

3D-VField: Adversarial Augmentation of Point Clouds for Domain Generalization in 3D Object Detection

Alexander Lehner^{*,○,1,2} Stefano Gasperini^{*,1,2} Alvaro Marcos-Ramiro² Michael Schmidt²
 Mohammad-Ali Nikouei Mahani² Nassir Navab^{1,3} Benjamin Busam¹ Federico Tombari^{1,4}

¹ Technical University of Munich ² BMW Group ³ Johns Hopkins University ⁴ Google

Abstract

As 3D object detection on point clouds relies on the geometrical relationships between the points, non-standard object shapes can hinder a method’s detection capability. However, in safety-critical settings, robustness to out-of-domain and long-tail samples is fundamental to circumvent dangerous issues, such as the misdetection of damaged or rare cars. In this work, we substantially improve the generalization of 3D object detectors to out-of-domain data by deforming point clouds during training. We achieve this with 3D-VField: a novel data augmentation method that plausibly deforms objects via vector fields learned in an adversarial fashion. Our approach constrains 3D points to slide along their sensor view rays while neither adding nor removing any of them. The obtained vectors are transferable, sample-independent and preserve shape and occlusions. Despite training only on a standard dataset, such as KITTI, augmenting with our vector fields significantly improves the generalization to differently shaped objects and scenes. Towards this end, we propose and share CrashD: a synthetic dataset of realistic damaged and rare cars, with a variety of crash scenarios. Extensive experiments on KITTI, Waymo, our CrashD and SUN RGB-D show the generalizability of our techniques to out-of-domain data, different models and sensors, namely LiDAR and ToF cameras, for both indoor and outdoor scenes. Our CrashD dataset is available at <https://crashd-cars.github.io>.

1. Introduction

With the established wide-spread progress of learning-based methods tackling a variety of perception tasks (e.g.,

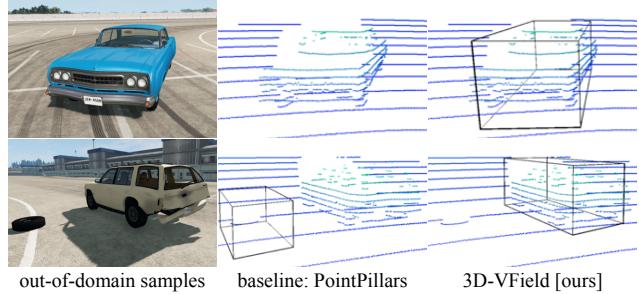


Figure 1. Predictions of PointPillars [18] trained on KITTI [13], without and with our adversarial augmentations on out-of-domain samples from the proposed CrashD dataset. CrashD comprises *rare* (top) and *damaged* (bottom) vehicles, resulting in natural adversarial examples [17]. As the models were applied to CrashD without fine-tuning, due to the different object shapes, the standard PointPillars delivered two false negatives and a false positive. Images used with courtesy of BeamNG GmbH.

object detection, semantic and panoptic segmentation), a recent trend denoted a focus shift towards ensuring the safe applicability of these powerful approaches in critical scenarios, such as autonomous driving and robotics [27]. This has led to the pursuit of improving the model robustness and generalization [12, 22, 37], especially against out-of-domain data, which can naturally occur in the real world [17]. Such approaches include domain adaptation [39] and generalization [37], uncertainty estimation [11], simulations [4], and adversarial alterations [35].

Since corner cases are difficult to be captured as they occur in a dynamic real-life scenario, current datasets include only a limited amount of them, if any [5], leaving most of these cases out-of-domain. However, taking care of corner cases is particularly important in safety-critical settings, where long-tail and out-of-distribution samples could lead to dangerous issues if not accounted for during training [5].

While several works have addressed some of these concerns on the imaging domain [4, 11, 16, 26], this is still

* The authors contributed equally.

○ Contact author: Alexander Lehner (alexanderlehner@tum.de).

Work partly sponsored by the German Federal Ministry for Economic Affairs and Energy (grant 19A19005B), VDA KI-Absicherung project.

mostly unexplored for 3D point clouds [35], also due to the inherent challenges of point clouds, as they are unordered, sparse and irregularly sampled. Nevertheless, as the output of 3D sensors (e.g., LiDAR, ToF cameras), point clouds are especially useful in high automation, where robustness and redundancy are intertwined with safety.

In this context, real non-standard objects, such as damaged and rare cars, or those from different regions, can lead to false negatives, as shown in Figure 1, since the inter-point geometry on which 3D detectors rely is different than usual. While these examples can naturally occur in the real-world [17], they can also be generated artificially with adversarial attacks [14]. This kind of approaches show the vulnerabilities of a model, which can then be addressed to improve robustness. Recent adversarial point cloud alteration methods [35] have tackled this problem to improve the generalization to out-of-distribution data. However, despite being effective attacks, existing adversarial deformation strategies [19, 40] are sample-specific, lack wide-applicability, and by being designed without considering a 3D sensor, are mostly unconstrained in space [19].

In this work, we substantially improve the generalization capability of 3D object detectors to out-of-domain data, bridging this gap by deforming point clouds during training. We propose 3D-VField: a novel adversarial augmentation method that learns to deform point clouds via widely-applicable and sample-independent vector fields (i.e., collections of vectors linked to a set of points in a given space). Our deformations preserve the overall object shape, only slide points along the view ray, and do not add or remove any points. After learning a vector field, we use it to alter objects as data augmentation. The main contributions of this paper can be summarized as follows:

- We raise awareness on natural adversarial examples, such as those represented by damaged and rare cars, around their ability to fool popular 3D object detectors.
- We propose 3D-VField: a sensor-aware adversarial point cloud deformation method based on vector fields able to increase the generalization of 3D object detectors to out-of-domain samples via data augmentation.
- We introduce and publicly release CrashD: a dataset of damaged and rare cars. Extensive experiments on four outdoor and indoor datasets, namely KITTI [13], Waymo [33], our CrashD, and SUN RGB-D [30], show the wide applicability of our approach.

2. Related Work

Our work is about adversarial augmentation to improve the generalization of 3D object detectors for point clouds. In this section we provide a brief overview of existing approaches in these neighboring fields.

2.1. Improving Generalization

Generalization to unseen data is a highly desirable property for any learning-based approach [37]. Unseen data includes any samples on which a model has not been trained on, comprising both out-of-domain and in-domain data (e.g., validation set), depending on the size of the domain shift. In particular, domain generalization deals with improving the performance on a target domain, without any knowledge about it [37], in contrast to domain adaptation which has access to the target data [39]. These works can be grouped in two broad categories: those acting on the model itself, and those operating on the input data.

Among the former category, model regularization strategies are commonly used to reduce overfitting [31] or address domain generalization [3]. Estimating the model uncertainty was also found beneficial for out-of-domain data [11]. Moreover, specific architectures can be found via search algorithms to improve robustness [22].

A different category of works targets generalization by manipulating the input data. Towards this end, it is possible to leverage pretraining and multi-task learning to improve on out-of-distribution samples [2]. Additionally, synthetic data can be included to increase the accuracy on rare classes [4]. Data augmentation methods [16, 32, 45] also belong to this category. Among these, there are adversarial approaches, which extended the training data with altered inputs learned in an adversarial fashion as a way to improve generalization [26, 35, 36].

The method we propose in this work addresses domain generalization (i.e., does not use any target information) and belongs to the data category, specifically to the adversarial approaches, which are detailed in Section 2.2.

2.1.1 Generalization for 3D Object Detection

In the context of generalization, some works addressed the task of 3D object detection, which is also the focus of this work. Simonelli et al. [29] created virtual views normalizing the objects with respect to their distance, to better generalize to samples at different depths in the image domain. Tu et al. [35] improved the generalization towards cars with roof-mounted objects, via adversarial examples on LiDAR point clouds. Wang et al. [39] used domain adaptation to fill the gap between vehicles from multiple countries and different LiDAR sensors.

2.2. Adversarial Examples

Adversarial examples are input alterations designed to lead a model to false predictions [14, 34]. A variety of works explored adversarial examples in the image domain [9, 23, 24, 41, 44], where pixel perturbations imperceptible to humans are able to fool the target model. Alaifari et al. [1] deformed images using a different adversarial vector

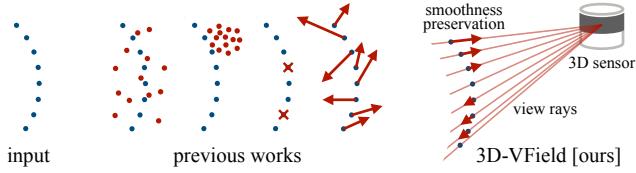


Figure 2. Adversarial deformations introduced by previous works, compared to ours. Other methods add, drop or move points with minor constraints. Ours only slides points along the view ray, while preserving shapes and occlusions.

field learned for each sample. Wang et al. [38] proposed adversarial morphing fields to alter image pixels spatially and fool classifiers. However, this topic is still mostly unexplored on point clouds, especially those captured by 3D sensors (e.g., LiDAR, ToF camera).

2.2.1 Adversarial point clouds

Adversarial methods for 3D point clouds can be grouped in three categories: generation if they add points, removal if they remove points, and perturbation if points are only shifted. Then we present the methods from the perspective of generalization to out-of-domain samples.

Generation and removal Xiang et al. [40] pioneered adversarial point clouds proposing a series of methods, some of which added points to fool the shape recognition. Cao et al. [8] showed the vulnerability of LiDAR-based methods against adversarial objects added to the scene. Similarly, Tu et al. [35] added adversarial meshes on top of cars. A different line of works explored sensor attacks, adding points by means of a spoofing device [7]. Conversely, removal methods adversarially learn to discard a few critical points [43].

