GaussRender: Learning 3D Occupancy with Gaussian Rendering

Supplementary Material

A. Datasets

SurroundOcc-nuScenes [54] and Occ3D-nuScenes [50] are derived from the nuScenes dataset [3]. nuScenes provides 1000 scenes of surround view driving scenes in Boston and Singapore split in three sets train/val/test of size 700/150/150 scenes. Each comprises 20 seconds long and is fully annotated at 2Hz using one ground truth from 5 radars, 6 cameras at resolution 900 × 1600 pixels, one LiDAR, and one IMU. From the LiDAR annotations, SurroundOcc-nuScenes [54] derived a 3D grid of shape $[200, 200, 16]$ with a range in $[-50, 50] \times [-50, 50] \times [-5, 3]$ meters at a spatial resolution of $[0.5, 0.5, 0.5]$ meters, annotated with 17 different classes, 1 representing empty and 16 for the semantics. The Occ3d-nuScenes [50] dataset has a lower voxel size of 0.4 meters in all directions while keeping the same voxel grid shape with a range of $[-40, 40] \times [-40, 40] \times [-1, 5.4]$ meters. It contains 18 classes: 1 representing empty, 16 for the semantics, and 1 for others.

SSCBench-Kitti360 [31] is derived from the Kitti360 dataset [34]. Kitti360 consists of over 320k images shot by 2 front cameras at a resolution 376 × 1408 pixels and two fisheye cameras in surburban areas covering a driving distance of 73.7km. Only one camera is used in the 3D occupancy task. SSCBench-Kitti360 [31] annotates for each sequence a voxel grid of shape $[256, 256, 32]$ with a range in $[0, 51.2] \times [-25.6, 25.6] \times [-2, 4.4]$ meters at a voxel resolution of 0.2 in all directions. The provided voxel grid is annotated with 19 classes: one is used to designate empty voxels, and the 18 other are used for the semantic classes.

B. Models and implementation details

We integrate our rendering module and associated loss into three different models: **SurroundOcc** [54] (multi-scale voxel-based approach), **TPVFormer** [15] (triplane-based approach), and **Symphonies** [19] (voxel-with-instance query-based approach). Each model is retrained using the same training setting, following the optimization parameters from SurroundOcc. No extensive hyperparameter searches are conducted on the learning rate; the goal is to demonstrate that the loss can be integrated at minimal cost into existing pipelines. All models are trained for 20 epochs on 4 A100 or H100 GPUs with a batch size of 1, using an AdamW optimizer with a learning rate of $2e^{-4}$ and a weight decay of 0.01. For each combination of models and datasets, we evaluate existing checkpoints if provided; otherwise, we report the scores from previous papers when available or we re-train the models. Note that we used the official checkpoint for Symphonies [19] while noticing there is a discrepancy in IoU / mIoU between the reported value in the paper and the actual one of the official checkpoint, as explained in their GitHub issue [1].

Data	Model	Tr. time (HH:MM)		Memory usage (GB)	
Sur.Occ-nusc	TPVFormer	21:44		25.3GB	
	w/ GaussRender	24:00	+10.4%	28.1GB	+11.1%
	SurroundOcc	26:38		23.0GB	
	w/ GaussRender	29:19	+10.5%	24.2GB	+5.2%
SSCB.K.360	TPVFormer	7:02		29.3GB	
	w/ GaussRender	8:16	+14.0%	31.7GB	+8.2%
	SurroundOcc	11:12		15.5GB	
	w/ GaussRender	11:56	+6.1%	17.6GB	+13.5%

Table 7. **Training time and GPU memory usage** across models and datasets without or with our module using four renderings per scene, two for BeV (ground truth and predictions), and two for another camera (ground truth and predictions). Test performed on a 40GB A100.

C. Computational cost

Our module introduces a computation overhead for each rendering it performs. For a given input scene, we generate two views (one 'cam' and one 'bev'), and for each view, we render both the predictions and the ground truth, resulting in a total of four renderings per iteration.

