

GaussRender: Learning 3D Occupancy with Gaussian Rendering

Loïck Chambon^{1,2} Eloi Zablocki² Alexandre Boulch² Mickaël Chen³ Matthieu Cord^{1,2}

¹ValeoAI, Paris, France. ²Sorbonne University, Paris, France. ³Hcompany.ai, Paris, France.

Abstract

Understanding the 3D geometry and semantics of driving scenes is critical for safe autonomous driving. Recent advances in 3D occupancy prediction have improved scene representation but often suffer from visual inconsistencies, leading to floating artifacts and poor surface localization. Existing voxel-wise losses (e.g., cross-entropy) fail to enforce visible geometric coherence. In this paper, we propose GaussRender, a module that improves 3D occupancy learning by enforcing projective consistency. Our key idea is to project both predicted and ground-truth 3D occupancy into 2D camera views, where we apply supervision. Our method penalizes 3D configurations that produce inconsistent 2D projections, thereby enforcing a more coherent 3D structure. To achieve this efficiently, we leverage differentiable rendering with Gaussian splatting. GaussRender seamlessly integrates with existing architectures while maintaining efficiency and requiring no inference-time modifications. Extensive evaluations on multiple benchmarks (SurroundOcc-nuScenes, Occ3D-nuScenes, SSCBench-KITTI360) demonstrate that GaussRender significantly improves geometric fidelity across various 3D occupancy models (TPVFormer, SurroundOcc, Symphonies), achieving state-of-the-art results, particularly on surface-sensitive metrics such as RayIoU. The code is open-sourced at <https://github.com/valeoai/GaussRender>.

1. Introduction

Understanding the 3D geometry and semantics of driving scenes from multiple cameras is both a fundamental challenge and a critical requirement for autonomous driving. This problem is central to perception tasks such as object detection [21, 29, 32, 35, 38, 63], agent forecasting [6, 8, 25, 44, 45, 56, 57, 60], and scene segmentation [1, 5, 9–12, 47]. 3D occupancy prediction [15, 19, 33, 52, 54] has emerged as a task to evaluate how well models capture the spatial structure and semantics of a scene.

The challenge in 3D occupancy prediction lies in achieving geometrically coherent reasoning from multi-view im-

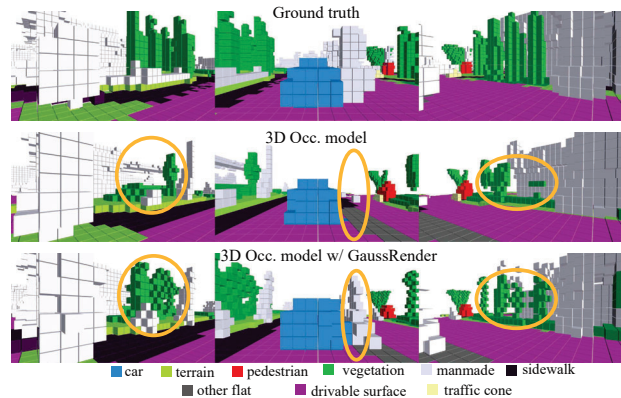


Figure 1. **Comparison of rendered 3D predictions.** Standard 3D Occupancy models trained on per-voxel losses result in physically implausible predictions (e.g., floating voxels, poorly localized surfaces, highlighted with orange ellipses) that maintain high 3D IoU but fail to produce visually consistent predictions. GaussRender enforces multi-view consistency, eliminating artifacts through learning projective constraints.

ages. Existing methods [15, 16, 30, 54] typically optimize per-voxel predictions using standard 3D losses, such as cross-entropy, Dice [48], or Lovász [2]. However, these losses treat all voxels equally and do not enforce spatial consistency between neighboring voxels, leading to visual artifacts, as noticed in prior works [36] and illustrated in Fig. 1. Such artifacts (floating voxels, disjoint surfaces, and misaligned boundaries) have a low impact on voxel-based segmentation losses, which give the same weight to all voxels. The consequences extend beyond mere visual artifacts: poor surface localization and unrealistic floating objects may significantly undermine downstream tasks such as free-space estimation or motion planning. The underlying issue is that conventional supervision methods lack mechanisms to penalize unrealistic spatial arrangements, a deficiency that becomes especially apparent when projecting 3D artifacts into 2D views. This observation motivates our key insight: integrating projective consistency into the training objective encourages the model to learn consistent, physically and visually plausible geometries.