Perturbation Xiang et al. [40] also proposed the first two adversarial perturbation approaches. One is the iterative gradient L2 attack, which is an adaptation of PGD from the image domain [20], optimizing for a minimal deformation constrained by the L2 norm. Another approach is the Chamfer attack, which uses the Chamfer distance (CD) between the original and the deformed object to decrease the perceptibility of the attack [19]. The CD is measured by averaging the sum of the distances of the nearest neighbor from each point of the original point cloud to the deformed one. Using this distance function encourages point shifts across the surface of the object. Our method is closely related to the iterative gradient L2 attack, but we do not learn a vector for each point of each sample. Instead, we learn a sample-independent vector field and introduce further constraints to improve our deformations. Liu et al. [19] investigated perturbations more noticeable than the ones of Xiang et al., while producing continuous shapes by altering neighboring points accordingly. Cao et al. [6] 3D printed

adversarial objects to fool multi-modal (LiDAR and camera) detectors.

Generalization Several works on adversarial point clouds were proposed targeting the ModelNet dataset [15, 19, 40], which comprises a set of synthetic 3D point clouds resembling various object shapes. Since ModelNet was not created with a 3D sensor, these foundation works often produce unrealistic outputs [19, 40], that were not intended to improve the generalization of the models, but rather set the basis for adversarial attacks on point clouds [40]. Additionally, these mechanisms are sample-specific, making their applicability limited [15, 19, 40]. Instead, Tu et al. [35] explored the impact on LiDAR object detection of meshed objects, such as canoes and couches, synthesized on top of a car roof. Moreover, they attacked these meshes in an adversarial fashion, and used them to defend the detector, thereby improving its robustness and generalization capability to unseen samples with roof-mounted objects.

Our work sets itself apart from all sample-specific methods [1, 19, 40, 43], as we construct a single highly transferrable and generic set of perturbations. Similar to the work of Tu et al. [35], we aim to improve the generalization to out-of-domain samples. However, compared to theirs, as can be seen in Figure 2, we do not add any points, making ours a perturbation method. Additionally, unlike Tu et al., by not making any assumptions on the object nor the kind of sensor, our method has a wider applicability, from indoor to outdoor settings. Plus, we improve realism by taking into account occlusion constraints, which were ignored so far, and making our deformations sensor-aware, as we only shift points along the sensor ray. Additionally, our method differs from all the ones above also because it generates adversarial point clouds via transferable learned vector fields, which has not been explored yet.

3. Method

We now illustrate our method, based on deforming point clouds to account for natural object variations, thereby improving the generalization of 3D object detectors to out-of-domain data via adversarial augmentation. As shown in Figure 3, we achieve this by adversarially learning a vector field (Section 3.1). Once trained, this vector field can be frozen and then applied to any previously seen or unseen objects, after scaling it to match the target size and constraining the points movement to preserve shapes and occlusions (Section 3.2). We apply it to deform all objects of its class, which we use as data augmentation (Section 3.3).

3.1. Adversarially learned vector field

We create a lattice of uniformly spaced 3D vectors within a 3D bounding box. Since the aim is to perturb the point cloud without adding or removing points, vectors are an immediate representation of this set of point shifts. This

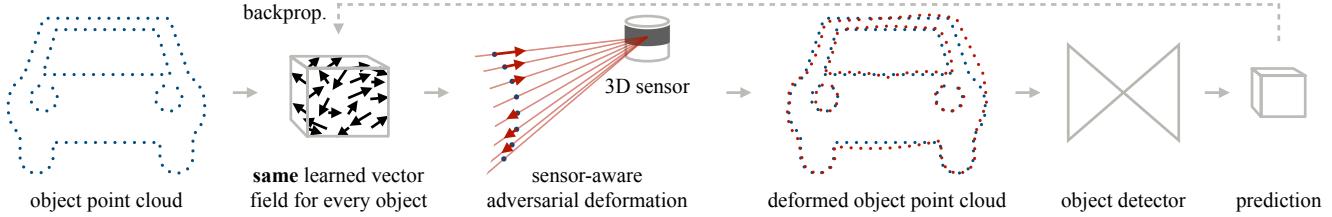


Figure 3. Overview of the proposed 3D-VField. We first learn a vector field adversarially to plausibly deform objects, taking constraints into account. The modified scenes are later used as augmentations to improve the generalization to unseen object shapes.

allows for both compactness and transferability, since the same learned vector field can be applied to any target object. To construct such a vector field, we discretize the space of a default bounding box B_o with a step size t to obtain root coordinates f in 3D space and assign an empty vector $v = (x, y, z)$ to each root. B_o is defined by width w , height h , length l , orientation angle α and its center $c = (x, y, z)$.

Adversarial loss We use a binary cross entropy loss to suppress all relevant bounding box proposals, following [35]. We consider a proposal as relevant if the prediction confidence score $s > 0.1$. \mathcal{Q} is the set of relevant proposal q , where each q has a confidence score s . We minimize s , weighed by the 3D IoU with the the ground truth q^* :

$$\mathcal{L}_{\text{adv}} = \sum_{q, s \in \mathcal{Q}} -\text{IoU}(q^*, q) \log(1 - s). \quad (1)$$

By repeatedly reducing the confidence score while training the vector field, the detector misses the object or predicts a misaligned box. During training, we apply the same vector field to each target object in every scene, minimizing the loss on the whole dataset. At each optimization step, the vectors are updated, resulting in differently deformed point clouds of target objects, which eventually lead to different predictions. As \mathcal{L}_{adv} smoothly converges, the performance of the detector, against which the vector field is optimized, decreases. Once trained, the vectors can be used for data augmentation.

3.2. Objects Deformation

Before applying a vector field, we scale it to match the target object size. Manipulating the points through these vectors, we constrain their movement as described below.

Optical ray consistency To help generalization and preserve the sensor's physical constraints when generating deformations, we employ a simple sensor model in which the 3D points can only be moved across the optical ray. We first compute the ray u_i between the 3D sensor and each point p_i , which determines the deformation direction for each point. Then we calculate the deformation vectors r_i , for each p_i by projecting its nearest vector v_i onto the ray u_i . Points are therefore only moved by r_i .

Regularizing the deformations We limit the perturbation of the points by restricting the vectors with $\|v\|_\infty < \epsilon$ following the standard PGD L_∞ attack [20]. We then ensure shape smoothness along the object surface by sampling multiple k neighboring vectors to move a given 3D point. For each j -th nearest neighbor we calculate the euclidean distance d_{ij} between each point p_i of the object and its nearest vector v_{ij} from the vector field. The final shift m_i of each point is calculated by weighting the deformation vectors r_{ij} with their corresponding distance d_{ij} :

$$m_i = \frac{\sum_{j=1}^k d_{ij} r_{ij}}{k} \quad (2)$$

This allows for a more gradual depth difference between neighboring points, as neighboring vectors with opposite directions would lead to almost no movement of the affected point. Thus, shape smoothness is preserved and less irregular deformations are produced.

Relative rotation We found that using a single vector field for all objects present in the dataset leads to very low amounts of deformation. Due to the various object poses, its vectors would be pointing in all directions, decreasing its efficacy. We circumvent this and allow for a larger degree of alignment between neighboring vectors, by first clustering all the objects in the dataset w.r.t. the relative orientation between object and sensor, and then learning G different fields, one for each cluster.

3.3. Adversarial Data Augmentation

During training of the object detector, we perturb the input point clouds by using the adversarially learned vector fields as data augmentation. This increases the robustness, given that the learned deformations are structurally-consistent, and are therefore more capable than standard augmentations (e.g., scaling, flip, rotation) of resembling out-of-domain car shapes, such as vehicles from a different country [39]. We increase the variability by learning N different vector fields for each of the G rotations (Section 3.2). During training, we randomly select only one object in the scene, and we deform it with a randomly chosen vector field out of the N possible ones for its relative rotation. This high

variability ensures that the model learns both normal and deformed objects, and that each sample can be deformed differently across training, thereby preventing overfitting to specific deformations.

4. Experiments and Results

4.1. Experimental Setup

Datasets We conducted our experiments on four different datasets. Three of them are autonomous driving LiDAR-based: KITTI [13], the Waymo Open Dataset [33], and the proposed synthetic CrashD, which we introduce below. Additionally, we apply our method also on the indoor SUN RGB-D dataset [30], showing its wide applicability. **KITTI** is a popular 3D object detection benchmark recorded in Germany. We adopted a standard split [18], which comprises 3712 training and 3769 validation LiDAR point clouds, where we used the *car* class, reporting on the standard *easy*, *moderate* and *hard*. We evaluated models trained on KITTI (without any fine-tuning) on Waymo and our CrashD to assess the generalization capability of the models to out-of-domain data, particularly critical for autonomous driving. The **Waymo** dataset is a challenging large-scale collection of real scenes recorded in various locations of the USA. It is highly diverse with different weather and illumination conditions, such as rain and night. Furthermore, in the Supplementary Material we show the wide-applicability of our techniques on time-of-flight (ToF) cameras with the **SUN RGB-D** dataset.

CrashD dataset To quantify the generalizability on out-of-domain samples, we produced a synthetic dataset named CrashD. As this includes various types of cars, such as normal, old, sports and damaged, it comprises a variety of plausible vehicle shapes, thereby serving as a valuable out-of-domain test. Specifically, the crashes are individually generated with a realistic simulator [21] and distinguished depending on the intensity, namely *light*, *moderate*, *hard*, as well as the kind of damage: *clean* (i.e., undamaged), *linear* (i.e., frontal or rear), and *t-bone* (i.e., lateral). The randomly and automatically generated 15340 scenes were captured by a 64-beam LiDAR configured to mimic the KITTI one. Each scene presents between 1 and 5 vehicles, with visible damages, before being repaired and placed at the same locations to collect the *clean* set, resulting in a total of 46936 cars. We are releasing this data publicly, as an out-of-domain evaluation benchmark for models trained on KITTI [13], Waymo [33] or similar datasets. Further details can be found in the Supplementary Material.