As reported in Tab. 7, training with GaussRender incurs a modest increase in memory and computation time (10%), while using high-resolution renderings. This overhead can be further reduced by pre-selecting camera locations, allowing annotation renderings (the two ground-truth renderings) to be pre-processed in advance. Additionally, lower rendering resolutions can be used if needed.

While GaussRender introduces a small per-iteration cost, it actually accelerates learning. A key observation is that models using GaussRender reach the same performance level as their baseline counterpart 17% faster. Overall, despite a minor increase in computational overhead, GaussRender ultimately reduces the total training time required to achieve comparable or superior performance.

D. BeV metrics.

Additionally, we evaluate some Bird's-Eye-View (BeV) metrics — critical for downstream motion forecasting and

[1] https://github.com/hustvl/Symphonies/issues/5

Dataset	Model	IoU$^{\text{BeV}}$ (↑)	mIoU$^{\text{BeV}}$ (↑)
SurroundOcc-nuSc [54]	TPVFormer [15]	58.16	28.48
	w/ GaussRender	59.20 +1.04	**28.73** +0.25
	SurroundOcc [54]	58.60	28.26
	w/ GaussRender	**60.55** +1.95	28.64 +0.38
Occ3D-nusc [50]	TPVFormer [15]	52.95	29.72
	w/ GaussRender	54.35 +1.40	30.26 +0.54
	SurroundOcc [54]	53.52	28.98
	w/ GaussRender	**55.65** +2.13	**30.50** +1.52

Table 8. **Impact of GaussRender on BeV metrics.** Comparison of BeV metrics (IoU$^{\text{BeV}}$, mIoU$^{\text{BeV}}$) across datasets. Best results per dataset/metric are in bold with green performance deltas.

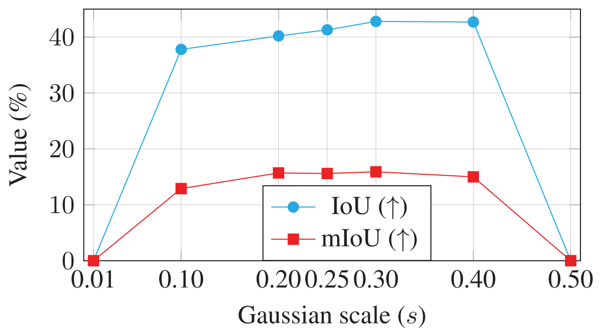


Figure 4. **Impact of fixed Gaussian scales on 3D mIoU and IoU** using TPVFormer [15] trained using only L_{2D} without L_{3D} on 20% of Occ3d-nuScenes [50] validation dataset.

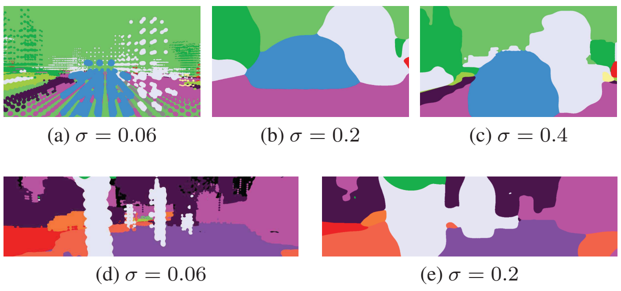


Figure 5. **Visualization of different Gaussianized voxels for different datasets and scales**. The first row represents data from Occ3d-nuScenes [50] and the second and third rows are from SSCBench-Kitti360 [31].

planning — measuring spatial accuracy on the horizontal plane using IoU$^{\text{BeV}}$ and mIoU$^{\text{BeV}}$ that capture different aspects of spatial understanding.

For this study, we compute the orthographic BeV image for both the prediction and the ground truth, following the rendering procedure of Sec. 3.3 for ground truth. The class assigned to the pixel p is the one corresponding to the maximal value of C_p. Then, we compute IoU (binary occupancy, full vs empty) and mIoU (semantic occupancy) by comparing the two images.

Our analysis is presented in Tab. 8. We observe that the use of GaussRender enhances both metrics simultaneously, with systematic gains associated with different combinations of datasets and models. Both TPVFormer and SurroundOcc show significant improvements across all datasets: Occ3d-nuScenes and SurroundOcc-nuScenes and evaluations. This evaluation highlights that the use of GaussRender not only improves 3D occupancy predictions but also enhances consistency with BeV and sensor observations.

