3D-VField: Learning to Adversarially Deform Point Clouds for Robust 3D Object Detection

Alexander Lehner[*,○,1,2] Stefano Gasperini[*,1,2] Alvaro Marcos-Ramiro[2] Michael Schmidt[2]
Mohammad-Ali Nikouei Mahani[3] Nassir Navab[1,4] Benjamin Busam[1] Federico Tombari[1,5]

[1] Technical University of Munich [2] BMW Group [4] Johns Hopkins University [5] 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 on out-of-distribution 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 taking into account deformed point clouds during training. We achieve this with 3D-VField: a novel method that plausibly deforms objects via vectors 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 transferrable, sample-independent and preserve shape smoothness and occlusions. By augmenting normal samples with the deformations produced by these vector fields during training, we significantly improve robustness against differently shaped objects, such as damaged/deformed cars, even while training only on KITTI. Towards this end, we propose and share open source 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 high generalizability of our techniques to out-of-domain data, different models and sensors, namely LiDAR and ToF cameras, for both indoor and outdoor scenes.

1. Introduction

With the established wide-spread progress of learning-based methods tackling a variety of perception tasks (e.g., 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 [28]. This has led to the pursuit of improving the model robustness against out-of-domain data, which can naturally occur in the real world [17]. Such approaches include domain adaptation [40], uncertainty estimation [12], simulations [4], and adversarial perturbations [36].

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, 12, 16, 27], this is still mostly unexplored for 3D point clouds [36], also due to the

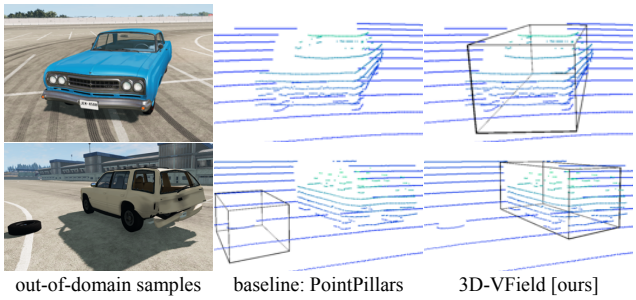

Figure 1. Predictions of PointPillars [18] trained on KITTI [13], without and with our adversarial augmentations on out-of-domain samples from the proposed CrashD dataset. CrashD comprises rare (top) and damaged (bottom) vehicles, resulting in natural adversarial examples [17]. As the models were transferred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

[*] The authors contributed equally.
[○] Contact author: Alexander Lehner (*alexander.lehner@tum.de*).
[3] Work done while at BMW Group.

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, 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 [36] have tackled this problem to improve the generalization to out-of-distribution data. However, despite being effective attacks, existing adversarial deformation strategies [20, 41] are sample-specific, lack wide-applicability, and by being designed without considering a 3D sensor, are mostly unconstrained in space [20].

In this work, we substantially improve the generalization capability of 3D object detectors to out-of-distribution data, bridging this gap by deforming point clouds during training. We propose 3D-VField: a novel adversarial 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-distribution data.
- 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 [34], our CrashD and SUN RGB-D [31], show the wide applicability of our approach.

2. Related Work

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

2.1. Improving Generalization

Generalization to unseen data is a highly desirable property for any learning-based approach [38]. Unseen data includes any samples on which a model has not been trained on, comprising both out-of-distribution 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 [38], in contrast to domain adaptation which has access to the target data [40]. 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 [32]. Other works explored regularization [3] or estimating the model uncertainty [12] to improve domain generalization. Moreover, specific architectures can be found via search algorithms to improve robustness [23].

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, 33, 46] also belong to this category. Among these, there are adversarial training approaches, which add altered inputs learned in an adversarial fashion as a way to improve generalization [27, 36, 37].

The method we propose in this work belongs to the data category, and 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. [30] 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. [36] improved the generalization towards cars with roof-mounted objects, via adversarial examples on LiDAR point clouds. Wang et al. [40] 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, 35]. An adversarial training incorporates these examples into the training data, improving the robustness against such inputs. Although a variety of works explored adversarial examples in the image domain [9, 24, 25, 42, 45], where pixel perturbations imperceptible to humans are able to fool the target model, this

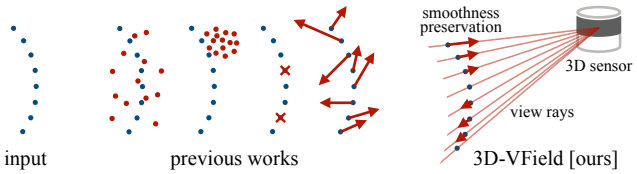

Figure 2. Adversarial deformations introduced by previous works, compared to ours. Other methods add, drop or move points with minor constraints. Ours only slides points along the view ray, while preserving shape smoothness.

topic is still mostly unexplored on point clouds, especially those captured by 3D sensors (e.g., LiDAR, ToF camera). Alaifari et al. [1] deformed images using a different adversarial vector field learned for each sample. Wang et al. [39] proposed adversarial morphing fields to alter image pixels spatially and fool classifiers.

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. [41] 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. [36] 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 [44].

