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

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

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

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 [1], Waymo [8], 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 [9], 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.

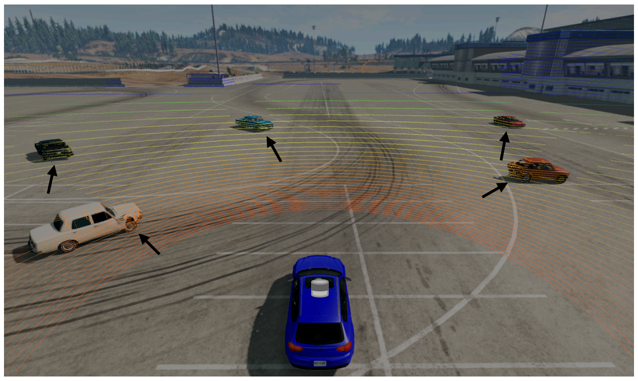
Figure 6. LiDAR scene setup of CrashD. For each car, a black arrow indicates its damaged area, which is ensured to be visible from the sensor viewpoint. *Image used with courtesy of BeamNG GmbH.*

A.1.2 Driving Simulator

CrashD was generated using a driving simulator developed by BeamNG [4], 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 [1].

* The authors contributed equally.
◦ Contact author: Alexander Lehner (*alexander.lehner@tum.de*).

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

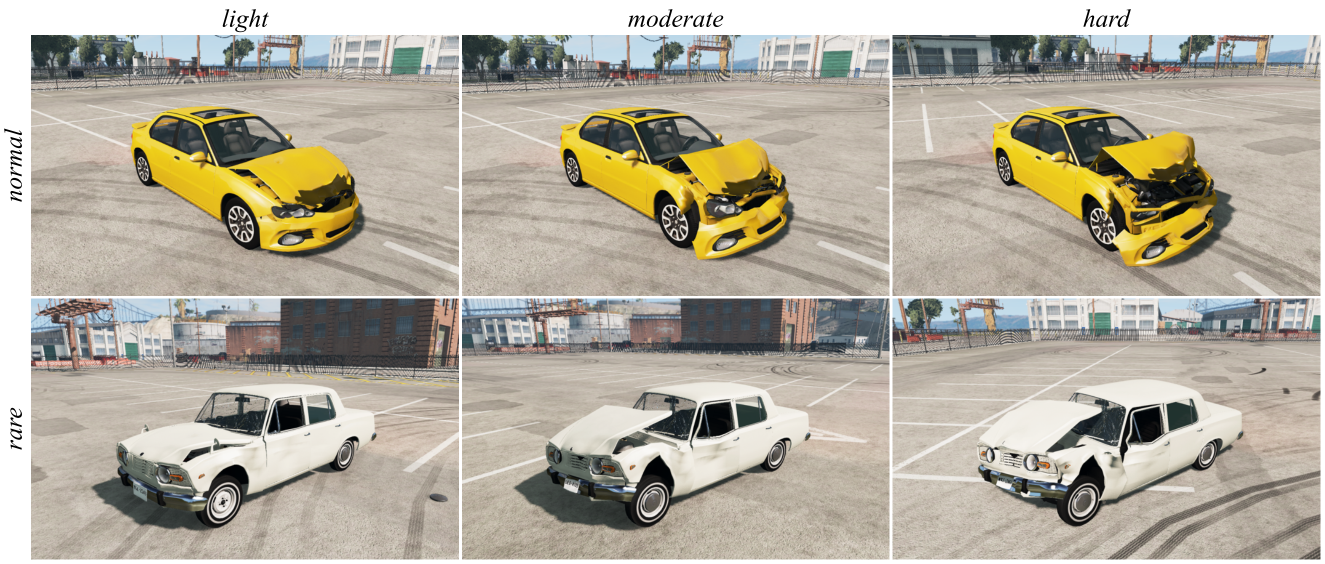

Figure 7. Comparison of *linear* damage intensities for *normal* and *rare* cars of CrashD. For each type of car, the accidents were created by the same hitting vehicle, coming from the same angle. It can be seen that the *hard* crash compromised the structure of the weaker *rare* car, while the *normal*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

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.