E. Ablations

E.1. Gaussian scaling

An important parameter in our rendering process is the fixed size of the Gaussians representing voxels. To study its impact, we train a TPVFormer [15] model on Occ3d-nuScenes [50], varying the Gaussian scale for both ground-truth and predicted renderings. We train models using only the 2D rendering losses (Eq. 6), excluding the usual 3D voxel losses to isolate the effect of scale on rendering metrics.

Our results, shown in Fig. 4, highlight the importance of the Gaussian scale. If the Gaussians are too large, only a few will cover the image, and the loss will be backpropagated mainly from the nearest ones. If they are too small, gaps appear between voxels, leading to sparse activations and a model that renders mostly the empty class, yielding poor metrics.

Theoretically, the optimal size should correlate with the voxel size. For Occ3d-nuScenes and SurroundOcc-nuScenes, the optimal scale is $s = 0.25$ and $s = 0.20$, while for SSCBench-KITTI360, it is $s = 0.1$. This aligns with our intuition: a voxel should be represented by a spherical Gaussian with a standard deviation such that $2s = c$, where c is the voxel side.

Qualitatively, Fig. 4 shows the effect of scale on rendering, confirming the need for a balanced Gaussian size to avoid either sparse activation or excessive concentration on nearby elements.

E.2. Loss balance

We investigate the importance of the balance λ between the 2D loss and the 3D loss. If the weight of the 2D loss is too high, there is a risk of optimizing the image rendering at the expense of voxel predictions. Conversely, if the 2D loss is too low, its contribution to the learning process may be overlooked. To analyze this, we vary the contribution λ of the 2D loss and study the impact on the final metrics, as reported in Fig. 6. Based on the training results of TPV-Former [15] on a subset of Occ3D-nuScenes [50], we set

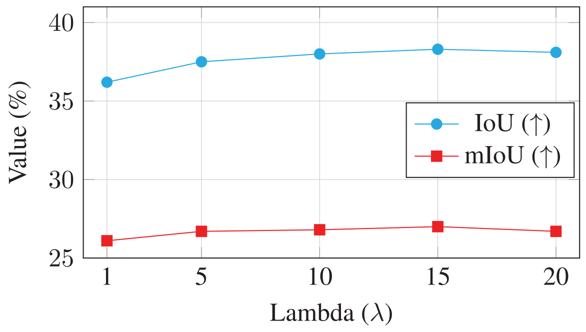


Figure 6. **Impact of different the contribution λ of L_{2D} on 3D semantic occupancy performance.** The architecture used is TPVFormer [15]. Models are trained using a combination of L_{2D} and L_{3D} with varying λ values and evaluated on 3D IoU and mIoU. We train and evaluate the models on 20% of Occ3D-nuScenes [50] training and validation datasets.

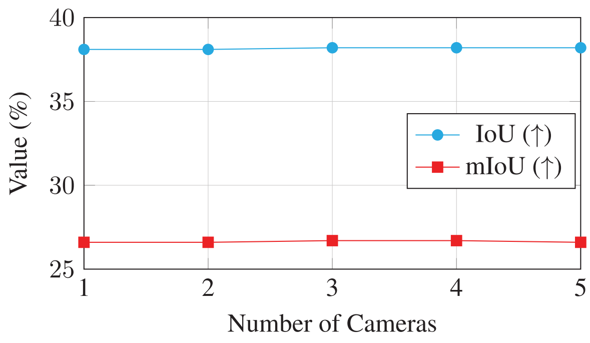


Figure 7. **Impact of the number of cameras on 3D semantic occupancy performance.** The architecture used is TPVFormer [15]. Models are trained with varying numbers of cameras and evaluated on 3D IoU and mIoU using 20% of Occ3d-nuScenes training and validation set.

the weight to $\lambda = 15$.

