A. Robust Fusion Benchmark

We publish the robust fusion dataset, benchmark, data format and instructions at our website https://anonymous-benchmark.github.io/robust-benchmark-website.

Benchmark documentation. We show the dataset documentation and intended uses in https://anonymous-benchmark.github.io/robust-benchmark-website/documentation.html.

Benchmark maintenance. We provide data download links (Google Drive & Baidu YunPan) for users in https://anonymous-benchmark.github.io/robust-benchmark-website/download.html. We will maintain the data for a long time and check the data accessibility in a regular basis.

Benchmark and code. The codebases used in our benchmark are open-source. See our GitHub repository for more details in https://github.com/anonymous-benchmark/lidar-camera-robust-benchmark.

Case Summary. In Fig. 2, we visualize a common design of the autonomous driving perception system, whose main components include the camera and LiDAR sensor, and an on-device computer. Based on our experience, each step of the aforementioned system can encounter certain failures or disruptions, and yield noisy data that are drastically different from the normal clean data. We identify three categories of cases and briefly discuss the potential reasons and consequences in Tab. 5.

Autonomous driving datasets. Our robustness benchmarks nuScenes-R and Waymo-R are generated from two large-scale popular datasets for autonomous driving, nuScenes and Waymo.

nuScenes is a large-scale autonomous-driving dataset for 3D detection, consisting of 700, 150 and 150 scenes for training, validation, and testing, respectively. Each frame contains one point cloud and six calibrated images that cover 360 fields-of-view. For 3D detection, the main metrics are mean Average Precision (mAP) and nuScenes detection score (NDS). The mAP is defined by the BEV center distance with thresholds of 0.5m, 1m, 2m, 4m, instead of the IoUs of bounding boxes. NDS is a consolidated metric of mAP and other metric scores, such as average translation error and average scale error.

Waymo Open Dataset is an another large-scale dataset for autonomous driving, which contains 798 training, 202 validation, and 150 testing sequences. Each sequence has about 200 frames with LiDAR points and camera images, which are collected by five LiDAR sensors and five pinhole cameras. The official metrics are mean Average Precision (mAP) and mean Average Precision weighted by Heading (mAPH). The mAP and mAPH are defined based on the 3D IoU with the threshold of 0.7 for vehicles and 0.5 for pedestrians and cyclists. The measures are reported based on the distances from objects to sensor, i.e., 0-30m, 30-50m and >50m, respectively. Besides, two difficulty levels, LEVEL 1 (boxes with more than five LiDAR points) and LEVEL 2 (boxes with at least one LiDAR point), are considered.

Broader Impacts Statement and Limitations. This paper studies robust LiDAR-camera fusion for 3D object detection. Since the detection explored in this paper is for generic objects and does not pertain to specific human recognition, so we do not see potential privacy-related issues.

This research composes a toolkit that can transform an autonomous driving dataset into a robustness benchmark by simulating real-world noisy data cases. However, the generated simulated data and real data still have some gaps. To overcome the limitation, we plan to collect and provide noisy data cases in realistic scenarios, which requires long-term efforts as they do not often occur. And though the main purpose of this work is to create a robustness benchmark, we nonetheless provide a simple method, which fine-tunes the model on these robustness scenarios, and show that it moderately improves the robustness of current methods. However, there is still a large performance gap when compared to the results of the clean settings, which demonstrates that the autonomous vehicles desire more effective and reliable robust fusion methods or training strategies.

B. Implementation Details

In order to make the experiment results reproducible, we further provide detailed experimental settings for each method in this paper. The models are mostly trained and evaluated on servers with 8 NVIDIA GeForce RTX 3090 GPUs.

B.1. Benchmark Setup

We summarized an overview of our final benchmark setup in Tab. 6. Note that, we only investigate one failure case at a time, and do not create a robust benchmark that has multiple malfunctions at the same time, if more than one setting is considered, the mean performance is calculated as Eq. (1). And we only simulate noisy data cases by altering the image and LiDAR data, the ground-truth annotation will remain the same as the 3D position of the object in the surrounding worlds will not change when the sensors malfunction. And note that both LiDAR and camera modality can encounter the temporal misalignment issue.