We introduce GaussRender, a module that bridges the gap through differentiable rendering of 3D occupancy predictions. Our key idea is to enforce spatial consistency by projecting both predicted and ground-truth voxels into 2D camera views using Gaussian splatting [23]. These projections serve, during training, as a supervision signal, allowing us to penalize 3D configurations that produce poor 2D projections, thereby enforcing a more coherent and geometrically plausible 3D structure. To achieve this, we apply two complementary supervision signals: (1) a semantic rendering loss that enforces local semantic coherence, and (2) a depth rendering loss that penalizes occlusion-disrupting artifacts. These rendering-based losses are applied alongside standard 3D supervision, such as cross-entropy, ensuring compatibility with existing occupancy learning frameworks. GaussRender enables rendering from arbitrary viewpoints. Its flexibility helps reduce occlusions, for instance, by leveraging elevated viewpoints that are less affected by horizontal obstructions from ground objects.

We validate our approach on multiple datasets, including SurroundOcc-nuScenes [54], Occ3D-nuScenes [50], and SSCBench-KITTI360 [31]. Our experiments demonstrate that GaussRender significantly improves geometric fidelity across diverse architectures — ranging from multi-scale voxel-based models (SurroundOcc [54]), tri-plane models (TPVFormer [15]) to hybrid query-voxel models (Symphonies [19]). Across all settings and datasets, GaussRender consistently improves performance on classical metrics such as IoU and mIoU. Moreover, when evaluated using surface- and artifact-sensitive metrics, such as RayIoU [36], the improvements are even more pronounced. This is because the projective constraints enforced by our method promote surface continuity — ensuring that neighboring voxels agree when rendered from any viewpoint, thereby eliminating floating artifacts and discontinuities. Crucially, our approach is plug-and-play, integrating seamlessly with existing 3D occupancy frameworks without requiring any architectural modifications. Thanks to our efficient Gaussian rendering proxy, GaussRender incurs minimal computational and memory overhead compared to NeRF-based alternatives [17, 43].

The key contributions include:

- A rendering-based module that enforces semantic and geometric consistency in 3D occupancy prediction, eliminating visual and spatial artifacts through projective constraints,
- A fast and memory-efficient loss implementation-based on Gaussian splatting without architectural modifications during training and having no impact during inference,
- A camera positioning strategy that amplifies supervision signals in geometrically complex regions.
- Improves results on three standard benchmarks for many models, with particular gains in RayIoU.

2. Related work

2.1. Learning 3D semantic geometry from cameras

Reconstructing unified 3D scenes from multi-camera systems must address three interconnected challenges: multi-view cameras create perspective limitations requiring occlusion reasoning, lifting 2D image features to metric 3D space demands geometric and semantic 2D-to-3D transfer, and merging multi-view inputs into volumetric representations like voxels incurs cubic memory growth [55]. Modern methods address these challenges through structured intermediate representations.

Discrete methods rely on regular 3D grids to stay as close as possible to the desired voxel output. Bird’s-eye-view (BeV) projections [27, 51] significantly reduce memory cost but introduce a substantial bias in the intermediate representation due to height compression. Instead, tri-plane features [15] and tensor decompositions [62] map any 3D coordinate onto distinct planes or vectors performing interpolation to get distinct features. As opposed to BeV approaches, they do not suffer from height compression while still operating on a compressed representation. Octrees [39] preserves a full multi-scale 3D representation, coarse in uniform areas and fine where details are needed, resulting in a fast and memory-efficient representation [54]. Continuous architectures have emerged, using a set of Gaussians to represent the scene [16, 18] which are then discretized to obtain the voxelized output map. Furthermore, to better capture the overall instance semantics and scene context, query-based methods have been introduced [19]. They facilitate interactions between image and volume features using sparse instance queries.