Perturbation Xiang et al. [41] 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 [21], 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 perceptibilty of the attack. [20]. The CD is measured by averaging over the sum 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. [20] 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, 20, 41], 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 [20, 41], that were not intended to improve the generalization of the models, but rather set the basis for adversarial attacks on point clouds [41]. Additionally, these mechanisms are sample-specific, making their applicability limited [15, 20, 41]. Instead, Tu et al. [36] 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, 20, 39, 41], as we construct a single highly transferrable and generic set of perturbations. As we aim to improve the generalization to out-of-distribution samples, ours is similar to that of Tu et al. [36], but 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., as we do not make 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 shape variations, thereby improving the generalization of 3D object detectors on out-of-domain data. 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 shape smoothness 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

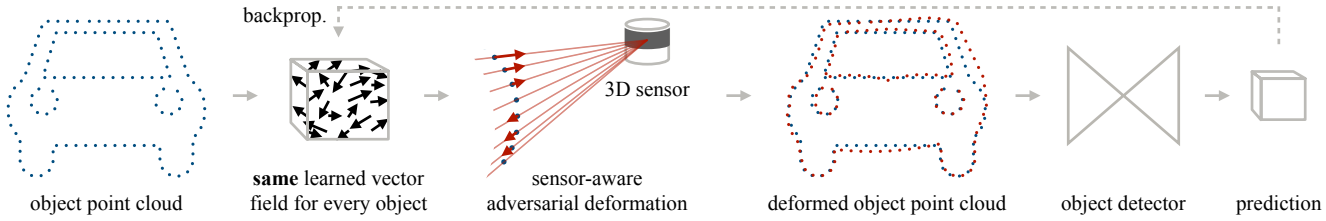

Figure 3. Overview of the proposed 3D-VField. We first learn a vector field adversarially, then apply it to plausibly deform objects, taking constraints into account. The modified scenes are then used as augmentations to improve the generalization to unseen object shapes.

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_v with a step size s to obtain root coordinates $\boldsymbol{c}$ in 3D space, and assign an empty vector $\boldsymbol{v}$ to each root. B_v 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 following Tu et al. [36]. This reduces the prediction confidence score c of a relevant box y from the set of relevant proposals $\mathcal{Y}$ of a 3D object detector, according to its IoU with the ground truth y^*:

$$\mathcal{L}_{\text{adv}} = \sum_{y,c \in \mathcal{Y}} -\text{IoU}(y^*, y) \log(1 - c). \tag{1}$$

where we consider a proposal y relevant if $c > 0.1$.

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 $\mathcal{L}_{\text{adv}}$ smoothly converges, the performance of the detector, against which the vector field is optimized, decreases. Once trained, the vectors can be used as 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 preserve the sensor physical constraints when generating deformations and therefore help generalization, we employ a simple sensor model in which the 3D points can only be moved across the optical ray. We first compute the ray $\boldsymbol{l}_i$ between the 3D sensor and each point $\boldsymbol{p}_i$, which determines the deformation direction for each point. Then we calculate the deformation vectors $\boldsymbol{r}_i$, for each $\boldsymbol{p}_i$ by projecting its nearest neighbor $\boldsymbol{v}_i$ onto the ray $\boldsymbol{l}_i$. Points are therefore only moved by $\boldsymbol{r}_i$.

Regularizing the deformations We limit the perturbation of the points by restricting the vectors with $\|\boldsymbol{v}\|_\infty < \epsilon$ following the standard PGD L_∞ attack [21]. 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 $\boldsymbol{p}_i$ of the object and its nearest vector $\boldsymbol{v}_{ij}$ from the vector field. The final shift $\boldsymbol{s}_i$ of each point is calculated by weighting the deformation vectors $\boldsymbol{r}_{ij}$ with their corresponding distance d_{ij}:

$$\boldsymbol{s}_i = \frac{\sum_{j=1}^{k} d_{ij} \boldsymbol{r}_{ij}}{k} \tag{2}$$

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-distribution car shapes, such as vehicles from a different country [40]. 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

Method	KITTI AP *easy*	KITTI AP *mod.*	KITTI AP *hard*	KITTI ASR ↑	KITTI deformation CD ↓	KITTI deformation SR ↓	→ Waymo AP	→ CrashD AP *normal* *clean*	→ CrashD AP *normal* *crash*	→ CrashD AP *rare* *clean*	→ CrashD AP *rare* *crash*
baseline [18]	**88.24**	77.11	74.55	-	-	-	40.86	65.20	43.67	34.14	22.48
iter. grad. L2 [41]	86.24	76.92	73.84	*95.9	0.14	1.36	39.86	58.65	41.86	35.92	23.69
Chamfer att. [20]	87.15	77.05	74.07	***99.8**	0.30	**0.39**	40.54	56.84	39.56	36.29	24.73
[ours] not learned	86.81	76.36	68.80	10.1	**0.07**	0.85	41.62	62.94	44.06	33.19	21.75
[ours] unleashed	87.11	76.82	73.68	97.7	0.24	1.48	40.95	60.43	46.03	39.30	27.55
[ours] ray constr.	87.06	76.35	68.89	59.5	0.13	0.82	41.03	59.82	44.60	39.92	29.16
[ours] 3D-VField	87.05	**77.13**	**75.55**	63.4	0.12	0.76	**44.61**	**67.95**	**52.87**	**43.40**	**30.37**

Table 1. Evaluation of methods trained on KITTI [13] towards out-of-distribution data (without any fine-tuning), namely Waymo validation set [34] and our CrashD datasets, as well as on the KITTI validation set. All methods are based on PointPillars [18], and all except the baseline use adversarial examples (on which ASR, CD and SR are measured) for data augmentation, resulting in the reported APs. →: transfer from KITTI. *: being sample-specific, the deformation had to be trained on the validation set of KITTI.

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 [34], and the proposed synthetic CrashD, which we introduce below. Additionaly, we apply our method also on the indoor SUN RGB-D dataset [31], 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 common *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, we showed the wide-applicability of our techniques with the **SUN RGB-D** dataset. This posed a completely new set of challenges compared to the three driving datasets: indoor furniture objects captured by depth cameras such as time-of-flight (ToF). We trained on all 10 classes, but we selected one at a time for adversarial training. 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*).

