MetaBEV: Solving Sensor Failures for 3D Detection and Map Segmentation <Supplementary Material>

Chongjian Ge* Junsong Chen* Enze Xie Zhongdao Wang
Lanqing Hong Huchuan Lu Zhenguo Li Ping Luo

We provide a detailed PyTorch-style pseudo-code to describe two key components in our MetaBEV, including the cross-modal attention and the M^{2}oE structure. Then we elaborate on the comprehensive sensor corruptions utilized in our evaluation, as well as the detailed training configurations employed for BEV 3D detection and map segmentation. Moreover, we present additional experiments and visualizations to further validate the effectiveness of MetaBEV. We also provide a video demo to illustrate how MetaBEV performs under various sensor failures. Finally, we emphasize that the code and trained models used in our research will be made publicly available for future studies.

A. Algorithms

We present the PyTorch-style pseudo-code for two key components in MetaBEV, i.e., the cross-modal attention and M^{2}oE, in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Cross-Attention Modules.

```
import torch
import torch.nn as nn

class CrossAttention(nn.Module):
  def __init__(self, d_model=264, n_heads=8, n_points=4, lidar_flag=False, camera_flag=False, **kwargs):
    super().__init__()

    if self.camera_flag:
       self.offsets_camera = nn.Linear(d_model, n_heads * n_points * 2)
       self.attention_weights_camera = nn.Linear(d_model, n_heads * n_points)

    if self.lidar_flag:
       self.offsets_lidar = nn.Linear(d_model, n_heads * n_points * 2)
       self.attention_weights_lidar = nn.Linear(d_model, n_heads * n_points)

    self.value_proj = nn.Linear(d_model, d_model)
    self.output_proj = nn.Linear(d_model, d_model)


  def forward(self, query, reference_points, x, camera_flag=False, lidar_flag=False):
    value = self.value_proj(x)

    if camera_flag and lidar_flag:
       offsets_camera = self.offsets_camera(query)
       offsets_lidar = self.offsets_lidar(query)
       offsets = torch.cat([offsets_camera, offsets_lidar])
       attention_weights_camera = self.attention_weights_camera(query)
       attention_weights_lidar = self.attention_weights_lidar(query)
       attention_weights = torch.cat([attention_weights_camera, attention_weights_lidar], dim=3)
    elif camera_flag and not lidar_flag:
       offsets = self.offsets_camera(query)
       attention_weights = self.attention_weights_camera(query)
    if lidar_flag and not camera_flag:
       offsets = self.offsets_lidar(query)
       attention_weights = self.attention_weights_lidar(query)

    attention_weights = F.softmax(attention_weights)
    sampling_locations = reference_points + offsets

    # MSDeformAttnFunction is implemented by DeformableAttention
    output = MSDeformAttnFunction.apply(value, sampling_locations.float(), attention_weights,
    output = self.output_proj(output)
    return x
```