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 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 [4] 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

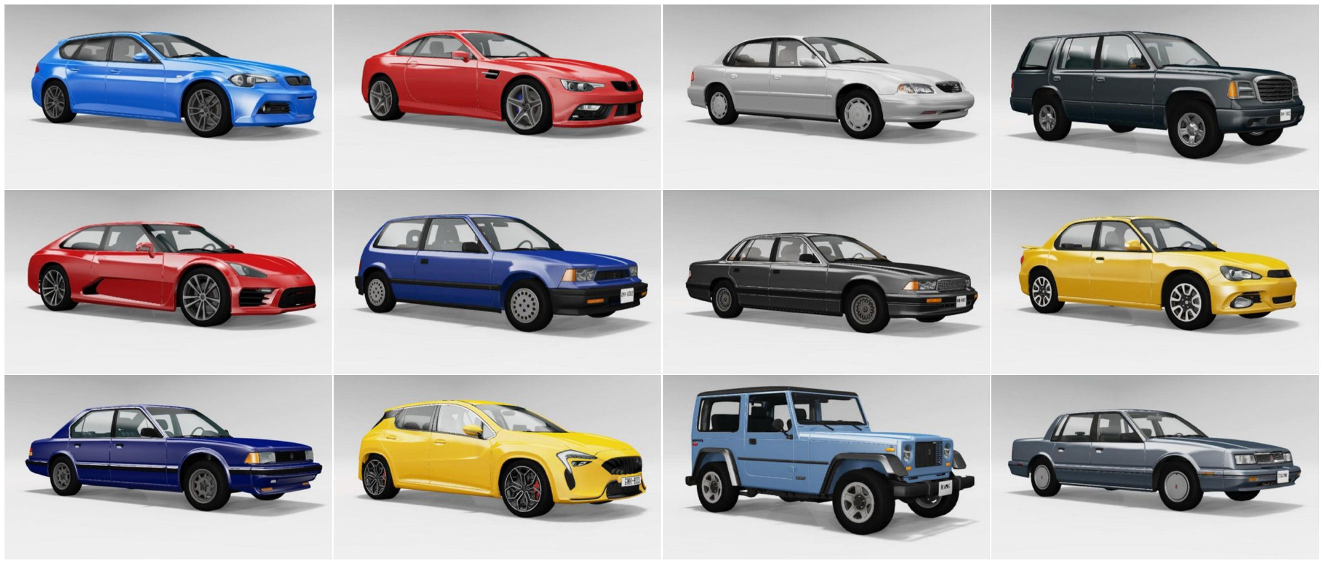
Figure 8. *Normal* cars of CrashD. These were classified as *normal* as they resemble the vast majority of cars on the road today in Germany, USA, and other locations where popular LiDAR datasets, such as KITTI [1] and Waymo [8], have been recorded.

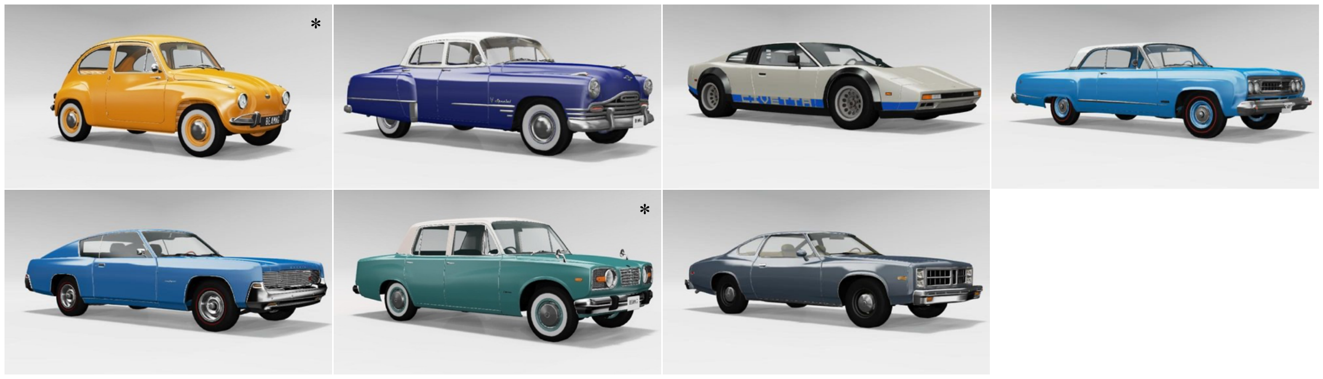

Figure 9. *Rare* cars of CrashD. These were classified as *rare* as they complement the *normal* (i.e., common) cars shown in Figure 8. In particular, *rare* ones resemble old cars from various regions, and also include a wedge-shaped sports car. * indicates cars that cannot hit other vehicles (due to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