CrashD dataset To quantify the generalizability on out-of-distribution samples, we produced a synthetic dataset named CrashD. As this includes a variety 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 [22] 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 with settings mimicking 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 open source, as an out-of-distribution evaluation benchmark for models trained on KITTI [13], Waymo [34] 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 an 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. [36] using the **attack success rate** (ASR) metric. It measures the percentage of objects that become false negatives after undergoing a perturbation. For the ASR, we consider an object detected if its IoU > 0.7. To measure the average deformation applied to each object we use the **Chamfer distance** (CD), computed between the original and the deformed point clouds.

Surface roughness metric Neighboring points of an object captured through a sensor usually do not differ too much in depth when the underlying surface is smooth. As deformations change that, we propose a new metric: the surface roughness (SR), measuring the maximal displacement of neighboring points compared to their original locations with respect to the 3D sensor. Be o an object with n points

→ CrashD			*normal, linear*			*normal, t-bone*			*rare, linear*			*rare, t-bone*		
			light	*mod.*	*hard*	*light*	*mod.*	*hard*	*light*	*mod.*	*hard*	*light*	*mod.*	*hard*
AP	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**
	crash	baseline [18]	46.5	33.8	28.6	57.9	54.9	40.2	26.7	22.9	15.4	31.2	23.3	15.4
		3D-VF [ours]	**54.3**	**46.6**	**40.6**	**65.3**	**60.2**	**50.2**	**33.4**	**31.0**	**21.5**	**41.7**	**33.0**	**22.1**

Table 2. Detailed evaluation of PointPillars [18] on our CrashD dataset according to the various accident types, and intensities, as well as the kinds of car. Comparison of the baseline with our 3D-VField (3D-VF).

and $\boldsymbol{t}_{ij}$ and $\boldsymbol{t}'_{ij}$ the line between two points p_i, p_j, p'_i, p'_j in the original point cloud and the deformed one respectively. The roughness is calculated by averaging over the maximum of the set of angles $A_5(\boldsymbol{t}_i, t'_i)$ between the lines of each point p_i and p'_i to its 5 nearest neighbors. Therefore the SR of an object is calculated as:

$$SR_o = \frac{1}{n}\sum_{i=1}^{n} \max A_5(\boldsymbol{t}_i, \boldsymbol{t}'_i) \tag{3}$$

Network architectures We use 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 [43] voxelizes the point cloud and uses a learned voxel feature encoding. Part-A^2 Net [29] is an extension of PointRCNN that also predicts intra-object part locations for improved accuracy. VoteNet [26] is based on PointNet++ and Hough voting. While the first three models are mostly used for autonomous driving, VoteNet is used in indoor settings.

Implementation details For our experiments, we construct each vector field B_v with $w = 1.8$m, $h = 1.6$m, $l = 4.6$m and a step size of $s = 20$cm resulting in 1656 vectors per vector field. If not stated otherwise, we group objects by relative rotations with $G = 12$ groups, and set $N = 6$. During the perturbation stage, we move points according to their $k = 2$ nearest vectors and deform only along the sensor ray. For the PGD optimization, we use Adam with a learning rate of 0.05. The distance threshold is set to $\epsilon = 30$cm. Each vector is randomly initialized form a uniform distribution with values between -1cm and 1cm. We trained all models using PyTorch and MMDetection3D [11] on a single NVIDIA Tesla V100 32GB GPU.

Prior works and baseline We focus on object detection and compare with other adversarial perturbations methods. For the LiDAR outdoor experiments, all methods are based on PointPillars [18], unless otherwise noted. We used PointPillars as baseline, the iterative gradient L2 [41] and the Chamfer attack [20] as point perturbation methods. For a fair comparison we trained all on the same KITTI dataset split [10], with $\epsilon = 30$cm, then we perturbed the point clouds as data augmentation with the same settings as ours (i.e., random selection of one object per scene to augment).

4.2. Quantitative Results

Comparison with related methods Table 1 shows the comparison between our 3D-VField and related approaches. In particular, we report other adversarial perturbation methods, such as the iterative gradient L2 [41] and the Chamfer attack [20], as well as the baseline PointPillars [18]. In terms of ASR, our approach is not as strong as the iterative gradient L2 [41] and the Chamfer attack [20] However, this is expected as ours are sample-independent vector fields, compared to their sample-specific point-to-point deformations. Due to this reason, their perturbations had to be learned directly on the **KITTI** validation set, on which the ASR was measured. Interestingly, the adversarial training of our 3D-VField did not reduce the overall AP performance compared to the strong baseline, while bringing numerous benefits. As demonstrated by Wang et al. [40], 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, for KITTI and Waymo respectively, as well as the 50% higher point density and the narrower field of view [34]. This test assesses the quality of the generated deformations with respect to real vehicle shapes found in a different country. Our 3D-VField delivered more than 9% relative improvement over the strong baseline, and the other perturbation methods, proving the benefit of our added sensor-awareness on real and challenging out-of-domain data. On the right of Table 1 we report

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 3. ASR ↑ on the validation set of KITTI for different grouping strategies and amount of vector fields.

Augment.	PointP. [18] AP	PointP. [18] ASR	Second [43] AP	Second [43] ASR	Part-A^2 [29] AP	Part-A^2 [29] 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 the transferrability of our deformations, as well as the efficacy of our adversarial augmentaions, on the validation set of KITTI. ASRs on Second and Part-A^2 are measured on vector fields trained on the defended PointPillars, to report the transferrability. w/o $\mathcal{L}_{adv}$: ours not learned.