Evaluation metrics We evaluated the object detection performance on the standard **AP**, with a 3D IoU threshold of 0.7 for KITTI and CrashD, 0.5 for Waymo, and the standard 0.25 for SUN RGB-D. To measure the quality of the adversarial perturbations we followed Tu et al. [35] using

the **attack success rate** (ASR) metric. It measures the percentage of objects that become false negatives after undergoing an adversarial alteration. For the ASR, we considered an object detected if its 3D IoU was larger than 0.7.

Network architectures We used four different 3D object detectors. PointPillars [18] voxelizes the scene in vertical columns (i.e., pillars) from the bird’s eye view, using PointNet for feature extraction. Second [42] voxelizes the point cloud and uses a learned voxel feature encoding. Part-A² Net [28] is an extension of PointRCNN that predicts intra-object part locations for improved accuracy. VoteNet [25] (Supplementary Material) is based on PointNet++ and Hough voting. While the first three are mostly used for autonomous driving, VoteNet is used indoor.

Implementation details We constructed each vector field within B_o with $w = 1.8\text{m}$, $h = 1.6\text{m}$, $l = 4.6\text{m}$ and a step size of $t = 20\text{cm}$ resulting in 1656 vectors per vector field. If not stated otherwise, we grouped objects by relative rotations with $G = 12$ groups, and set $N = 6$. During the perturbation stage, we moved points according to their $k = 2$ nearest vectors and deform only along the sensor ray. For the PGD optimization, we used Adam with a learning rate of 0.05. The distance threshold was set to $\epsilon = 30\text{cm}$. Each vector was randomly initialized from a uniform distribution with values between -1cm and 1cm. We trained all models using PyTorch and MMDetection3D [10] on a single NVIDIA Tesla V100 32GB GPU.

Prior works and baseline We focused on object detection and compared with other adversarial methods. All models were applied on PointPillars [18], unless otherwise noted. As point perturbation methods we used the iterative gradient L2 [40] and the Chamfer attack [19]. For generation we used [40] adding 10% and [43] removing 10% of the objects points. For a fair comparison, we trained all on the same KITTI dataset split [18], with $\epsilon = 30\text{cm}$, then we altered the point clouds as data augmentation with the same settings as ours (i.e., random selection of one object per scene to augment). Moreover, we combined ours with the domain adaptation statistical normalization (SN) strategy of [39]. Following [39], after computing the average box dimensions in the target datasets (i.e., Waymo and CrashD), we scaled the source (i.e., KITTI) point clouds within the ground truth boxes accordingly and fine-tuned the trained models with this altered target-aware source data.

4.2. Quantitative Results

Adversarial methods and generalization Table 1 shows the comparison between our 3D-VField and related adversarial approaches when applied on PointPillars [18] in the context of generalization. In particular, we report other adversarial perturbation methods, such as the iterative gradient L2 [40] and the Chamfer attack [19], adversarial generation [40], as well as adversarial removal [43]. Augment-

Architecture	Method	KITTI			ASR	AP	→ Waymo		→ CrashD			
		easy	AP mod.	hard			AP normal clean	AP normal crash	AP rare clean	AP rare crash		
PointPill. [18]	no augm. [18]	70.00	61.88	56.23	-	30.68	1.79	0.93	3.92	2.33		
	no obj. sampl. [18]	83.83	74.14	68.30	-	37.85	50.36	36.44	28.70	20.02		
	PointPillars [18]	88.24	77.11	74.55	-	40.86	65.20	43.67	34.14	22.48		
	iter. grad. L2 [40]	86.24	76.92	73.84	*95.9	39.86	58.65	41.86	35.92	23.69		
	Chamfer att. [19]	87.15	77.05	74.07	*99.8	40.54	56.84	39.56	36.29	24.73		
	advers. gener. [40]	86.12	76.39	73.18	*91.6	40.55	57.75	38.03	35.73	24.18		
	advers. remov. [43]	86.51	76.85	74.04	*86.1	40.32	66.52	48.88	41.42	28.10		
	3D-VField [ours]	87.05	77.13	75.55	63.4	44.61	67.95	52.87	43.40	30.37		
	SN dom. adapt. [39]	-	-	-	-	49.27	79.42	72.59	60.53	48.23		
	[ours] + SN [39]	-	-	-	-	51.32	92.14	87.28	86.26	76.42		
Second [42]	Second [42]	88.93	78.68	76.87	-	42.45	72.73	56.74	41.85	32.84		
	3D-VField [ours]	88.87	78.56	76.81	54.9	43.51	76.54	60.51	47.47	36.14		
Part-A ² [28]	Part-A ² [28]	89.60	79.16	78.52	-	49.76	83.05	63.25	74.03	52.33		
	3D-VField [ours]	89.65	79.26	78.62	50.5	56.08	88.80	73.80	81.10	61.34		

Table 1. Comparison of models trained on KITTI [13] towards out-of-domain data (without any fine-tuning), namely Waymo validation set [33] and our CrashD datasets, as well as on the KITTI validation set. Each method applies a data augmentation (for adversarial ones ASR is measured on their adversarial examples), or performs domain adaptation (only SN [39] in this work), resulting in the reported APs. →: transfer from KITTI. *: being sample-specific, the adversarial method had to be trained on the validation set of KITTI.

ing with the adversarial examples of our 3D-VField did not reduce the overall in-domain AP compared to PointPillars, but brought numerous benefits in terms of out-of-domain generalization. As demonstrated by Wang et al. [39], the transfer from KITTI to **Waymo** is particularly challenging due to the different shapes and sizes of the vehicles found in Germany and the USA, as well as the 50% higher point density and the narrower field of view [33]. This test assesses the quality of the generated deformations with respect to real vehicle shapes found in a different country. On Waymo our 3D-VField delivered more than 9% relative improvement over PointPillars and the other adversarial methods, and 13% over Part-A² [28], proving the benefit of our added sensor-awareness on real and challenging out-of-domain data. On the right of Table 1 we report the results on the proposed **CrashD**. It can be seen that despite the transfer from KITTI, the AP on *clean normal* cars is relatively high for all approaches, likely because those samples are not particularly difficult. However, when damaging those exact same vehicles and placing them at the same locations (*crash*), the detection performance dropped. This shows the effort required for the methods to relate these to the cars learned on KITTI, and proves them as natural adversarial examples. Similarly, with *rare* cars (i.e., old and sports cars), the AP dropped even more, quantifying the domain shift from *normal* vehicles. *Rare crash* cars, by combining the two out-of-domain aspects (i.e., rarity and damage),

were the hardest for all methods, reducing the AP from *normal clean* by up to two thirds (PointPillars). Nevertheless, our method improved significantly over the detectors and the other adversarial approaches for all transfers and categories. This can be attributed to our adversarial augmentations introducing diversity in the training data, while being sensor-aware. In particular, the sensor-awareness ensures that the deformed point clouds are still plausible, thereby better resembling possible out-of-domain samples, such as those of Waymo and CrashD. Among the other adversarial approaches, only removing points [43] improved generalization to CrashD, probably because it preserved the overall point clouds. Nevertheless, [43] was not beneficial on Waymo, which features denser point clouds and more challenging real scenes.

Combination with data augmentations As adversarial data augmentation, our 3D-VField is not alternative to different augmentation strategies, but can be applied in combinations with others. In Table 1 we show how common data augmentation techniques impact the detections for PointPillars [18]. Using no augmentations (no augm.) critically reduced the APs, especially on CrashD at IoU 0.7 (Table 1). At IoU 0.5, this resulted in an AP on *normal clean* of 65.59, while the baseline [18] delivered 98.91. Introducing standard augmentations (no obj. sampl., e.g., flip and rotation) improved, but adding the popular object sampling [18] (PointPillars) increased the APs further. On top, our aug-

# G	K. ASR \uparrow	K. mod.	\rightarrow Waymo	# vectors
1	55.08	77.32	40.43	10K
12	63.37	77.13	44.61	120K
360	44.84	77.06	40.30	3.6M

Table 2. Our 3D-VField trained on KITTI (K.) with varying amounts of relative rotations G . \rightarrow : transfer no fine-tuning.

mentations substantially improved all transfers, without decreasing the in-domain performance.

Combination with domain adaptation By addressing domain generalization, our approach does not use any target information. Therefore, ours is not alternative to domain adaptation methods [39], which make use of target data. However, similarly to other data augmentation strategies, our 3D-VField can be combined with domain adaptation techniques. As shown in Table 1, such combination further boosts the performance on challenging out-of-domain data. By altering the objects size via the statistical normalization (SN) of [39], the AP on Waymo increased. Constrained by the high amount of false positives and negatives, when combined with SN, ours retained a margin of over 2% compared to PointPillars with SN. Moreover, the AP on CrashD improved dramatically across all categories, especially for the hardest *rare crash* group. The results show how, despite a substantial increase in AP from PointPillars [18], SN alone did not reach the full potential of the detector. Only when combined with ours, the AP doubled (*normal crash*) and more than tripled (*rare crash*) over PointPillars, without using any extra target information. This shows the benefit of this combination, and reiterates the added value of incorporating adversarially deformed objects via data augmentation to improve generalization to out-of-domain samples.

Adversarial methods as attacks In terms of ASR (Table 1), our approach is not as strong as the other adversarial methods, namely the iterative gradient L2 [40], the Chamfer attack [19], adversarial generation [40] and removal [43]. However, this is expected as our vector fields are sample-independent, compared to their point-to-point deformations being sample-specific. Due to this reason, their alterations had to be learned directly on the KITTI validation set, on which the ASR was measured. Nevertheless, a very high ASR means the altered objects are unrecognizable, which does not aid generalization. The goal of our method is not having a detector fully miss the attacked objects (high ASR), but rather deforming them to improve the performance on out-of-domain data. Towards this end, the perturbed objects need to be at the same time altered enough to add diversity to the training data, and not be too far apart from the training distribution to avoid confusing the detector. We found this balance by learning our vector fields ad-

Method	KITTI		\rightarrow W.	\rightarrow CrashD	
	mod.	ASR		$n., clean$	$r., crash$
P.P. [18]	77.11	-	40.86	65.20	22.48
no learn	76.36	10.1	41.62	62.94	21.75
unleash	76.82	97.7	40.95	60.43	27.55
ray con.	76.35	59.5	41.03	59.82	29.16
full	77.13	63.4	44.61	67.95	30.37

Table 3. Ablation on the deformation constraints imposed by our method, compared to PointPillars (P.P.) [18]. Trained on KITTI. \rightarrow : transfer no fine-tun.; W.: Waymo.

versarily, while preserving the objects shape and the sensor realism with our added constraints.