E.3. Cameras

For a given input, we can position as many cameras as needed to render multiple views. In this experiment, we explore using multiple cameras by selecting from the six available views. While, in theory, more cameras could provide more accurate gradients, we observe in practice that it does not significantly impact the final results (Fig. 7). Since additional cameras introduce computational overhead, we opt to render from a single camera per iteration, changing its position across batches according to the strategy defined in Sec. 3.2.

E.4. Camera strategies

- **Sensor Strategy:** Cameras are placed at the original sensor locations and orientations as provided in the dataset.
- **Elevated Strategy:** Each camera is lifted and tilted downward. This modification increases the vertical field of view, providing a top-down perspective that reduces self-occlusion and captures a broader context of the scene.
- **Elevated + Around Strategy:** This strategy combines elevation and downward tilt with additional random displacements around the ego vehicle—up to half the maximum scene range. It allows observing the scene from novel angles while maintaining a consistent overhead viewpoint, improving the visibility of occluded voxels.
- **Fully Random Strategy:** Cameras are randomly placed throughout the scene, applying pitch and yaw perturbations and varying distances from the ego-vehicle. While this increases the diversity of viewpoints, it also introduces inconsistency and often places cameras in less informative positions (e.g., viewing empty space).
- **Dynamic Strategy:** Cameras are elevated and positioned at random distances along a circular ring. However, each camera is oriented to look at the element with the highest 3D cross-entropy. In other words, the camera focuses on a region where it made the largest prediction error. It appears that its supervision signal does not help much the training.

F. Scores detailed per class

The following tables give the detailed IoU and mIoU scores of the models studied for each dataset. Tab. 10 concerns the Occ3D-nuScenes dataset [50], Tab. 11 the SSCBench-KITTI360 dataset [31] and Tab. 9 the SurroundOcc-nuScenes dataset [54]. The variations by class appear to be due to learning variance, which is why it makes more sense to look at the overall IoU and RayIoU metrics, rather than looking for intrinsic reasons.

G. Qualitative results

In Fig. 8 and Fig. 9, we respectively present qualitative results on randomly selected scenes from SurroundOcc-nuScenes [54] and Occ3d-nuScenes datasets [50]. We also provide complete gifs in our github.

Model	IoU	mIoU	barrier ■	bicycle ■	bus ■	car ■	const. veh. ■	motorcycle ■	pedestrian ■	traffic cone ■	trailer ■	truck ■	drive. suf. ■	other flat ■	sidewalk ■	terrain ■	manmade ■	vegetation ■
MonoScene [4]	23.96	7.31	4.03	0.35	8.00	8.04	2.90	0.28	1.16	0.67	4.01	4.35	27.72	5.20	15.13	11.29	9.03	14.86
Atlas [42]	28.66	15.00	10.64	5.68	19.66	24.94	8.90	8.84	6.47	3.28	10.42	16.21	34.86	15.46	21.89	20.95	11.21	20.54
BEVFormer [32]	30.50	16.75	14.22	6.58	23.46	28.28	8.66	10.77	6.64	4.05	11.20	17.78	37.28	18.00	22.88	22.17	13.80	22.21
TPVFormer-lidar [15]	11.51	11.66	16.14	7.17	22.63	17.13	8.83	11.39	10.46	8.23	9.43	17.02	8.07	13.64	13.85	10.34	4.90	7.37
OccFormer [61]	31.39	19.03	18.65	10.41	23.92	30.29	10.31	14.19	13.59	10.13	12.49	20.77	38.78	19.79	24.19	22.21	13.48	21.35
GaussianFormer [18]	29.83	19.10	19.52	11.26	26.11	29.78	10.47	13.83	12.58	8.67	12.74	21.57	39.63	23.28	24.46	22.99	9.59	19.12
GaussianFormerv2 [16]	30.56	20.02	20.15	12.99	27.61	30.23	11.19	15.31	12.64	9.63	13.31	22.26	39.68	23.47	25.62	23.20	12.25	20.73
TPVFormer [15]	30.86	17.10	15.96	5.31	23.86	27.32	9.79	8.74	7.09	5.20	10.97	19.22	38.87	21.25	24.26	23.15	11.73	20.81
w/ GaussRender	32.05	**20.85**	20.2	13.06	**28.95**	30.96	**11.26**	**16.69**	13.64	10.57	12.77	22.58	40.69	23.49	**26.41**	24.97	14.41	22.94
(gain)	1.19	3.75	4.24	7.75	5.09	3.64	1.47	7.95	6.55	5.37	1.80	3.36	1.82	2.24	2.15	1.82	2.68	2.13
SurroundOcc [54]	31.49	20.30	20.59	11.68	28.06	30.86	10.70	15.14	**14.09**	**12.06**	**14.38**	22.26	37.29	23.70	24.49	22.77	14.89	21.86
w/ GaussRender	**32.61**	20.82	**20.32**	**13.22**	28.32	**31.05**	10.92	15.65	12.84	8.91	13.29	**22.76**	**41.22**	**24.48**	26.38	**25.20**	**15.31**	**23.25**
(gain)	1.12	0.52	-0.27	1.54	0.26	0.19	0.22	0.51	-1.25	-3.15	-1.09	0.50	3.93	0.78	1.89	2.43	0.42	1.39