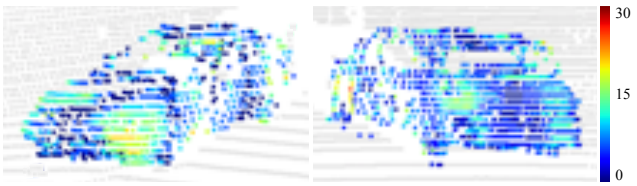

Figure 4. Color-coded deformations in cm learned by the proposed method. The perturbation does not affect every point, its magnitude is relatively low, and local smoothness is preserved.

the results on **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, the detection performance drops, showing the effort required to the methods to relate these to the cars learned on KITTI. Similarly, with rare cars (i.e., old and sports cars), the AP drops even more, quantifying the domain shift from normal vehicles. Nevertheless, our method improved significantly over the strong baseline and the other approaches. This can be attributed to our adversarial deformations providing a good trade-off between the degree of deformation (CD) and the roughness of the resulting surfaces (SR), 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.

Influence of types of crashes and vehicles In Table 2 we compare the AP transfer performance from KITTI to CrashD of our 3D-VField with the strong baseline [18] across various kinds of damages, with different intensities and types of cars. Our adversarial augmentation strategy outperformed PointPillars [18] across the board by a significant margin, especially on rare cars. In particular, with high intensity crashes (hard), the baseline [18] severely underperformed, reducing by half its AP 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 *crashed* counterparts. Interestingly, *rare* vehicles were more challenging to be detected than *normal damaged* 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.

Transferrability to other 3D detectors 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 [43] and Part-A^2 [29], 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.

Ablation study on deformation constraints As we introduced the sensor-awareness and the surface smoothness constraints to our deformations, we investigate the impact of these in terms of generalization to out-of-domain data. At the bottom of Table 1 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 (not learned) can already be a beneficial augmentation technique, as it improved the transfer to Waymo. Instead, removing all constraints, but learning the vector fields (unleashed) delivers a strong ASR of 97.7%, while producing rough and largely displaced deformations (high CD and SR). This significantly increases the AP on the CrashD *rare* cars. When deforming with sensor-awareness (ray-constr.), ASR, SR and CD reduce, but the AP on the most difficult transfer settings (i.e., *rare damaged*) improved. Our full model 3D-VField, adds the distance smoothing (Section 3.2) which further reduces SR, while 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.

Effect of number of vectors Our method learns only 1656 3D vectors to perturb objects as data augmentation. However, training with $G = 12$ and $N = 6$, the amount of vectors increases to 120K. Conversely, the sampling-based iterative gradient L2 [41] and the Chamfer [19] attacks required 10.9M and 12.6M vectors for training and validation

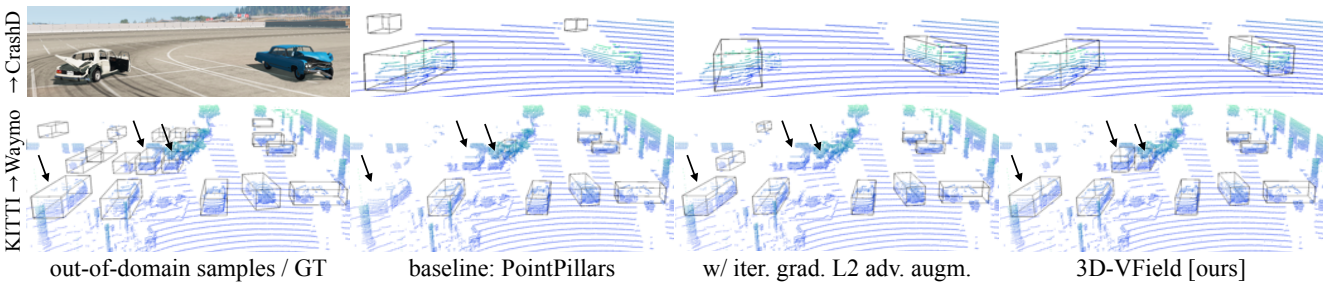

Figure 5. Predictions on challenging out-of-domain samples from the proposed CrashD (top) and Waymo [34] (bottom). Models based on PointPillars [18] trained on KITTI (without fine-tuning). Iterative gradient L2 [41] and ours trained with adversarial augmentation. *Image used with courtesy of BeamNG GmbH.*

sets respectively. This shows how easily applicable ours is, compared to theirs. In Table 3 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 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 result in contrasting vectors, hence reduced deformations on the object.

Augment.	*beds* AP	*beds* ASR	*sofas* AP	*sofas* ASR	*chairs* AP	*chairs* 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 5. AP and ASR ↓ on the validation set of SUN RGB-D [31], with a VoteNet [26] architecture. w/o $\mathcal{L}_{adv}$: ours not learned.

Indoor settings Table 5 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 [31]. Shifting the points with our 3D-VField produced a strong ASR, especially on *sofas* and *chairs*. Using the deformations as augmentation even improved the validation set AP, confirming the benefit of our techniques towards the generalization to unseen data, despite the rather different setting, sensor, objects, and model.

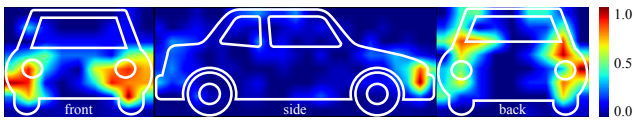

Figure 6. Color-coded contribution of each vector to the ASR.

4.3. Qualitative Results

In Figure 5 we compare the transfer predictions from KITTI to CrashD and Waymo [34] of the standard PointPillars [18], augmented with ours and the iterative gradient L2 adversarial approach [41], 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 the baseline, and our 3D-VField outperforming 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, the baseline ignored 3 recognizable cars with a high amount of points, the iterative gradient L2 missed 2 of them and detected 2 further ones, albeit with misaligned boxes. Instead, despite missing further ones, our method was able to recognize the visible cars.

Figure 4 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 6 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).

5. Conclusion

In this paper we presented 3D-VField: an adversarial deformation 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 that can be used as data augmenta-

tion. 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-distribution data, including various kinds of damaged cars.

