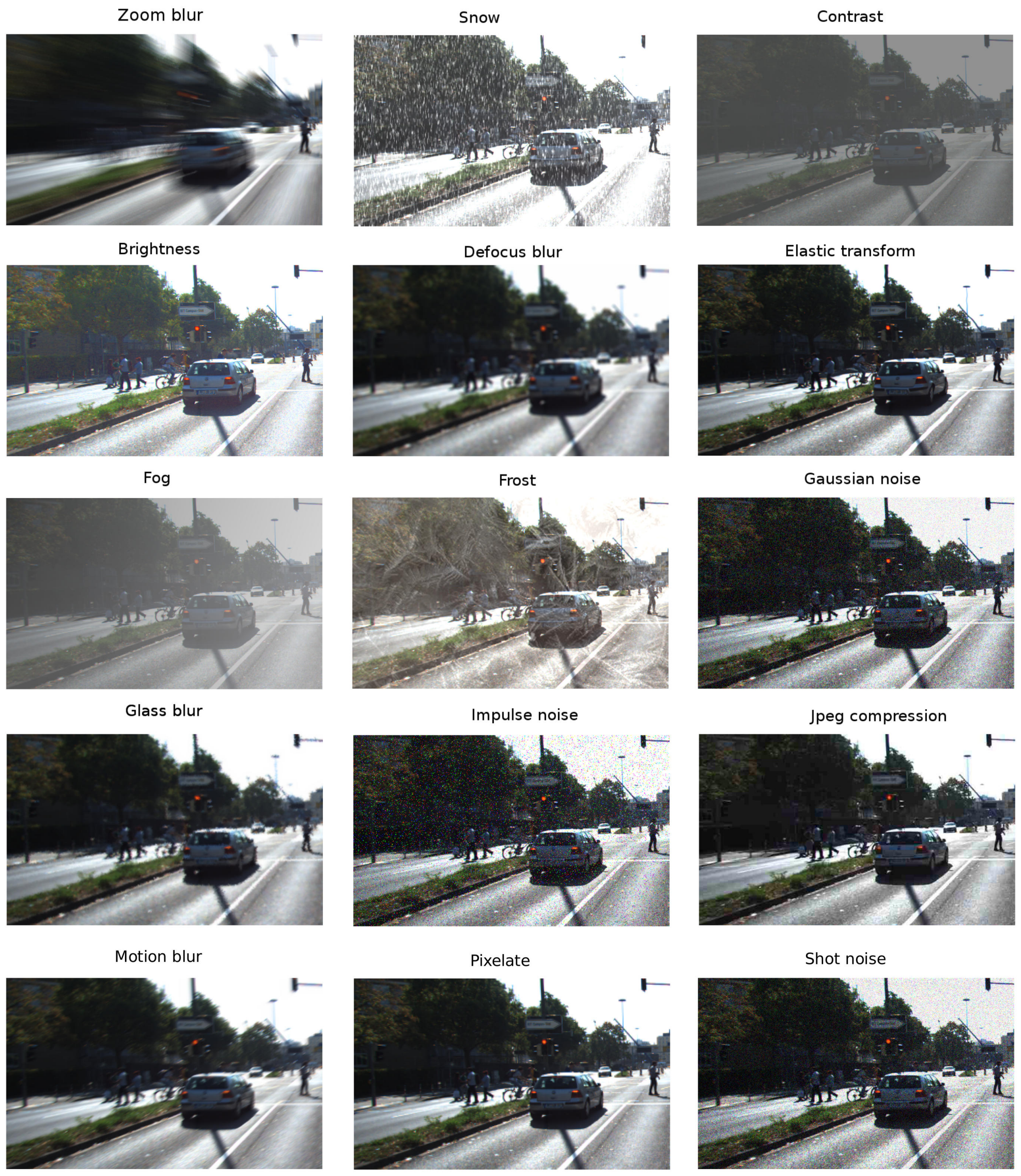

FIGURE 2.5 – An arbitrary image from KITTI-C for all different corruptions with severity level 2. We cropped the images to be able to illustrate all corruptions in limited space.

the nuScenes Night and Rainy subsets only to evaluate the model's robustness in diverse weather and lighting conditions, and we did not use them during training. We did not consider the scenes that their description contained both rainy and night. We used the Clear subset for training the model. We did the same procedure for breaking down the Waymo dataset to Clear, Night, and Rainy subsets. Table 2.2 provides the number of samples in our proposed sub-datasets.

	Clear	**Rainy**	**Night**
nuScenes	8,501	5,927	3,503
Waymo	27,365	2,498	2,641

TABLE 2.2 – The number of images and their corresponding point clouds in Waymo-Clear, Waymo-Rainy, Waymo-Night, nuScenes-Clear, nuScenes-Rainy, and nuScenes-Night.

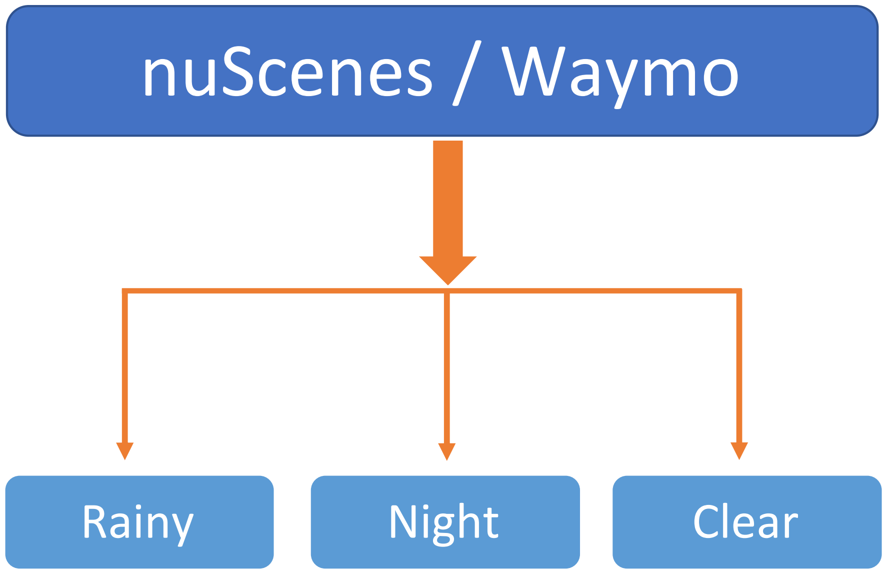

FIGURE 2.6 – The nuScenes dataset is divided into three subsets, including the Clear subset that contains scenes captured during the day in a good weather condition, the Night subset that includes scenes captured at night, and the Rainy subset for scenes in rainy weather. We did the same procedure for dividing the Waymo dataset into three subsets.

2.2 Model Architecture

2.2.1 Frustum PointNets

The Frustum PointNets [6] is one of the best models among multimodal methods that exploits both 3D point clouds and 2D images to detect objects of interest in a 3D environment. In this section, we present the model introduced by *Qi et al.* in [6] that we leveraged in this research to show the effectiveness of our robustness test datasets.

In comparison with 3D object detectors, 2D object detectors are investigated better, and there are larger datasets with labels for training them, which can help achieve better results and performance [8]. Hence, it is possible to use a 2D object detector's output to achieve better results in the case of multimodal 3D object detectors.

Frustum PointNets proposes leveraging a 2D object detector to predict 2D bounding boxes and categories of objects as the first step. Then it exploits these 2D bounding boxes for extracting frustums in point clouds for each object. In the second step, Frustum PointNets leverages these point clouds within the frustums for finding the 3D bounding boxes. In fact, utilizing the frustums helps reduce the search space in point clouds and improves the model's performance.

As illustrated in figure 2.7, the pipeline of the Frustum PointNets consists of several modules. First, by extruding 2D bounding boxes of objects of interest from a 2D object detector's outputs, the 3D bounding frustums of objects are extracted. Then, by using two variants of PointNet [38], in the 3D space trimmed by each of the 3D frustums, 3D object instance segmentation, and 3D bounding box regression are performed. The PointNets use multilayer perceptron (MLP) networks of various sizes. The segmentation network predicts the 3D mask of the object of interest. The regression network estimates the 3D bounding box, which covers the entire object even if only part of it is visible.

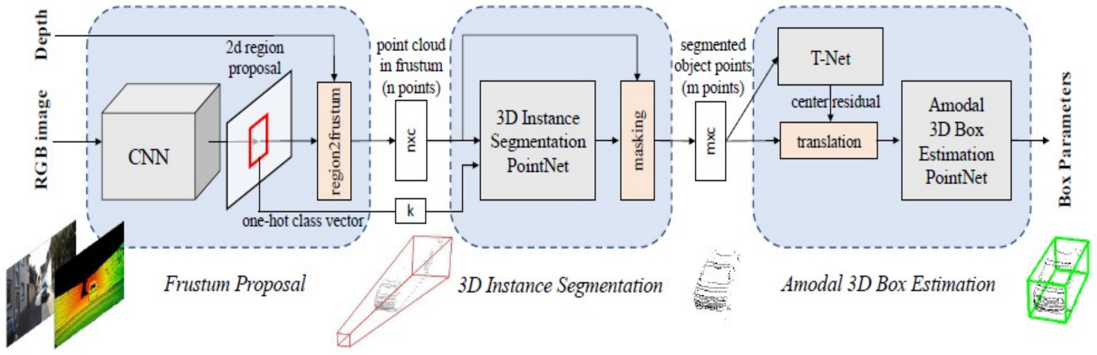

FIGURE 2.7 – The architecture of the Frustum PointNets pipeline for 3D object detection (figure reproduced from [6]).

3D Instance Segmentation PointNet

This module of the Frustum PointNets pipeline performs instance segmentation on the points within frustums. Since the objects are naturally separated in physical 3D space, segmentation in a 3D point cloud is much more natural and straightforward than in images where pixels from distant objects can be nearby [6]. The schematic illustration of this network is demonstrated in figure 2.8.

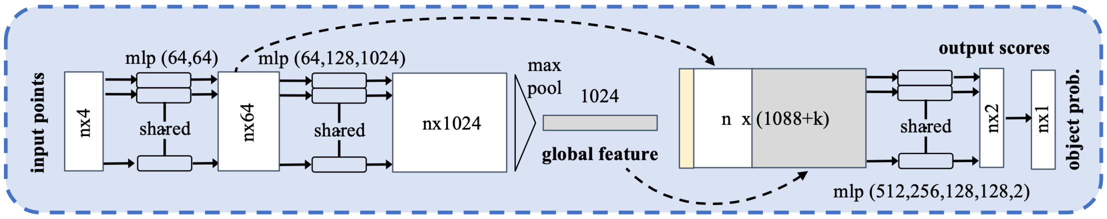

FIGURE 2.8 – The architecture of 3D Instance Segmentation PointNet. The input is a frustum point cloud with n points. The points can optionally include the intensity of reflection. The outputs are point-wise class prediction scores. k is the number of object categories (figure reproduced from [6]).

3D Box Estimation PointNet

The 3D Instance Segmentation module's output feeds to 3D Box Estimation PointNet, and this module estimates the 3D bounding box of the object. It leverages a box regression PointNet [38], a transformer network, and the T-Net simultaneously to obtain the oriented 3D bounding box of the object. The T-Net module predicts the center point of the 3D bounding box. The schematic overview of 3D Box estimation PointNet and T-Net are demonstrated in figures 2.9 and 2.10 respectively.

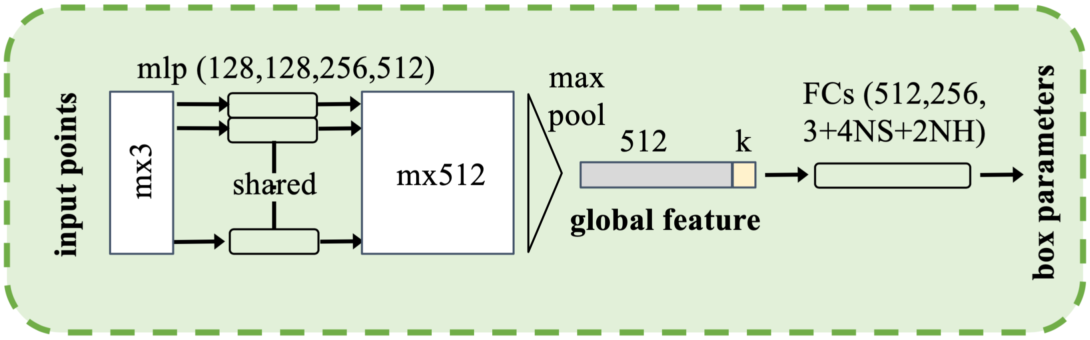

FIGURE 2.9 – The architecture of 3D Box Estimation PointNet. k is the number of classes. NS is the number of size templates, and NH is the number of equally split angle bins. The model classifies both sizes and headings (NS scores for size, NH scores for heading) and predicts residual values (figure reproduced from [6]).

Multi-task Losses

Localization and classification are two purposes of object detection. Under object detection evaluation metrics, the accuracy of localization is an essential calculable indicator [11]. Consequently, increasing localization accuracy can improve detection performance, notably. Designing a novel loss function to measure predicted boxes' accuracy is an effective way to increase localization accuracy.

