Supplementary Material

A. Implementation Details

This section gives additional details on our implementation to facilitate reproducibility. First, we describe our baseline configuration, which is dubbed as BEVDepth†. Afterwards, we provide further information on our X^{3}KD implementation and the ablation experiments.

A.1. Baseline Details

Data Processing: Our data processing pipeline for images is mainly adopted from the published Code of BEVDet4D [7][1]. We implemented the pipeline in our code and verified that the resulting preprocessed inputs exactly match up with the ones from our reimplementation. The pipeline starts with processing the 6 images from different cameras of resolution 900×1600. To obtain images of the desired resolution during the evaluation, the images are resized by a factor of 0.48, i.e., to a resolution of 432×768. We then crop it to the target resolution of 256×704 with a center crop in the width dimension and by cropping the top part in the height dimension (assuming objects of interest are not above a certain height). During training we apply random resizing with scale factors in the range $[0.38, 0.55]$, random rotations $[-5.4°, 5.4°]$, and random horizontal flipping. The augmentation parameters are chosen randomly per sample but are applied equally to all 6 images from the cameras. If we use a higher final resolution, we adapt all rescaling and cropping transformations outlined above by the ratio between the higher resolution and 256×704 in a straightforward fashion. After all image transformations, we normalize all pixel values with the mean and standard deviations from ImageNet [14].

We store the parameters of all transformations to reversely apply them to image pixels before projecting them into a point cloud representation in LSS [13]. During training, we also apply data augmentations in BEV before applying voxel pooling to the point cloud representation in LSS. We apply random rotations $[-22.5°, 22.5°]$, random rescaling $[0.95, 1.05]$ and random horizontal and vertical flipping. Ground truth bounding boxes are transformed accordingly. During evaluation, no transformations are applied in BEV.

Network Configuration: The (augmented) input images are passed through our camera-based 3D object detection (3DOD) network architecture, which is mainly adopted from BEVDepth [10]. The images are first passed through a backbone, which is either ResNet-based [6] or ConvNeXt-based [11] using the implementations from mmdetection[2] and mmclassification[3], respectively. From the four resulting feature maps, the last two are passed through the feature pyramid network architecture from [7] with 512 output channels, which are input to the LSS transformation [13].

In LSS, the architectural choices for the feature and depth prediction networks are adopted from BEVDepth [10]. We use a depth spacing of $1\,\mathrm{m}$ and a voxel grid with the range $[-51.2\,\mathrm{m}, 51.2\,\mathrm{m}]$ with a spacing of $0.8\,\mathrm{m}$, resulting in a 128×128 feature grid with 64 output channels for the lowest image resolution 256×704. For all higher image resolutions, we use a depth spacing of $0.5\,\mathrm{m}$ and a voxel grid with the range $[-51.2\,\mathrm{m}, 51.2\,\mathrm{m}]$ with a spacing of $0.4\,\mathrm{m}$, resulting in a 256×256 feature grid with 64 output channels. As in previous works [7, 10], we use the same feature extraction and LSS transformation for a second adjacent time frame. Feature maps from both frames are concatenated directly after LSS. We compensate for the movement between both frames in the point cloud representation as in [10][4]. Also, we randomly sample adjacent frames between the 2nd and 9th sweep before the current one during training while the 5th sweep before the current one is used during evaluation, which corresponds to roughly $0.5\,\mathrm{s}$ time difference.

The BEV feature grid is passed through a small ResNet-like backbone and another feature pyramid network whose architectures are adopted from [7]. The resulting feature map has 256 output channels. Finally, we pass the features through a CenterPoint head [16] with parameters chosen as in BEVDet4D [7] to predict dense class probabilities and bounding box regression parameters for the 10 classes of nuScenes: car, truck, construction vehicle, bus, trailer, barrier, motorcycle, bicycle, pedestrian, and traffic cone. The Gaussian Focal loss for optimizing the dense class-wise probability scores is applied over a sigmoid output weighed with a factor of 1, while the smooth L1 loss for the regression outputs is weighted with factor of 0.25.