References

[1] Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. ADef: An iterative algorithm to construct adversarial deformations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. 3

[2] 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

[3] 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

[4] 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

[5] Daniel Bogdoll, 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

[6] 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 42nd IEEE Symposium on Security and Privacy (IEEE S&P 2021)*, May 2021. 3

[7] 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, London United Kingdom, Nov. 2019. ACM. 3

[8] 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

[9] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, May 2017. ISSN: 2375-1207. 2

[10] 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. 6

[11] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 6, 15

[12] 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. *arXiv preprint arXiv:2110.01604*, 2021. 1, 2

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, pages 3354–3361. IEEE, 2012. 1, 2, 5, 12, 13, 14, 15, 16, 17, 18

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2

[15] Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. AdvPC: Transferable adversarial perturbations on 3D point clouds. In *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, 15, 16, 17

[19] Daniel Liu, Ronald Yu, and Hao Su. Extending Adversarial Attacks and Defenses to Deep 3D Point Cloud Classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283, Sept. 2019. ISSN: 2381-8549. 7, 15, 18

[20] Daniel Liu, Ronald Yu, and Hao Su. Adversarial shape perturbations on 3D point clouds. In *European Conference on Computer Vision*, pages 88–104. Springer, 2020. 2, 3, 5, 6, 16

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. 3, 4

[22] Pascale Maul, Marc Mueller, Fabian Enkler, Eva Pigova, Thomas Fischer, and Lefteris Stamatogiannakis. BeamNG.tech technical paper, 2021. 5, 12, 13

[23] Jisoo Mok, Byunggook Na, Hyeokjun 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. 2

[24] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2574–2582. IEEE Computer Society, 2016. 2

[25] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, Mar. 2016. 2

[26] 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. 6, 8, 19

[27] 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

[28] 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

[29] 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. 6, 7, 15, 16

[30] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Elisa Ricci, and Peter Kontschieder. Towards generalization across depth for monocular 3D object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 767–782. Springer, 2020. 2

[31] 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, 8, 19

[32] 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

[33] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE, 2019. 2

[34] 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, 12, 13, 14, 15, 16

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*, Feb. 2014. arXiv: 1312.6199. 2

[36] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically Realizable Adversarial Examples for LiDAR Object Detection. In *CVPR*, pages 13713–13722, Seattle, WA, USA, June 2020. IEEE. 1, 2, 3, 4, 5

[37] 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 32nd International Conference on Neural Information Processing Systems*, pages 5339–5349, 2018. 2

[38] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4627–4635. ijcai.org, 2021. 2, 12

[39] 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 28th ACM International Conference on Multimedia*, pages 1376–1385, 2020. 3

[40] 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, 6

[41] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3D Adversarial Point Clouds. In *CVPR*, pages 9128–9136, Long Beach, CA, USA, June 2019. IEEE. 2, 3, 5, 6, 7, 8, 15, 16, 18, 19

[42] Chaowei Xiao, Bo Li, Jun-yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating Adversarial Examples with Adversarial Networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 3905–3911, Stockholm, Sweden, July 2018. International Joint Conferences on Artificial Intelligence Organization. 2

[43] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, Oct. 2018. 6, 7, 15, 16

[44] 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

[45] 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, Sept. 2019. Conference Name: IEEE Transactions on Neural Networks and Learning Systems. 2

[46] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 2

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, Section A.3 reports more quantitative results, and Section A.4 shows deformations of both indoor and outdoor point clouds.

A.1. Details on the Proposed Dataset: CrashD

In this section we further describe the proposed dataset: CrashD. We refer the reader to the supplementary video to see 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-distribution data. It is meant to serve as a test benchmark for 3D detectors trained on KITTI [13], Waymo [34], 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 [38], 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 [22], 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), 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.

Figure 7. 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.*

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 8. 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 7. We considered a crash visible if the sensor was within 35