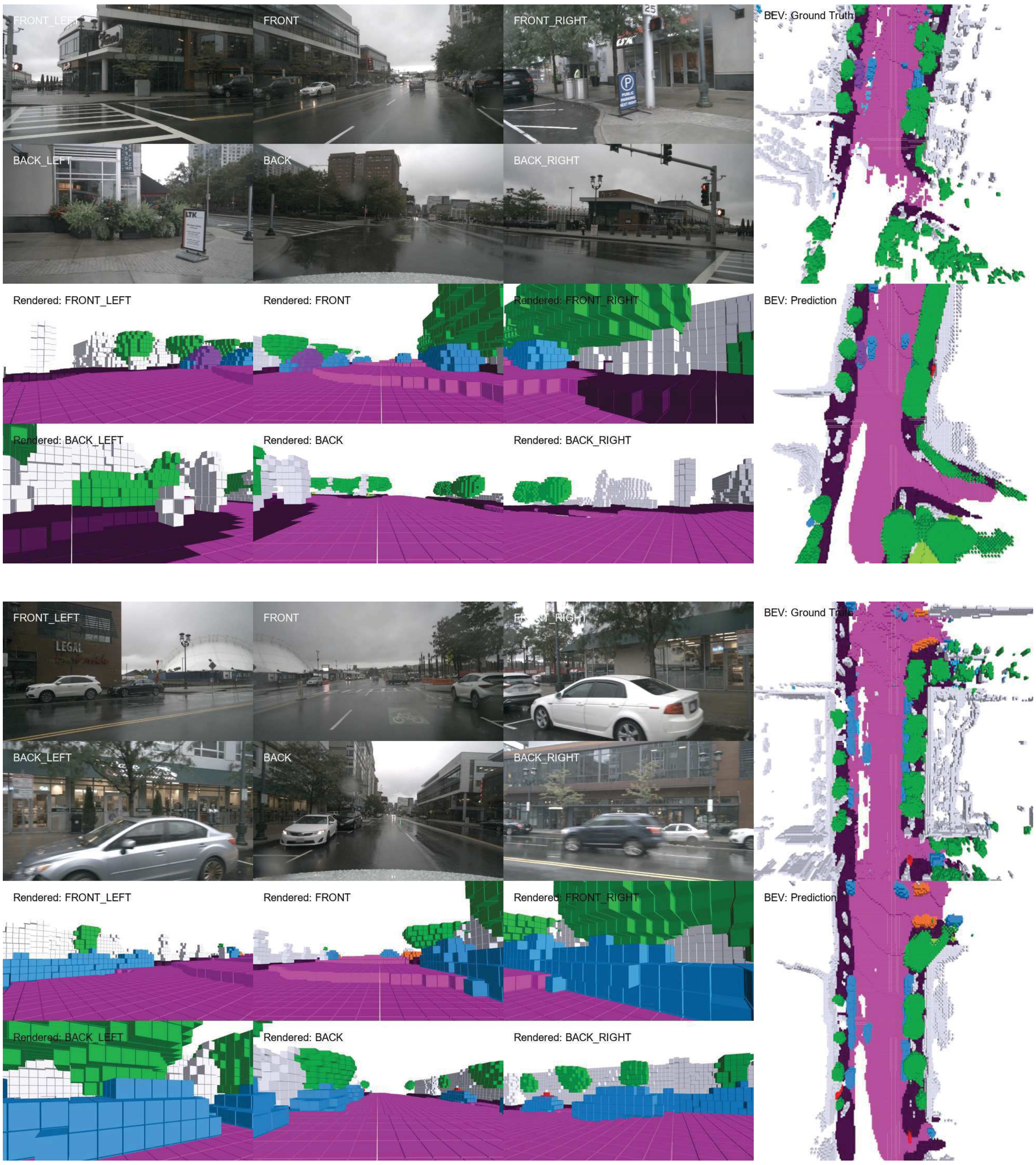
Table 9. **Semantic voxel occupancy results on the SurroundOcc-NuScenes [54] validation set.** The best results are in bold. Training models with our module GaussRender achieves state-of-the-art performance. Previous results are reported from [16].