Different 3D detectors In Table 1, we also compare the performance of our 3D-VField when paired with different 3D object detectors, namely PointPillars [18], Second [42], and Part-A² [28]. Remarkably, using the proposed adversarial augmentation improved the AP of Part-A² on Waymo by a large margin. The superiority of Part-A² over the other detectors can be attributed to its part-awareness [28], which might have set its focus on the most relevant object parts (e.g., wheels) and their relationships to identify cars also in out-of-domain settings. For Second [42], the performance on KITTI turned out lower than the one reported in [10], despite using the same settings and framework. This reduced AP affected both the baseline [42] and our approach. Nevertheless, adding our adversarial deformations significantly improved the generalization of all three detectors to out-of-domain data, despite training our vector fields solely against PointPillars. This shows the wide applicability and transferability of our techniques.

Specificity-generalization trade-off Table 2 shows that by varying the amount of relative rotations G , a trade-off arises between generalization, attack specificity (i.e., strength on individual samples by overfitting to the training data), and storage (i.e., amount of vectors). $G = 12$ offers a good balance. With the extreme $G = \#$ of objects, ours would become sample-specific, inheriting the weaker generalization capabilities of [19, 43]. While these methods needed to be trained on the validation set, allowing for high ASRs (Table 1), our vectors were learned on the training set. So with high G , ours overfitted on the training data, which is visible evaluating on the validation set. Our augmentation strategy learns only 1656 3D vectors to perturb objects. However, by training with $G = 12$ and $N = 6$, the amount of vectors increased to 120K. Conversely, the sample-specific iterative gradient L2 [40] and the Chamfer [19] attacks required 10.9M and 12.6M vectors for training and validation sets respectively. This shows the easy applicability of our 3D-VField.

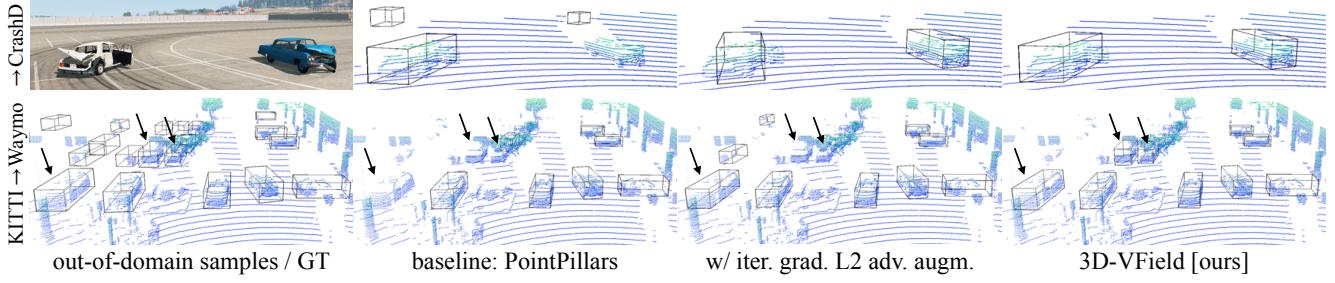


Figure 4. Predictions on challenging out-of-domain samples from the proposed CrashD (top) and Waymo [33] (bottom). Models based on PointPillars [18] trained on KITTI (without fine-tuning). Iterative gradient L2 [40] and ours trained with adversarial augmentation.

Ablation study on deformation constraints As we introduced the sensor-awareness and the surface smoothness constraints to our deformations, we investigate their impact in terms of generalization to out-of-domain data. In Table 3, we report this comparison when limiting the deformations to $\epsilon = 30$ cm. It can be seen that not learning the perturbations, but applying all our constraints (no learn) could already be a beneficial augmentation technique, as it improved the transfer to Waymo. Instead, removing all constraints, but learning the vector fields (unleash) delivered a strong ASR of 97.7%. This significantly increased the AP on the CrashD *rare* cars. When deforming with sensor-awareness (ray con.), ASR reduced, but the AP on the most difficult transfer settings (i.e., *rare crash*) improved. Our full model 3D-VField, adds the distance smoothing (Section 3.2) delivering superior transfer capabilities. Furthermore, increasing the maximum deformation ϵ to 40 or 60 cm, improved the ASR to 73.3% and 87.1%, but as augmentation decreased the AP on KITTI by 1% and 1.7%, respectively. This means that higher deformations do not generalize well, as their plausibility decreases, while 30 cm offers a good trade-off.

4.3. Qualitative Results

In Figure 4 we compare the transfer predictions from KITTI to CrashD and Waymo [33] of the standard PointPillars [18], augmented with ours and the iterative gradient L2 adversarial approach [40], which is the closest to ours in terms of adversarial deformation (Section 2). For CrashD, as seen in the quantitative results (Section 4.2), the iterative gradient L2 method delivered better detections compared to not using any adversarial augmentations [18], but our 3D-VField outperformed it, with a more aligned box for the left damaged car. The figure also shows the severity of the *hard* damages present in CrashD, and how adversarial augmentation helps to detect such challenging samples. For the difficult transfer KITTI → Waymo (Section 4.2), it can be seen that all methods had troubles detecting the cars with few points in the parking lot on the left. Furthermore, PointPillars [18] ignored 3 recognizable cars with a high amount

of points, while augmenting with the iterative gradient L2 caused missing 2 of them and detecting 2 further ones, albeit with misaligned boxes. Instead, despite missing further ones, our method was able to recognize these visible cars.

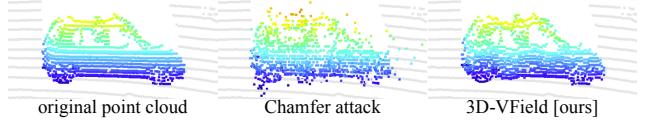


Figure 5. Example deformations by our method and the Chamfer attack [19] on a car of the KITTI validation set [13].

Figure 5 confirms that the strong ASR of the Chamfer attack [19] seen in Table 1 corresponds to unrecognizable objects. It also provides an example of the minor deformations introduced by our adversarial vector fields. By preserving the overall shape of the car and its surfaces, ours allowed for superior generalization to unseen data.

We refer to the **Supplementary Material** for more results on indoor settings, transferability, robustness against noise, detailed evaluations on CrashD, and various ablation studies on grouping and aggregation strategies, as well as the amount of deformed objects during training.

5. Conclusion

In this paper we presented 3D-VField: an adversarial augmentation method for point clouds to improve the object detection performance on natural adversarial examples and out-of-domain data, such as rare, damaged cars, or vehicles from different regions. Towards this end, 3D-VField produces plausible shapes used as data augmentation. Extensive experiments showed the high generalization and transferability of the proposed approach, from indoor to outdoor settings, on both real and synthetic data. Furthermore, we proposed and released CrashD: a new benchmark to challenge 3D object detectors on out-of-domain data, including various kinds of damaged and rare cars.

A. Supplementary Material

In this supplementary material we include further details and results. Specifically, Section A.1 describes the proposed CrashD out-of-domain dataset to a greater extent, Section A.2 provides additional implementation details, Sections A.3 and A.4 report more quantitative and qualitative results on outdoor data, while Section A.5 provides results on ToF camera data in indoor settings.

A.1. Details on the Proposed Dataset: CrashD

In this section we further describe the proposed dataset: CrashD. We refer the reader to the dataset webpage to see examples of the generated accidents and scenes.

A.1.1 Intended Use

This dataset was designed to evaluate the performance of LiDAR-based 3D object detectors on out-of-domain data. It is meant to serve as a test benchmark for 3D detectors trained on KITTI [13], Waymo [33], or similar datasets.

It should be noted, that CrashD is not intended for training and evaluating an object detector directly, since the generated LiDAR scenes do not include anything other than ground and cars. Therefore, training and evaluating on this dataset would be rather trivial, since the detector could learn that anything rising from the ground is a car, except for the relatively small spare parts separated by the accidents (e.g., the tire in Figure 1).

Nevertheless, reasonable uses of the proposed CrashD could include domain adaptation, transfer learning, and domain generalization [37], as well as synthetic-to-real transfers. Furthermore, it could be used to assess the damage of a vehicle, and also for uncertainty estimation or similar methods to detect out-of-distribution samples. Moreover, it could serve for point cloud reconstruction, or anomaly segmentation approaches comparing damaged and undamaged cars, since for each crashed vehicle in a scene we provide its repaired counterpart at the same location.

A.1.2 Driving Simulator

CrashD was generated using a driving simulator developed by BeamNG [21], which includes a realistic physics engine, allowing for realistic damages. It offers a Python interface to setup the scenarios programmatically. Furthermore, it features a variety of sensors, including a LiDAR with customizable settings. Therefore, we equipped the ego vehicle with a LiDAR that imitates the one used in KITTI [13].

A.1.3 Data Generation and Collection

We generated random accidents with random settings (e.g., hitting angle, distance, type of hitting car, type of hit car),



Figure 6. LiDAR scene setup of CrashD. For each car, a black arrow indicates its damaged area, which is ensured to be visible from the sensor viewpoint. *Image used with courtesy of BeamNG GmbH.*

and placed the cars randomly in the LiDAR scene. On each type of car (i.e., *normal* and *rare*), we applied 2 types of accidents (i.e., *linear* and *t-bone*), with 3 intensities each (i.e., *light*, *moderate* and *hard*). That results in 12 different categories of damaged cars and their 12 undamaged counterparts (i.e., *clean*), resulting in 24 categories overall. As the undamaged cars were placed at the exact same locations in the LiDAR scenes, they can be used as control group, to check the performance drop of a 3D detector when introducing the damages on the same cars.