Complementary approaches have been proposed to enhance 3D occupancy predictions beyond architectural choices. Temporal aggregation uses past frames to resolve occlusions and refine geometric details [26, 27, 43, 46, 59], with extensions to 4D forecasting for dynamic scenes [24, 40, 53, 58]. Self-supervised methods reduces dependency on 3D annotations by generating pseudo-labels from monocular depth and segmentation [7, 17], but suffer from scale miscalibrations and label mismatches between 2D and 3D labels. Lastly, supervised rendering with lidar reprojections [43, 49] has been introduced to provide accurate depths. While more precise than pseudo-annotations, lidar reprojections face challenges such as signal sparsity, occlusions, and misalignment between the lidar and cameras. In particular, [43, 49] require temporal supervision from adjacent frames to compensate for lidar sparsity and occlusions. In contrast, our method achieves accurate supervision without these constraints, allowing more flexible and efficient 3D occupancy learning across different architectures.

Despite architectural variations, all methods ultimately produce voxel grids for supervision. This common output

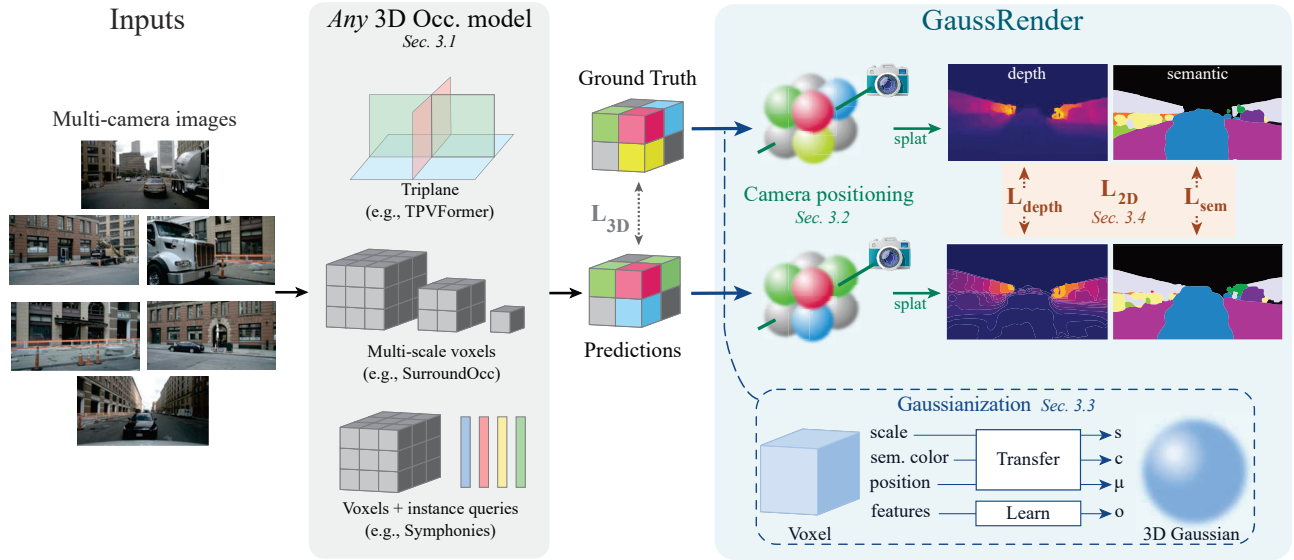


Figure 2. **Overview of GaussRender.** Our module enforces 3D-2D consistency via differentiable Gaussian rendering. First, both predicted and ground-truth voxel grids are ‘gaussianized’ by converting each voxel into a simple spherical Gaussian: the center μ is fixed at the voxel center, the scale s is a simple fixed scaling of the original voxel dimensions, and features are directly transferred as the semantic ‘classes’ c — with only the opacity o learned when voxel features are available. Next, virtual cameras are positioned in the scene (a fixed bird’s-eye view and a dynamic, arbitrarily placed camera as described in Sec. 3.2). The resulting 3D Gaussians are then projected into 2D using Gaussian splatting (Sec. 3.3), producing both semantic and depth renderings. These rendered views are compared against their ground-truth counterparts using an L1 penalty, ensuring enhanced spatial coherence and geometric consistency (Sec. 3.4).