Method	Input	mIoU	others ■	barrier ■	bicycle ■	bus ■	car ■	const. veh. ■	motorcycle ■	pedestrian ■	traffic cone ■	trailer ■	truck ■	drive. suf. ■	other flat ■	sidewalk ■	terrain ■	manmade ■	vegetation ■
MonoScene	Voxels	6.06	1.75	7.23	4.26	4.93	9.38	5.67	3.98	3.01	5.90	4.45	7.17	14.91	6.32	7.92	7.43	1.01	7.65
BEVDet	Voxels	19.38	4.39	30.31	0.23	32.26	34.47	12.97	10.34	10.36	6.26	8.93	23.65	52.27	24.61	26.06	22.31	15.04	15.10
OccFormer	Voxels	21.93	5.94	30.29	12.32	34.40	39.17	14.44	16.45	17.22	9.27	13.90	26.36	50.99	30.96	34.66	22.73	6.76	6.97
BEVStereo	Voxels	24.51	5.73	38.41	7.88	38.70	41.20	17.56	17.33	14.69	10.31	16.84	29.62	54.08	28.92	32.68	26.54	18.74	17.49
BEVFormer	Voxels	26.88	5.85	37.83	17.87	40.44	42.43	7.36	23.88	21.81	20.98	22.38	30.70	55.35	28.36	36.0	28.06	20.04	17.69
CTF-Occ	Voxels	28.53	8.09	39.33	20.56	38.29	42.24	16.93	24.52	22.72	21.05	22.98	31.11	53.33	33.84	37.98	33.23	20.79	18.0
RenderOcc	Lidar	23.93	5.69	27.56	14.36	19.91	20.56	11.96	12.42	12.14	14.34	20.81	18.94	**68.85**	33.35	42.01	43.94	17.36	22.61
RenderOcc	Voxels+Lidar	26.11	4.84	31.72	10.72	27.67	26.45	13.87	18.2	17.67	17.84	21.19	23.25	63.2	**36.42**	**46.21**	**44.26**	19.58	20.72
TPVFormer	Voxels	27.83	7.22	38.90	13.67	40.78	45.90	17.23	19.99	18.85	14.30	**26.69**	**34.17**	55.65	35.47	37.55	30.70	19.40	16.78
w/ GaussRender	Voxels	30.48	9.84	42.3	24.09	41.79	46.49	18.22	25.85	25.06	22.53	22.9	33.34	58.86	33.19	36.57	31.84	23.55	21.8
(gain)		2.65	2.62	3.40	10.42	1.01	0.59	0.99	5.86	6.21	8.23	-3.79	-0.83	3.21	-2.28	-0.98	1.14	4.15	5.02
SurroundOcc	Voxels	29.21	8.64	40.12	23.36	39.89	45.23	17.99	24.91	22.66	18.11	21.64	32.5	57.6	34.1	35.68	32.54	21.27	20.27
w/ GaussRender	Voxels	30.38	8.87	40.98	23.25	43.76	46.37	19.49	25.2	23.96	19.08	25.56	33.65	58.37	33.28	36.41	33.21	22.76	22.19
(gain)		1.17	0.23	0.86	-0.11	3.87	1.14	1.50	0.29	1.30	0.97	3.92	1.15	0.77	-0.82	0.73	0.67	1.49	1.92

Table 10. **Semantic voxel occupancy results on the Occ3D-nuScenes [50] validation set.** The best results are in bold. Training models with our module GaussRender achieves state-of-the-art performance. Previous results are reported from [16, 43].