B.2. More Implementation Settings

Here we detail some implementation settings.

CenterPoint. Following the original paper [59], we implement CenterPoint using VoxelNet [57, 67, 68] encoder. For experiments on nuScenes, we set the detection range to $[-51.2m, 51.2m]$ for the X and Y axis, and $[-5m, 3m]$ for Z axis. Voxel size is set as $(0.1m, 0.1m, 0.2m)$.

Group	Reason	Consequent Case
Noisy LiDAR	Damaged LiDAR sensors	Missing of corresponding point inputs
	Installation limitation of LiDAR	Limited LiDAR field-of-view
	Low reflection rate of objects	LiDAR object failure
Noisy camera	Damaged camera sensors	Missing of corresponding image inputs
	Camera lens occlusion	Lens Occlusion
Ill-synchronization	Vibration during driving	Spatial misalignment
	Loose physical mounting	Spatial misalignment
	Instability of on-device computer	Temporal misalignment
	Temporary insufficient cable bandwidth	Temporal misalignment
	Sensor connection failure	Temporal misalignment

Table 5. **Common reasons for noisy data cases**. Based on the realistic experiences, we report various reasons that cause the noisy data cases.

Group	Noise	Setting
Noisy LiDAR	Limited LiDAR FOV	Keep the points that satisfy $\theta \in (-\pi/3, \pi/3)$
	LiDAR Object Failure	Randomly drop the points within a bounding box with a probability of 0.5
Noisy camera	Missing of Camera Inputs	Drop front camera & only keep front camera
	Occlusion of Camera Lens	Paste a random mask on each image
Ill-synchronization	Spatial Misalignment	Add random rotation (1° to 5°) and translation (0.5cm to 1.0cm) noise
	Temporal Misalignment	Stuck 50% frames (discrete selection & consecutive selection)

Table 6. **An overview of the final benchmark setup.**

And the model is trained for 20 epochs. The waymo model uses a detection range of $[-75.2m, 75.2m]$ for the X and Y axis, and $[-2m, 4m]$ for the Z axis. Voxel size is kept as $(0.1m, 0.1m, 0.15m)$. The model is trained for 12 epochs. Codebase: https://github.com/open-mmlab/mmdetection3d/tree/master/configs/centerpoint

DETR3D. Following the original paper [54], the model consists of a ResNet [10] feature extractor, a FPN, and a DETR3D detection head. We use ResNet101 with deformable convolutions [8] in the 3rd stage and 4th stage. The FPN [25] takes features output by the ResNet and produces 4 feature maps whose sizes are 1/8, 1/16, 1/32, and 1/64 of the input image sizes. On the nuScenes dataset, the model is trained for 12 epochs in total. On the Waymo dataset, we made some small changes to transfer the model, e.g, Waymo has only five camera inputs, and the model is trained for 24 epochs. Codebase: https://github.com/WangYueFt/detr3d

PointAugmenting. PointAugmenting proposes a new multi-modal data augmentation. In this paper, we only focus on the robustness of the fusion method. Thus, when training PointAugmenting, we do not use the extra multi-modal data augmentation mentioned in PointAugmenting. On the nuScenes dataset, we use PointPillars [16] as the LiDAR stream and DLA34 [62] as the image stream. We use the pretrained DLA34 model from CenterTrack [65]. On the Waymo dataset, we use PointPillars [16] as the LiDAR stream and ResNet50 [10] with FPN [25] as the image stream. We only use the P_2 feature from FPN as the image feature. The image feature extractor is trained for 36 epochs on Waymo of the 2D detection task. Codebase: https://github.com/VISION-SJTU/PointAugmenting