Algorithm 2 MoE.

```
import torch
import torch.nn as nn

# HMoE: Hard MoE
class MultiTaskExpertMlp(nn.Module):
   def __init__(self, in_features=256, hidden_features=None, out_features=None, act_layer=nn.GELU(),
            drop=0., det_flag=False, seg_flag=False, **kwargs):
      super().__init__()
      self.det = det_flag
      self.seg = seg_flag
      self.multi_tasks = det_flag and seg_flag

      if self.multi_tasks:
         self.det_expert = nn.Sequential(
            nn.Linear(in_features, hidden_features), act_layer, nn.Dropout(drop),
            nn.Linear(hidden_features, out_features), nn.Dropout(drop)
         )
         self.seg_expert = nn.Sequential(
            nn.Linear(in_features, hidden_features), act_layer, nn.Dropout(drop),
            nn.Linear(hidden_features, out_features), nn.Dropout(drop)
         )
         self.fuse = nn.Sequential(nn.Linear(out_features*2, out_features),)
         self.norm = nn.LayerNorm(out_features)
      elif self.seg:
         self.seg_expert = nn.Sequential(
            nn.Linear(in_features, hidden_features), act_layer, nn.Dropout(drop),
            nn.Linear(hidden_features, out_features), nn.Dropout(drop)
         )
      elif self.det:
         self.det_expert = nn.Sequential(
            nn.Linear(in_features, hidden_features), act_layer, nn.Dropout(drop),
            nn.Linear(hidden_features, out_features), nn.Dropout(drop)
         )

   def forward(self, x):
      b, n, c = x.shape
      if self.multi_tasks:
         det_fea = self.det_expert(x)
         seg_fea = self.seg_expert(x)
         fuse_fea = self.norm(self.fuse(torch.cat([det_fea, seg_fea], dim=2)))
         return fuse_fea
      elif self.det:
         x = self.det_expert(x)
         return x
      elif self.seg:
         x = self.seg_expert(x)
         return x


# RMoE: Routing MoE
from tutel import moe as tutel_moe
from tutel import net

class MultiTaskRouteMoEMlp(nn.Module):
   def __init__(self, in_features=256, hidden_features=None, out_features=None, drop=0., act_layer=nn.GELU(),
        **kwargs):
      super().__init__()
      self.act = act_layer
      gate_type = kwargs['gate_type']
      experts = kwargs['experts']

      # Implemented using Microsoft-tutel: https://github.com/microsoft/tutel
      self._moe_layer = tutel_moe.moe_dplayer(
         gate_type=gate_type,
         experts={'type': experts.type, 'count_per_node': experts.count_per_node,
                'hidden_size_per_expert': hidden_features, 'activation_fn': lambda x: self.act(x),
                'dropout_fn': lambda x: nn.Dropout(drop)(x)},
         model_dim=out_features,
         scan_expert_func=None,
         group=net.create_groups_from_world(group_count=1).data_group, # used for DistributedDataParallel mode
         seeds=(1, dist.get_rank() + 1, 1),
         a2a_ffn_overlap_degree=experts.a2a_ffn_overlap_degree,
         parallel_type=experts.parallel_type,
         use_2dh=experts.use_2dh,
      )

   def forward(self, x, gate_index):

      return self._moe_layer(x, gate_index=gate_index)
```

B. Implementation Details

B.1. Datasets

We evaluate MetaBEV on nuScenes [1], a large-scale multi-modal dataset for 3D detection and map segmentation, which contains high-resolution sensor data collected from diverse urban driving scenes, including a vast amount of annotated images, LiDAR scans, and 3D annotations. The dataset is generally split into 700/150/150 scenes for training/validation/testing. It contains data from multiple sensors, including six cameras, one LiDAR, and five radars. For camera inputs, each frame consists of six views of the surrounding environment at one specific timestamp. For LiDAR inputs, nuScenes collects points cloud reflected by the surroundings within 360 degrees. NuScenes is widely employed for the evaluation on object detection, tracking, and semantic segmentation. In this work, we not only evaluate our model on complete input modalities from camera and LiDAR but also conduct an in-depth investigation of the robustness of MetaBEV by utilizing pre-existing datasets to simulate and evaluate various corruptions. Details are illustrated as follows.

B.2. Corruptions

For sensor corruptions, we introduce a total of six common corruptions, which are detailed in the following:

1. Limited Field (LF): Nuscenes collects LiDAR point cloud data from a 360-degree perspective. However, due to hardware malfunctions, there may be certain angles where the point cloud data is missing, resulting in data collection that is less than 360 degrees. In this article, we follow [7] to simulate missing LiDAR data at different angles. The cases include 360, 240, 180, and 120 degrees.

2. Missing Objects (MO): In the real world, differences in object color and surface material, among other factors, can result in a reduction in the number of reflection points detected by the LiDAR system. In this work, we simulate missing object points by probabilistically removing point cloud data from objects at a specific rate. The selected rates are set to 0.0, 0.1, 0.3, 0.5, 0.7, and 1.0.

3. Beam Reduction (BR): Beams Reduction (BR) is a phenomenon that occurs in sensing systems where the available power supply or sensor processing capability is limited. In such systems, it is not possible to process the full set of data collected by the sensor, which leads to a reduction in the number of beams that are processed. In our work, we select 1, 4, 8, 16, and 32 beams for evaluation.