Training Details: For training, we use the AdamW op-

[1] https://github.com/HuangJunJie2017/BEVDet

[2] https://github.com/open-mmlab/mmdetection

[3] https://github.com/open-mmlab/mmclassification

[4] https://github.com/Megvii-BaseDetection/BEVDepth

timizer [12] with standard parameters and a weight decay of 0.01. We train all models for 24 epochs with an initial learning rate of $2 \cdot 10^{-4}$ and 500 warm-up iterations. We reduce the learning rate to $2 \cdot 10^{-5}$ after 16 epochs and again to $2 \cdot 10^{-6}$ after 22 epochs. We select the last checkpoint after 24 epochs for evaluation and do not use early stopping.

A.2. X^{3}KD Details

LiDAR Teacher Configuration: Our LiDAR-based 3DOD teacher model is strongly based on the implementation of TransFusion [1][5] which we reimplement in our framework. It first voxelizes the point cloud and uses the sparse encoder from SECOND [15] to generate 3D sparse features. After flattening the height dimension into the channel dimension, we apply the same BEV encoder-decoder network and CenterPoint head and loss functions used in the camera-based 3DOD model. We train the model for 20 epochs using the AdamW optimizer, an initial learning rate of 10^{-4} and the standard cyclic learning rate schedule over 20 epochs from mmdetection3d[6]. We additionally apply gradient clipping at a value of 0.1. During training, we apply random rescaling $[0.9, 1.1]$, random rotations $[-45°, 45°]$, random horizontal and vertical flipping, point shuffling, and object sampling augmentations. The last 5 epochs are trained without object sampling.

Cross-modal Output-stage Distillation (X-OD): During knowledge distillation, we use LiDAR point clouds corresponding to the multi-camera image input to the camera-based 3DOD model. The point cloud is processed with the same BEV augmentations as the point cloud representation inside LSS. Subsequently, it is passed through the LiDAR-based 3DOD model to generate outputs corresponding to the camera-based 3DOD model outputs. Recent work has shown that this technique is usually more effective than pre-computing pseudo labels [2]. If the output resolutions of the LiDAR-based and camera-based 3DOD models do not match, we resize the LiDAR-based 3DOD model's outputs using nearest neighbor interpolation to match the camera model outputs.

Cross-modal Feature-stage Distillation (X-FD): We use the features directly after the LSS transform of the camera-based 3DOD model. We pass these features through a convolutional layer, batch normalization, and ReLU activation with kernel size 3×3 and 32 output channels. Afterward, we apply another convolution with kernel size 1×1, one output channel, and a subsequent ReLU activation such that the output can only contain positive values. This output is the predicted mean feature activation $\hat{\boldsymbol{h}}$ in the X-FD loss function. We compare this prediction against the LiDAR model's flattened sparse features in BEV $\tilde{\boldsymbol{f}}^{\mathrm{BEV}}$, whose absolute values are averaged over the channel dimension yielding $\tilde{\boldsymbol{h}}$. If the spatial extent of $\tilde{\boldsymbol{h}}$ and $\hat{\boldsymbol{h}}$ do not match, we interpolate $\tilde{\boldsymbol{h}}$ in nearest neighbour fashion to match the spatial resolution of $\hat{\boldsymbol{h}}$.

[5]https://github.com/XuyangBai/TransFusion
[6]https://github.com/open-mmlab/mmdetection3d