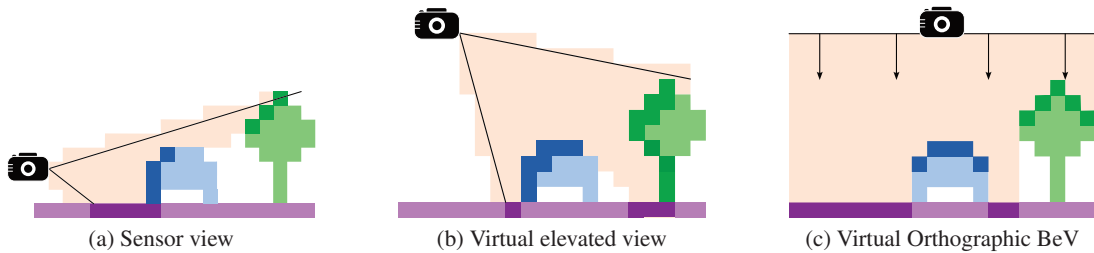


Figure 3. **Arbitrary camera positioning.** Virtual cameras can be freely placed in the scene, enabling view constraints in occluded regions. The sensors’ cameras do not provide access to the rear of objects, leading to occlusion issues. GaussRender can place the cameras arbitrarily within the scene, allowing them to be elevated to reach previously hidden areas.

space lets GaussRender enhance any 3D occupancy model with a simple rendering loss applied to the voxel output.

2.2. Differentiable rendering of 3D representations

Supervising 3D predictions through 2D projections requires efficient differentiable rendering methods. While traditional differentiable rendering methods handle 3D modalities like point clouds and meshes [22, 37], recent approaches focus on neural rendering and Gaussian-based methods.

NeRF-based methods [41] perform volume rendering by predicting a volumetric density and applying ray-based integration. RenderOcc [43] constructs a NeRF-style volume representation supervised by semantic LiDAR projections.

It leverages lidar-derived 2D labels (depth and semantics) for 3D supervision but inherits lidar’s sparsity. SelfOcc [17] uses signed distance fields for occupancy prediction and applies differentiable volume rendering to synthesize depth and semantic views. Self-supervision is enforced using multi-view consistency from video sequences. However, NeRF-based rendering is computationally expensive, especially with high ray-sampling resolutions. Additionally, its reliance on image quality and occlusion mitigation necessitates auxiliary supervision from temporal frames [43, 49].

Gaussian splatting provides an efficient alternative to NeRF by representing 3D scenes as a set of Gaussians [23]. In 3D semantic occupancy prediction, GaussianOcc

[7] projects a voxel-based Gaussian representation camera, using adjacent views for self-supervision, with optional 2D segmentation to refine the predicted occupancy. GSRender [49] uses lidar reprojection and temporal consistency to help the learning process. GaussTR [20] extends Gaussian representation learning with foundation models (e.g., CLIP, Grounded-SAM) to enable open-vocabulary occupancy prediction. Unlike other self-supervised methods, it does not use temporal supervision but relies on heavy pre-trained vision models. A key limitation of these approaches is their tight coupling of Gaussian representations with the model architecture, imposing a fixed representation that limits flexibility. In contrast, our approach introduces a Gaussian rendering loss applicable to any model, without requiring the underlying 3D representation to be Gaussian.

All aforementioned methods are restricted to rendering from recorded camera perspectives, as they require RGB images to estimate pseudo-labels [7, 17, 20] or rely on semantic LiDAR reprojections [43, 49]. In contrast, as GaussRender renders both the ground truth and the predictions, it is not constrained by fixed camera perspectives or temporal supervision, enabling rendering from any viewpoint. Moreover, GaussRender does not enforce a specific 3D feature representation, and operates at prediction-level, making it adaptable to diverse architectures.