MVX-Net. Following the original paper [42], we use PointPillars [16] as the LiDAR stream and ResNet50 [10] with FPN [25] as the image stream. At the fusion stage, we project each LiDAR point to all images from different views to acquire the corresponding image features from the deep networks. Then, we average the features on the channel dimension and then concatenate all features from different views with the original LiDAR point feature before the downstream task. On the nuScenes dataset, we set the detection region of interest to $[-50m, 50m]$ for the X and Y axis, and $[-5m, 3m]$ for the Z axis. The pillar size is kept as $[0.25m, 0.25m]$. The image feature extractor is trained for 36 epochs on nuImage [3]. On Waymo dataset, we set the detection range to $[-74.88m, 74.88m]$ for the X and Y axis, and $[-2m, 4m]$ for the Z axis. The pillar size is hold as $[0.32m, 0.32m]$. The image fea-

Approach	Modality	nuScenes-R (mAP / NDS)						
		P_C	10%	30%	50%	70%	90%	100%
CenterPoint [59]	L	56.8 / 65.0	56.6 / 64.9	55.9 / 64.4	54.7 / 63.7	52.4 / 62.2	45.6 / 58.1	28. 4/ 48.5
PointAugmenting [50]	LC	46.9 / 55.6	46.8 / 55.5	46.3 / 55.2	45.0 / 54.3	43.2 / 52.7	36.0 / 49.1	21.3 / 39.4
MVX-Net [42]	LC	61.0 / 66.1	60.8 / 65.9	60.2 / 65.6	59.1 / 65.1	57.1 / 63.9	51.3 / 60.6	34.0 / 51.1
TransFusion [1]	LC	66.9 / 70.9	66.7 / 70.7	66.1 / 70.4	65.0 / 69.7	62.8 / 68.4	56.1 / 64.4	34.6 / 53.6

Table 7. **Results of different dropping proportion setting of the LiDAR object failure case**.

Approach	Modality	nuScenes-R (mAP / NDS)						
		P_C	0.1	0.3	0.5	0.7	0.9	1.0
CenterPoint [59]	L	56.8 / 65.0	50.2 / 61.4	40.4 / 56.0	28.4 / 48.5	18.3 / 41.6	8.9 / 31.9	5.0 / 23.7
PointAugmenting [50]	LC	46.9 / 55.6	38.7 / 50.6	30.0 / 45.2	21.3 / 39.4	13.6 / 33.7	5.1 / 27.9	2.3 / 24.0
MVX-Net [42]	LC	61.0 / 66.1	55.7 / 63.3	44.4 / 57.4	34.0 / 51.1	23.3 / 44.3	12.8 / 36.5	7.0 / 29.5
TransFusion [1]	LC	66.9 / 70.9	59.9 / 67.1	46.5 / 60.0	34.6 / 53.6	21.2 / 43.6	12.9 / 36.0	7.3 / 28.3

Table 8. **Results of different dropping probability setting of the LiDAR object failure case**.

ture extractor is trained for 36 epochs on Waymo of the 2D detection task. Codebase: https://github.com/open-mmlab/mmdetection3d/tree/master/configs/mvxnet

TransFusion. Following the original paper [1], we implemented TransFusion. For nuScenes, we use the DLA34 [62] of the pretrained CenterNet as the 2D backbone and keep its weights frozen during training. We set the image size to 448 × 800, which performs comparably with full resolution (896 × 1600). VoxelNet [57, 67] is chosen as our 3D backbone and the voxel size is set to $(0.075m, 0.075m, 0.2m)$. We set the detection range to $[-54.0m, 54.0m]$ for the X and Y axis, and $[-5m, 3m]$ for Z axis. The waymo model uses a detection range of $[-75.2m, 75.2m]$ for the X and Y axis, and $[-2m, 4m]$ for the Z axis. We set the image size to 640 × 960 and the voxel size to $(0.1m, 0.1m, 0.15m)$. We use ResNet50 with FPN as the image stream and it is trained for 36 epochs on Waymo of the 2D detection task. Codebase: https://github.com/XuyangBai/TransFusion/