Cross-modal Feature-stage Adversarial Training (X-AT): After concatenating the refined features $\tilde{\boldsymbol{f}}^{\mathrm{REF}}$ and $\hat{\boldsymbol{f}}^{\mathrm{REF}}$ from LiDAR-based 3DOD model and camera-based 3DOD model, respectively, we pass them through a gradient reversal layer which only affects the backward pass and a discriminator whose architecture is strongly adopted from [9]. We use three blocks of convolutional layers with kernel size 4×4, stride 2, and 256 output channels, as well as subsequent instance normalization and Leaky ReLU activation with slope 0.2 for negative values. Afterwards, we apply a convolutional layer with kernel size 4×4 and 1 output channel with subsequent Sigmoid activation function, yielding the modality classification output $\hat{s}$. Note that $\hat{s}$ is a map where every entry can be interpreted as the probability that an input patch of features is generated from the LiDAR-based (and not from the camera-based) 3DOD model. The binary labels $\boldsymbol{s}$ are created straightforwardly because we always know which model generated the input to the discriminator. When backpropagating the gradients from the binary cross entropy loss between $\hat{s}$ and $\boldsymbol{s}$, they are reversed in the gradient reversal layer directly before they reach the camera-based 3DOD model.

Cross-task Instance Segmentation Distillation (X-IS): For pretraining the instance segmentation models, we use the training split of the nuImages dataset, which contains about $60,000$ images containing instance segmentation labels for the 10 classes also used for the 3DOD task. We train the instance segmentation model for 12 epochs using plain stochastic gradient descent with an initial learning rate of $2 \cdot 10^{-2}$, a momentum of 0.9, and a weight decay of 10^{-4}. The learning rate is reduced to $2 \cdot 10^{-3}$ and $2 \cdot 10^{-4}$ at epochs 8 and 11, respectively. During 3DOD model training, we use the augmented input images and pass them through the instance segmentation teacher model following the strategy from [2]. Using the same prediction head architecture in both the instance segmentation teacher and the additional PV head of the 3DOD model ensures that the output shapes are equal and can be easily compared inside loss functions.

A.3. Ablation Details

Waymo Dataset Experiments: We mainly adopt the nuScenes configuration for Waymo and only apply the changes made necessary by the sensor setup and task definition of Waymo. First, images from different cameras have differing shapes, so we first pad all images with zeros to the shape 1280×1920. During the evaluation, we resize images by 0.4 and crop to the final resolution of 384×736. During training, we randomly resize by factors $[0.33, 0.45]$ and keep all other image augmentations as on nuScenes. In the

Model	Pre.	X-OD	X-FD	X-AT	X-IS	*mATE*↓	*mASE*↓	*mAOE*↓	*mAVE*↓	*mAAE*↓	*mAP*↑	*NDS*↑
BEVDepth[†]	✓	✗	✗	✗	✗	0.644	0.274	0.455	0.350	0.211	37.7	49.5
X-OD	✓	✓	✗	✗	✗	0.634	0.276	<u>0.451</u>	0.343	<u>0.204</u>	38.1	50.0
X-FD	✓	✗	✓	✗	✗	<u>0.626</u>	<u>0.270</u>	0.475	**0.333**	0.209	38.2	50.0
X-AT	✓	✗	✗	✓	✗	0.654	0.273	**0.439**	<u>0.337</u>	0.207	37.9	49.8
X-IS	✓	✗	✗	✗	✓	0.635	0.273	0.462	0.350	<u>0.204</u>	<u>38.7</u>	<u>50.1</u>
X^{3}KD$_{\text{all}}$	✓	✓	✓	✓	✓	**0.615**	**0.269**	0.471	0.345	**0.203**	**39.0**	**50.5**
LiDAR Teacher	NA	NA	NA	NA	NA	0.301	0.257	0.298	0.256	0.195	59.0	66.4

Table 1. **Ablation study of X^{3}KD on the nuScenes validation set**: We incrementally add our proposed cross-modal feature distillation (X-FD), adversarial training (X-AT) and output distillation (X-OD) as well as our cross-task instance segmentation distillation (X-IS). All X^{3}KD variants in the top part are solely based on multi-camera images during inference. **Backbone weights are initialized with weights pretrained for instance segmentation on nuImages (Pre.).** Best numbers in boldface, second best underlined.