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 [1], 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 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 [1] to Waymo [8], 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

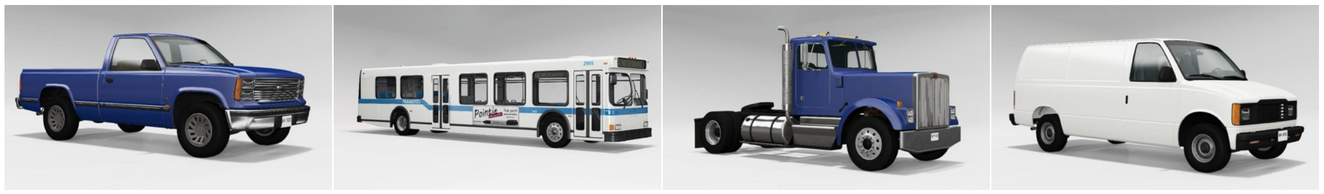

Figure 10. These vehicles can only hit others and are not detectable objects, as they do not fit the KITTI [1] criteria for being a car, so they would not get recognized by a model transferred from KITTI.

vast majority of vehicles on the road today in the countries where common LiDAR datasets were recorded, such as Germany and USA, for KITTI [1] and Waymo [8] 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).

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 [1] *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 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 [10] we minimize our adversarial loss $\mathcal{L}_{\text{adv}}$ constraining the deformation $\boldsymbol{m}$ for each point $\boldsymbol{p}$ with $||\boldsymbol{m}||_2 < \epsilon$, with $\epsilon = 30$ cm.

Chamfer attack For the Chamfer attack [3] we used the Chamfer distance to measure the gap between the original and perturbed point clouds, which is given by:

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

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

$$\mathcal{L}_{\text{cha}} = \mathcal{L}_{\text{adv}} + \lambda \mathcal{C}(\boldsymbol{p} + \boldsymbol{m}, \boldsymbol{p}) \tag{2}$$

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

Adversarial removal For the removal attack we follow [12] 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 [10].

Adversarial generation We follow [10] 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 [10] solely on the added points. Thus shifting them to decrease the detection quality.

Transfer to Waymo To evaluate the transfers to Waymo [8], 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

	PointP. [2]		Second [11]		Part-A^2 [6]	
Adv.aug.	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-A^2 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 [2] are highly effective also on rather different architectures such as Second [11] and Part-A^2 [6], 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. [2]	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.

In Table 5 we report the performance of PointPillars [2] 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 [2, 6, 11] across the board by a significant margin, especially on *rare* cars. In particular, with high intensity crashes (*hard*), the baselines [2, 6, 11] 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 [2]. The values also confirm the superiority of Part-A^2 [6] over the other 3D detectors, as seen in Table 1.

Correct and wrong detections on CrashD Table 7 reports a comparison of PointPillars [2] 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 [2] 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 [2].

→ CrashD			*normal, linear* *light*	*mod.*	*hard*	*normal, t-bone* *light*	*mod.*	*hard*	*rare, linear* *light*	*mod.*	*hard*	*rare, t-bone* *light*	*mod.*	*hard*
PointP. [2]	clean	baseline [2]	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 [2]	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**
Second [11]	clean	baseline [11]	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 [11]	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 [6]	clean	baseline [6]	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 [6]	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 [2], Second [11], and Part-A^2 [6] trained on KITTI [1] 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 baseline [2]	3D-VF [ours]	IoU 0.5 baseline [2]	3D-VF [ours]	IoU 0.7 baseline [2]	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 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 [2], trained on KITTI [1] and transferred to CrashD without any fine-tuning. For reference, the total amount of cars in CrashD is 46936.

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 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 [1] 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 [8] 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 dis-

# augm. objects	KITTI *mod.*	→ W.	→ CrashD *n.,clean*	*r.,crash*
[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; →: transfer without any fine-tuning; W.: Waymo; *n.*: *normal*; *r.*: *rare*.

tance 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.

Aggregation strategies Table 10 shows the effect of different aggregation strategies of vectors when applying the deformations on the cars of KITTI [1]. 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 the deformation magnitude. In fact, the high ASR of sum with 12 G, was due to larger perturbations.