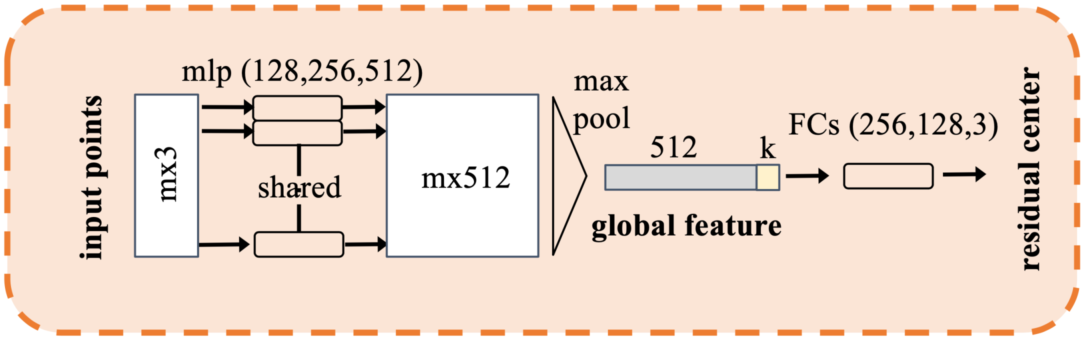

Figure 2.10 – The architecture of T-Net. This module regresses residuals for the 3D bounding box center position. It uses the object point cloud as input. T-Net is actually a modified version of the PointNet [38] classification network (figure reproduced from [6]).

At the same time, all three modules i.e. 3D instance segmentation PointNet, T-Net, and box estimation PointNet, are optimized with multi-task losses [6]. In equation 2.1, $\mathcal{L}_{seg}$ represents the loss for 3D Instance Segmentation PointNet module, $\mathcal{L}_{c1-reg}$ is for T-Net and $\mathcal{L}_{c2-reg}$ is for center regression of box estimation network. $\mathcal{L}_{h-cls}$ and $\mathcal{L}_{h-reg}$ are losses for heading angle prediction. $\mathcal{L}_{s-cls}$ and $\mathcal{L}_{s-reg}$ are for box size. We use Softmax for classification tasks and Huber loss for regression tasks.

$$\mathcal{L}_{multi-loss} = \mathcal{L}_{seg} + \lambda(\mathcal{L}_{c1-reg} + \mathcal{L}_{c2-reg} + \mathcal{L}_{h-cls} + \mathcal{L}_{h-reg} + \mathcal{L}_{s-cls} + \mathcal{L}_{s-reg} + \gamma\mathcal{L}_{corner}) \quad (2.1)$$

The corner loss ($\mathcal{L}_{corner}$) aims to penalize the center, size, and yaw angle estimates simultaneously to achieve optimal 3D bounding box estimation. For this purpose, we calculate the sum of the L1 distances between the eight corners of the ground truth 3D bounding box and the predicted one [6]. λ and γ are weights for 3D bounding boxes loss and corners loss, respectively.

2.3 Evaluation Metrics

Evaluation metrics are used to assess the quality of deep learning models. Evaluating deep learning models and algorithms is necessary for any research project. Using evaluation metrics helps us to ensure that the model is operating correctly and optimally. Various evaluation metrics are available to evaluate the performance of a model. One of the principal metrics for evaluating the performance of a model is Average Precision (AP).

Average precision is derived from precision and recall. AP is usually evaluated in a category-specific manner and computed for each object category separately. The AP metric makes a summary of the shape of the precision-recall curve. For this purpose, we leverage the modified definition of AP metric by KITTI benchmark [4, 42] which is the mean precision at a set of 40 equally spaced recall levels :

$$AP = \frac{1}{40} \sum_{r \in \{\frac{1}{40}, \frac{2}{40}, \ldots, 1\}} \rho_{interp}(r) \tag{2.2}$$

Following the [56], we consider that the precision at each recall level r is interpolated by taking the maximum precision measured for which the corresponding recall exceeds r :

$$\rho_{interp}(r) = \max_{r^{'}:r^{'} \geq r} \rho(r^{'}) \tag{2.3}$$

For each object category, we obtain the precision-recall curve from a method's ranked output. In equation 2.4, for a given threshold, the recall is the number of true positive predictions over the sum of false negatives and true positives. The number of true positive predictions over the number of all predictions is precision.

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \tag{2.4}$$

Where TP is the abbreviation for True Positive, which shows the number of objects that the model predicted correctly, and FP is False Positive and shows the number of detected objects that do not have a corresponding ground truth and the prediction is false. FN is False Negative and shows the number of objects in the ground truth that the model could not detect.

The usual way to evaluate object detection methods is the average precision (AP) [8]. Therefore, we adopt this approach in our work. Considering intersection over union (IoU), we can judge the regression quality estimation with the IoU between the predicted bounding box and its corresponding assignment ground truth box.

2.3.1 Assessing the Robustness

In this work, we have introduced datasets for evaluating the robustness of models. To examine our proposal's effectiveness, we have chosen a state-of-the-art multimodal model, trained it on different datasets, and then evaluated and compared its robustness with our robustness test datasets.

We exploited two different approaches to make our datasets for testing robustness. In the first scenario, following [10, 40], we first built a synthesized dataset based on the KITTI dataset, KITTI-C, which contains corrupted images by applying various synthetic corruptions. Overall, we used 15 different corruptions, each with five different severity levels, to fully evaluate the models' robustness. Although this method gives us a good overview of our model's capability facing corrupted images, the lack of considering real scenes is its drawback.

In order to assess the performance of the model on KITTI-C, we follow [10] and leverage **relative performance under corruption** *(rPC)* in equation 2.6. The **mean performance under corruption** *(mPC)* for N_C corruptions, each one with N_S severities is :

$$mPC = \frac{1}{N_C} \sum_{c=1}^{N_C} \frac{1}{N_S} \sum_{s=1}^{N_S} P_{c,s} \tag{2.5}$$

Where the $P_{c,s}$ is the model's performance under corruption c with severity s, and in this work, we use AP metric as discussed in the previous section for evaluating the performance of the model. In equation 2.6, P_{clean} is the model's performance on clean data, which is the original dataset, before applying any corruption.

$$rPC = \frac{mPC}{P_{clean}} \tag{2.6}$$

To have a realistic evaluation of robustness, we proposed four other datasets that contain images from large-scale nuScenes and Waymo datasets captured in diverse weather and lighting conditions. The annotations of nuScenes and Waymo datasets contain records that annotators have written short descriptions about each scene. Using these descriptions, we have separated scenes of nuScenes and Waymo captured during the night as the datasets for evaluating extreme lighting conditions' performance.

The nuScenes dataset has descriptions for scenes that are captured during rainy days. Therefore, we leveraged these descriptions to build the nuScenes-Rainy dataset, while Waymo has no descriptions for weather conditions, so for Waymo, we separated rainy scenes manually. We use these subsets of nuScenes and Waymo for measuring the robustness of the multimodal model, Frustum PointNets, in the rainy weather for real scenes. In the table 2.3, the number of objects in each labeled category for all our subsets of nuScenes and Waymo is provided.

	Clear		**Rainy**		**Night**	
	nuScenes	**Waymo**	**nuScenes**	**Waymo**	**nuScenes**	**Waymo**
Car	36,885	221,788	34,649	26,599	12,036	17,798
Pedestrian	17,231	131,783	6,002	13,412	4,738	6,720
Cyclist	906	3,419	445	60	601	117

TABLE 2.3 – Labeled objects of interest counts in Waymo-Clear, Waymo-Rainy, Waymo-Night, nuScenes-Clear, nuScenes-Rainy, and nuScenes-Night.

In all cases, during training, our model has not seen the images from our proposed robustness test datasets. This condition is a prerequisite for our method to assess a given model's robustness accurately and provide meaningful results.

Chapitre 3

Experimental Results and Discussion

In this chapter, different experiments and evaluations are conducted to verify the effectiveness of our proposed robustness test datasets to assess the robustness of Frustum PointNets as a sample model. Moreover, quantitative and qualitative results are provided. At the end of this chapter, we have a comprehensive discussion of the results and future work.

We created KITTI-C dataset by applying various common corruptions to the standard KITTI dataset, as discussed in detail in chapter two. Then we leveraged all subsets of the KITTI-C dataset to evaluate the robustness of the Frustum PointNets model.

We have built nuScenes-Rainy, nuScenes-Night, Waymo-Rainy, and Waymo-Night subsets of nuScenes and Waymo datasets. Then we evaluated the Frustum PointNets model on these proposed subsets to test its robustness in case of natural adverse weather and lighting conditions.

3.1 Preparing Data

One of the difficulties when evaluating a model's performance on different datasets is linked to the fact that different datasets use different sensor setups. Moreover, the annotation formats of datasets are different. We have converted all sub-datasets of nuScenes and Waymo to KITTI benchmark format to cope with this problem.

To this end, we leveraged the captured images by the front camera in nuScenes and Waymo datasets. Furthermore, the KITTI benchmark considers three different levels of difficulty for objects of interest, include Easy, Moderate, and Hard, according to their size on the 2D images and their occlusion and truncation levels. Following [57], for nuScenes and Waymo datasets, we exploited the depth ranges of objects of interest to determine their level of difficulty for the detection task. Therefore, we replaced the pixel size, occlusion, and truncation thresholds criteria on 2D bounding boxes in the KITTI benchmark with 30, 50, 70 meters on object depths for Easy, Moderate, and Hard levels of difficulty for Waymo and nuScenes sub-datasets.

3.2 Evaluating the Robustness

In this section, we leveraged the proposed approach to assess the robustness of the Frustum PointNets model. We exploited similar 2D object detector architecture for predicting the 2D bounding boxes of the objects of interest in the 2D images in all experiences. Moreover, we have used the same architectures for the 3D Instance Segmentation PointNet and 3D Box Estimation PointNet modules in all tests. Following [6], we set the weights for 3D bounding boxes loss and corners loss in multi-task losses in equation 2.1 to $\lambda = 1$ and $\gamma = 10$ respectively.

3.2.1 Evaluation of Frustum PointNets on KITTI-C

In this experience, we have exploited KITTI dataset for training the Frustum PointNets model, and then we have evaluated the robustness of the trained model by our KITTI-C dataset. For 2D detector, we used well-known model, Faster R-CNN [19] with Resnet [58] for extracting the features. The Resnet backbone of the 2D detector of the pipeline is pre-trained on Imagenet classification dataset [48] and COCO object detection dataset [59] and further, it is fine-tuned on the KITTI 2D object detection dataset to classify and predict 2D bounding boxes of instances.

Classes/APs	AP_{2D}			AP_{BEV}			AP_{3D}		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	95.79	88.68	80.02	81.68	74.26	72.71	73.97	63.19	60.04
Pedestrian	77.65	74.07	66.37	67.40	58.97	51.63	63.73	52.18	45.05
Cyclist	81.58	58.61	56.96	67.07	44.88	43.32	64.65	43.71	41.58

TABLE 3.1 – Results for Average Precision (AP in %) in 2D, 3D, and BEV scenarios on the KITTI test set without any corruption (clean KITTI). The results for easy, moderate, and hard difficulties for Car, Pedestrian, and Cyclist classes are provided.

Classes/(mPC,rPC)	$2D$ **(mPC/rPC)**			BEV **(mPC/rPC)**			$3D$ **(mPC/rPC)**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	64.60/0.67	56.30/0.63	50.68/0.63	61.73/0.75	52.56/0.71	47.11/0.65	56.10/**0.76**	45.67/0.72	41.14/0.68
Pedestrian	46.35/0.59	41.17/0.55	36.60/0.46	40.85/0.60	34.97/0.59	30.37/0.59	37.91/0.59	32.12/0.61	27.99/0.62
Cyclist	38.52/0.47	26.98/0.46	25.99/0.45	32.51/0.48	22.50/0.50	21.14/0.49	30.96/0.48	21.42/0.49	20.21/0.48

TABLE 3.2 – Results for **mean performance under corruption** *(mPC in %)* and **relative performance under corruption** *(rPC)* in equations 2.6 and 2.5 in 2D, 3D, and BEV scenarios on the KITTI-C. The results for easy, moderate, and hard difficulties for Car, Pedestrian, and Cyclist classes are provided.

Since the ground truth for the KITTI test set is not available, and the access to the test server is limited, following [45, 55], we divided 7481 training images and their corresponding point clouds of the training data of KITTI into a training set and a test set, which results in 3712 data samples for training and 3769 data samples for test. The split avoids samples from the same sequence being included in both the training and the test subsets.

We used the training subset of the KITTI to train the 2D object detector and Frustum PointNets models. We extracted frustums of the objects of interest during the training stage by leveraging ground truth 2D bounding boxes for Car, Pedestrian, and Cyclist categories. We leveraged the trained model for detecting the objects in the test subset.

The average precision for 3D detection and bird's eye view (BEV) detection is calculated using the KITTI benchmark's official evaluation method. We provided the obtained results for the evaluation of the trained model on the KITTI dataset in table 3.1. We denoted average precision (AP) for 2D, 3D and BEV detection tasks by AP_{2D}, AP_{3D} and AP_{BEV} respectively.

To assess the robustness of the model against synthetic corruptions, we leveraged our KITTI-C dataset. The summary of the detection results for Car, Pedestrian, and Cyclist categories with moderate difficulty for all 15 corruptions with various severities is demonstrated in figures 3.2, 3.4, and 3.3 for AP_{2D}, AP_{BEV}, and AP_{3D} respectively. The table 3.2 shows the results for mean performance under corruption *(mPC)* and relative performance under corruption *(rPC)* in equations 2.6 and 2.5 in 2D, 3D, and BEV scenarios for Car, Pedestrian, and Cyclist classes with different difficulties.

3.2.2 Evaluation of Frustum PointNets on nuScenes-Rainy and nuScenes-Night

To evaluate the robustness of Frustum PointNets on subsets of nuScenes dataset that include rainy and night scenes, we first trained the 2D detector on the nuScenes-Clear dataset. This subset of the nuScenes dataset contains only clear scenes captured during the daytime while the weather is not rainy.

The nuScenes-Clear dataset consists of 8501 images captured by the front camera among six cameras used to build the original nuScenes dataset. The related point clouds to these scenes are also available. After training the 2D detector as the first step of the pipeline used in the Frustum PointNets model, we used the ground truth 2D bounding boxes to extract frustums of instances for training the model to predict 3D bounding boxes.