We generated the accidents as follows. For each of the 12 categories of damages, we randomly selected 5 cars of the corresponding vehicle type (i.e., *normal*, *rare*), and 1 hitting vehicle. The hitting vehicle crashed into each of the 5 cars, getting repaired before each crash. We then repeated this process at least 64 times for each of the 12 categories, generating more than 3840 different accidents.

Furthermore, within each category, we used several random parameters, resulting in a high amount of possible damages. The intensity was determined by the distance from which the hitter starts, so the higher the distance, the higher the speed at which it will hit the target (i.e., one of the 5 cars). The effect of different intensities on the two types of cars for a *linear* crash can be seen in Figure 7. For each intensity type, there was a random variable determining a variation of the distance at which the hitter was placed. Then, the hitting angle and the side (i.e., front or back for *linear*, and left or right for *t-bone*) were also randomized. Overall, this covered 360 degrees for each type of car and intensity.

Each batch of 5 cars, after being hit, was randomly placed in the LiDAR scene, such that the damaged area was visible from the sensor viewpoint, as shown in Figure 6. We considered a crash visible if the sensor was within 35

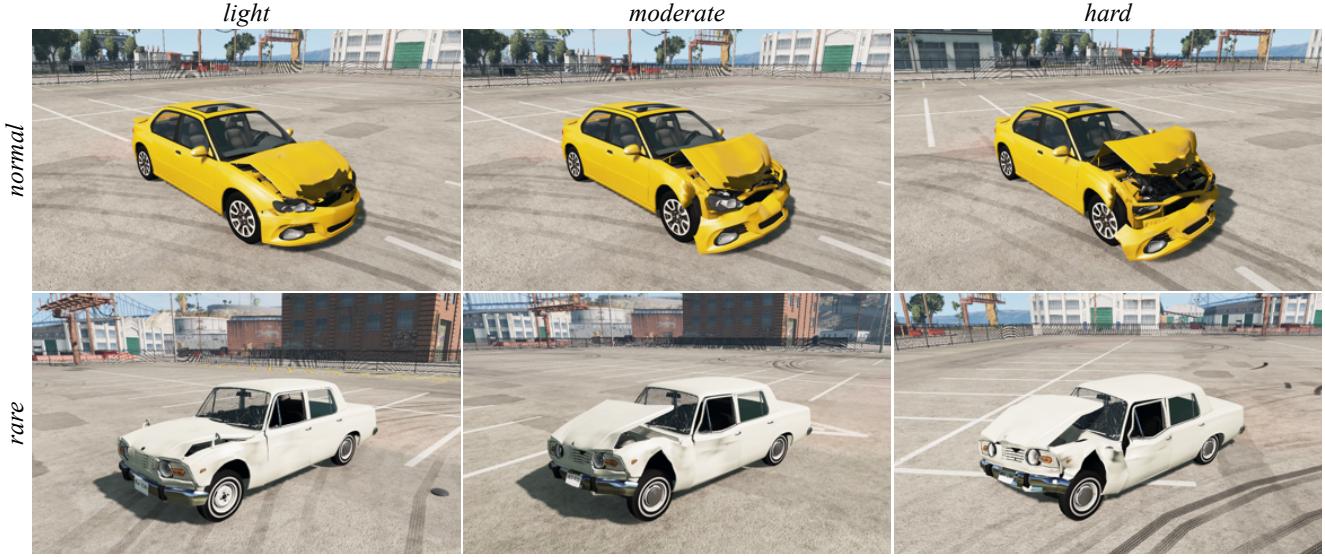


Figure 7. Comparison of *linear* damage intensities for *normal* and *rare* cars of CrashD. For each type of car, the accidents were created by the same hitting vehicle, coming from the same angle. It can be seen that the *hard* crash compromised the structure of the weaker *rare* car, while the *normal* car absorbed the impact differently, leaving the cabin unchanged.

degrees from the hitting angle. This ensured that a car classified as damaged is represented by a deformed point cloud. Moreover, if the damaged part was not visible from the sensor, the car was discarded from the batch.

This was due to a series of reasons, resulting in the lack of control over the rotation of the damaged car within the LiDAR scene. In particular, BeamNG setup the simulator [21] such that if a vehicle is rotated programmatically, it gets automatically repaired. Plus, depending on the dynamics of a crash, a damaged vehicle could rotate following the impact. So, as we reduced the LiDAR scene to the front 180 degrees, we had to discard some cars to be sure that they were not classified as damaged if their impacted area was not visible. To avoid that crashes with a set of hitting angles could systematically not be placed in the scene, we randomly rotated the whole accident scenarios.

For each batch of 5 cars, we recorded 10 frames with the cars with visible damages (between 1 and 5), where we randomized the distance from the sensor, as well as the angle around it. Moreover, again to avoid that a vehicle is considered damaged if the affected area is not visible, we excluded occlusions considering only 25 angles around the sensor, and preventing two cars from occupying the same one. This resulted in 750 possible different locations in the scene. With this setup, a given vehicle might be discarded in one frame if the angles from which its damage is visible are occupied by other cars, but might appear in a subsequent frame if it gets placed beforehand.

Furthermore, we put the objects only in the front, motivated by the front-facing setup of KITTI [13], thereby facilitating transfers from KITTI to the proposed CrashD. To-

wards this end, we positioned the vehicles from 10 to 40 meters away from the LiDAR, around its front 180 degrees. As shown in Figure 6, the scene features a large parking lot, where no object is located, other than the cars. We selected a totally empty parking lot (lacking poles, trees, or anything else), to fully focus on the task at hand, providing test data for evaluating the generalization capability of a method to different object shapes. Instead, having distracting elements (e.g., trees) in the scene, could have led to a different kind of transfer evaluation (e.g., the ability of recognizing cars compared to other objects in the scene), which goes beyond the scope of this dataset. Nevertheless, in the main paper, as well as in additional results in this supplementary material, we also show a transfer from KITTI [13] to Waymo [33], which features real complex scenes, with trees and other objects, thereby challenging the 3D detector in a different way compared to transferring to the proposed CrashD.

A.1.4 Vehicles

The simulator offers a variety of fictional vehicles, which are shown in Figures 8, 9 and 10. In particular, the 12 *normal* cars used are shown in Figure 8, resembling the vast majority of vehicles on the road today in the countries where common LiDAR datasets were recorded, such as Germany and USA, for KITTI [13] and Waymo [33] respectively. Figure 9 shows the 7 *rare* cars used for CrashD, including older cars from Europe, USA and Asia, as well as a wedge-shaped sports car. Among older cars, the simulator features different muscle cars, and also a very small car (at the top left of Figure 9).



Figure 8. *Normal* cars of CrashD. These were classified as *normal* as they resemble the vast majority of cars on the road today in Germany, USA, and other locations where popular LiDAR datasets, such as KITTI [13] and Waymo [33], have been recorded.



Figure 9. *Rare* cars of CrashD. These were classified as *rare* as they complement the *normal* (i.e., common) cars shown in Figure 8. In particular, *rare* ones resemble old cars from various regions, and also include a wedge-shaped sports car. * indicates cars that cannot hit other vehicles (due to their low speed and weight), but can only be hit by others.

The significant gap between the two types of cars can be seen by comparing the *normal* and *rare* vehicles in Figures 8 and 9 respectively. Specifically, considering the *normal* cars resemble those from KITTI and Waymo, the shapes of the *rare* ones are rather different, posing a substantial challenge for any LiDAR-based 3D object detector transferring on this dataset from those two others. Analogously, detecting the cars with the various deformations resulting from the accidents, which can be seen in Figure 7, pose a different, but also significant challenge for a detector trained on KITTI, Waymo, or a similar dataset.

Since the KITTI [13] *car* annotations do not include vans, trucks, pickups and busses, we excluded these from the detectable vehicles of CrashD. Nevertheless, these vehicles were part of the pool of hitting vehicles, and they

are shown in Figure 10. Hitting vehicles also included all the ones shown in Figure 8, as well as those in Figure 9. However, we excluded the 2 cars marked with * due to their relatively low speed and weight, which would have not provided an accident as intense as those caused by the other vehicles, thereby altering the data distribution along the intensity types (i.e., *light*, *moderate*, *hard*). In spite of that, the 2 with * were part of the detectable vehicles.

A.1.5 Dataset Statistics

In total, the proposed CrashD includes 46936 cars, half of which are damaged and half are not, as the LiDAR scenes were repeated with and without damages. *Normal* cars are 23314, while *rare* ones are 23622, again half of each is dam-



Figure 10. These vehicles can only hit others and are not detectable objects, as they do not fit the KITTI [13] criteria for being a car, so they would not get recognized by a model transferred from KITTI.

aged. 8124 cars were hit by *light* accidents, 7453 *moderate* and 7891 *hard*. 11530 were affected by a *linear* crash, while 11938 by a *t-bone*. Due to the vehicle placement in the LiDAR scene being dependent on the damage visibility, cars undergoing a *linear* crash were more likely to be included from a frontal or rear perspective (including 3/4 views), while *t-bone* ones were only included from the sides.

A.2. Additional Implementation Details

Iterative gradient L2 attack For this attack [40] we minimize our adversarial loss \mathcal{L}_{adv} constraining the deformation \mathbf{m} for each point \mathbf{p} with $\|\mathbf{m}\|_2 < \epsilon$, with $\epsilon = 30$ cm.