Method	IoU	mIoU	car ■	bicycle ■	motorcycle ■	truck ■	other veh. ■	person ■	road ■	parking ■	sidewalk ■	other grnd. ■	building ■	fence ■	vegetation ■	terrain ■	pole ■	traf.-sign ■	other struct. ■	other obj. ■
MonoScene *	37.87	12.31	19.34	0.43	0.58	8.02	2.03	0.86	48.35	11.38	28.13	3.32	32.89	3.53	26.15	16.75	6.92	5.67	4.20	3.09
VoxFormer *	38.76	11.91	17.84	1.16	0.89	4.56	2.06	1.63	47.01	9.67	27.21	2.89	31.18	4.97	28.99	14.69	6.51	6.92	3.79	2.43
OccFormer *	40.27	13.81	22.58	0.66	0.26	9.89	3.82	2.77	54.30	13.44	31.53	3.55	36.42	4.80	31.00	**19.51**	7.77	8.51	6.95	4.60
SurroundOcc	38.51	13.08	21.31	0.0	0.0	6.05	4.29	0.0	53.88	12.56	30.89	2.57	34.93	3.59	29.03	16.98	5.61	6.66	4.39	2.62
w/ GaussRender	38.62	13.34	21.61	0.0	0.0	6.75	4.5	0.0	53.64	11.93	30.24	2.67	35.01	4.55	29.81	17.32	6.19	8.49	4.8	2.59
(gain)	0.11	0.26	0.30	0.0	0.0	0.70	0.21	0.0	-0.24	-0.63	-0.65	0.10	0.08	0.96	0.78	0.34	0.58	1.83	0.41	-0.03
Symphonies (official checkpoint)	43.40	17.82	26.86	4.21	4.92	14.19	7.67	**16.79**	57.31	13.60	35.25	4.58	39.20	**7.96**	34.23	19.20	8.22	**16.79**	6.03	6.03
w/ GaussRender	**44.08**	**18.11**	**27.37**	3.24	**5.12**	**14.69**	**8.76**	16.70	**58.05**	**13.87**	**35.70**	**4.76**	**40.09**	7.88	**34.76**	19.20	**8.22**	16.49	**8.64**	**6.50**
(gain)	+0.68	+0.29	+0.51	-0.97	+0.20	+0.50	+1.09	-0.09	+0.74	+0.27	+0.45	+0.18	+0.89	-0.08	+0.53	0.00	0.00	-0.30	+2.61	+0.47

Table 11. **Semantic voxel occupancy results on the SSCBench-KITTI360 [31] test set.** The best results are in bold. Training models with our module GaussRender achieves state-of-the-art performance. Previous results are reported from [19].



⬛ ego vehicle 🟪 driveable surface 🟦 car 🟨 bus 🟪 truck 🟩 terrain 🟩 vegetation 🟪 sidewalk ⬜ other flat 🟥 pedestrian ⬜ bicycle

⬜ manmade 🟨 motorcycle 🟧 barrier 🟦 construction vehicle 🟫 trailer 🟨 traffic cone

Figure 8. **Qualitative predictions of a SurroundOcc [54] model trained with GaussRender on the SurroundOcc-nuScenes [54] dataset.** We display the six input camera images (top left), the rendered predictions (bottom left), the BeV ground-truth (top right) and BeV prediction (bottom left). The scene is randomly selected from the validation set and we show predictions at two different timesteps.