We have trained the 2D object detector for 100 epochs with batch size 64. We exploited the model, which is pre-trained on the KITTI dataset, to use transfer learning advantages. For training the 2D detector, we used Stochastic Gradient Descent (SGD) optimizer with the learning rate equal to 0.0025 and momentum equal to 0.9. We set the weight decay equal to 0.0001. We trained the 3D Instance Segmentation PointNet and 3D Box Estimation PointNet modules for 60 epochs with suggested hyper-parameters in [6].

The evaluation results of the Frustum PointNets on nuScenes-Rainy, nuScenes-Night, and test set of nuScenes-Clear for 2D, BEV and 3D scenarios are provided in tables 3.3, 3.4, and 3.5. The figure 3.5 demonstrates the 3D detection results for two samples from the nuScenes-Rainy dataset. In the figure 3.6, results of detection of 3D bounding boxes for two samples from the nuScenes-Night are illustrated.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	73.87	65.66	56.96	28.41	23.24	18.53	28.66	23.10	20.71
Pedestrian	73.29	58.91	50.99	21.80	18.21	14.52	20.22	15.74	14.63
Cyclist	76.18	60.72	56.72	2.73	2.67	2.78	9.09	9.09	9.09

TABLE 3.3 – The evaluation results (AP_{2D} in %) of the Frustum PointNets on nuScenes-Rainy, nuScenes-Night, and test set of nuScenes-Clear for 2D detection.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	35.18	26.62	23.24	12.72	9.89	8.27	26.80	21.88	19.80
Pedestrian	34.75	26.88	24.63	10.07	8.12	6.88	13.19	10.77	10.60
Cyclist	16.18	12.11	11.25	0.07	0.07	0.07	3.03	3.03	3.03

TABLE 3.4 – The evaluation results (AP_{BEV} in %) of the Frustum PointNets on nuScenes-Rainy, nuScenes-Night, and test set of nuScenes-Clear for BEV detection.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	20.75	14.85	13.30	3.50	2.67	2.67	16.89	13.35	12.64
Pedestrian	22.97	18.00	16.41	3.57	3.19	3.19	10.20	9.09	9.09
Cyclist	6.36	5.90	5.76	0.05	0.05	0.05	0.70	0.70	0.70

TABLE 3.5 – The evaluation results (AP_{3D} in %) of the Frustum PointNets on nuScenes-Rainy, nuScenes-Night, and test set of nuScenes-Clear for 3D detection.

3.2.3 Evaluation of Frustum PointNets on Waymo-Rainy and Waymo-Night

In this section, we provide the results of applying our method on Waymo sub-datasets. To this end, first, we have trained our 2D object detector and Frustum PointNets on Waymo-Clear. Then we evaluated the trained model on Waymo-Rainy and Waymo-Night sub-datasets.

We have trained Faster R-CNN as our 2D detector for 60 epochs with batch size 64. We leveraged Stochastic Gradient Descent (SGD) as the optimizer with the learning rate equal to 0.0025, momentum 0.9, and weight decay 0.0001. Then we exploited this trained model for predicting the 2D bounding boxes in all experiments on Waymo sub-datasets. We leveraged the ground truth 2D bounding boxes to extract frustums and trained the 3D Instance Segmentation PointNet and 3D Box Estimation PointNet modules on extracted frustums for 60 epochs with hyper-parameters following [6].

In the tables 3.6, 3.7, and 3.8, the evaluation results of the Frustum PointNets on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for 2D, BEV, and 3D scenarios are provided. The figure 3.7 illustrates the 3D detection results for two samples from Waymo-Rainy. In figure 3.8, the 3D detection results of two samples from Waymo-Night are demonstrated.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	90.75	90.66	89.64	60.22	60.52	50.46	13.41	14.56	9.09
Pedestrian	79.91	81.08	80.95	51.39	50.19	41.61	9.09	9.09	9.09
Cyclist	84.31	85.96	87.66	17.67	17.67	17.67	-	-	-

Table 3.6 – The evaluation results (AP_{2D} in %) of the Frustum PointNets on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for 2D detection.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	57.00	59.84	53.57	41.92	42.77	35.25	13.98	16.09	15.54
Pedestrian	52.10	57.17	56.52	26.97	26.03	21.01	9.09	9.09	9.09
Cyclist	50.24	50.91	54.39	17.67	17.67	17.67	-	-	-

Table 3.7 – The evaluation results (AP_{BEV} in %) of the Frustum PointNets on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for BEV detection.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	34.42	40.08	35.67	24.51	23.60	17.40	10.76	14.45	9.09
Pedestrian	45.81	52.92	46.73	19.59	19.27	17.66	9.09	9.09	9.09
Cyclist	44.32	48.11	49.63	17.67	17.67	17.67	-	-	-

Table 3.8 – The evaluation results (AP_{3D} in %) of the Frustum PointNets on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for 3D detection.

3.3 Discussion and Future Work

As we can see in the table 3.1, the detection results for the Car category is better than the results for the Pedestrian and Cyclist categories. The reason could be the fact that there are more car instances (28,742 instances) than pedestrians and cyclists instances (4,492 and only 1,627 instances for pedestrian and cyclist, respectively) in the KITTI dataset. Therefore, during training, the model learns to detect cars more accurately than pedestrians and cyclists. Moreover, in the same distance, a car is a bigger object than a pedestrian or a cyclist, so it has more points in point clouds, and the model can estimate its bounding box more precisely.

In the table 3.2, we can observe that the rPC of our trained model for 3D and BEV object detection for all difficulties are greater than rPC of the 2D object detector. Therefore, by leveraging the synthetic corruptions method, we found that the 3D object detector is more robust than the 2D object detector, due to the point clouds, as one of the modalities of the input of our multimodal object detector, are not affected by inducing the corruptions to images. Moreover, in all cases, rPC for the Car class is better than rPC for the Pedestrian and the Cyclist classes. In fact, it follows the pattern of results in the table 3.1.

We can observe in plots of detailed results across different corruption types in the figures 3.2, 3.3, and 3.4 that the model performance degraded by increasing the level of severity of corruptions. However, for some corruptions, such as zoom blur, the degradation is severe, while for some others, like elastic transform, the model is more robust. These results can help to select appropriate data augmentation during the training of the model.

The difference between the degradation rate of the results for various corruptions depends on the nature of the corruption and the method that the model leverages to extract the features. The CNNs, as the backbone of the first stage of Frustum PointNets, are vulnerable to some types of corruptions such as Gaussian and blur noises more than elastic transform or brightness. The reason could be the fact that convolution operation performs locally on nearby pixels. Therefore, for corruptions such as noises, although only some pixels change severely, it has a drastic negative impact on CNN's result. In contrast, for corruptions such as elastic transform and brightness, the nearby pixels change at the same rate, and they degrade the image globally. Hence they are less problematic for CNN to extract the features, and the model is less sensitive to the increase of severity for these corruptions.

In figure 3.9, we demonstrated the evaluation results for class Car with easy and hard difficulties for all 15 corruptions with various severity levels. As we can see in this figure, in all cases, the zoom blur and snow corruptions have the most negative impact on the results. Due to the importance of the textures for object recognition by CNNs compared to the shape of the objects [60], the severe impact on the objects' texture by zoom blur and snow corruptions leads to a severe decrease in results.

Furthermore, in the figure 3.9, we can surprisingly observe that the AP_{3D} for class Car with easy difficulty, slightly increased for defocus blur and glass blur corruptions with severity one and two. The reason could be the positive impact of these corruptions with low severity on estimating the 2D bounding boxes that leads to better frustum extraction in 3D space and achieve better 3D detection results than the evaluation results on the clean KITTI test set.

In our experience, we exploited simple data augmentation including horizontal flip of images and random translation and scaling of 2D bounding boxes. The most critical weakness of leveraging synthetic corruption to measure robustness is that we can always improve the model's performance against a specific corruption by using that corruption for data augmentation during training the model. Therefore, measuring robustness by exploiting the artificial corruptions can not give us an accurate estimation of the model's robustness for practical applications.

In figure 3.1, we can observe the impact of increasing the corruption severity for a sample with synthetic snow corruption. At a higher level of severity, the model has missed some of the objects. For instance, in the highest severity, the model could not detect any object. Although in severity with level four, the model detected a car, it could not estimate the heading angle correctly, causing a severe problem in the practical implementation of the model in a self-driving car. More qualitative detection results for other corruptions are provided in the appendix A.

To evaluate the model's robustness in case of real-world corruptions such as adverse weather or lighting, we have leveraged nuScenes-Rainy, nuScenes-Night, Waymo-Rainy, and Waymo-Night datasets. According to the results, we can observe that the model's performance in diverse weather and lighting conditions decreased significantly. For example, as we can see in table 3.5, the 3D average precision (AP_{3D}) of the model on the nuScenes-Clear test set for the Car class with moderate difficulty is 14.85% while in the rainy weather it decreased to 2.67%, and at night, it is 13.35%.

As we can see in the tables 3.6, 3.7, and 3.8, the Frustum PointNets model that is trained on Waymo-Clear dataset could not detect any instance from the Cyclist category in Waymo-Night. Furthermore, we can observe in tables 3.3, 3.4, and 3.5 that the 2D, BEV, and 3D average precisions of the Cyclist class are heavily degraded for nuScenes-Rainy and nuScenes-Night. The reason could be the lack of a sufficient number of cyclist instances in the rainy weather and night time. Moreover, the pedestrian and cyclist are smaller objects than cars, and diverse weather and lighting conditions impact the detection of these objects severely.

As we mentioned in the preparing data section, we have used only the depth of objects in the 3D space to determine their level of detection difficulty for nuScenes and Waymo. However, we observed that this approach is not always accurate without considering occlusion and truncation of instances. For example, in table 3.8, the AP_{3D} of the Pedestrian class with moderate difficulty for the Waymo-Clean dataset is 52.92% while it is 45.81% for the easy difficulty. In other words, we may have a heavily occluded and truncated nearby object in the 3D scene, which is a problematic instance for detecting by the model.

The multimodal model that we have exploited for examining our approach has a cascade architecture, which means that each stage's output feeds to the next stage. More precisely, if the first stage of Frustum PointNets, the 2D object detector, could not detect an instance in the 2D image, the model would not detect the 3D bounding box for that instance. For example, in the figures 3.6 and 3.8 that belong to nuScenes-Night and Waymo-Night respectively, some objects are not detected due to inability of the 2D detector to detect those objects in 2D images.

The need to estimate the third dimension in 3D space can cause a drop in results when we compare the 3D detection with 2D detection results. Furthermore, by comparing the 2D and 3D results, the 3D detection results for Waymo and nuScenes decreased more than the KITTI dataset due to a higher probability of consisting of heavily occluded objects. For example, the average number of car instances per image is 8.1 for Waymo-Clean, while it is 3.84 for the KITTI dataset. Therefore, it is likely that we have more nearby objects in Waymo-Clean than KITTI. When two extracted frustums are close, it is hard for the Instance Segmentation module of Frustum PointNets to segment them precisely because the model supposes that there is one object in each frustum. Moreover, in the nuScenes dataset, the point clouds are more sparse than the point clouds of KITTI and Waymo and it impacts 3D detection results. The nuScenes dataset exploited the 32-beam LiDAR for acquiring the point clouds, while KITTI and Waymo leveraged 64-beam LiDAR.

On the other hand, the model's performance dropped significantly in rainy weather due to the impact of rain on the performance of the LiDAR [37]. For instance, in the figures 3.5 and 3.7 that illustrate the detection results for samples from nuScenes-Rainy and Waymo-Rainy datasets, we can observe that the model detected some objects. However, it could not estimate their 3D bounding boxes accurately.

To show the effectiveness of our proposed method in practical applications, we have conducted an example experiment to demonstrate how evaluating the robustness with proposed datasets containing real images captured on rainy days and at night can help choose the right direction during training.

In this experiment, we exploited augmentation to enrich our training data for the Waymo-Clear dataset. To this end, we have applied all corruptions with severity two that introduced for making KITTI-C on 1000 random images from the test set of the Waymo-Clean dataset. We trained the model that we leveraged in previous experiments on this sub-dataset with the same parameters. Next, we evaluated the trained model on Waymo-Rainy, Waymo-Night, and Waymo-Clean test set while excluding the images we exploited for training. The results are provided in tables 3.9, 3.10, and 3.11 for AP_{2D}, AP_{BEV}, and AP_{3D} respectively.

In this example experiment, we observed that by exploiting this set of corruptions for augmenting the dataset and fine-tuning the model, the evaluation results on the Waymo-Night for the Car class increased remarkably. In contrast, the results for Waymo-Rainy decreased slightly while the results for the Waymo-Clean test set decreased significantly. According to these results, we can conclude that the exploited augmentation approach in this experiment improved the robustness of the trained model for samples captured at night while decreasing the robustness for samples captured on rainy days. Therefore, according to this conclusion, we can choose an appropriate approach for training the model for various practical situations.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	88.51	88.45	81.09	59.29	59.97	50.67	25.45	24.49	16.78
Pedestrian	71.66	72.01	71.45	50.19	50.28	42.26	17.07	9.09	9.09
Cyclist	67.76	68.07	69.71	12.09	12.09	12.09	-	-	-

Table 3.9 – The evaluation results (AP_{2D} in %) of the Frustum PointNets after leveraging augmentation, on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for 2D detection.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	54.79	58.11	52.96	38.83	41.11	34.27	24.54	24.03	16.40
Pedestrian	48.05	54.93	53.98	25.46	26.10	21.14	15.85	9.09	9.09
Cyclist	41.17	41.06	46.40	12.09	12.09	12.09	-	-	-

Table 3.10 – The evaluation results (AP_{BEV} in %) of the Frustum PointNets after leveraging augmentation, on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for BEV detection.