Model	Pre.	$\lambda^{\text{X-AT}}$	*mAP*↑	*NDS*↑
BEVDepth[†]	✓	0	37.7	49.5
	✓	1	37.6	49.4
X-AT	✓	10	**37.9**	**49.8**
	✓	20	37.8	49.7

Table 2. **Ablation on the X-AT loss weight**. We initialize all models with backbones pre-trained on nuImages (Pre.).

Model	Pre.	$\lambda^{\text{X-FD}}$	*mAP*↑	*NDS*↑
BEVDepth[†]	✓	0	37.7	49.5
	✓	2	37.8	49.6
X-FD	✓	10	**38.2**	**50.0**
	✓	25	38.1	49.7
	✓	100	37.3	48.8

Table 3. **Ablation on the X-FD loss weight**. We initialize all models with backbones pre-trained on nuImages (Pre.).

Model	Pre.	$\alpha^{\text{3D-bbox}}$	*mAP*↑	*NDS*↑
BEVDepth[†]	✓	NA	37.7	49.5
	✓	0.8	37.9	49.8
X-OD	✓	0.6	**38.1**	**50.0**
	✓	0.3	38.0	49.9

Table 4. **Ablation on the X-OD ground truth selection threshold**. We initialize all models with backbones pre-trained on nuImages (Pre.).

Model	Pre.	$\alpha^{\text{2D-bbox}}$	*mAP*↑	*NDS*↑
BEVDepth[†]	✓	1.0	37.7	49.5
	✓	0.8	37.9	49.8
	✓	0.5	38.0	50.0
X-IS	✓	0.2	**38.7**	**50.1**
	✓	0.1	38.2	49.9

Table 5. **Ablation on the X-IS ground truth selection threshold**. We initialize all models with backbones pre-trained on nuImages (Pre.).

LSS view transformation, we use a depth spacing of $1\,\text{m}$ and a voxel grid with the range $[-76.8\,\text{m}, 76.8\,\text{m}]$ with a spacing of $1.2\,\text{m}$, resulting in a 128×128 feature grid. All other parameters and augmentations are as on nuScenes. Further, the 5 cameras of the Waymo dataset only provide a $252°$ field of view. Therefore, we make sure to filter bounding boxes that are not visible in any of the cameras during training and evaluation (indicated by the metadata of a bounding box). Also, we adapt the CenterPoint head to output predictions for only 3 classes of bounding boxes, i.e., vehicles, pedestrians, and cyclists.

RADAR-based 3DOD Experiments: The data loading and preprocessing of the RADAR point cloud input data is adopted from FUTR3D [3][7] and in principle works as the preprocessing for the LiDAR point cloud. While the RADAR point cloud is much sparser than the one from the LiDAR sensor, it contains additional information per point, such as velocity. Our RADAR-based 3DOD model's feature extraction is adopted from the LiDAR model's feature extraction. We concatenate features from both modalities after the view transformation/projection to BEV for the camera-RADAR fusion-based model. The subsequent processing in BEV remains unchanged.

[7]https://github.com/Tsinghua-MARS-Lab/futr3d

B. Additional Experiments

In this section, we provide further analysis regarding our chosen hyperparameters and the effectiveness of X^{3}KD on varying backbones.

B.1. Hyperparameter Analysis

Effect of Pre-Training: We provide additional ablation studies with a backbone pre-trained on the nuImages dataset in Table 1. We observe that adding all of our contributions individually consistently improves the baseline BEVDepth[†] and our complete X^{3}KD method achieves the best results. This is consistent with the main paper's observations for backbones pre-trained on ImageNet. We further observe that, if we pre-train the backbone for instance segmentation, the achieved improvement is 1.0 and 1.3 points in *mAP* and *NDS*, respectively. This improvement is more minor compared to using backbones pre-trained only on ImageNet. Nevertheless, X^{3}KD can provide meaningful additional supervision signals to guide the multi-camera 3DOD model towards a better convergence.