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

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 [1], and applied to

	Aggregation						
	-	sum		average		distance	
k	1	2	3	2	3	2	3
$G = 1$	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 ↑ on the validation set of KITTI [1] 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 [2].

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 ↑	44.1	46.3	**53.0**	49.6

Table 11. ASR ↑ on the validation set of KITTI [1] for different step sizes of the vector field grid. A smaller step size increases the amount of vectors. All configurations are based on PointPillars [2], 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 a comparison of the deformations applied by each method to a set of cars from KITTI [1]. We included related works, such as the Chamfer attack [3] and the iterative gradient L2 approach [10], as well as variations of the proposed 3D-VField. The Chamfer attack [3] 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 [10] 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

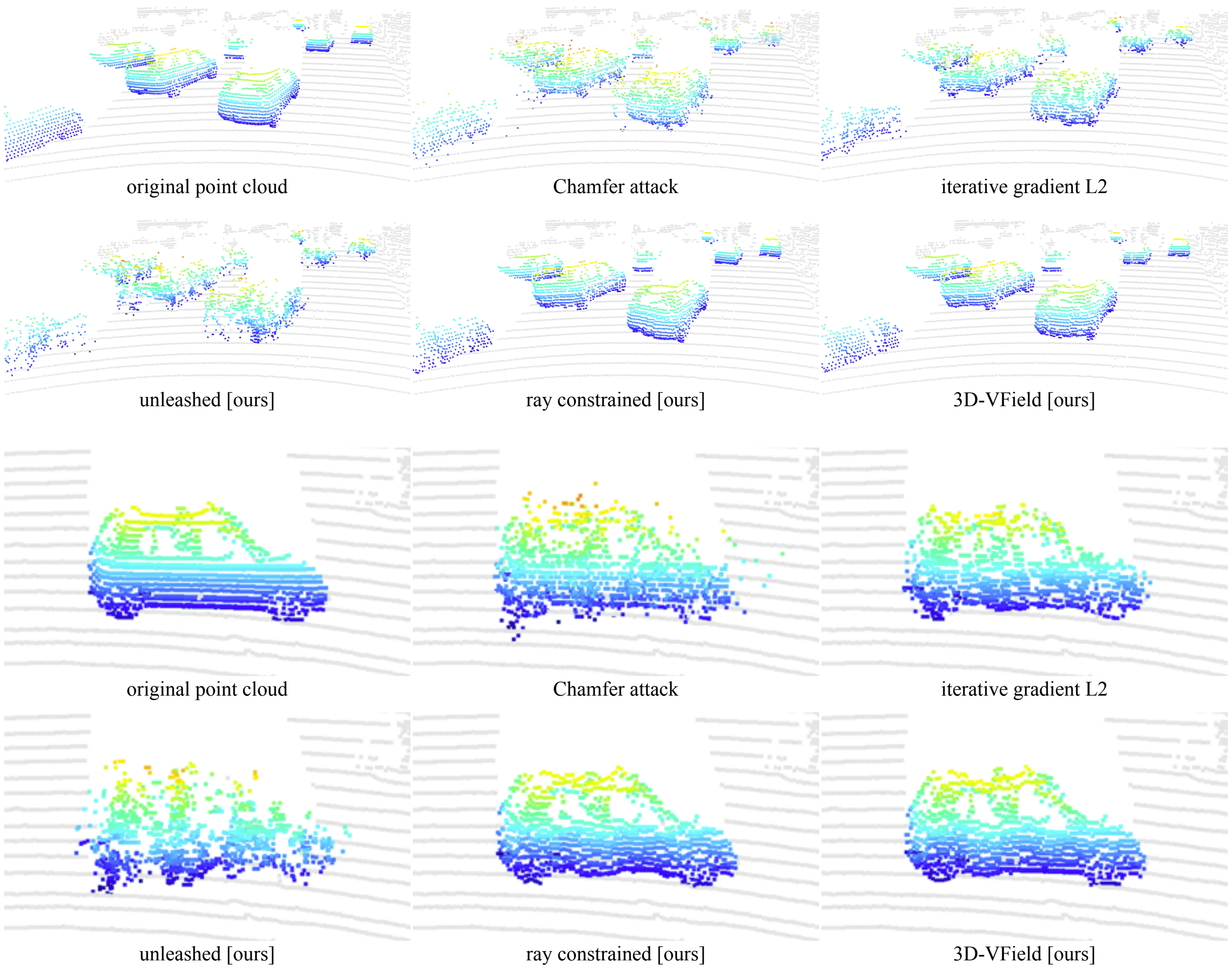

Figure 11. Comparison of adversarial perturbations on a set of cars from two different point clouds of KITTI [1]. The effect of the Chamfer attack [3], the iterative gradient L2 [10], and multiple variations of our approach are shown. It can be seen that our 3D-VField preserves the shape of the original point cloud better than the other approaches.