4. View Noise (VN): It is a phenomenon whereby the captured view contains random variations or distortions that are not representative of the actual scene. VN can arise due to various factors, including sensor noise, electrical interference, atmospheric conditions, or compression artifacts. In this paper, we have adopted a strategy to evaluate the effectiveness of our approach by simulating VN through the replacement of 0 to 6 views with randomly generated noise.

5. View Drop (VD): It refers to a real-world scenario where a portion of the visual scene is not captured by cameras, leading to a loss of information. VD can arise due to various factors such as incorrect camera positioning or hardware failure. In a similar manner to VN, we involve substituting zero-initialized inputs for up to six missing views during evaluation.

6. Obstacle Occlusion (OO): It refers to a perceptual phenomenon that occurs when objects within a scene are partially obscured from view by obstructions or occlusions. To simulate this phenomenon, we use a set of predefined masks to perform alpha blending with the camera views, thereby generating an occluded view.

B.3. Training Details

To train on the full modalities, we adopt the following configurations. To increase the diversity of training samples, we use standard image and LiDAR data augmentation strategies from MMDetection3D [2]. We also utilize AdamW [5] as our optimizer with a weight decay of 0.05 and a cyclical learning rate schedule [6]. To balance the object classes during data sampling, we use CBGS [8]. Generally, we train for 26 epochs for 3D detection and 20 epochs for segmentation on 8 A100 GPUs to achieve the best results.

For multi-task learning, we load the multi-modality model pre-trained on the 3D detection task and insert the segmentation head into the BEV-Evolving encoder. We then fine-tune the overall network with another 10 epochs. To balance multiple training objectives, we rescale the weights for detection and segmentation losses with 10 and 1, respectively and these hyper-parameters follow the setting in BEVFusion [4].

For the switched-modality training, we begin by loading a task-specific pre-trained model. During each training iteration, we randomly input sensor information from either camera, LiDAR, or both in a proportion uniformly configured as 1/3, 1/3, 1/3.

Task	Epoch	Vanilla training			Switched Modality Training		
		Lidar-Missing	Camera-Missing	Full	Lidar-Missing	Camera-Missing	Full
BEV-Map Segmentation	20(vanilla)+8(SMT)	36.5	27.0	70.4	54.4 **(+17.9)**	53.7 **(+16.7)**	68.5 (-1.9)
BEV-Map Segmentation	20(Vanilla)+20(SMT)	36.5	27.0	70.4	63.4 **(+26.9)**	58.3 **(+31.3)**	69.2 (-1.2)

Table 1. **Map segmentation results of Switched Modality Training with longer epochs**. Comparing the results of eight-epochs training in the main paper, we found that training for a longer schedule (e.g, 20 epochs) further improves MetaBEV's performance in map segmentation.

C. Additional Experiments

In this chapter, Sec. C.1 first demonstrates the performance of using Switched-Modality Training scheme to train MetaBEV on map segmentation for longer epochs (Table 1 *vs*. Table 5(d) in main paper). Secondly, Sec. C.2 presents a more comprehensive analysis of the zero-shot and in-domain performance of MetaBEV on various corruption types compared to the results reported in Table 3. In addition, Sec. C.3 illustrates the effectiveness of our map segmentation approach in various real-world scenarios, like snow, fog scenes *etc*. Finally, Sec. C.4 provides additional visualizations to facilitate further comparisons.

C.1. Additional Results on Switch Modality Training

In our main paper, the results of map segmentation in Table 5(d) are achieved via training MetaBEV using the Switched-Modality training (SMT) strategy for 8 epochs. The epoch number here is set to be the same as the full modality fine-tuning schedule, and MetaBEV already demonstrates satisfactory performance even when a single modality is absent (*e.g*., LiDAR or Camera). To further verify the performance of MetaBEV, we train it using SMT for total 20 epochs. The experimental results presented in Table 1 demonstrate that our model's performance could be significantly improved on both single and full modalities with longer training schedules. For instance, as shown in Table 1, MetaBEV achieves a remarkable 63.4% and 58.3% mIoU on camera-only inputs and LiDAR-only inputs. These results consistently outperform the model trained with only 8 epochs by 7.0% and 4.6% mIoU, respectively.