Class	**Clear**			**Rainy**			**Night**		
	easy	**moderate**	**hard**	**easy**	**moderate**	**hard**	**easy**	**moderate**	**hard**
Car	31.69	38.20	30.50	22.42	24.47	16.26	20.57	15.22	13.76
Pedestrian	41.10	49.72	44.82	18.36	18.78	17.46	9.09	9.09	9.09
Cyclist	38.58	37.95	43.17	12.09	12.09	12.09	-	-	-

Table 3.11 – The evaluation results (AP_{3D} in %) of the Frustum PointNets after leveraging augmentation, on Waymo-Rainy, Waymo-Night, and test set of Waymo-Clear for 3D detection.

In this work, we have introduced robustness test datasets for assessing the model's robustness for rainy days and at night. The proposed method's future development needs to consider more diverse weathers that a self-driving car may experience in practice, such as snowy or foggy weather. Moreover, we require evaluating the robustness of a model for autonomous vehicle applications in more challenging conditions, including rainy or foggy nights. Furthermore, the robustness test dataset needs to involve more diverse geographical locations to become a reliable method to assess the robustness.

In the multimodal models' designing process, it is necessary to pay attention to the model's architecture to prevent the model's inability to provide results in case of loss of a modality. To this end, the model requires to extract features from different modalities in parallel and fuse them to obtain the final results. Moreover, the robustness test dataset can be expanded to include sub-datasets to evaluate the robustness against abnormally in point clouds or misaligned sensors from different modalities.

We leveraged the proposed evaluation metric by the KITTI benchmark. This evaluation method is appropriate for assessing the model's performance in mild weather and lighting conditions such as sunny days due to the need for high IoU thresholds for considering a detected object as true positive, such as the 0.7 IoU threshold for Car class. However, to evaluate the robustness, we can modify the thresholds or propose new evaluation metrics to cover robustness evaluation requirements.

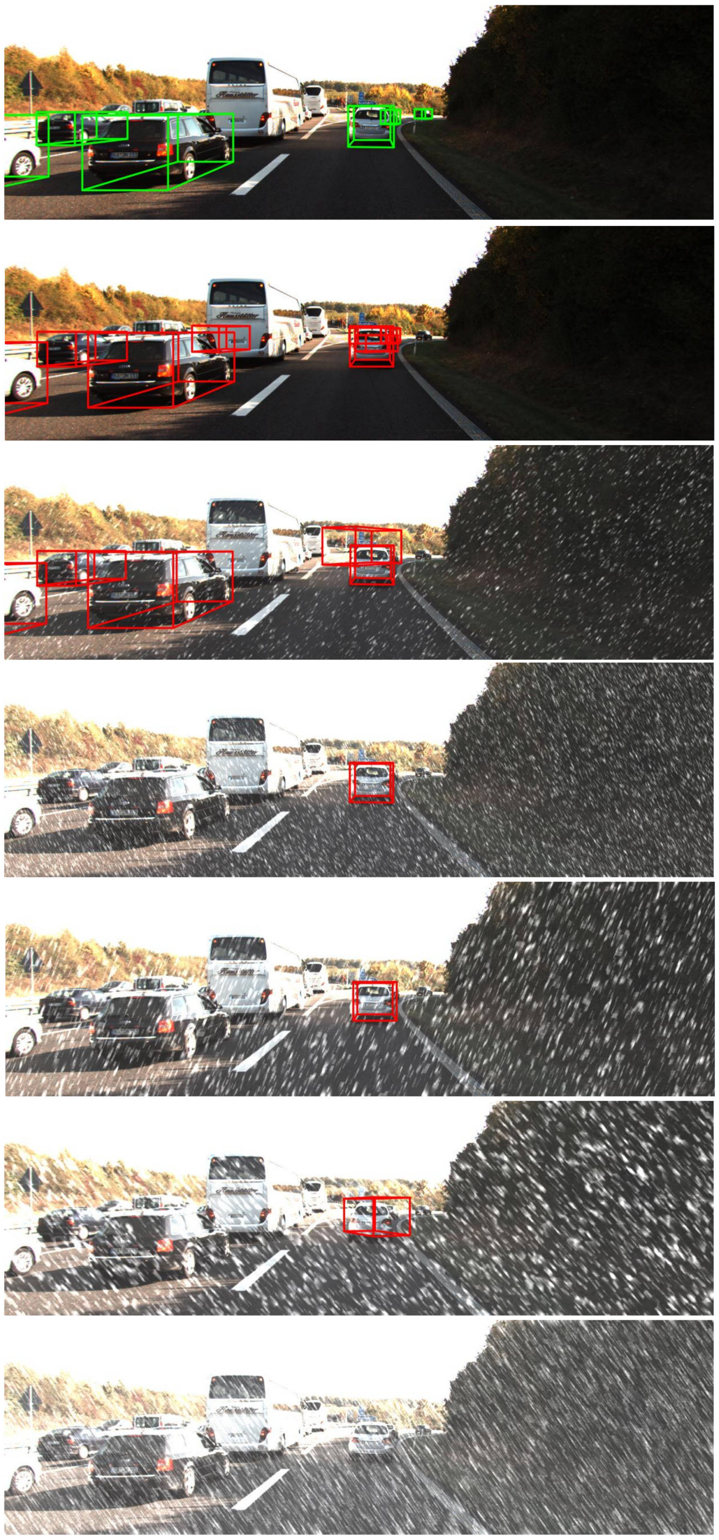

FIGURE 3.1 – Ground truth and detection results for a sample from the KITTI-C dataset for snow corruption. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5. In this example, the model could not detect any object in the corrupted scene with maximum severity.

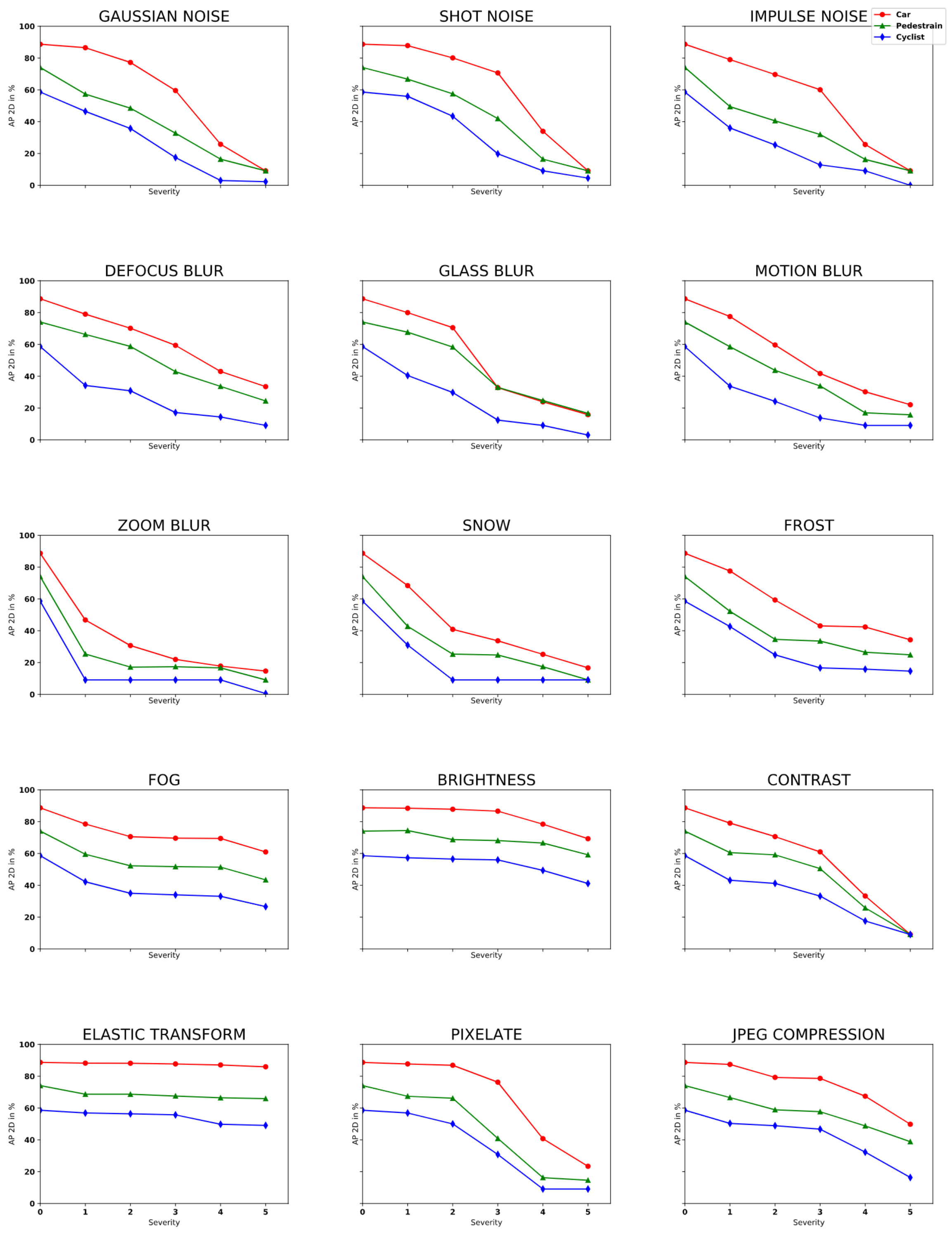

FIGURE 3.2 – The KITTI-C evaluation results of Frustum PointNets for Car, Pedestrian, and Cyclist classes with moderate difficulty for all 15 corruptions with various severities for 2D detection (AP_{2D} in %).

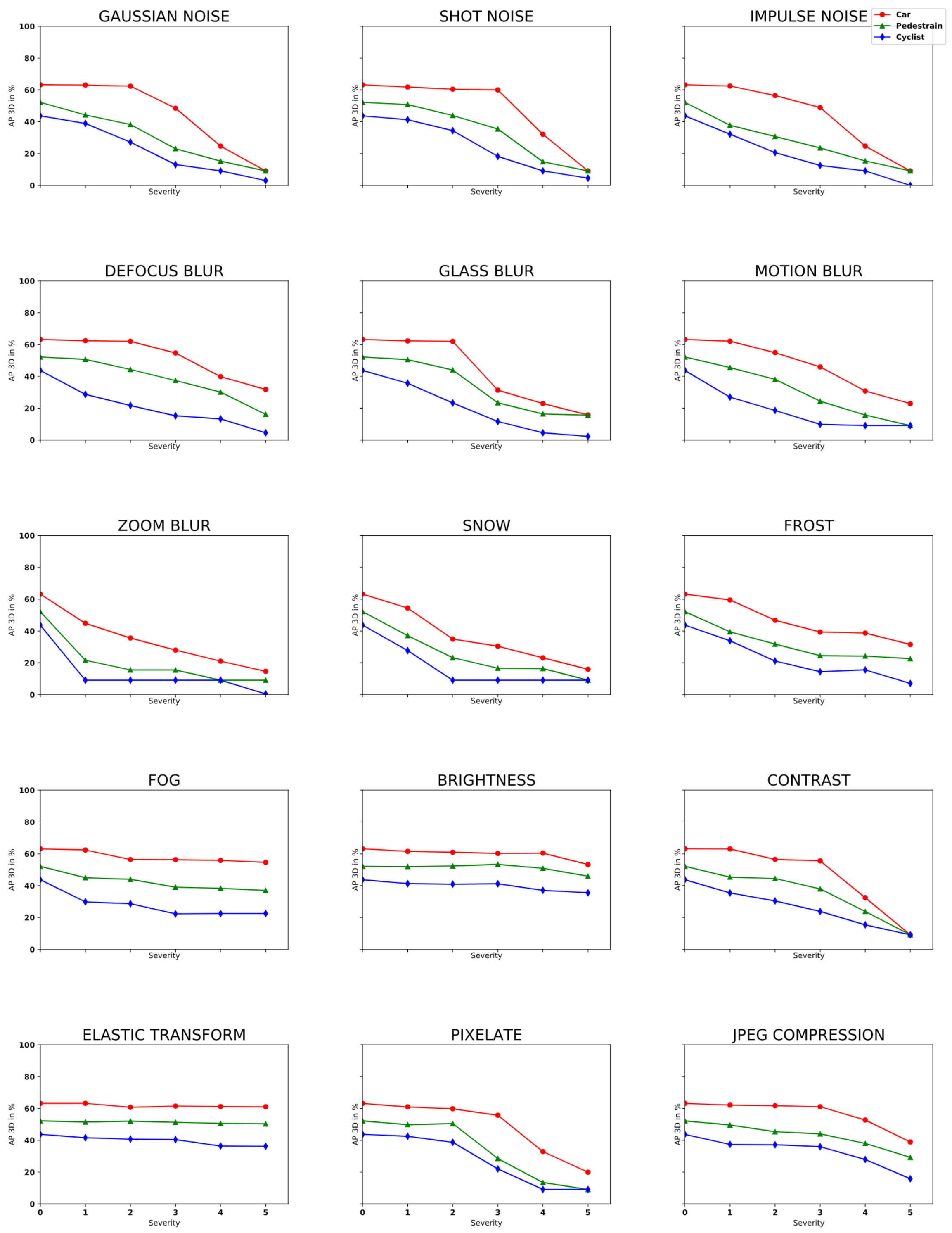

FIGURE 3.3 – The KITTI-C evaluation results of Frustum PointNets for Car, Pedestrian, and Cyclist classes with moderate difficulty for all 15 corruptions with various severities for 3D detection (AP_{3D} in %).