Chamfer attack For the Chamfer attack [19] we used the Chamfer distance to measure the gap between the original and perturbed point clouds, which is given by:

$$\mathcal{C}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2 \quad (3)$$

for two sets X and Y . We perturb by minimizing:

$$\mathcal{L}_{\text{cha}} = \mathcal{L}_{\text{adv}} + \lambda \mathcal{C}(\mathbf{p} + \mathbf{m}, \mathbf{p}) \quad (4)$$

with λ set to 0.1 and the amount of deformation constrained by $\mathcal{C}(\mathbf{p} + \mathbf{m}, \mathbf{p}) < \epsilon$, with $\epsilon = 30$ cm. It should be noted that single deformations vectors could lead to perturbations larger than 30 cm, since what is bounded is the overall Chamfer distance and not single vectors. This attack led to only a small amount of perturbed points, but the ones that moved showed large displacements.

Adversarial removal For the removal attack we follow [43] and remove 10% of the *critical points* of an object. These are those input points that if removed, the prediction changes. We estimate them as those with the highest deformation magnitude from the iterative gradient L2 attack [40].

Adversarial generation We follow [40] adding 10% of the objects points. We initialize their location as that of the *critical points* (see removal). We then perform the iterative gradient L2 attack [40] solely on the added points. Thus shifting them to decrease the detection quality.

Transfer to Waymo To evaluate the transfers to Waymo [33], we used the standard KITTI evaluation. Therefore, the LiDAR scene was cut until 70 m in front of

the ego vehicle and 40 m to both sides. We also lowered the whole point cloud and ground truth bounding boxes by 1.6 m, to match the KITTI coordinates and ground plane.

A.3. Additional Outdoor Quantitative Results

A.3.1 Transferability of the Vector Fields

Adv.aug.	PointP. [18]		Second [42]		Part-A ² [28]	
	AP	ASR	AP	ASR	AP	ASR
none	77.1	63.4	79.2	54.9	79.2	50.5
w/o \mathcal{L}_{adv}	76.4	60.0	77.2	52.5	79.3	47.4
[ours]	77.1	21.8	78.1	18.3	79.3	18.7

Table 4. Moderate AP and ASR ↓ across different models, showing transferability and efficacy of our deformations, on the validation set of KITTI. ASRs on Second and Part-A² are measured on vector fields trained on the defended PointPillars, to report the transferability. Adv.aug.: adversarial augmentation; w/o \mathcal{L}_{adv} : ours not learned.

Table 4 shows the high transferability of our adversarial deformations to other 3D object detectors. It can be seen that perturbations learned on PointPillars [18] are highly effective also on rather different architectures such as Second [42] and Part-A² [28], maintaining up to 86% ASR across the models. Table 4 reports also the benefit of our adversarial augmentation strategy against our deformations. The perturbed point clouds targeting PointPillars are effective also to defend the other models.

A.3.2 Robustness against noise

Method	-10%	-5%	0%	+5%	+10%
PointP. [18]	70.51	70.88	77.11	67.36	65.27
[ours]	71.45	71.75	77.13	69.57	65.86

Table 5. KITTI validation moderate AP under various % of removed and added points within the cars bounding boxes.

→ CrashD		normal, linear			normal, t-bone			rare, linear			rare, t-bone			
		light	mod.	hard	light	mod.	hard	light	mod.	hard	light	mod.	hard	
PointP. [18]	clean	baseline [18]	59.6	64.4	60.6	65.5	73.7	67.3	33.5	33.8	27.7	37.5	35.1	37.3
		3D-VF [ours]	61.8	64.2	62.0	72.4	76.7	70.6	39.6	41.1	35.0	49.6	47.4	47.7
Second [42]	clean	baseline [42]	67.0	68.6	68.7	76.1	81.1	75.0	39.3	43.8	37.5	43.7	42.5	44.4
		3D-VF [ours]	71.3	75.4	73.1	79.3	82.4	77.7	40.9	47.5	41.5	52.8	49.2	53.0
Part-A ² [28]	clean	baseline [28]	60.1	46.4	43.0	72.0	65.6	53.3	36.1	37.8	28.8	40.1	31.4	22.9
		3D-VF [ours]	64.8	50.4	44.9	75.5	69.4	58.1	38.4	37.0	29.1	49.3	37.7	25.4
Part-A ² [28]	clean	baseline [28]	77.9	82.7	78.4	86.6	87.6	85.2	71.5	72.7	73.7	78.3	72.9	75.1
		3D-VF [ours]	85.6	86.2	86.0	91.3	93.2	90.5	80.0	81.6	79.8	83.7	79.3	82.2
Part-A ² [28]	crash	baseline [28]	71.1	58.6	49.3	79.7	64.3	56.5	61.7	55.5	49.0	67.0	48.6	32.2
		3D-VF [ours]	81.1	69.4	63.3	87.3	75.8	65.9	74.9	69.1	59.0	74.5	53.8	36.7

Table 6. Detailed AP comparison of PointPillars [18], Second [42], and Part-A² [28] trained on KITTI [13] and transferred to the proposed CrashD without any fine-tuning. The evaluation is shown according to the various accident types, and intensities, as well as the kinds of car. Baseline indicates the standard method, while [ours] shows the impact of our adversarial augmentation strategy.

→ CrashD		IoU 0.1			IoU 0.5			IoU 0.7		
		baseline [18]	3D-VF [ours]	baseline [18]						
normal, clean	TP ↑	11547	11651	11539	11638	8571	8894			
	FP ↓	4069	419	4077	432	7045	3176			
	FN ↓	110	6	118	19	3086	2763			
normal, crash	TP ↑	11485	11642	11391	11562	6770	7620			
	FP ↓	4550	772	4644	852	9265	4794			
	FN ↓	172	15	266	95	4887	4037			
rare, clean	TP ↑	11761	11805	11747	11790	6091	7528			
	FP ↓	4700	316	4714	331	10370	4593			
	FN ↓	50	6	64	21	5720	4283			
rare, crash	TP ↑	11724	11804	11566	11680	4688	6011			
	FP ↓	4742	590	4900	714	11778	6383			
	FN ↓	87	7	245	131	7123	5800			

Table 7. Impact of our adversarial augmentation on the main categories of the proposed CrashD according to true positives (TP), false positives (FP) and false negatives (FN) at different IoU thresholds. The models were based on PointPillars [18], trained on KITTI [13] and transferred to CrashD without any fine-tuning. For reference, the total amount of cars in CrashD is 46936.

In Table 5 we report the performance of PointPillars [18] with and without our adversarial augmentation strategy. For this set of experiments, at inference time we randomly added and removed points within the cars bounding boxes according to the percentages reported in the table. Both models were the same as in the rest of this work, simply evaluated with this setup. Thanks to the improved generalization provided by our vector fields, the augmented model

was more robust against such noise. Our augmentation acts as regularization during training, allowing the model to learn more meaningful features independent of specific points. This led to a constant gap between 5 and 10% removal. Conversely, randomly adding points is not realistic from the sensor perspective, since occlusions and its physical properties are not respected. Due to this reason, both models suffered more when adding points, than removing.

A.3.3 Detailed transfer to CrashD

Evaluation by categories In Table 6, we show a detailed evaluation of the various 3D object detectors along the different sub-categories of the proposed CrashD, with various kinds of damages, different intensities and types of cars. Our adversarial augmentation strategy outperformed all detectors [18, 28, 42] across the board by a significant margin, especially on *rare* cars. In particular, with high intensity crashes (*hard*), the baselines [18, 28, 42] severely underperformed, reducing by half their APs on cars undergoing a *t-bone* accident. This can be due to the large point displacement introduced by the impacts, especially with weaker old cars. Conversely, our 3D-VField, as it was trained on sensor-aware deformations, was more robust against these damages, delivering a smaller decrease from the *clean* cars to their *crash* counterparts. Interestingly, *rare* vehicles were often more challenging to be detected than *normal crash* ones. This can be attributed to an accident typically affecting only a local region of a vehicle, leaving the rest of it untouched and detectable, compared to a rare design which has an impact on the whole object point cloud, making it in general harder to be recognized. Comparing the same cars with and without damages (*crash* and *clean*) shows that the former are significantly more difficult for every detector, due to the different resulting shapes. All detectors substantially benefited from our adversarial augmentations, despite training the vector fields solely against PointPillars [18]. The values also confirm the superiority of Part-A² [28] over the other 3D detectors, as seen in Table 1.

Correct and wrong detections on CrashD Table 7 reports a comparison of PointPillars [18] without and with our adversarial augmentations on CrashD, according to the number of true positives, false positives and false negatives, depending on the main categories of the proposed dataset, at different IoU thresholds. It can be seen that the baseline [18] had a strong tendency towards over-predicting the amount of objects in the scene, resulting in a high number of false positives. In fact, even with a low IoU threshold of 0.1, over 30% of the boxes predicted by the baseline did not match any car in the scene. At the same time, it completely ignored several cars, both damaged and undamaged, resulting in false negatives. On the other hand, as seen already in the main paper showing the APs, the proposed 3D-VField delivered a significantly better detection rate, vastly reducing the amount of false positives and negatives, despite being based on the same architecture and settings as the baseline [18].

A.3.4 Ablation Studies

Amount of deformed objects In Table 8 we report the effect of augmenting various amounts of objects during training. Specifically, augmenting more objects in each scene

# augm. objects	KITTI	\rightarrow W. <i>mod.</i>	\rightarrow CrashD	
	<i>n., clean</i>		<i>r., crash</i>	
[ours] 1 obj.	77.13	44.61	67.95	30.37
[ours] 50% obj.	76.31	39.60	53.99	23.63
[ours] 100% obj.	59.30	32.84	38.29	14.83

Table 8. Models trained on KITTI, augmented with our adversarial technique. In each row, the amount of objects augmented at training time in each scene changes. The chosen number of augmented objects was 1. *mod.*: moderate difficulty; \rightarrow : transfer without any fine-tuning; W.: Waymo; *n.*: *normal*; *r.*: *rare*.

did not help generalization, as it made difficult to recognize standard objects. Augmenting all cars means the detector never learns a normal vehicle, making it rather hard to identify one at inference time. This can be seen in the AP drop on KITTI [13] from augmenting half of the cars, to all of them. Instead, augmenting a single object allowed to retain the same AP on KITTI, while significantly improving it on the out-of-domain Waymo [33] and the proposed CrashD.