C.2. Detailed Comparisons on Sensor Corruptions

As described in Sec. B, we formulate different degrees for each type of corruption. In the main paper, we employ the most severe (or representative) degree on each specific corruption for evaluation. To demonstrate the robustness of MetaBEV more comprehensively, we evaluate MetaBEV on 3D detection and map segmentation under various degrees of corruptions. The experimental results shown in Table 2 prove that MetaBEV outperforms BEVFusion [4] in zero-shot evaluation for various degrees of corruptions, with the exception of Beam Reduction, where MetaBEV performs comparably. Besides, the carefully designed architecture facilitates effective training of MetaBEV on various degree corruption patterns, making it more capable of generalization and demonstrating stronger overall performance during in-domain evaluations. Specially, when dropping 6 views, MetaBEV surpasses BEVFusion by 39.4% mIoU (43.2% *v.s.* 4.1%). It indicates MetaBEV is able to process arbitrary modalities for strong robustness.

C.3. Additional Results on Map Segmentation

For segmentation, there is a representative benchmark [3] proposing overall sixteen corruptions to evaluate the zero-shot robustness of models, including different illumination, input noise, etc. In this appendix, to further test the robustness of MetaBEV, we selected 12 most commonly-encountered corruptions mentioned in [3], including cameras blur (i.e., defocus blur, glass blur, Gaussian blur), lighting conditions (i.e, brightness, contrast, saturation, JPEG compression), weather (i.e., snow, spatter, fog, frost), *etc*., and generated nuScenes-C for testing the performance of MetaBEV. For comparisons, we select the representative method BEVFusion [4] for testing. We adopt two metrics, including the mean Intersection over Union (mIoU) for zero-shot segmentation, and the retention ratio ρ between the performance on full modalities and corrupted modalities, which is calculated as follows:

$$\rho = \frac{F_{mIoU}}{C_{mIoU}}, \tag{1}$$

where F_{mIoU} is the mIoU tested on the full modalities, while C_{mIoU} is the one calculated on the corrupted inputs. Figure 1 and Table 3 show the comparison results, which indicate that MetaBEV almost consistently surpasses BEVFusion [4] on the metrics of mIoU and retention ρ.

Methods	Evaluation	Limited Field [-180,180]		Limited Field [-120,120]		Limited Field [-90,90]		Limited Field [-60,60]		Obstacle Occlusion w/o Occlusion		Obstacle Occlusion w Occlusion	
		NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU
BEVFusion [4]	zero-shot	71.4	62.3	52.8	51.9	49.6	49.6	41.6	47.2	71.4	62.3	68.6	45.7
MetaBEV		71.5	69.3	57.5	63.4	52.6	62.3	47.0	61.1	71.5	69.3	70.0	61.1
BEVFusion [4]	in-domain	65.3	59.6	48.8	57.0	47.1	56.3	42.6	55.5	70.2	60.5	69.7	59.3
MetaBEV		71.1	65.8	61.7	63.1	58.2	62.4	54.3	62.1	70.8	70.6	70.3	70.2

(a) Experimental comparisons on *Limited Field* and *Obstacle Occlusion*.

Methods	Evaluation	Missing Objects 0.0 rate		Missing Objects 0.1 rate		Missing Objects 0.3 rate		Missing Objects 0.5 rate		Missing Objects 0.7 rate		Missing Objects 1.0 rate	
		NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU
BEVFusion [4]	zero-shot	71.4	62.3	70.6	62.3	68.8	62.2	67.1	62.2	65.2	62.1	62.1	62.1
MetaBEV		71.5	69.3	70.8	69.2	69.3	69.2	67.6	69.2	65.9	69.2	62.6	69.2
BEVFusion [4]	in-domain	70.3	62.0	69.6	62.0	69.5	62.0	67.6	62.0	67.7	61.9	65.2	61.9
MetaBEV		70.9	69.9	70.6	69.9	70.3	69.9	69.7	69.9	69.4	69.9	68.5	69.9

(b) Experimental comparisons on *Missing Objects*.