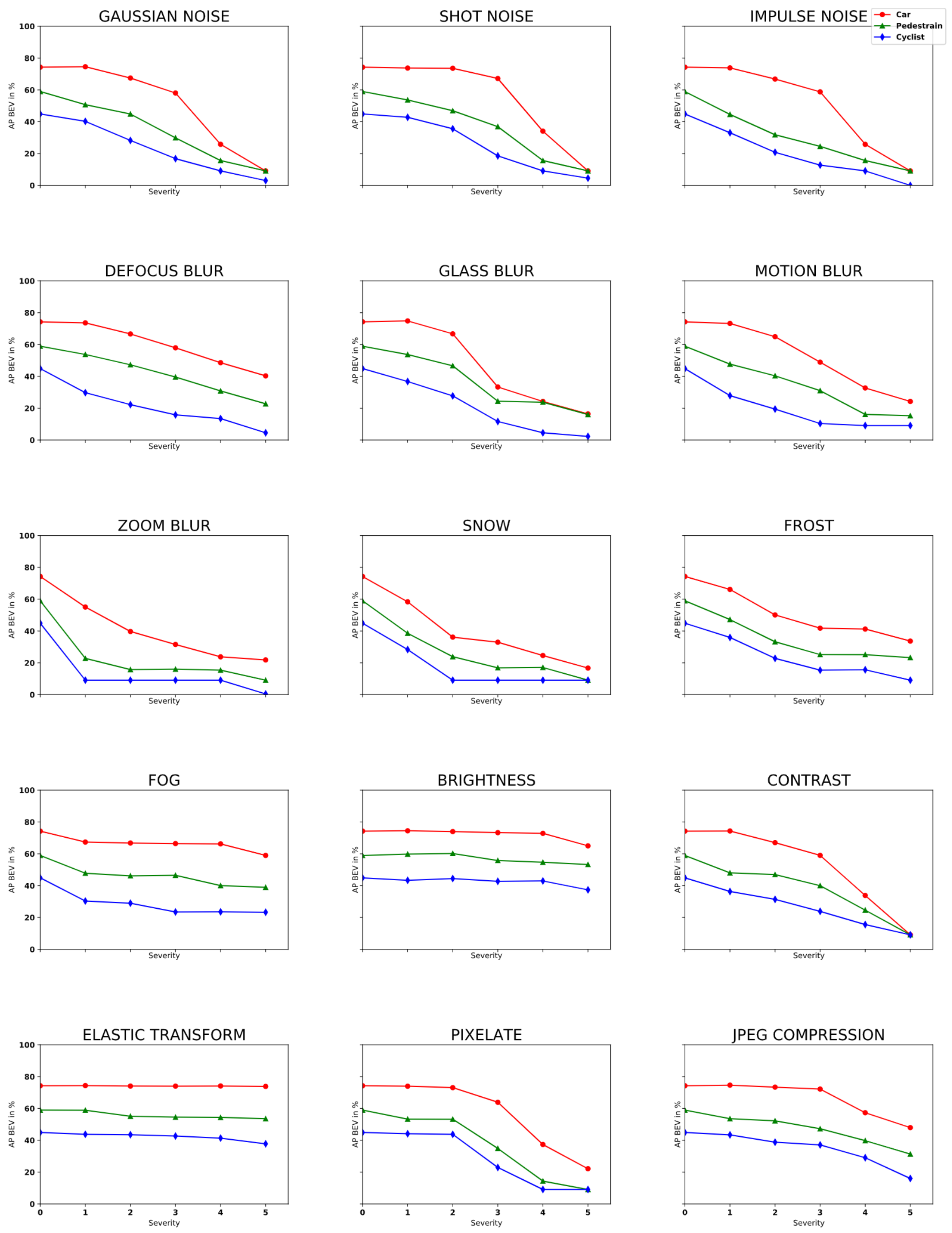

FIGURE 3.4 – The KITTI-C evaluation results of Frustum PointNets for Car, Pedestrian, and Cyclist classes with moderate difficulty for all 15 corruptions with various severities for Bird's Eye View (BEV) detection (AP_{BEV} in %).

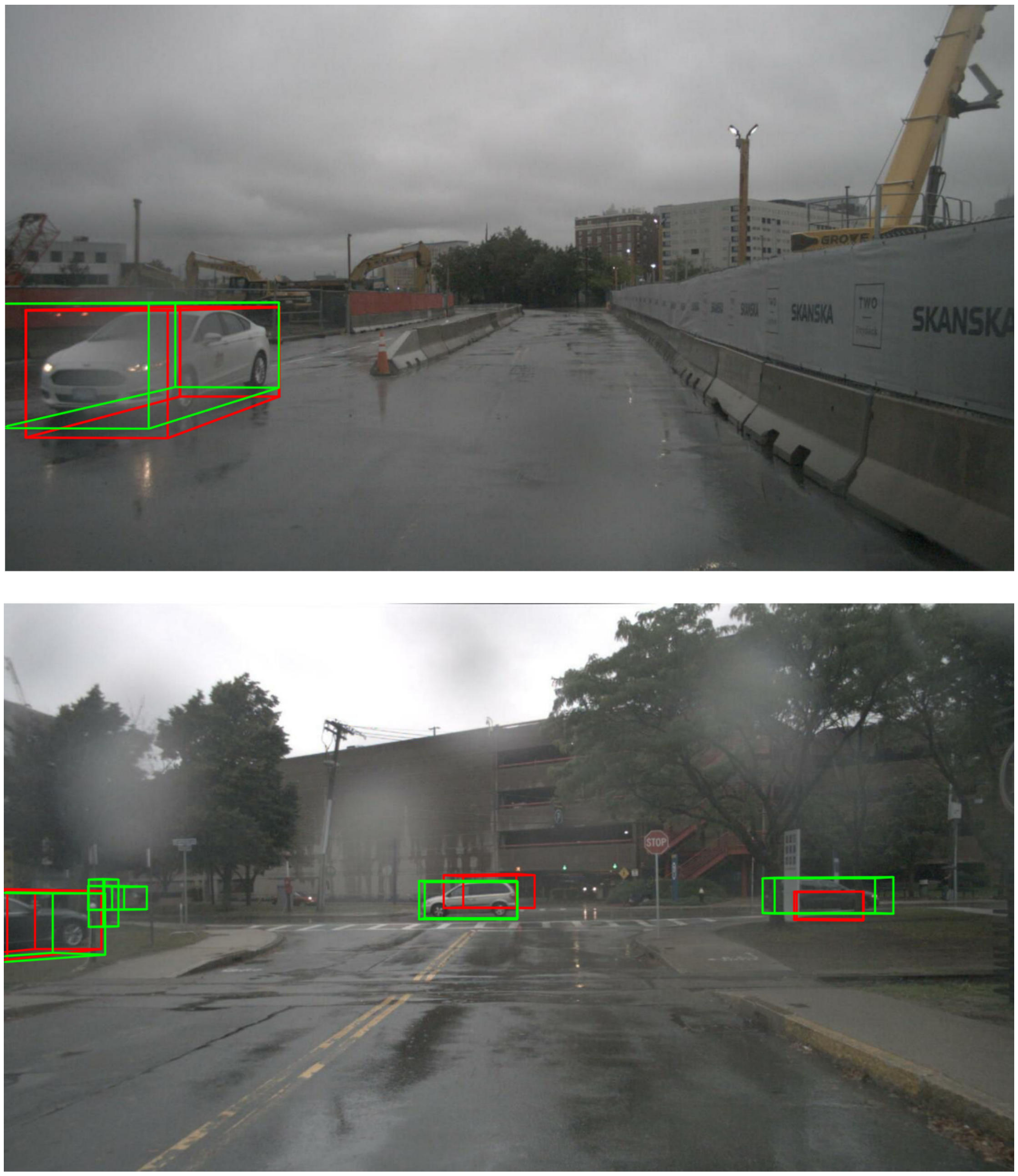

FIGURE 3.5 – Ground truth and detection results for a sample from nuScenes-Rainy. Green bounding boxes represent the ground truth, and the red bounding boxes represent the detection results.

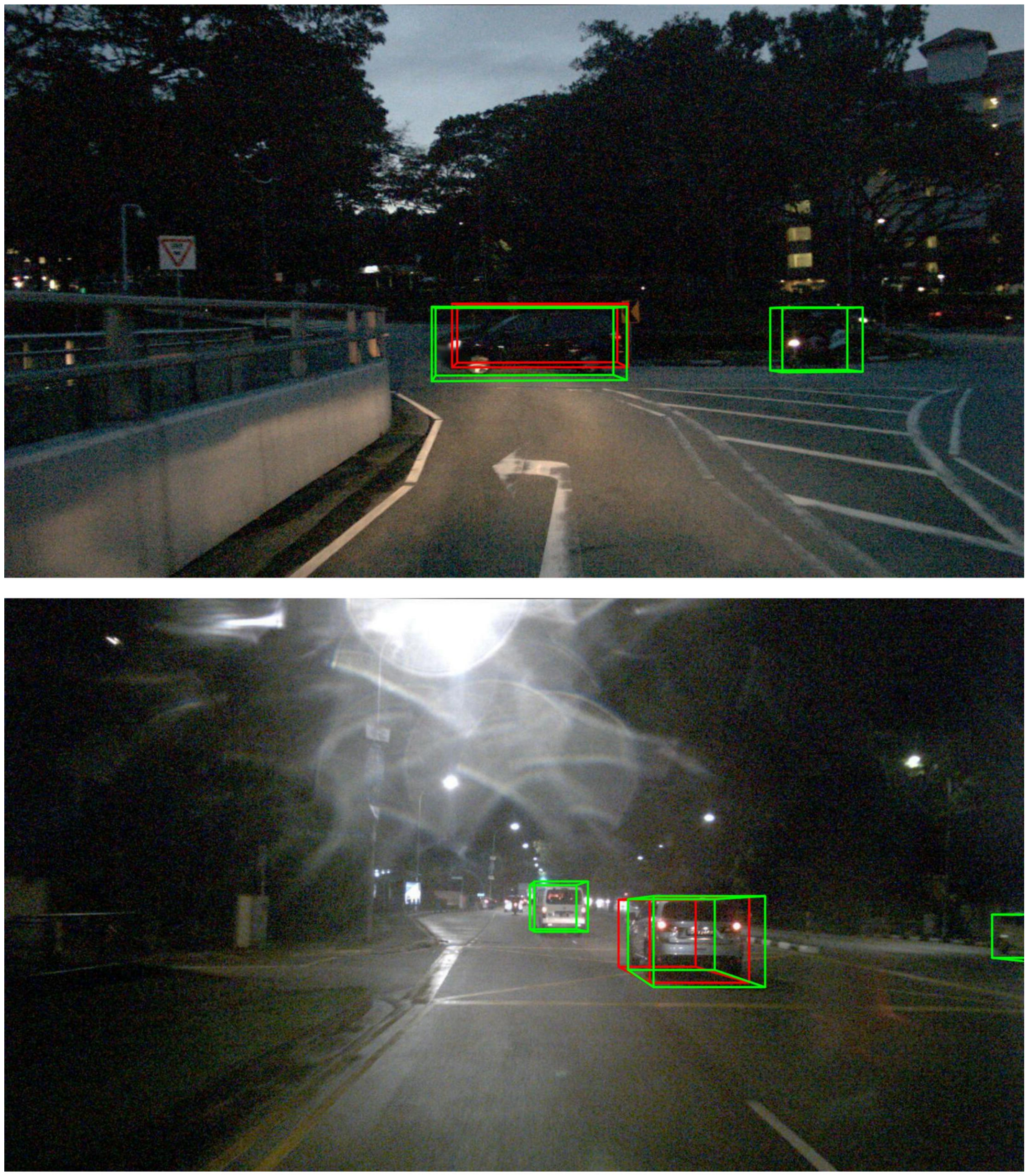

FIGURE 3.6 – Ground truth and detection results for a sample from nuScenes-Night. Green bounding boxes represent the ground truth, and the red bounding boxes represent the detection results.

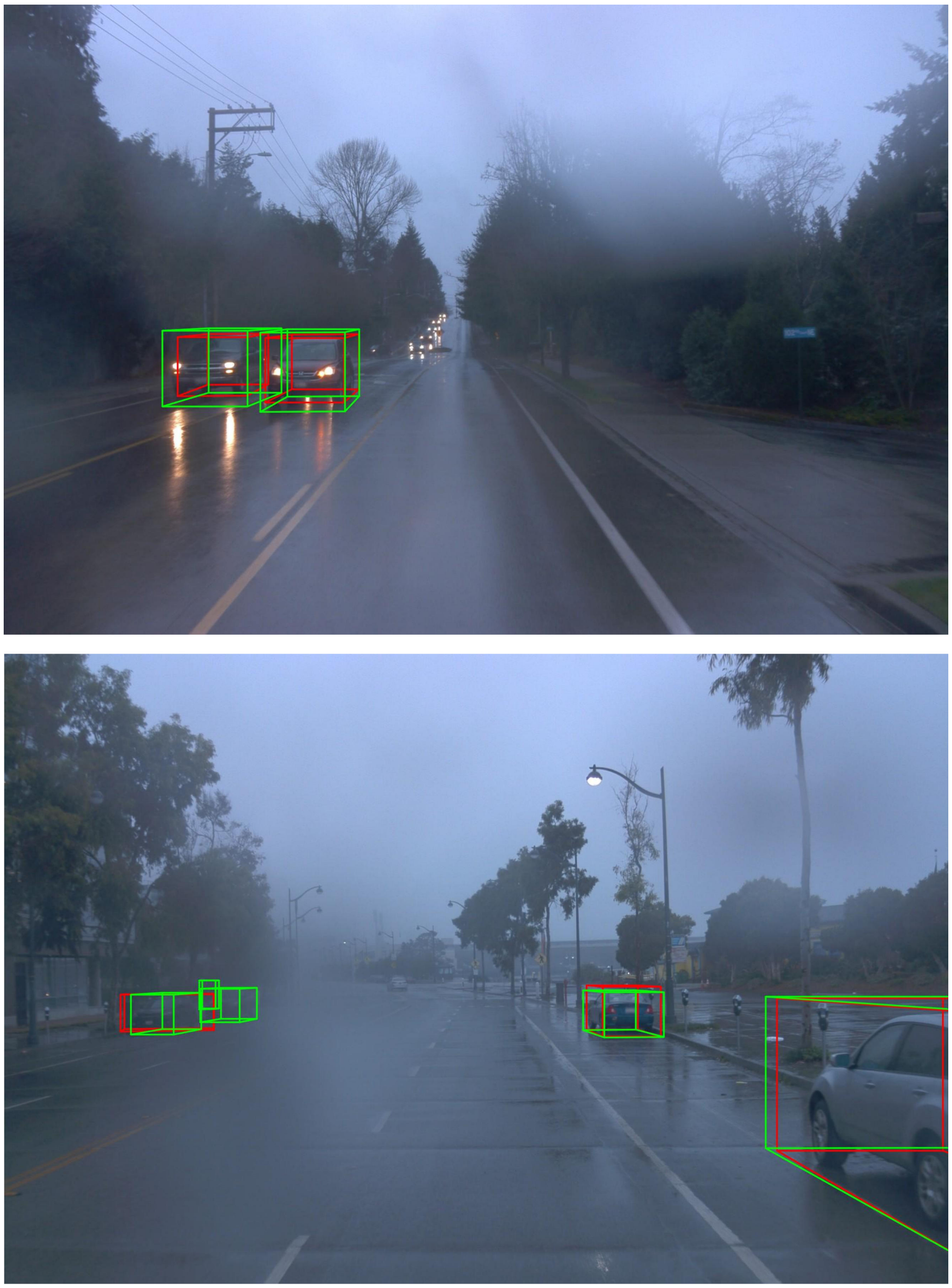

Figure 3.7 – Ground truth and detection results for samples from Waymo-Rainy. Green bounding boxes represent the ground truth, and the red bounding boxes represent the detection results.