Grouping strategies In Table 9 we show the impact of varying amounts of learned vector fields on the ASR, according to different distinguishing criteria. We compare the chosen relative rotation (Section 3.2) with selecting by distance of the object to the sensor or number of object points. Relative rotation delivered superior ASR, as it favors the mutual alignment between neighboring vectors. In contrast, less vector fields (i.e., 1 and 6) or different criteria resulted in contrasting vectors, reducing the object deformation.

Grouping	1-ASR	6-ASR	12-ASR	18-ASR
distance	55.1	56.2	57.3	57.6
nr. points	55.1	56.9	56.0	57.1
rel. rotation	55.1	59.2	63.4	63.7

Table 9. ASR \uparrow on the validation set of KITTI for different grouping strategies and amount of vector fields.

Aggregation strategies Table 10 shows the effect of different aggregation strategies of vectors when applying the deformations on the cars of KITTI [13]. It can be seen how the different amount of groupings (G) and neighboring vectors (k) considered for each point shift affected the adversarial performance of the method (ASR). In general, all deformations in the table were restricted to a maximum of $\epsilon = 30$ cm. The amount of learned vector fields G had an impact on the ASR of each aggregation strategy. For example, sum was more effective with 12 G than 1 G, since the vectors of the 12 fields were better aligned to each other than those of the single field (Section 4.2), so summing them increased

k	-	Aggregation						
		sum		average		distance		
G = 1	1	46.3	44.4	33.4	45.4	52.0	50.3	47.0
G = 12		59.6	76.5	80.3	61.9	62.4	63.4	59.6

Table 10. ASR \uparrow on the validation set of KITTI [13] for different aggregation strategies and number of neighbors (k) involved in each deformation, for both number of groups $G = 1$ and $G = 12$. All configurations are based on PointPillars [18].

the deformation magnitude. In fact, the high ASR of sum with 12 G , was due to larger perturbations.

Grid step size In Table 11 we show the impact of different step sizes t of the vector field grid. A larger step size, results in a coarser grid, which in turn means less vectors for each field. Intuitively, with more vectors, each would be more specific for a given point shift, but less generalizable to others. So, each vector would overfit to its training points. There is in fact a trade-off between the amount of vectors and the generalizability of the learned vector field, as seen in Table 2. That can be seen by the ASR, as the vectors were learned on the training set of KITTI [13], and applied to its validation set, on which the values are reported. Downscaling t from 20 to 5 cm, significantly reduced the ASR. Conversely, increasing t to 30 cm worsened their generalization. Therefore, $t = 20$ cm was chosen as the grid step size, offering a good trade-off between the vector specificity and generalizability, as shown by the ASR.

Step size	5 cm	10 cm	20 cm	30 cm
ASR \uparrow	44.1	46.3	53.0	49.6

Table 11. ASR \uparrow on the validation set of KITTI [13] for different step sizes of the vector field grid. A smaller step size increases the amount of vectors. All configurations are based on PointPillars [18], with $G = 1$.

A.4. Additional Outdoor Qualitative Results

In this section we provide qualitative results of the learned deformations.

A.4.1 Deformations on KITTI

Figure 11 shows the deformations learned by our method. It can be seen that only local areas are affected, and the cars preserved their overall shapes with smoothly deformed parts.

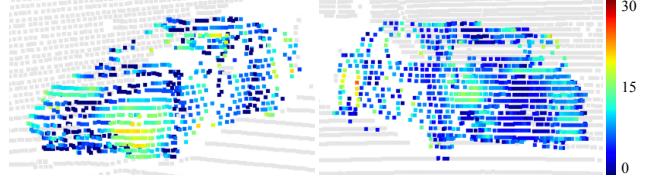


Figure 11. Color-coded deformations in cm learned by the proposed method. The perturbation does not affect every point, its magnitude is relatively low, and local smoothness is preserved.

Figure 12 shows the effect of each vector of the adversarial field to the ASR. It can be seen that the most affected was the front bumper, which can easily be deformed with an accident. The side of the car is mostly unaffected, probably due to the relatively limited amount of vehicles visible from the side in KITTI. Interestingly, the model has learned to avoid the areas without points (e.g., the windows).

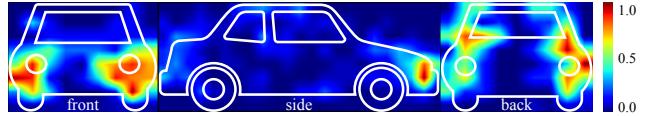


Figure 12. Color-coded contribution of each vector to the ASR, in percentage.

Figure 13 shows a comparison of the deformations applied by each method to a set of cars from KITTI [13]. We included related works, such as the Chamfer attack [19] and the iterative gradient L2 approach [40], as well as variations of the proposed 3D-VField. The Chamfer attack [19] shifted some points far away while many remained close to the original location, resulting in an almost perfect ASR. However, this came at the cost of rather obvious perturbations. The iterative gradient L2 [40] method also achieved a highly effective ASR (Table 1), but with significantly less evident deformations. As expected from the high ASR (Table 3), our unconstrained (unleashed) method delivered substantially perturbed objects, even more distorted than those produced by the Chamfer attack. Applying the ray constraint allowed for less perturbed (and less effective ASR), but more recognizable objects. It can be seen how this constraint alone impacts the realism of the deformations, by comparing it to the unleashed version. Moreover, aggregating neighboring vectors via distance weighting in our full approach (Section 3.2), further improved the resemblance of the object to the original point cloud. Although the difference is subtle, this can be appreciated comparing the rear wheel, the floor, and the windows of the car in the bottom half of Figure 13. Thanks to the realism and the smooth alterations of the points visible in the figure, training with our deformations allowed for superior transfer performance to

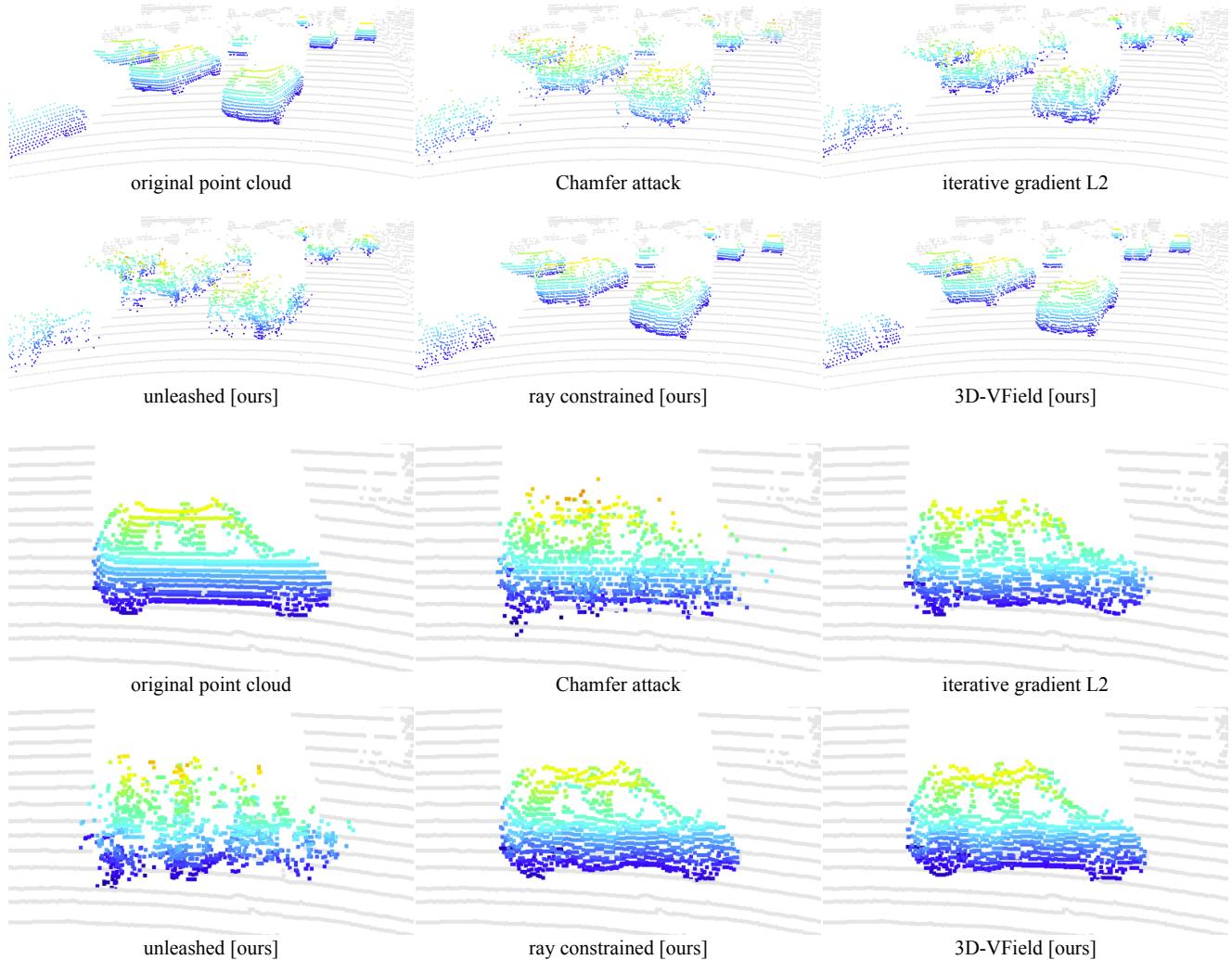


Figure 13. Comparison of adversarial perturbations on a set of cars from two different point clouds of KITTI [13]. The effect of the Chamfer attack [19], the iterative gradient L2 [40], and multiple variations of our approach are shown. It can be seen that our 3D-VField preserves the shape of the original point cloud better than the other approaches.

challenging out-of-domain data (Table 1).