Methods	Evaluation	Beam Reduction 32 beams		Beam Reduction 16 beams		Beam Reduction 8 beams		Beam Reduction 4 beams		Beam Reduction 1 beams	
		NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU
BEVFusion [4]	zero-shot	71.4	62.3	64.6	59.3	61.5	56.7	58.3	55.3	33.1	43.8
MetaBEV		71.5	69.3	65.0	67.1	61.2	64.7	57.7	64.3	32.2	57.9
BEVFusion [4]	in-domain	70.1	61.1	66.7	59.7	65.8	59.2	64.9	59.0	51.2	54.9
MetaBEV		71.3	68.2	68.0	67.0	67.2	66.5	66.6	66.3	54.6	62.0

(c) Experimental comparisons on *Beam Reduction*.

Methods	Evaluation	View Drop 1 drop		View Drop 2 drops		View Drop 3 drops		View Drop 4 drops		View Drop 5 drops		View Drop 6 drops	
		NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU
BEVFusion [4]	zero-shot	70.8	55.0	70.0	47.4	69.4	38.4	68.6	28.2	68.2	17.1	67.5	4.1
MetaBEV		71.0	67.2	70.4	64.6	70.0	61.8	69.5	57.6	68.9	51.8	68.3	43.5
BEVFusion [4]	in-domain	68.4	57.8	68.3	56.9	68.3	56.0	68.2	54.8	68.3	54.1	68.3	52.8
MetaBEV		70.2	68.2	70.0	68.2	69.8	68.1	69.6	68.0	69.4	68.0	69.3	67.9

(d) Experimental comparisons on *View Drop*.

Methods	Evaluation	View Noise 1 noise		View Noise 2 noise		View Noise 3 noise		View Noise 4 noise		View Noise 5 noise		View Noise 6 noise	
		NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU	NDS	mIoU
BEVFusion [4]	zero-shot	70.7	56.9	69.8	51.4	69.1	45.5	68.2	39.1	67.8	33.1	66.9	25.8
MetaBEV		70.9	67.0	70.4	64.4	70.0	61.4	69.5	57.2	68.9	51.8	68.3	45.1
BEVFusion [4]	in-domain	69.3	57.4	69.1	56.7	68.8	55.8	68.7	54.9	68.5	54.2	68.3	53.1
MetaBEV		70.2	68.0	70.0	68.0	69.8	67.9	69.6	67.8	69.4	67.6	69.2	67.5

(e) Experimental comparisons on *View Noise*.

Table 2. **Experimental comparisons on sensor corruptions with various degrees.** Texts in **blue** denote the specific corruption degrees. MetaBEV consistently outperforms BEVFusion [4] on various sensor corruptions in both zero-shot and in-domain tests.

Methods	Clear	Blur				Digital				Weather			
		Motion	Defoc	Glass	Gauss	Bright	Contr	Satur	JPEG	Snow	Spatt	Fog	Frost
BEVFusion [4]	62.9	37.1 (58.9%)	25.4 (40.4%)	36.0 (57.1%)	29.1 (46.3%)	49.3 (78.3%)	39.2 (62.2%)	43.3 (68.7%)	31.4 (49.8%)	30.7 **(48.7%)**	48.7 (77.4%)	48.3 (76.8%)	31.0 (49.3%)
MetaBEV	**70.4**	**49.6 (70.4%)**	**40.7 (57.9%)**	**51.5 (73.2%)**	**43.8 (62.2%)**	**57.9 (82.2%)**	**47.1 (66.9%)**	**48.6 (69.1%)**	**41.7 (59.2%)**	**31.6** (44.9%)	**55.3 (78.5%)**	**56.7 (80.5%)**	**35.8 (50.8%)**

Table 3. **Experimental comparisons on map segmentation under different camera noises.**.