Model	*Backbone*	*Resolution*	*mAP*↑	*NDS*↑
BEVDepth†	ResNet-50	256 × 704	35.9	47.2
X^{3}KD (Ours)	ResNet-50	256 × 704	**39.0**	**50.5**
BEVDepth†	ConvNeXt-T	256 × 704	38.3	50.8
X^{3}KD (Ours)	ConvNeXt-T	256 × 704	**39.7**	**51.9**
BEVDepth†	ResNet-101	640 × 1600	42.8	53.6
X^{3}KD (Ours)	ResNet-101	640 × 1600	**46.1**	**56.7**

Table 6. **Ablation on varying backbones**: We show that X^{3}KD improves ResNet-based as well as ConvNeXt-based models on top of BEVDepth† (re-implementation of [10]).

X^{3}KD Hyperparameters: We further want to provide insights into our hyperparameter choices for X^{3}KD in Tables 2 to 5. First, we analyze the influence of the loss weights $\lambda^{\text{X-AT}}$ in Table 2. We observe that small loss weights tend to have no apparent effect on the performance, which is very similar to the baseline result. Increasing the loss weight to our chosen hyperparameter $\lambda^{\text{X-AT}} = 10$ improves performance over the baseline. Further increasing the weight yielded no further improvements. Similar observations could be made for the feature-stage distillation loss weight $\lambda^{\text{X-FD}}$ in Table 3, where the optimal weight is again $\lambda^{\text{X-FD}} = 10$. The cross-task instance segmentation distillation and cross-modal output distillation both involve losses, which are transferred from training with ground truth to training with pseudo labels. However, the losses have been investigated in [5, 16]. We therefore keep loss weights $\lambda^{\text{X-OD}} = 1$ and $\lambda^{\text{X-IS}} = 1$ whiteout further optimization. We do, however, optimize the ground truth selection thresholds $\alpha^{\text{3D-bbox}}$ and $\alpha^{\text{2D-bbox}}$ in Tables 4 and 5, respectively. Predictions above these thresholds are considered bounding boxes inside the respective losses. For $\alpha^{\text{3D-bbox}}$ in Table 4, we observe that we improve over the baseline regardless of the chosen threshold, while best results are achieved for $\alpha^{\text{2D-bbox}} = 0.6$. Similar results are obtained for $\alpha^{\text{2D-bbox}}$ in Table 5, where the optimal value $\alpha^{\text{3D-bbox}} = 0.2$ is obtained.

B.2. Varying Backbones

We provide an overview of the effectiveness of X^{3}KD for different backbones in Table 6. We observe significant improvements for ResNet-based models, the backbone we used to determine our method hyperparameters. When applying X^{3}KD to a model using the ConvNeXt-T backbone with unchanged hyperparameters, we also observe improvements in terms of *mAP* and *NDS*, though the more minor improvement compared to ResNet-based backbones might indicate that there is room for improvement through hyperparameter tuning.

B.3. Application on other Baselines

An important additional question is whether our method can be transferred to other baseline approaches. In this regard, our cross-task distillation (X-IS) enhances the image feature extraction, on which many camera-based 3DOD methods rely. Our output-level distillation (X-OD) replaces the ground truth in the 3DOD losses with the LiDAR model's output, which is also transferable. Regarding feature-level distillation by X-FD and X-AT, the LiDAR model's guiding 3D features are projected to BEV space, which can be adapted to other 2D representations.To demonstrate the transferability of X^{3}KD to other baseline approaches, we combine X^{3}KD with BEVDet [8] and BEVDet4D [7] in Table 7, showing significant improvements by using X^{3}KD.

model	BEVDet	BEVDet + X^{3}KD	BEVDet4D	BEVDet4D + X^{3}KD
NDS↑	37.9	41.3	45.7	49.1

Table 7. Combination of X^{3}KD with BEVDet and BEVDet4D.

model	BEVDepth	X-AT	X-FD	X-OD	X-IS	**X^{3}KD**
training time	54 h	65 h	65 h	67 h	73 h	78 h
GPU memory	39 GB	44 GB	44 GB	44 GB	48 GB	52 GB
NDS↑	47.2	48.1	48.5	48.7	50.1	50.5

Table 8. Training complexity/performance trade-off when training on 4 NVIDIA A100 GPUs with a batch size of 16 per GPU.