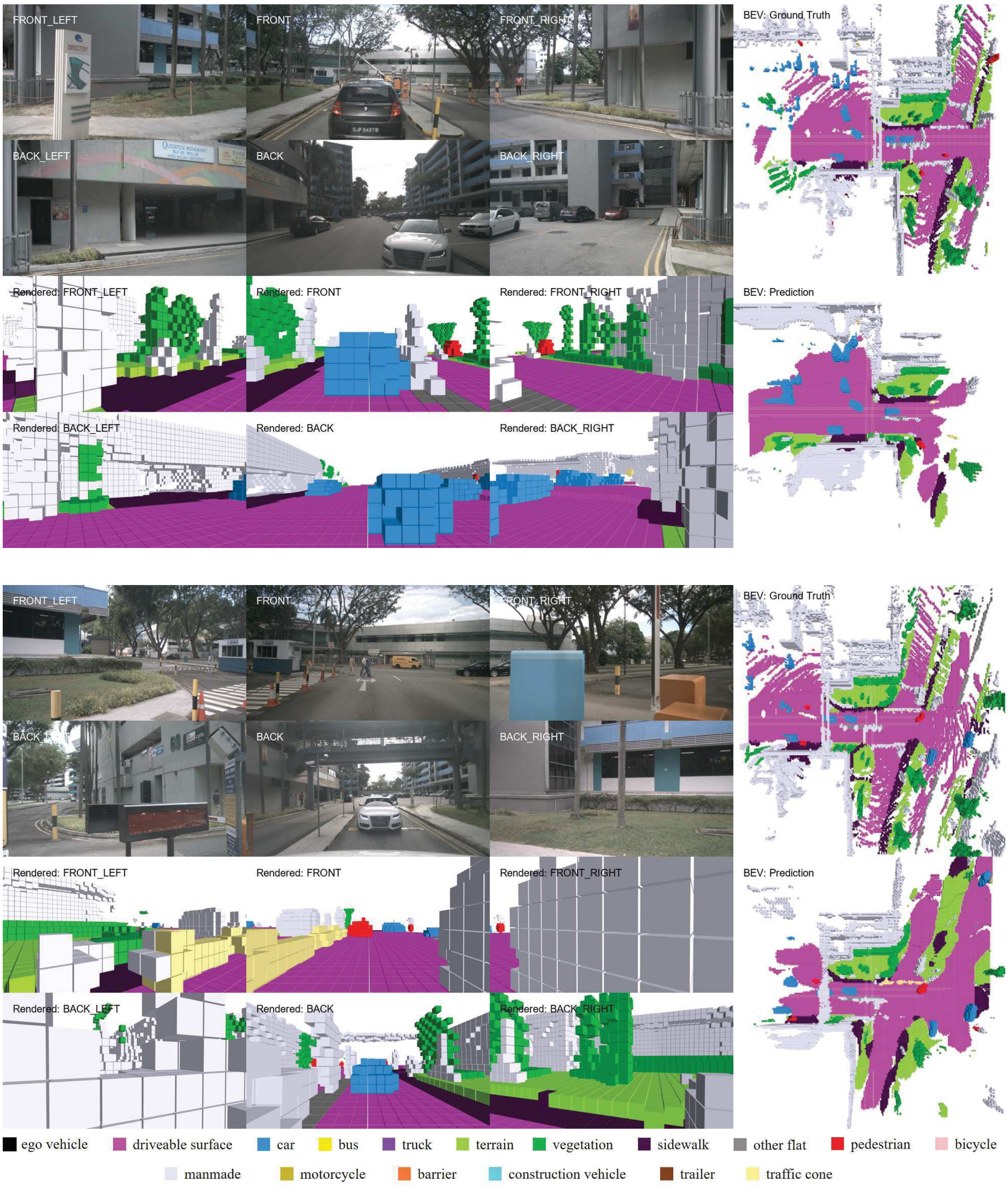


Figure 9. **Qualitative predictions of a TPVFormer [15] model trained with GaussRender on the Occ3d-nuScenes [50] dataset.** We display the six input camera images (top left), the rendered predictions (bottom left), the BeV ground-truth (top right) and BeV prediction (bottom left). The scene is randomly selected from the validation set and we show predictions at two different timesteps.