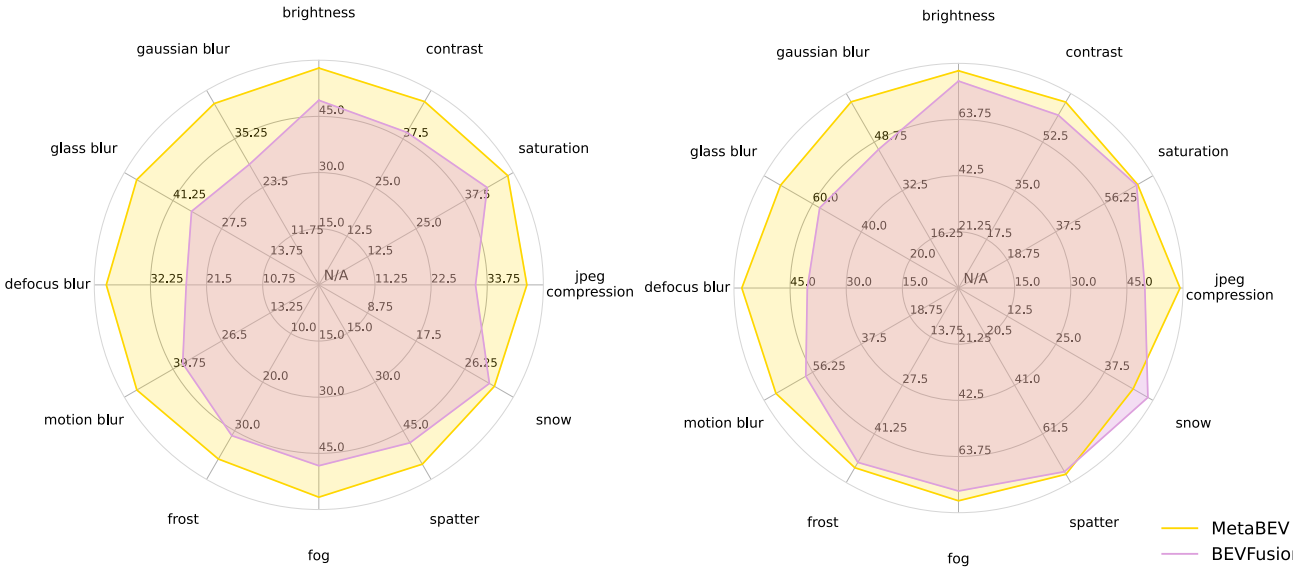

Figure 1. **Map Segmentation results comparison under camera noises.** We show the map segmentation mIoU (**left**) and the corresponding retention (**right**) under different weather, blur and digital conditions to show the zero-shot performance in the real-world environment. Details could be found in Table 3 and better view in color on screen.

C.4. Additional Visualizations

We present additional 3D prediction results in Figure 2. It could be observed that MetaBEV performs robustly under various kinds of sensor failures. For example, we when the camera is totally missing, BEVFusion [4] can barely generate map segmentation, while MetaBEV is still capable of producing satisfactory results. The same phenomenon could also be found under the LiDAR-missing scenarios. Besides, for sensor corruptions, MetaBEV is also more robust than BEVFusion [4]. For instance, when *View Noise* occurs as shown in the fifth column, BEVFusion tends to omit some objects (*e.g.*, the vehicles in the first row and the pedestrian in the fourth row), while MetaBEV is almost able to cover all the objects in the ground truth. Finally, we provide a video demo to better illustrate MetaBEV under all the above-mentioned sensor failure cases.