B.4. Training Complexity Analysis

In the following, we analyze the additional training complexity induced through X^{3}KD. We can observe from Table 7 that although two additional teacher networks are introduced the overall training time increases only by about 50% as we do not need to calculate gradients for those additional networks. The complexity of pretraining segmentation and LiDAR teacher models can be omitted as one could use off-the-shelf models, e.g., from mmdetection3D [4]. We also observe that adding only single contributions from ours provides interesting trade-offs between performance and additional training complexity. However, also note that X^{3}KD induces no extra compute load in inference time, which is vital for sustainable large-scale deployment, e.g., for vehicle fleets.

C. Qualitative Analysis

We provide additional qualitative examples for X^{3}KD in comparison to the baseline BEVDepth† and the ground truth (GT) in Fig. 1. In the first example from the top, we observe that the orientation of trucks and trailers (red and green boxes) is not well predicted by the baseline BEVDepth† and some objects are not detected at all. While the predictions of X^{3}KD are also not perfect, most objects are detected and the orientation of objects is better aligned, which is also apparent by comparing the X^{3}KD output to the GT. We attribute the better detection to the additional guidance from instance segmentation in PV, while a better orientation and prediction in bird's eye view (BEV) is likely due to guidance from the LiDAR-based 3DOD model. More examples of objects that are difficult to detect are given in the second and third example. In the second example, the bicycle (pink box) is quite far away and appears rather small in the front