BEVFusion. Following the implementation details in original paper [24] [2], we conduct BEVFusion with Dual-Swin-Tiny [23] as 2D bakbone for image-view encoder. TransFusion-L [1] are chosen as the LiDAR stream and 3D detection head. The image size is set to 448 × 800 and the voxel size following the official settings of the LiDAR stream [1, 16, 59]. And the training consists of two stages: i) First train the LiDAR stream and camera stream with multi-view image input and LiDAR point clouds input, respectively. Specifically, we train both streams following their LiDAR official settings in MMDetection3D [7]; ii) Then train BEVFusion for another 9 epochs that inherit weights from two trained streams. And no data augmentation (i.e., flipping, rotation, or CBGS [68]) is applied when multi-view image input is involved. Codebase: https://github.com/ADLab-AutoDrive/BEVFusion.

[2]Since BEVFusion doesn't provide results on Waymo and limited computing resource, we only present it on Tab. 1 and Tab. 2.

C. More Experiments

C.1. More Results of LiDAR object failure

In the case of LiDAR object failure, we choose randomly selecting 50% boxes and dropping the points within them because we would think dropping all objects seemed unrealistic. Dropping 100% points within a bounding box also seems extreme, hence we provide additional settings that dropping from 0% to 100% of the number of points within a box in Tab. 7. We can observe that, the performance has a drastic drop from 90% to 100%, while dropping 50% points only decreases by 1-2 points in mAP.

And we also explore the dropping probability setting in Tab. 8, in which a trend can be observed that the performance degradation of all methods is linear to the dropping probability. Thus, to reduce the load of the benchmarks, we only consider 50% possibility as the final benchmark setting.

C.2. Robust Finetuning

Though the main contribution of this work is to provide a systematic overview of different aspects of the perception system and construct a robustness benchmark, we nonetheless provide a simple yet effective baseline method, robustness finetuning, to improve the robustness of fusion method. We select the MVX-Net to study the effectiveness of our method, as it has the most balanced LiDAR and camera robust performance.

Study the individual noisy case of robustness finetuning. We provide a simple baseline method by treating our toolkit as a data augmentation method to enrich the training data as the first attempt to improve the robustness of

Approach	Aug	P_C	mP_R	R	Lidar Stuck	Lidar FOV	Lidar Object	Camera Stuck	Camera Missing	Camera Calib	Camera Occlusion
nuScenes-R (mAP / NDS)											
MVX-Net [42]	None	61.0 / 66.1	37.7 / 53.2	0.62 / 0.81	35.2 / 51.4	17.6 / 43.1	34.0 / 51.1	48.3 / 58.8	32.7 / 50.6	50.8 / 59.9	45.5 / 57.6
	LiDAR-stuck	58.4 / 64.2	35.6 / 51.5	0.61 / 0.80	**39.1 / 52.1**	17.8 / 42.8	34.2 / 51.4	42.4 / 54.9	28.2 / 47.6	46.6 / 56.9	41.0 / 54.8
	FOV	54.2 / 59.1	33.9 / 49.9	0.63 / 0.84	33.3 / 49.5	**18.4 / 44.0**	30.3 / 47.4	41.4 / 50.8	27.7 / 47.0	45.7 / 56.0	40.5 / 54.9
	Object	56.8 / 60.9	36.5 / 51.4	0.63 / 0.84	**39.1 / 52.1**	17.8 / 42.8	33.1 / 49.9	44.3 / 53.7	29.9 / 48.5	48.2 / 57.5	42.8 / 55.1
	Camera-stuck	58.6 / 64.7	37.9 / 53.2	0.65 / 0.82	31.0 / 48.9	17.2 / 42.9	32.7 / 50.0	**51.3 / 60.4**	35.2 / 51.8	52.0 / 60.9	45.7 / 57.6
	Missing	60.9 / 66.0	40.1 / 54.4	0.66 / 0.82	34.9 / 51.1	18.3 / 43.6	**34.1 / 50.8**	48.7 / 59.1	**45.1 / 57.1**	51.1 / 60.2	**48.2 / 58.9**
	Calib	60.0 / 65.4	38.6 / 53.6	0.64 / 0.82	33.6 / 50.3	17.8 / 43.3	33.6 / 50.7	50.0 / 59.6	34.6 / 51.3	**54.9 / 62.2**	45.8 / 57.6