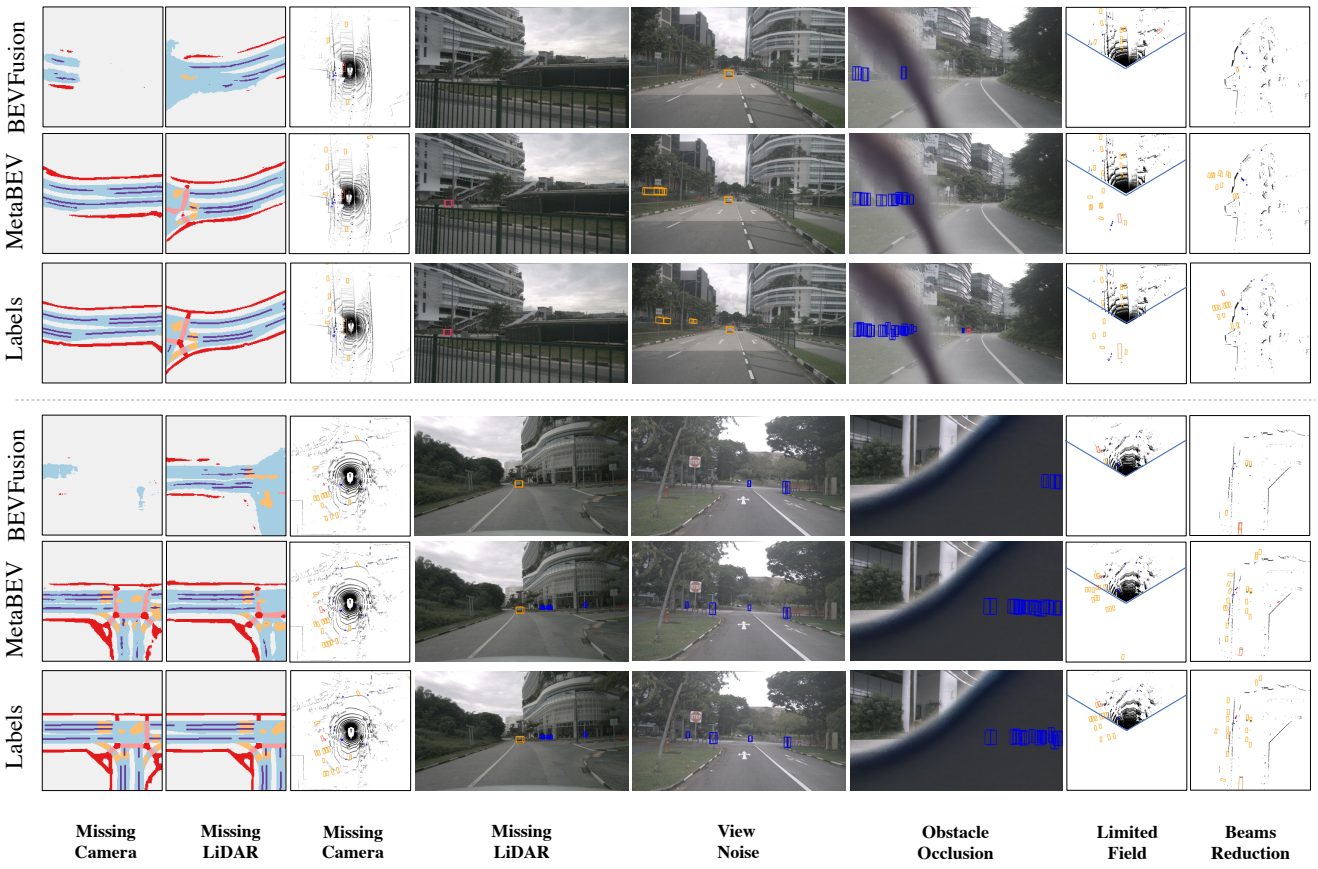

Figure 2. **Additional 3D prediction results under various sensor failures.**

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

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

[3] Christoph Kamann and Carsten Rother. Benchmarking the robustness of semantic segmentation models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[4] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542*, 2022.

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[6] Leslie N Smith. Cyclical learning rates for training neural networks. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2017.

[7] Kaicheng Yu, Tang Tao, Hongwei Xie, Zhiwei Lin, Zhongwei Wu, Zhongyu Xia, Tingting Liang, Haiyang Sun, Jiong Deng, Dayang Hao, et al. Benchmarking the robustness of lidar-camera fusion for 3d object detection. *arXiv preprint arXiv:2205.14951*, 2022.

[8] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.