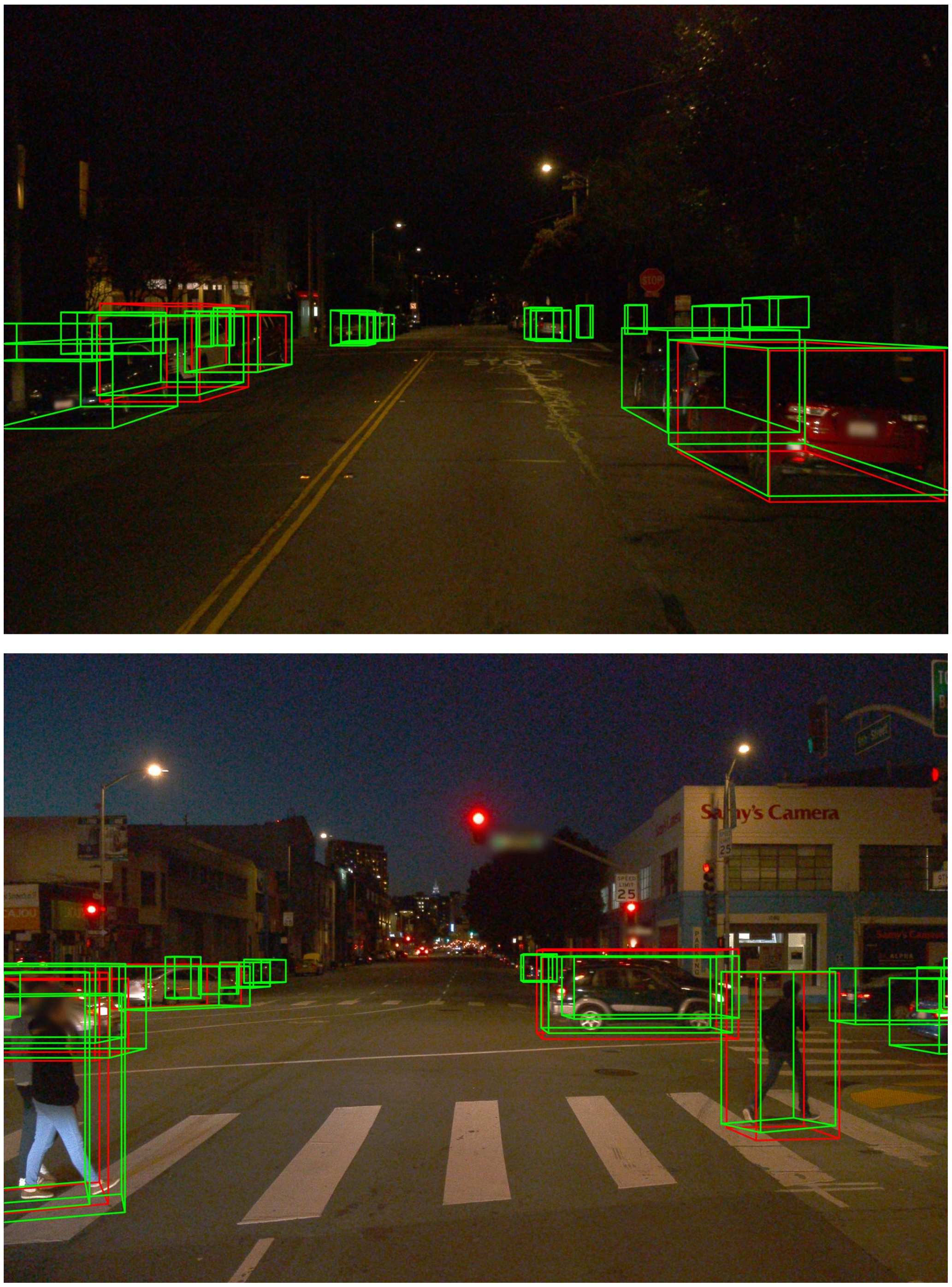

FIGURE 3.8 – Ground truth and detection results for samples from Waymo-Night. Green bounding boxes represent the ground truth, and the red bounding boxes represent the detection results.

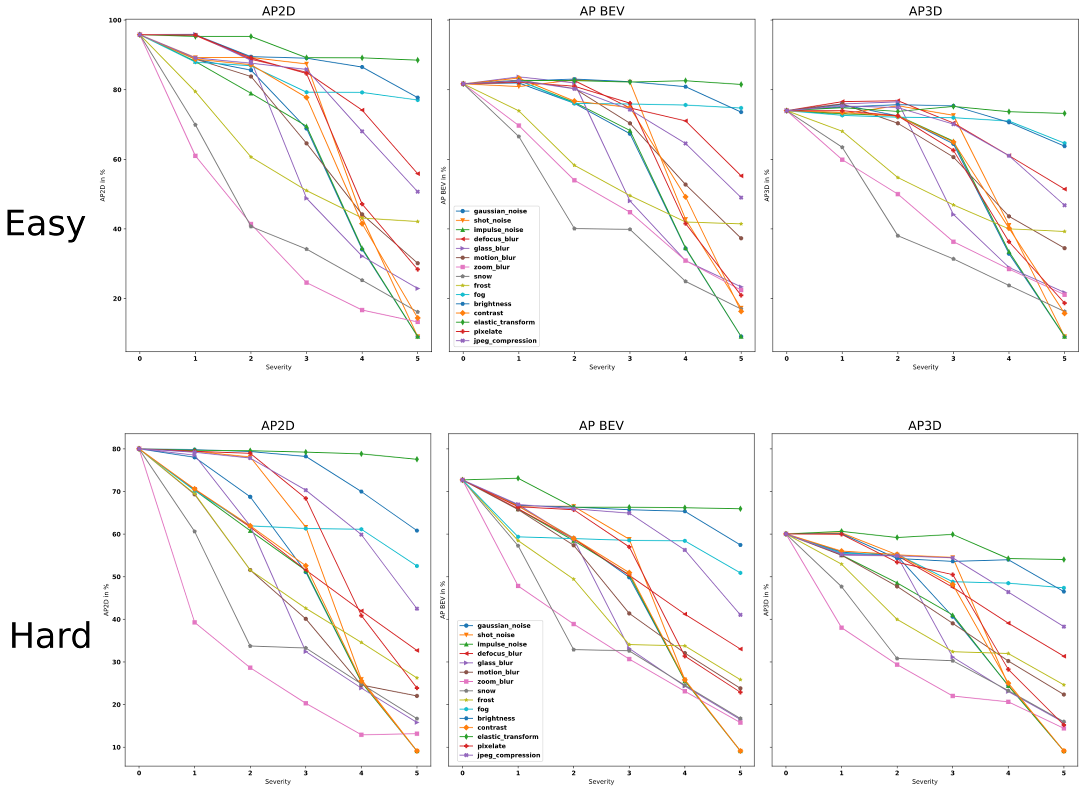

FIGURE 3.9 – Illustration of evaluation results (AP_{2D}, AP_{BEV}, AP_{3D} in %) on KITTI-C for class Car with Easy and Hard difficulties for all 15 types of corruptions with five severity levels. Zero severity belongs to evaluation results on the clean KITTI test set.

Conclusion

In this research, to assess the robustness of a multimodal 3D object detector in the autonomous vehicle scenario, we proposed a new method by introducing two different types of robustness test datasets including artificially corrupted images and real challenging scenarios as scenes captured at night or in rainy weather.

We conducted extensive experiences to investigate our method's effectiveness on a trained model to evaluate its robustness in diverse conditions. Moreover, we evaluated the robustness of a 3D multimodal object detector against synthetic corruptions by leveraging the mean performance under corruption and relative performance under corruption on our KITTI-C dataset and provided the results for demonstrating the impacts of individual corruption with different severity levels.

We observed that detecting objects of interest in the adverse weather and lighting conditions appears to be a challenging task, resulting in drastic drops in accuracy levels. We can conclude that when the trained multimodal 3D object detector is tested on the datasets involve rainy or night scenes, in order to evaluate the robustness, it failed dramatically.

The proposed robustness test datasets in this thesis are created for evaluating the robustness of a model in specific conditions including rainy weather and night conditions that are important and problematic for self-driving car applications. However, we can generalize the method for other different conditions, such as snowy weather, by leveraging the data that was captured during that specific condition.

The main principle for this application-specific robustness test method is collecting problematic real-world data with the same distribution while the collected test dataset has sufficient diversity. This robustness evaluation method can be useful for guiding design choices by estimating the robustness of different models for a specific application.

Overall, considerable progress is required before self-driving cars can operate reliably in mixed urban traffic, heavy rain and snow, in different daytime and lighting conditions, and assessing the robustness of a 3D perception system in these diverse conditions is a necessity.

Annexe A

Qualitative Results For Different Corruptions

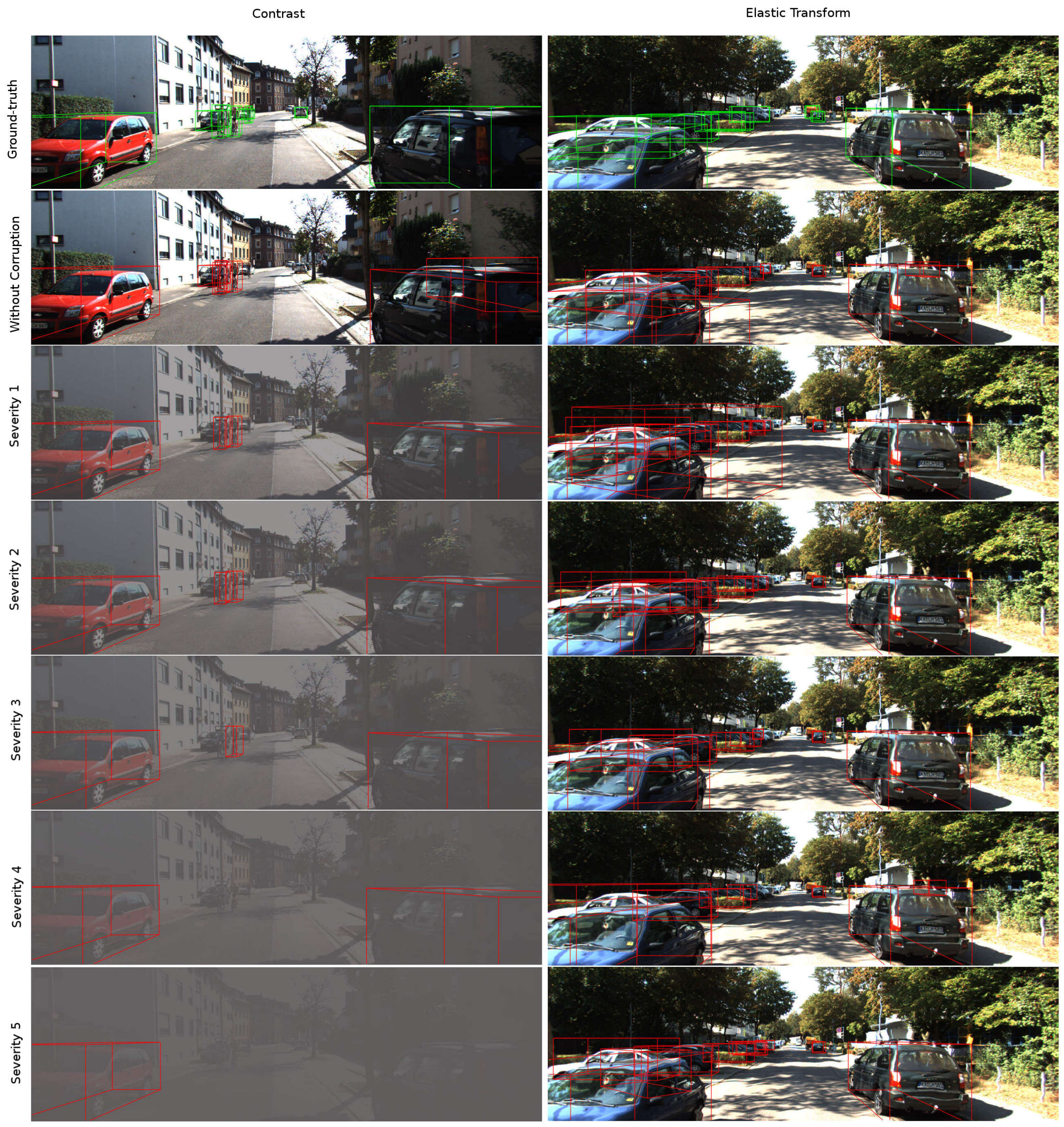

FIGURE A.1 – Ground truth and detection results for two samples from the KITTI-C dataset for contrast and elastic transform corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