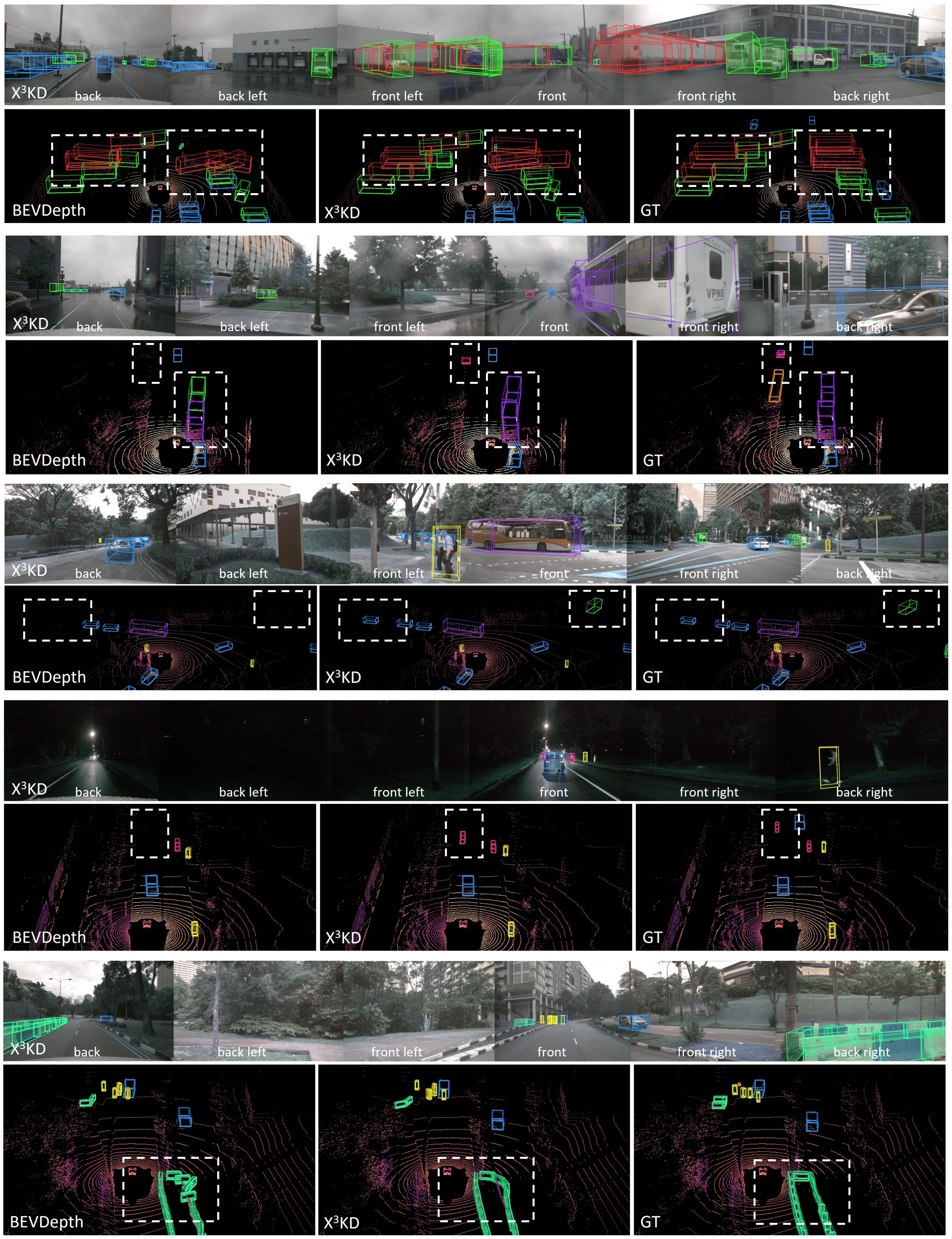

Figure 1. **Qualitative results on nuScenes**: We show the multi-camera input (top) and bounding box visualizations (bottom). We compare ResNet-101-based X^{3}KD$_{\text{all}}$ to BEVDepth[†] and the ground truth (GT) for a resolution of 640 $\times$ 1600. Best viewed on screen and in color.

camera image. Similarly, in the third example the truck (green box) and the car (blue box) appear small and the car is partially occluded. Guidance from instance segmentation in perspective view can help to detect these rather difficult examples. This improved detection behavior also generalizes to adverse weather conditions as can be seen in the fourth example. Again, a bicycle is detected by X^{3}KD but not by the baseline BEVDepth†. Finally, the fifth example provides additional evidence on how the guidance from the LiDAR-based 3DOD teacher improves the translation and orientation characteristics of the predicted bounding boxes. The barriers (light green boxes) in BEVDepth† are oriented rather random, while there is a structurally meaningful orientation observable for X^{3}KD.

References

[1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection With Transformers. In *Proc. of CVPR*, pages 1090–1099, 2022. 2

[2] Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *Proc. of CVPR*, pages 10925–10934, 2022. 2

[3] Xuanyao Chen, Tianyuan Zhang, Yue Wang, Yilun Wang, and Hang Zhao. FUTR3D: A Unified Sensor Fusion Framework for 3D Detection. *arXiv preprint arXiv:2203.10642*, 2022. 3

[4] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 4

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. of CVPR*, pages 2961–2969, 2017. 4

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*, pages 770–778, 2016. 1

[7] Junjie Huang and Guan Huang. BEVDet4D: Exploit Temporal Cues in Multi-camera 3D Object Detection. *arXiv preprint arXiv:2203.17054*, 2022. 1, 4

[8] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View. *arXiv preprint arXiv:2112.11790*, 2021. 4

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-To-Image Translation With Conditional Adversarial Networks. In *Proc. of CVPR*, pages 1125–1134, 2017. 2

[10] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. BEVDepth: Acquisition of Reliable Depth for Multi-view 3D Object Detection. In *Proc. of AAAI*, pages 1–9, 2023. 1, 4

[11] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proc. of CVPR*, pages 11976–11986, 2022. 1

[12] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. of ICLR*, pages 1–18, 2018. 2

[13] Jonah Philion and Sanja Fidler. Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D. In *Proc. of ECCV*, pages 194–210, 2020. 1

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1

[15] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, 2018. 2

[16] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. In *Proc. of CVPR*, pages 11784–11793, 2021. 1, 4