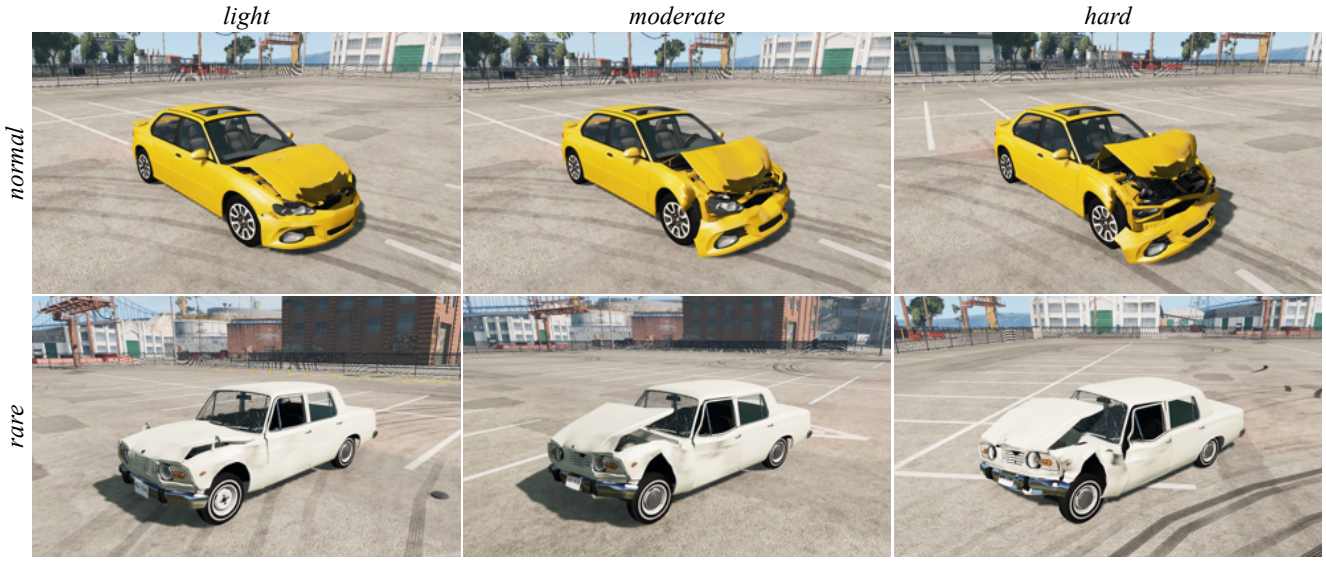

Figure 8. Comparison of *linear* damage intensities for *normal* and *rare* cars of CrashD. For each type of car, the accidents were created by the same hitting vehicle, coming from the same angle. It can be seen that the *hard* crash compromised the structure of the weaker *rare* car, while the *normal* car absorbed the impact differently, leaving the cabin unchanged. *Images used with courtesy of BeamNG GmbH.*

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 [22] 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. Towards this end, we positioned the vehicles from 10 to 40 meters away from the LiDAR, around its front 180 degrees. As shown in Figure 7, 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 [34], 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 9, 10 and 11. In particular, the 12 *normal* cars used are shown in Figure 9, 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 [34] respectively. Figure 10 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 10).

Figure 9. *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 were popular LiDAR datasets, such as KITTI [13] and Waymo [34], have been recorded. *Images used with courtesy of BeamNG GmbH.*

Figure 10. *Rare* cars of CrashD. These were classified as *rare* as they complement the *normal* (i.e., common) cars shown in Figure 9. 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. *Images used with courtesy of BeamNG GmbH.*

The significant gap between the two types of cars can be seen by comparing the *normal* and *rare* vehicles in Figures 9 and 10 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 8, 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 11. Hitting vehicles also included all the ones shown in Figure 9, as well as those in Figure 10. However, we excluded the 2 cars marked with *. These 2 were excluded 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 were part of the detectable vehicles.

Figure 11. 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 transferring from KITTI. *Images used with courtesy of BeamNG GmbH.*

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 damaged. 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 [41] we minimize our adversarial loss $\mathcal{L}_{\text{adv}}$ while constraining the deformation $\boldsymbol{v}$ for each point $\boldsymbol{p}$ with $\|\boldsymbol{v}\|_2 < \epsilon$, 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:

$$\mathcal{C}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y||_2 \tag{4}$$

for two sets X and Y. To learn the deformations, we minimized:

$$\mathcal{L}_{\text{cam}} = \mathcal{L}_{\text{adv}} + \lambda \mathcal{C}(\boldsymbol{p} + \boldsymbol{v}, \boldsymbol{p}) \tag{5}$$

with λ set to 0.1 and the amount of deformation constrained by $\mathcal{C}(\boldsymbol{p} + \boldsymbol{v}, \boldsymbol{p}) < \epsilon$, 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.

Transfer to Waymo To evaluate the transfer of the models from KITTI [13] to Waymo [34], 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 Quantitative Results

A.3.1 Generalization of Different 3D Detectors

Table 6 shows the detection performance of Part-A^2 [29] and Second [43] trained on KITTI [13] with and without our adversarial augmentation techniques, and transferred to both Waymo [34] and the proposed CrashD. Their APs are reported alongside the one of PointPillars [18], which were already shown in the main paper. Part-A^2 resulted in a significantly stronger and more robust 3D detector compared to PointPillars and Second. In fact, the performance of the base Part-A^2 did not degrade as much as the base models of PointPillars and Second, when transferring to Waymo, or the proposed CrashD. Furthermore, Part-A^2 showed a superior ability to detect challenging *rare* cars of CrashD, compared to PointPillars, nearly doubling the AP for both *clean* and *crash*. This can be attributed to Part-A^2 objects part-awareness and part-aggregation stages [29], which might have set the focus of the model on the most relevant parts to identify cars within LiDAR point clouds, such as wheels, ground clearance, general shape with bonnet and cabin, as well as the relationship between these parts. The performance of Second [43] on KITTI [13] is lower than the one reported in [11], as we could not reproduce the AP reached in [11], despite using the same settings and framework. This reduced performance affected both the baseline and our approach. Nevertheless, incorporating our adversarial deformations into the training pipeline, significantly improved the generalization of all 3 object detectors when transferring to out-of-domain data. This confirms again the wide-applicability and transferrability of our techniques, since the vector fields used for deforming the objects were only trained against PointPillars [18].

Detailed transfer to CrashD In Table 7, we show a more detailed evaluation of the various 3D object detectors along the different sub-categories of the proposed CrashD. Again, the values confirm the superiority of Part-A^2 [29] over the other 3D detectors. Comparing the same cars with and without damages (crash and clean) shows that the latter are significantly more difficult for every detector, due to the different resulting shapes. Moreover, applying our adversarial deformations trained against PointPillars [18] to the other 3D detectors, substantially increased their robustness

Method		KITTI AP *easy*	KITTI AP *mod.*	KITTI AP *hard*	→ Waymo AP	→ CrashD AP *normal* *clean*	→ CrashD AP *normal* *crash*	→ CrashD AP *rare* *clean*	→ CrashD AP *rare* *crash*
PointPillars [18]	baseline [18]	**88.24**	77.11	74.55	40.86	65.20	43.67	34.14	22.48
	iter. grad. L2 [41]	86.24	76.92	73.84	39.86	58.65	41.86	35.92	23.69
	Chamfer att. [20]	87.15	77.05	74.07	40.54	56.84	39.56	36.29	24.73
	3D-VField [ours]	87.05	**77.13**	**75.55**	**44.61**	**67.95**	**52.87**	**43.40**	**30.37**
Part-A^2 [29]	baseline [29]	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**	**56.08**	**88.80**	**73.80**	**81.10**	**61.34**
Second [43]	baseline [43]	**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	**43.51**	**76.54**	**60.51**	**47.47**	**36.14**

Table 6. Comparison of models trained on KITTI [13] and evaluated on the validation set of KITTI, as well as out-of-distribution samples from the Waymo validation set [34] and the proposed CrashD (without fine-tuning). This table is an extension of Table 1, as it includes various 3D object detector, namely PointPillars [18], Part-A^2 [29] and Second [43].