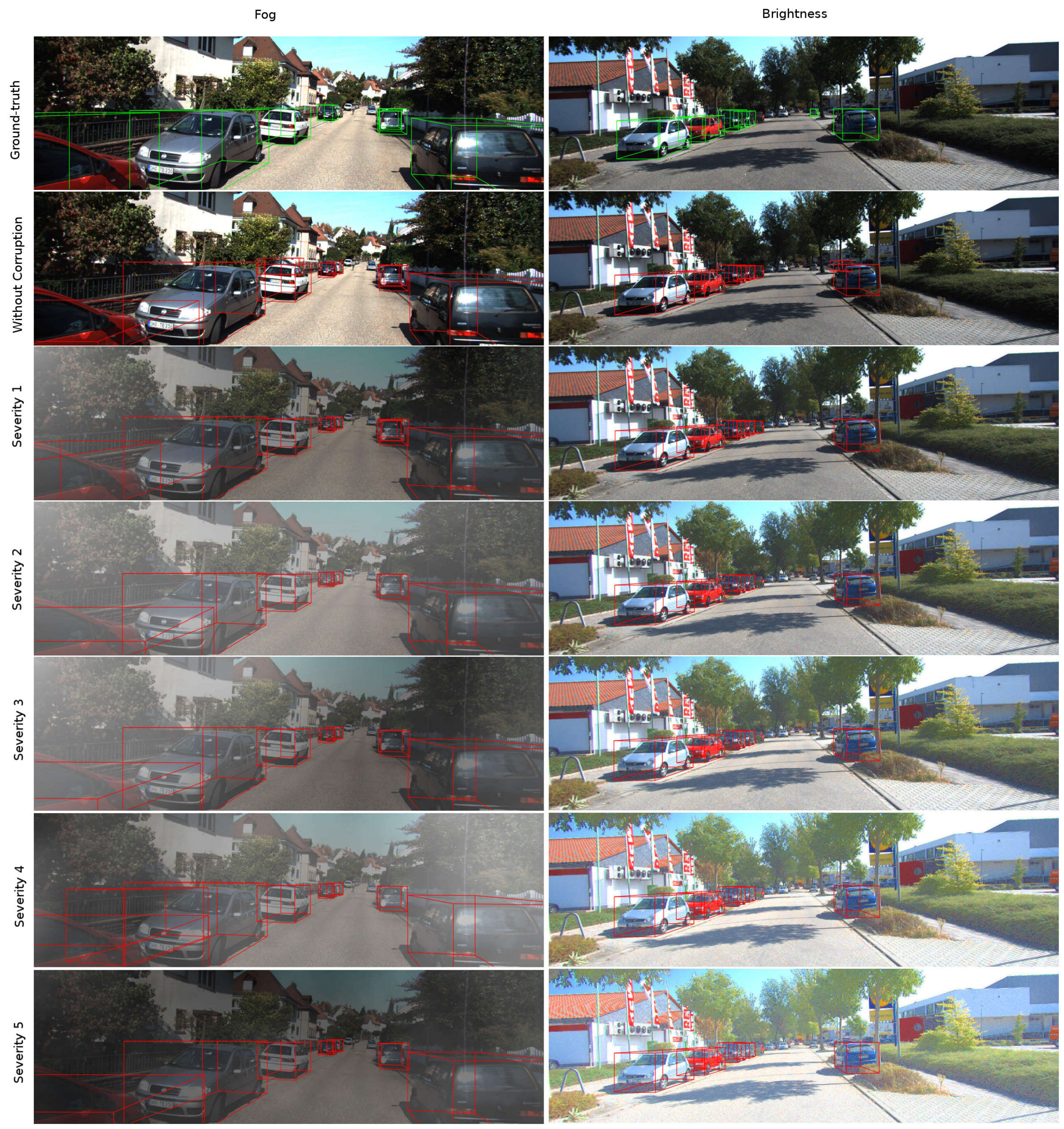

FIGURE A.2 – Ground truth and detection results for two samples from the KITTI-C dataset for fog and brightness corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

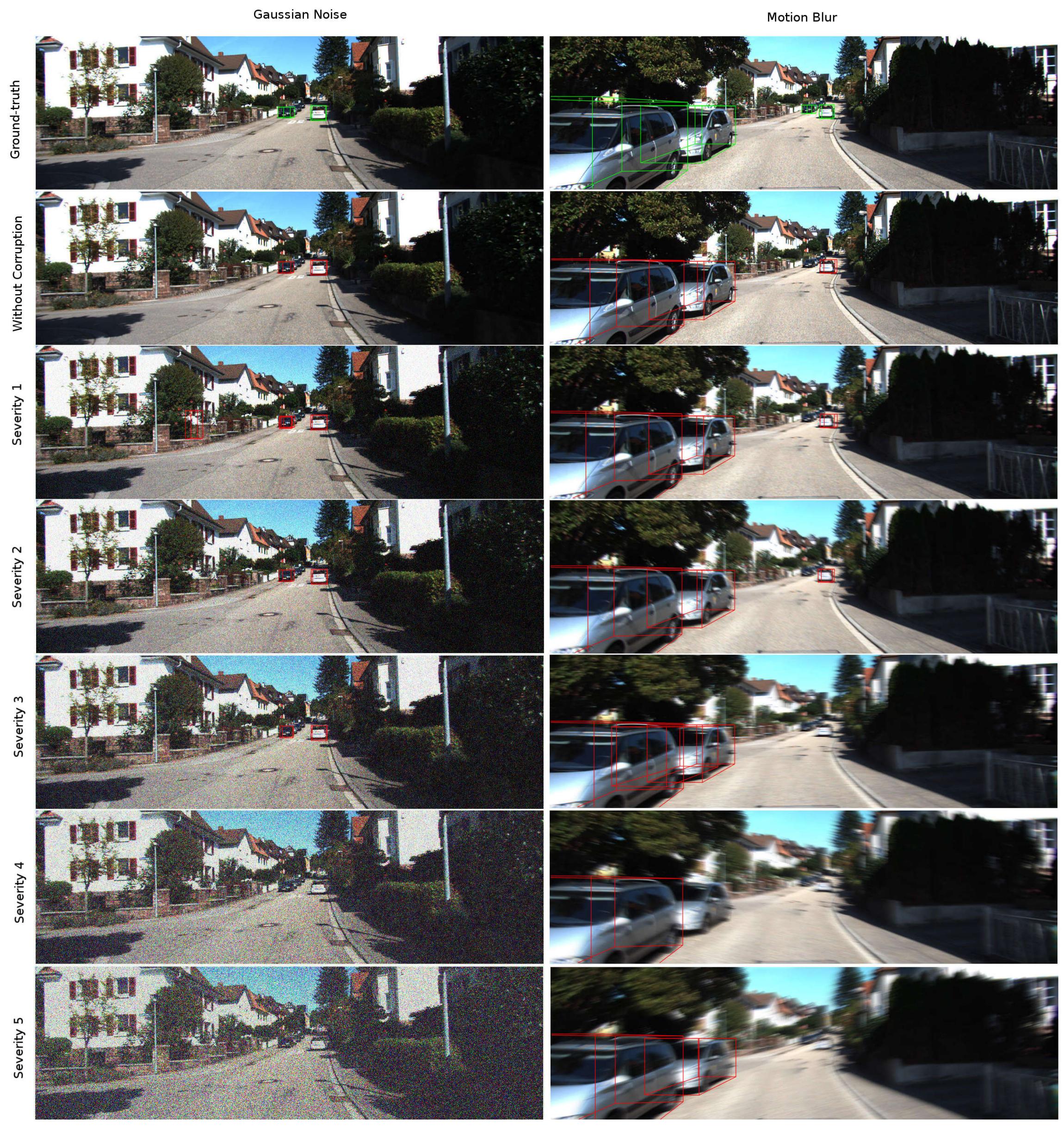

FIGURE A.3 – Ground truth and detection results for two samples from the KITTI-C dataset for Gaussian noise and motion blur corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

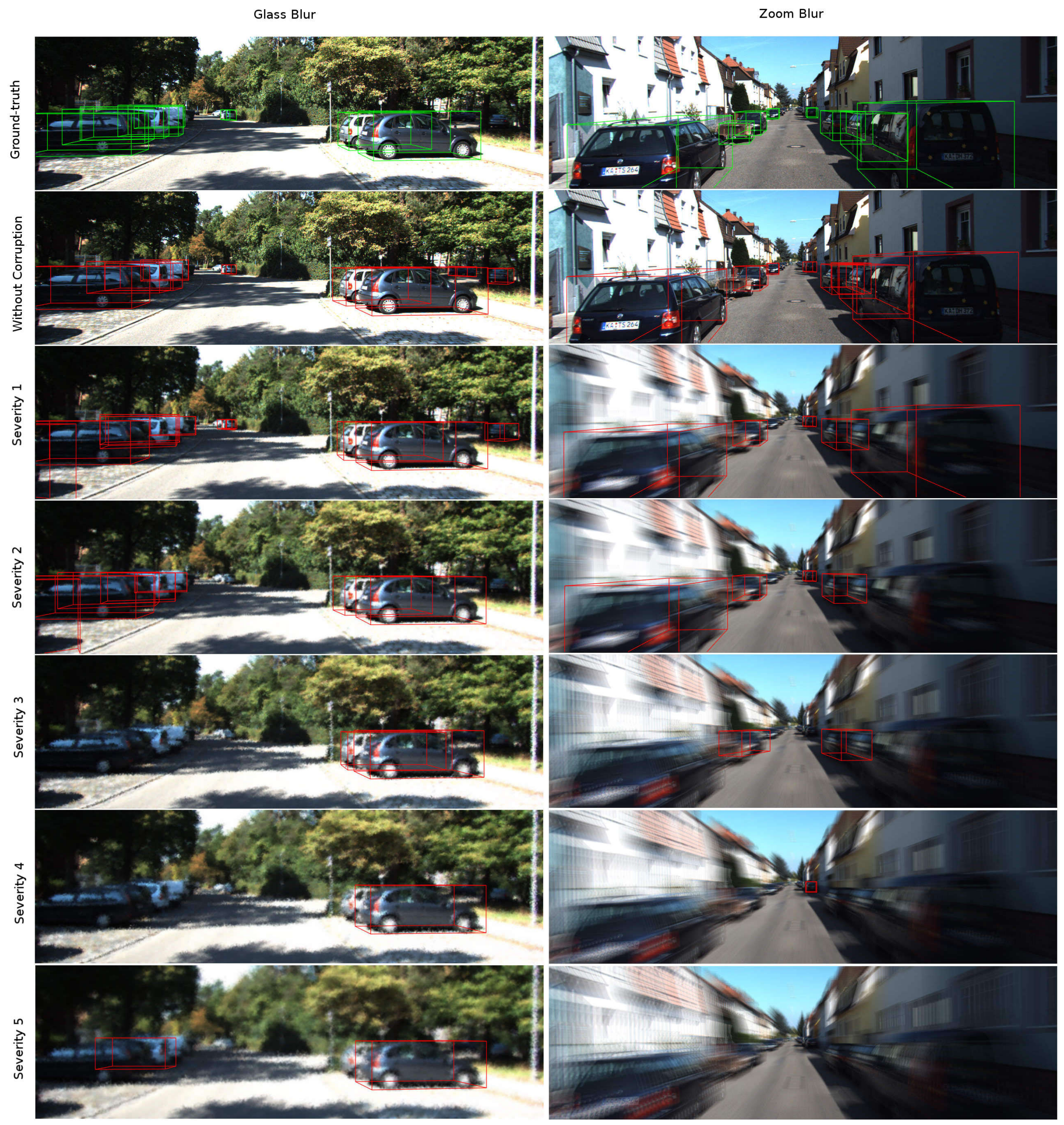

FIGURE A.4 – Ground truth and detection results for two samples from the KITTI-C dataset for glass blur and zoom blur corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