A.5. Results on Indoor Data

A.5.1 Experimental setup

In these experiments we used the SUN RGB-D dataset [30], which posed a completely new set of challenges compared to the three driving datasets. SUN RGB-D contains indoor furniture objects captured by depth cameras such as time-of-flight (ToF), as opposed to driving scenes captured by a LiDAR. We trained on all 10 classes, but we selected one at a time for learning our vector fields. In particular, we report on the classes *bed*, *sofa* and the highly diverse *chair*, as they are the ones where deformations are more plausible compared to others (e.g., *table*). In this setting, we apply

our method on a VoteNet [25] architecture. Moreover, we followed the same setup as for the outdoor experiments, except that we reduced the maximum deformation ϵ to 10 cm, making it more plausible in indoor settings.

A.5.2 Quantitative Results

Table 12 shows the wide applicability of our deformation and augmentation strategies when applied to point clouds from depth sensors capturing furniture objects from SUN RGB-D [30]. Shifting the points with our 3D-VField produced a strong ASR against the not adversarially augmented models (none), especially on *sofas* and *chairs*. Using the deformations as augmentation even improved the AP on the validation set, confirming the benefit of our techniques to

Adv.aug.	<i>beds</i>		<i>sofas</i>		<i>chairs</i>	
	AP	ASR	AP	ASR	AP	ASR
none	85.6	49.7	67.4	70.6	77.4	70.9
w/o \mathcal{L}_{adv}	85.2	41.1	67.5	65.4	76.9	62.1
[ours]	86.0	19.7	68.5	34.8	77.5	39.6

Table 12. AP and ASR ↓ on the validation set of SUN RGB-D [30], with a VoteNet [25] architecture. Adv.aug.: adversarial augmentation; w/o \mathcal{L}_{adv} : ours not learned.

wards the generalization to unseen data, despite the rather different setting, sensor, objects, and architecture. Furthermore, defending with our adversarial augmentations significantly reduced the ASR, showing the gained robustness against deformed objects.

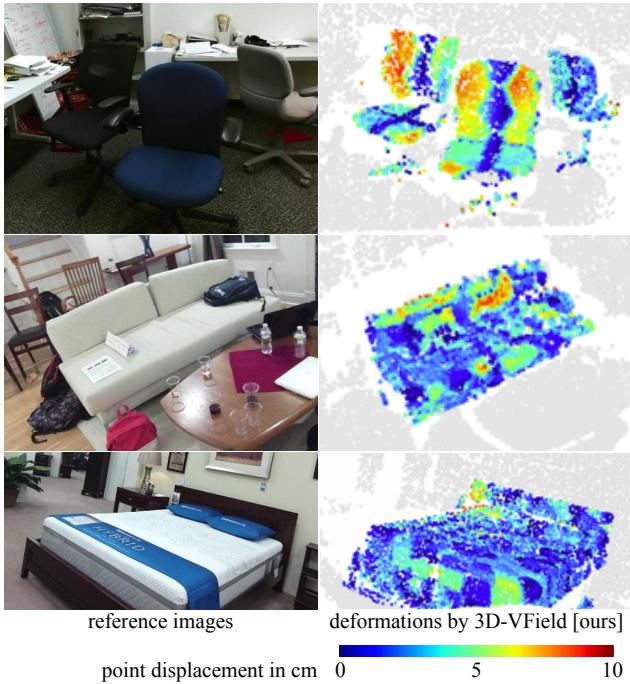


Figure 14. Color-coded deformations applied by the proposed 3D-VField on various objects of the SUN RGB-D dataset [30]. The color corresponds to the shift of each point in centimeters, limited to a maximum of 10 cm. Adversarial deformations learned against VoteNet [25].

A.5.3 Qualitative Results

In Figure 14 we show the deformations learned by our method against VoteNet [25] on three different categories of objects from SUN RGB-D [30], namely *chairs*, *sofas*, and *beds*. It can be seen that the overall shape of each ob-

ject is preserved, with minor perturbations applied. In this indoor setting, such alterations could resemble the presence of pillows, a blanket, or simply a different design of the object.

References

- Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. ADef: An iterative algorithm to construct adversarial deformations. In *Proceedings of the International Conference on Learning Representations*, 2019. 2, 3
- Isabela Albuquerque, Nikhil Naik, Junnan Li, Nitish Keskar, and Richard Socher. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv preprint arXiv:2003.13525*, 2020. 2
- Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in Neural Information Processing Systems*, 31:998–1008, 2018. 2
- Sara Beery, Yang Liu, Dan Morris, Jim Piavis, Ashish Kapoor, Neel Joshi, Markus Meister, and Pietro Perona. Synthetic examples improve generalization for rare classes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 863–873, 2020. 1, 2
- Daniel Bogdall, Jasmin Breitenstein, Florian Heidecker, Maarten Bieshaar, Bernhard Sick, Tim Fingscheidt, and Marius Zollner. Description of corner cases in automated driving: Goals and challenges. In *IEEE/CVF International Conference on Computer Vision Workshop*, pages 1023–1028, 2021. 1
- Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and LiDAR: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 176–194, 2021. 3
- Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on LiDAR-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2267–2281, 2019. 3
- Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint arXiv:1907.05418*, 2019. 3
- Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 39–57, 2017. 2
- MMDetection3D Contributors. MMDetection3D: Open-MMLab next-generation platform for general 3D object detection. <https://github.com/open-mmlab/mmdetection3d>, 2020. 5, 7
- Stefano Gasperini, Jan Haug, Mohammad-Ali Nikouei Mahani, Alvaro Marcos-Ramiro, Nassir Navab, Benjamin Busam, and Federico Tombari. CertainNet: Sampling-free uncertainty estimation for object detection. *IEEE Robotics and Automation Letters*, 7(2):698–705, 2021. 1, 2

- [12] Stefano Gasperini, Patrick Koch, Vinzenz Dallabetta, Nassir Navab, Benjamin Busam, and Federico Tombari. R4Dyn: Exploring radar for self-supervised monocular depth estimation of dynamic scenes. In *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, pages 751–760, 2021. 1
- [13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1, 2, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16
- [14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2015. 2
- [15] Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. AdvPC: Transferable adversarial perturbations on 3D point clouds. In *Proceedings of the European Conference on Computer Vision*, pages 241–257. Springer, 2020. 3
- [16] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021. 1, 2
- [17] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021. 1, 2
- [18] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1, 5, 6, 7, 8, 12, 13, 14, 15
- [19] Daniel Liu, Ronald Yu, and Hao Su. Adversarial shape perturbations on 3D point clouds. In *Proceedings of the European Conference on Computer Vision*, pages 88–104. Springer, 2020. 2, 3, 5, 6, 7, 8, 12, 15, 16
- [20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations*, 2018. 3, 4
- [21] Pascale Maul, Marc Mueller, Fabian Enkler, Eva Pigova, Thomas Fischer, and Lefteris Stamatogiannakis. BeamNG.tech technical paper, 2021. 5, 9, 10
- [22] Jisoo Mok, Byunggoon Na, Hyekjun Choe, and Sungroh Yoon. AdvRush: Searching for adversarially robust neural architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12322–12332, 2021. 1, 2
- [23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582. IEEE, 2016. 2
- [24] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European Symposium on Security and Privacy*, pages 372–387, 2016. 2
- [25] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep Hough voting for 3D object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 5, 16, 17
- [26] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020. 1, 2
- [27] Martin Rabe, Stefan Milz, and Patrick Mader. Development methodologies for safety critical machine learning applications in the automotive domain: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 129–141, 2021. 1
- [28] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 5, 6, 7, 12, 13, 14
- [29] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Elisa Ricci, and Peter Kortschieder. Towards generalization across depth for monocular 3D object detection. In *Proceedings of the European Conference on Computer Vision*, pages 767–782. Springer, 2020. 2
- [30] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015. 2, 5, 16, 17
- [31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2
- [32] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1262–1270, 2019. 2
- [33] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 2, 5, 6, 8, 9, 10, 11, 12, 14
- [34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014. 2
- [35] James Tu, Mengye Ren, Sivabal Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for LiDAR

- object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13713–13722, 2020. [1](#), [2](#), [3](#), [4](#), [5](#)
- [36] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 5339–5349, 2018. [2](#)
- [37] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4627–4635, 2021. [1](#), [2](#), [9](#)
- [38] Run Wang, Felix Juefei-Xu, Qing Guo, Yihao Huang, Xiaofei Xie, Lei Ma, and Yang Liu. Amora: Black-box adversarial morphing attack. In *Proceedings of the ACM International Conference on Multimedia*, pages 1376–1385, 2020. [3](#)
- [39] 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. [1](#), [2](#), [4](#), [5](#), [6](#), [7](#)
- [40] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3D adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9128–9136, 2019. [2](#), [3](#), [5](#), [6](#), [7](#), [8](#), [12](#), [15](#), [16](#)
- [41] Chaowei Xiao, Bo Li, Jun-yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating Adversarial Examples with Adversarial Networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3905–3911, July 2018. [2](#)
- [42] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, Oct. 2018. [5](#), [6](#), [7](#), [12](#), [13](#), [14](#)
- [43] Jiancheng Yang, Qiang Zhang, Rongyao Fang, Bingbing Ni, Jinxian Liu, and Qi Tian. Adversarial attack and defense on point sets. *arXiv preprint arXiv:1902.10899*, 2019. [3](#), [5](#), [6](#), [7](#), [12](#)
- [44] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019. [2](#)
- [45] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *Proceedings of the International Conference on Learning Representations*, 2021. [2](#)