→ CrashD			*normal, linear* *light*	*normal, linear* *mod.*	*normal, linear* *hard*	*normal, t-bone* *light*	*normal, t-bone* *mod.*	*normal, t-bone* *hard*	*rare, linear* *light*	*rare, linear* *mod.*	*rare, linear* *hard*	*rare, t-bone* *light*	*rare, t-bone* *mod.*	*rare, t-bone* *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**
	crash	baseline [18]	46.5	33.8	28.6	57.9	54.9	40.2	26.7	22.9	15.4	31.2	23.3	15.4
		3D-VF [ours]	**54.3**	**46.6**	**40.6**	**65.3**	**60.2**	**50.2**	**33.4**	**31.0**	**21.5**	**41.7**	**33.0**	**22.1**
Part-A^2 [29]	clean	baseline [29]	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**
	crash	baseline [29]	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**
Second [43]	clean	baseline [43]	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**
	crash	baseline [43]	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**

Table 7. Detailed AP comparison of PointPillars [18], Part-A^2 [29] and Second [43] 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. This table is an extension of Table 2.

to differently shaped vehicles, such as the ones in the proposed out-of-distribution CrashD.

Correct and wrong detections on CrashD Table 8 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].

→ CrashD		IoU 0.1 baseline [18]	IoU 0.1 3D-VF [ours]	IoU 0.5 baseline [18]	IoU 0.5 3D-VF [ours]	IoU 0.7 baseline [18]	IoU 0.7 3D-VF [ours]
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 8. 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.

G	aggregation	k	ASR ↑	CD ↓	SR ↓
1	-	1	46.32	0.1316	0.69
	sum	2	44.43	0.1117	0.68
	sum	3	33.35	**0.1020**	**0.60**
	average	2	45.35	0.1258	0.66
	average	3	**51.98**	0.1260	**0.60**
	distance	2	50.32	0.1272	0.66
	distance	3	46.97	0.1234	0.61
12	-	1	59.59	0.1255	0.77
	sum	2	76.45	0.1451	0.86
	sum	3	**80.34**	0.1599	0.89
	average	2	61.87	0.1209	0.67
	average	3	62.37	0.1178	**0.63**
	distance	2	63.37	0.1214	0.76
	distance	3	59.57	**0.1170**	0.65

Table 9. ASR ↑ CD ↓ and SR ↓ 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].

A.3.2 Ablation Studies

In Tables 9 and 10 we report two additional ablation studies.

Aggregation strategies Table 9 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 ϵ =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 than those of the single field (Section 4.2), so summing them increased the deformation magnitude. In fact, the high ASR of sum with 12 G, came at the cost of higher CD and SR, resulting in larger and less smooth perturbations. Average resulted as a valid alternative to the chosen distance weighting (Section 3.2), offering a good trade-off between ASR and degree of deformation (CD and SR). However, distance aggregation was chosen for its stronger ASR when paired to 12 G and 2 neighbors.

Step size	ASR ↑	CD ↓	SR ↓
5 cm	44.09	**0.11**	0.98
10 cm	46.26	0.12	0.80
20 cm	**52.99**	0.13	0.67
30 cm	49.58	0.13	**0.64**

Table 10. ASR ↑ CD ↓ and SR ↓ on the validation set of KITTI [13] for different step sizes of the vector fields grids. A smaller step size increases the amount of vectors. All configurations are based on PointPillars [18], with G=1.

Grid step size In Table 10 we show the impact of different step sizes s of the vector field grid. A larger step size, results in a coarser grid, which in turn means less vectors