Table 9. **The study of the individual noisy case for robustness finetuning.**

Approach	Modality	P_C	Overall mP_R	Overall R	Lidar mP_R	Lidar R	Camera mP_R	Camera R
nuScenes-R (mAP / NDS)								
MVX-Net [42]	LC	61.0 / 66.1	37.7 / 53.2	0.62 / 0.81	26.4 / 47.3	0.43 / 0.72	44.3 / 56.7	0.73 / 0.86
MVX-Net + finetune	LC	59.4 / 65.0	40.9 / 54.8	0.69 / 0.84	24.6 / 46.1	0.41 / 0.71	49.1 / 59.2	0.83 / 0.91
Waymo-R(L2 mAP / L2 mAPH)								
MVX-Net [42]	LC	59.7 / 54.1	44.3 / 40.1	0.74 / 0.74	28.3 / 25.5	0.47 / 0.47	56.4 / 51.0	0.94 / 0.94
MVX-Net + finetune	LC	59.5 / 54.0	47.7 / 43.1	0.80 / 0.80	27.8 / 25.2	0.47 / 0.47	57.6 / 52.1	0.97 / 0.96

Table 10. **Robust finetuning results of MVX-Net**.

performance. Specifically, we use the toolkit to transform the training data. We first analyze the effects of each individual transformation, as shown in Tab. 9, in which rows represent models finetuned on different augmentations. It's worth noticing that the transformation of camera lens occlusion case is not included during finetuning stage, since the masks for simulating the occlusion is only available during validation.

From Tab. 9, we can find that, most of the models achieve the best performance on the transformation they are fineuned on, but the models do not generalize to other types of noisy cases well. And it's worth noting that the model fineuned with missing case also achieves the best performance on occlusion case, which is consistent with previous works [13, 34], demonstrating that cameras dropout improves robustness against camera input corruption. Additionally, finetuning models with noisy transformation directly degrades the performance on clean data. Moreover, compared with MVX-Net baseline trained on clean data, finetuning with each noisy LiDAR data decreases the average performance mP_R, while finetuning with noisy camera data improves it. We hypothesize the reason is that the noisy LiDAR data brings wrong supervision during training and the fusion mechanism heavily relies on the LiDAR input.

A simple baseline to improve the robustness. From the practice in Tab. 9, we discover that transforming all data into the noisy format will significantly decrease the performance on the clean setting. To this end, we propose an augmentation policy with two cascaded probabilities for training: i) the augmentation probability p_a, decides whether to apply augmentation on the original data; ii) the probability distribution p_o, decides the probability for a certain augmentation from all robustness cases. As for existing fusion methods like MVX-Net, the LiDAR modality is the main modality and the camera modality is auxiliary. Thus, the sampling probabilities of noisy LiDAR transformations are set to be zero and the sampling probabilities of remaining noisy camera transformations, i.e., camera-stuck, missing of camera input and noisy calibration, are set to be $1/3$. And the augmentation probability p_a is set to 0.5.

We report the results in Tab. 10. We can observe that, though applying such robustness training slightly deteriorates the performance on the clean dataset, it moderately improves the robustness, where the mean robustness R improves from 0.62 and 0.81 to 0.69 and 0.84 in terms of mAP and NDS on nuScenes-R , and from 0.74 to 0.80 in terms of L2 mAP and mAPH on Waymo-R. However, we can still see a large gap between the robust benchmark and the clean ones, evidencing there is an actual research gap in this research direction.