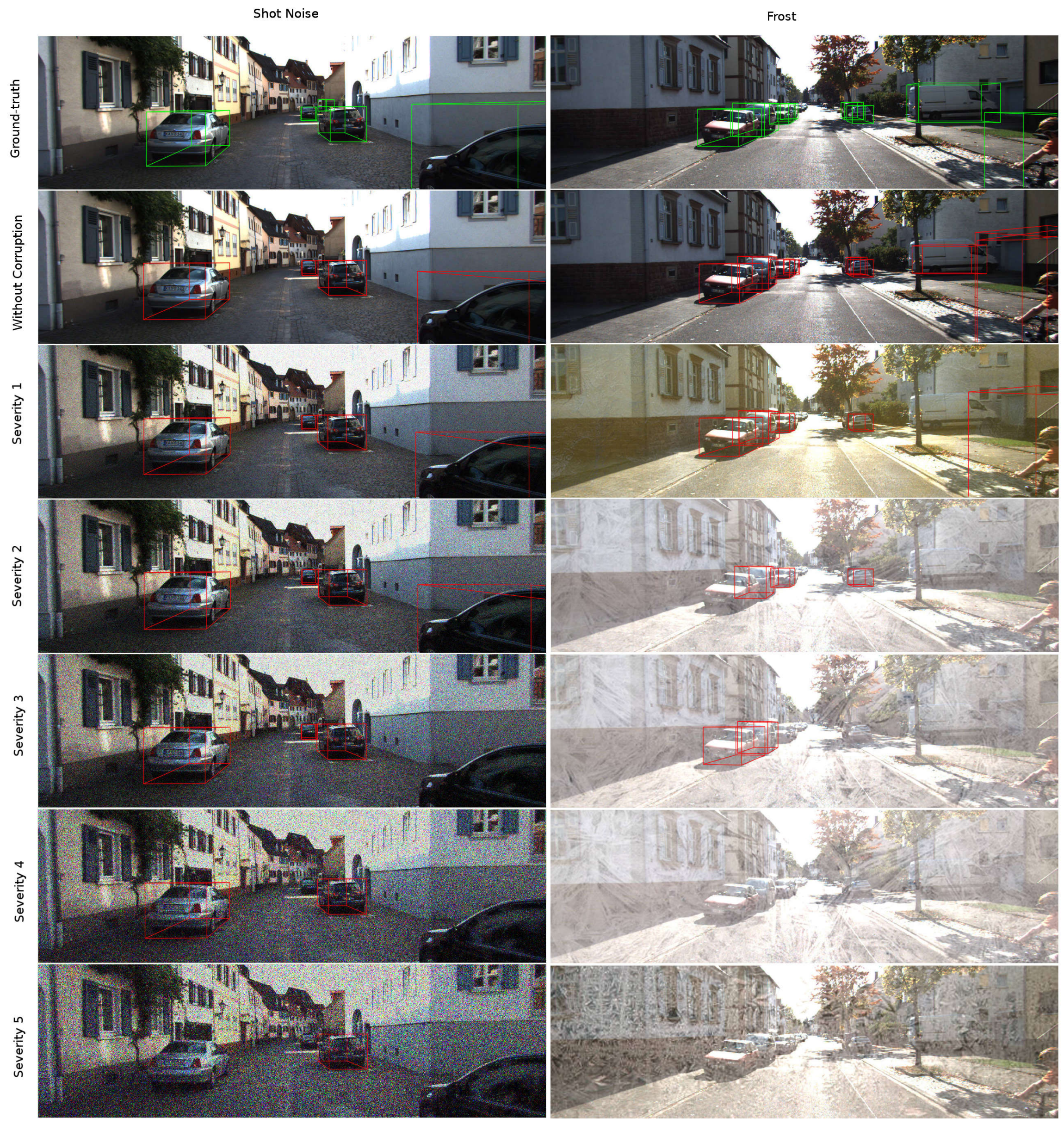

FIGURE A.5 – Ground truth and detection results for two samples from the KITTI-C dataset for shot noise and frost corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

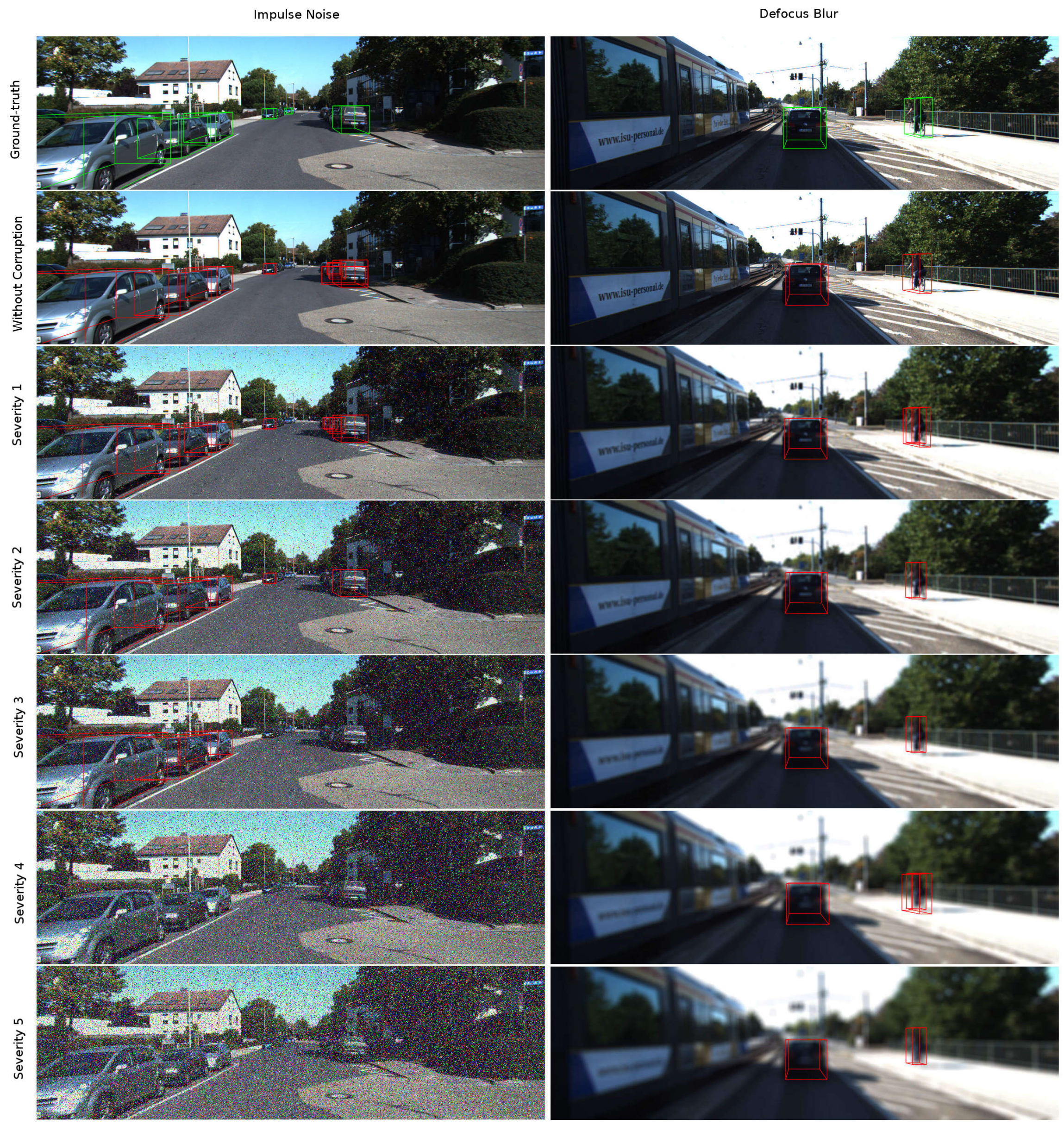

FIGURE A.6 – Ground truth and detection results for two samples from the KITTI-C dataset for impulse noise and defocus blur corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

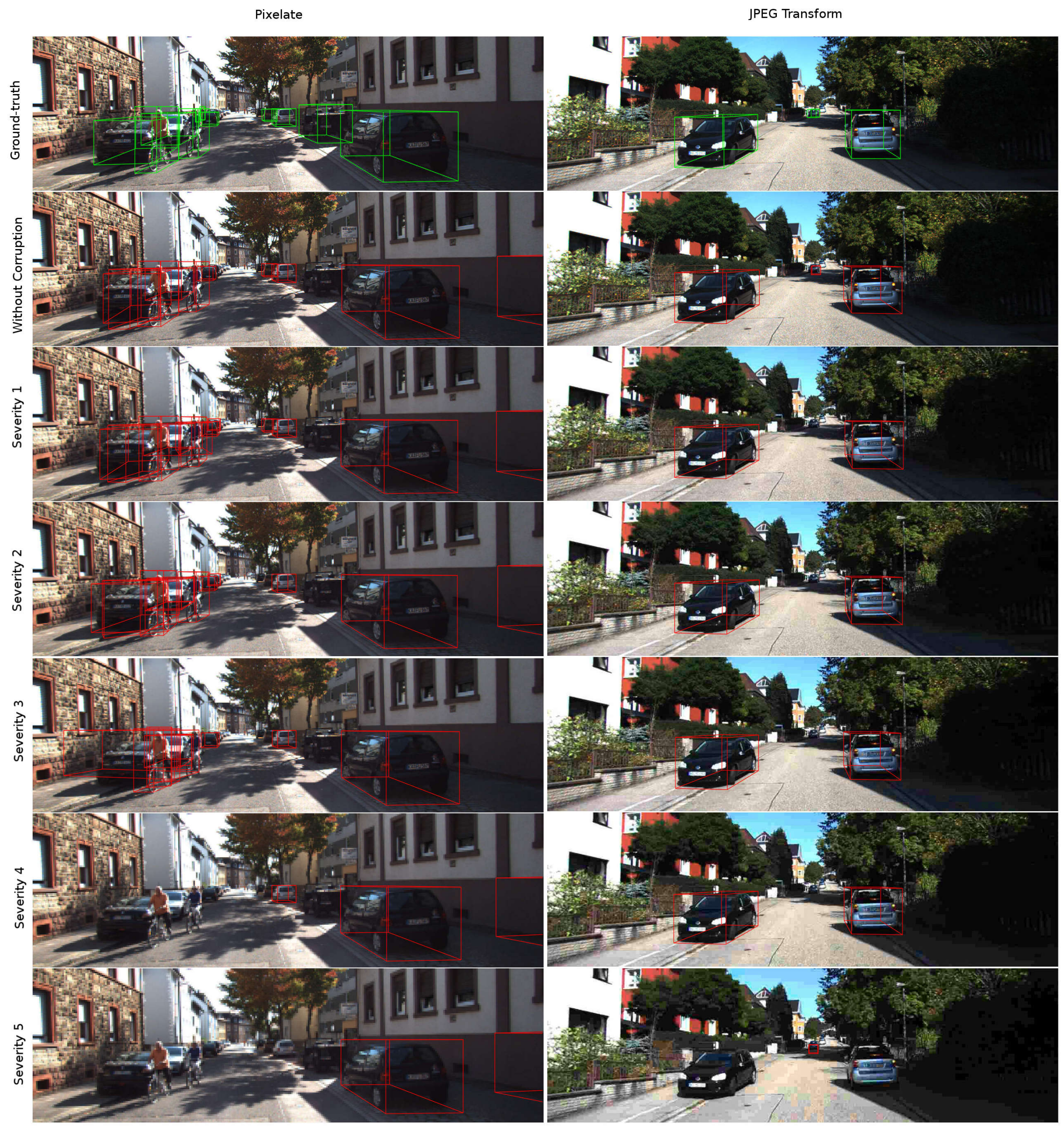

FIGURE A.7 – Ground truth and detection results for two samples from the KITTI-C dataset for pixelate and JPEG transform corruptions. From top to bottom : ground truth, without corruption, and corruption with severity 1 to 5.

Bibliographie

[1] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving : Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[2] Todd Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[3] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. SHREC'11 track : shape retrieval on non-rigid 3d watertight meshes. In *Proceedings of the 4th Eurographics conference on 3D Object Retrieval*, EG 3DOR'11, pages 79–88. Eurographics Association, 2011.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving ? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[5] 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. *arXiv preprint arXiv :1903.11027*, 2019.

[6] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[7] Eduardo Arnold, Omar Y Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10) :3782–3795, 2019.

[8] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving : Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[9] Linfeng Zhang, Muzhou Yu, Tong Chen, Zuoqiang Shi, Chenglong Bao, and Kaisheng Ma. Auxiliary training : Towards accurate and robust models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 372–381, 2020.

[10] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection : Autonomous driving when winter is coming. *arXiv preprint arXiv :1907.07484*, 2019.

[11] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7 :128837–128868, 2019.

[12] Bradley Rettler and Andrew M. Bailey. Object. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2017 edition, 2017.

[13] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection : A survey. *International journal of computer vision*, 128(2) :261–318, 2020.

[14] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu. Advanced deep-learning techniques for salient and category-specific object detection : A survey. *IEEE Signal Processing Magazine*, 35(1) :84–100, 2018.

[15] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2) :154–171, 2013.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9) :1904–1916, 2015.

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[21] Joseph Redmon and Ali Farhadi. Yolo9000 : Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[22] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*, 2018.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[24] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd : Deconvolutional single shot detector. *arXiv preprint arXiv :1701.06659*, 2017.

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[27] Prabhu Ramachandran and Gaël Varoquaux. Mayavi : 3d visualization of scientific data. *Computing in Science & Engineering*, 13(2) :40–51, 2011.

[28] Yin Zhou and Oncel Tuzel. Voxelnet : End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[29] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor : Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[30] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. Roarnet : A robust 3d object detection based on region approximation refinement. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2510–2515. IEEE, 2019.

[31] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 1–8. IEEE, 2018.

[32] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion : Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.

[33] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, and Dongpu Cao. Deep learning for image and point cloud fusion in autonomous driving : A review. *arXiv preprint arXiv :2004.05224*, 2020.

[34] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning : A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2) :423–443, 2018.

[35] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving : A survey. *IEEE Access*, 8 :2847–2868, 2019.

[36] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving : Common practices and emerging technologies. *IEEE Access*, 8 :58443–58469, 2020.

[37] Jacek Wojtanowski, Marek Zygmunt, Mirosława Kaszczuk, Z Mierczyk, and Michal Muzal. Comparison of 905 nm and 1550 nm semiconductor laser rangefinders' performance deterioration due to adverse environmental conditions. *Opto-Electronics Review*, 22(3) :183–190, 2014.

[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet : Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ : Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[40] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[41] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv :1901.09960*, 2019.

[42] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1991–1999, 2019.

[43] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[44] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019.

[45] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[46] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std : Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.

[47] Jeet Mohapatra, Pin-Yu Chen, Sijia Liu, Luca Daniel, et al. Towards verifying robustness of neural networks against semantic perturbations. *arXiv preprint arXiv :1912.09533*, 2019.

[48] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[49] Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D'Amour, Dan Moldovan, et al. On robustness and transferability of convolutional neural networks. *arXiv preprint arXiv :2007.08558*, 2020.

[50] Markus Braun, Sebastian Krebs, Fabian B. Flohr, and Dariu M. Gavrila. Eurocity persons : A novel benchmark for person detection in traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[51] Jungwook Lee, Sean Walsh, Ali Harakeh, and Steven L Waslander. Leveraging pre-trained 3d object detection models for fast ground truth generation. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2504–2510. IEEE, 2018.

[52] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, page 0278364920979368, 2020.

[53] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960, 2018.

[54] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics : The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[55] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[56] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2) :303–338, 2010.

[57] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa : Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[59] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco : Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[60] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture ; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv :1811.12231*, 2018.