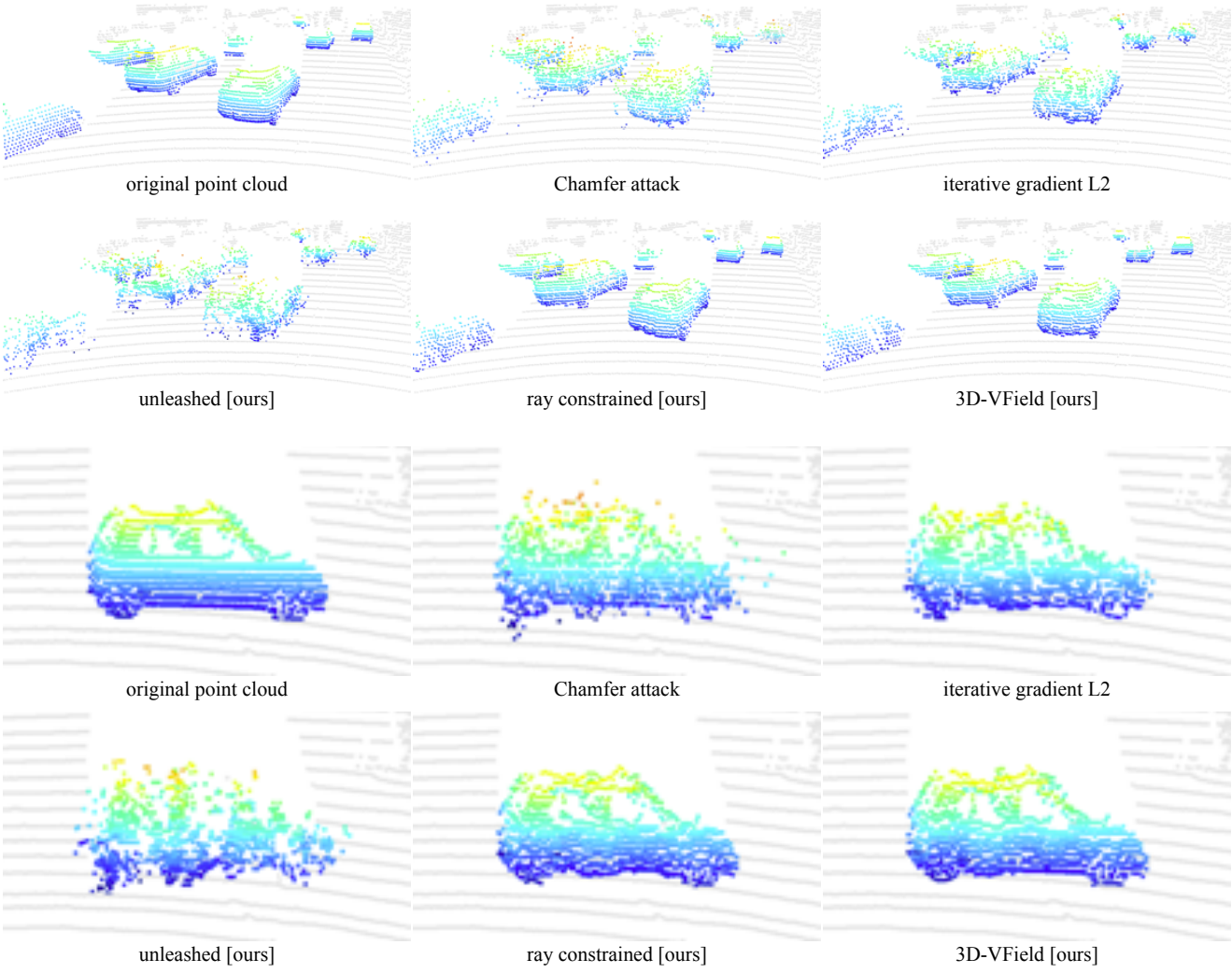

Figure 12. Comparison of adversarial perturbations on a set of cars from two different point clouds of KITTI [13]. The effect of the Chamfer attack [19], the iterative gradient L2 [41], and multiple variations of our approach are shown. It can be seen that our 3D-VField preserves the shape of the original point cloud better than the other approaches.

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. 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. Down-scaling s from 20 to 5 cm, significantly reduced the ASR. Conversely, increasing s to 30 cm made the deformations smoother (lower SR), but worsened their generalization via the ASR. Therefore, s =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.

A.4. Additional Qualitative Results

In this section we provide qualitative results of the deformations applied by each adversarial approach. We refer the reader to the supplementary video for more comparisons of the 3D detection performance when using our method as data augmentation on Waymo and CrashD.

A.4.1 Outdoor: Deformations on KITTI

Figure 12 shows a comparison of the deformations applied by each method to a set of cars from KITTI [13]. We included both related works, such as the Chamfer attack [19] and the iterative gradient L2 approach [41], as well as variations of the proposed 3D-VField. The Chamfer attack [19] shifted some points far away while many remained close to

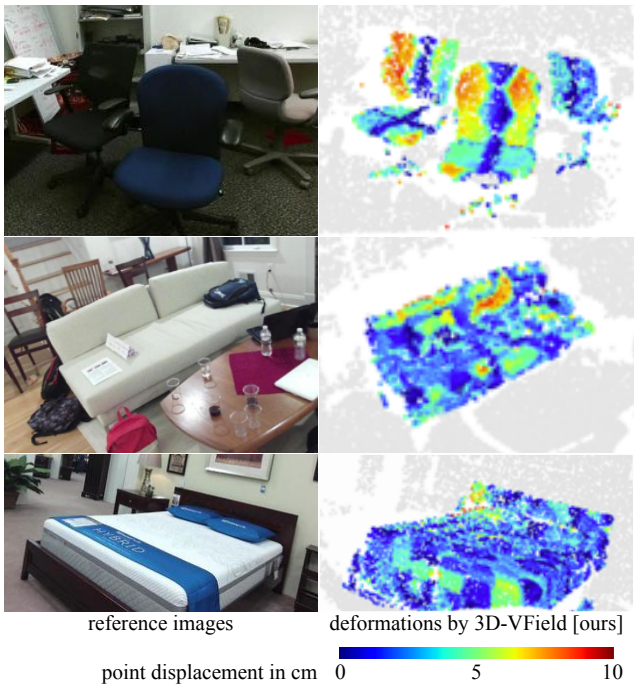

Figure 13. Color-coded deformations applied by the proposed 3D-VField on various objects of the SUN RGB-D dataset [31]. The color corresponds to the shift of each point in centimeters, limited to a maximum of 10. Adversarial deformations learned against VoteNet [26].

the original location, resulting in an almost perfect ASR. Since the SR is computed as an average over all points, the Chamfer attack shows the lowest roughness, despite producing rather obvious perturbations. The iterative gradient L2 [41] method also achieved a highly effective ASR (Table 1), but with significantly less evident deformations. As expected from the high SR (Table 1), 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 12. 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 challenging out-of-domain data (Table 1).

A.4.2 Indoor: Deformations on SUN RGB-D

In Figure 13 we show the deformations learned by our method against VoteNet [26] on three different categories of objects from SUN RGB-D [31], namely *chairs*, *sofa*, and *bed*. It can be seen that the overall shape of each object 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.