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 11. 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).

Figure 12 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 13 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, prob-

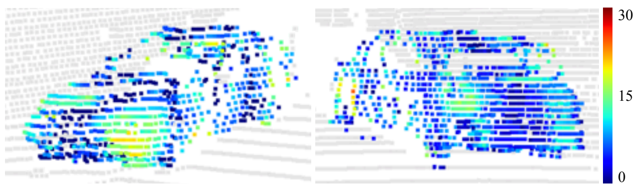

Figure 12. Color-coded deformations in cm learned by the proposed method. The perturbation does not affect every point, its magnitude is relatively low, and local smoothness is preserved.

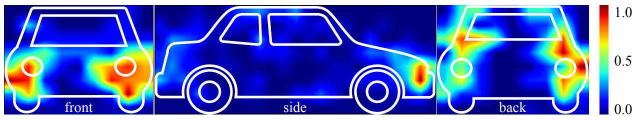

Figure 13. Color-coded contribution of each vector to the ASR, in percentage.

ably 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).

A.5. Results on Indoor Data

A.5.1 Experimental setup

In these experiments we used the SUN RGB-D dataset [7], 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 [5] 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

	beds		*sofas*		*chairs*	
Adv.aug.	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 $\downarrow$ on the validation set of SUN RGB-D [7], with a VoteNet [5] architecture. Adv.aug.: adversarial augmentation; w/o $\mathcal{L}_{adv}$: ours not learned.

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 [7]. 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 towards 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.

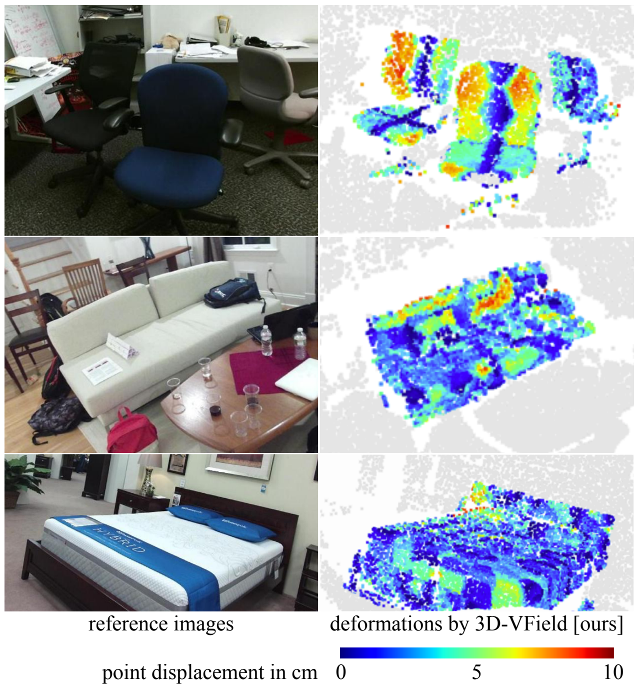

Figure 14. Color-coded deformations applied by the proposed 3D-VField on various objects of the SUN RGB-D dataset [7]. The color corresponds to the shift of each point in centimeters, limited to a maximum of 10 cm. Adversarial deformations learned against VoteNet [5].

A.5.3 Qualitative Results

In Figure 14 we show the deformations learned by our method against VoteNet [5] on three different categories of objects from SUN RGB-D [7], namely *chairs*, *sofas*, and *beds*. 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.

References

[1] 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, 3, 4, 6, 7, 8

[2] 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. 5, 6, 7

[3] 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. 4, 7, 8

[4] Pascale Maul, Marc Mueller, Fabian Enkler, Eva Pigova, Thomas Fischer, and Lefteris Stamatogiannakis. BeamNG.tech technical paper, 2021. 1, 2

[5] 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. 9

[6] 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] 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. 9

[8] 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. 1, 3, 4, 6

[9] 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

[10] 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. 4, 7, 8

[11] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, Oct. 2018. 5, 6

[12] 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. 4