C.3. A Simple Analysis of Late Fusion

Though the late fusion methods are not the ideal research direction in the future due to the complex post-processing technique. We nonetheless provide an analysis of non-deep-learning fusion strategies, simply splices results before non-maximum suppression(NMS) and weighted bounding boxes fusion [43]. We consider LSS [34] with Dual-Swin-Tiny [23] as our camera branch and PointPillars [16] as our LiDAR branch and 3D detection head. Not surprisingly,

Approach	Single-Modality		Multi-Modality		
	Camera(C)	LiDAR(L)	Fusion(LC)	Fusion w/o LiDAR(C)	Fusion w/o Camera(L)
UVTR [17]	31.4 / 40.1	60.8 / 67.6	65.4 / 70.2	4.0 / 22.0 (-68.7%)	57.5 / 65.8 (-6.3%)
BEVFusion(MIT) [27]	33.3 / 40.2	64.7 / 69.3	68.5 / 71.5	0.4 / 9.8 (-86.3%)	62.5 / 68.1 (-4.8%)
BEVFusion(Alibaba) [24]	22.7 / 26.1	64.9 / 69.9	67.9 / 71.0	0.1 / 1.7 (-97.6%)	56.2 / 63.5 (-10.5%)
FUTR3D [6]	31.3 / 40.1	60.8 / 67.6	64.2 / 68.0	0.2 / 10.5 (-84.6%)	25.3 / 44.8 (-34.1%)

Table 11. **Robustness against missing modality**. Fusion w/o LiDAR represents the scenario when LiDAR modality is severely missing, while Fusion w/o Camera means missing camera modality.

Approach	Modality	nuScenes (mAP / NDS)
PointPillars [16]	L	32.7 / 48.9
Lift-Splat-Shoot (LSS) [34]	C	23.0 / 31.2
Simply splices results before NMS	LC	35.7 / 47.6
Weighted boxes fusion [43]	LC	38.7 / 50.9
BEVFusion [24]	LC	53.5 / 60.4

Table 12. **A simple analysis of late fusion strategies**.

compared to deep fusion method, simple late fusion strategy achieved unsatisfactory fusion results as shown in Tab. 12.

C.4. Robustness Against Missing Modality

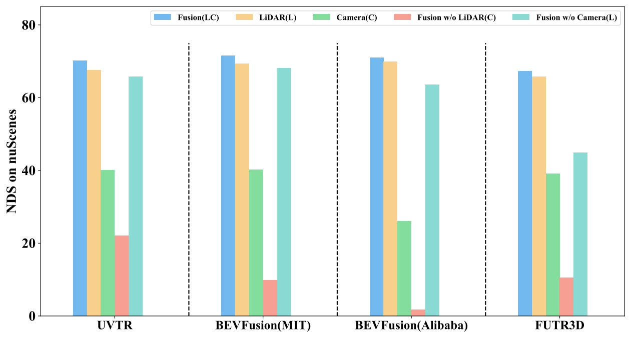

Figure 9. **Robustness against missing modality**. Fusion w/o LiDAR represents the scenario when LiDAR modality is severely missing, while Fusion w/o Camera means missing camera modality.

From Tab. 2, we can see that the robustness of models against noisy LiDAR cases is worse than the one against noisy camera cases. Here we further explore the robustness of more advanced fusion methods [6,17,24,27] against more severe robust scenarios, i.e., missing modality, in Tab. 11 and Fig. 9. And the values reflect mAP/NDS on nuScenes dataset. As shown, most of these concurrent fusion methods still maintain somehow good performance when camera modality is missing. When it comes to severely missing LiDAR modality, all of them almost result in failure. Moreover, when one modality is severely missing, the performances of fusion models are even worse than that of the single modality model of the other.

The fusion mechanism of most of the current popular fusion-based 3D object detection methods rely heavily on accurate LiDAR input. Thus, if the LiDAR sensor input is missing, current fusion methods fail to produce meaningful results. And in general, we believe an ideal sensor fusion framework should be able to do the following: i) given both modality data, it can significantly surpass the performance of single modality methods; ii) when there is a disruption of one modality, the performance should not be worse than the single modality method of the other. Currently, this approach is handled by using comprehensive post-processing techniques of the perception system. We hope our robust benchmark can be a tool for the community to fully exploit this research direction to develop truly robust methods that can be deployed on realistic vehicles.