3. GaussRender

We present GaussRender, a plug-and-play rendering module that enhances 3D occupancy models through efficient, differentiable Gaussian rendering. First, we define 3D-2D consistency and how to enforce it during training (Sec. 3.1). We then outline our camera placement strategy (Sec. 3.2) and describe the rendering process, which projects 3D voxels into 2D images using Gaussian splatting (Sec. 3.3). Finally, we detail our 2D rendering loss (Sec. 3.4).

3.1. Enforcing 3D-2D consistency

Vision-to-3D semantic occupancy models [15, 19, 54] take a set of N images, $I = \{I_i\}_{i=1}^N$, and predict a 3D semantic grid $O \in [0, 1]^{X \times Y \times Z \times C}$, where C is the number of semantic classes and (X, Y, Z) defines the spatial resolution. The standard pipeline consists of three steps:

- **Feature Extraction:** Each image is processed by a backbone network to extract 2D features $F = \{F_i\}_{i=1}^N \in \mathbb{R}^{d_{img}}$, where d_{img} is the feature dimension.
- **3D Lifting:** The features are lifted into a 3D representation (e.g., voxels, tri-planes) using cross- and self-attention mechanisms.
- **Voxel Prediction:** The 3D representation is converted into a voxel grid, and standard losses (e.g., cross-entropy, Lovász, Dice) are computed against the ground truth.

While the standard 3D losses (denoted as L_{3D}) ensure overall occupancy alignment with the ground truth, they treat each voxel independently and ignore spatial consistency. This limitation can lead to artifacts such as floating voxels or misaligned surfaces [36]. To overcome these issues, we introduce a 2D rendering loss, L_{2D} , which provides additional supervision by comparing rendered views of the predicted occupancy with corresponding ground-truth images. This loss enforces spatial coherence in the 2D projection, thereby penalizing 3D configurations that produce poor renderings. The overall training loss is defined as:

$$L = L_{3D} + \lambda L_{2D}, \quad (1)$$

where $\lambda \in \mathbb{R}^+$ is a weighting factor for the 2D loss.

In the following subsections, we define L_{2D} by addressing two key aspects: (1) the placement of rendering cameras in the scene (Sec. 3.2), and (2) the differentiable rendering of 3D occupancy via Gaussian splatting (Sec. 3.3).

3.2. Camera Placement Strategy

GaussRender is not restricted to the original sensor or adjacent frames [17, 43, 49], it renders images from arbitrary camera positions within the scene. This flexibility allows us to position virtual cameras anywhere in the 3D space. Fig. 3 illustrates how virtual cameras generate complementary constraints (both semantic and depth) to enhance voxel consistency.

The placement of these virtual cameras is crucial and has been extensively studied. Several strategies have been found depending on the objective:

- **Visible Voxels:** If the goal is to accurately reconstruct only the visible portions of the scene [31, 50], the rendering cameras should be placed close to the original sensor positions, as illustrated in Fig. 3 (a) in order to constraint the visible voxels. This placement ensures that the inferred 3D structure remains consistent with the sensor’s view.
- **Holistic 3D Reconstruction:** For complete 3D reconstruction, including occluded regions [54], cameras should be positioned more diversely. Exploring different viewpoints provides additional supervision for occluded areas, heavily penalizing aberrant configurations. In practice, we generate the virtual views with a simple rule: the camera is (1) elevated along z axis with respect to its original position, (2) randomly translated in a close range to the ego-vehicle on the xy plane. The resulting camera increases the field of view, looking at both visible and occluded parts of the scene (Fig. 3 (b)).

Additionally, given the importance of bird’s-eye view (BeV) understanding in autonomous driving, we systematically include a fixed virtual orthographic BeV camera, illustrated in Fig. 3 (c). The BeV camera offers an orthogonal and complementary perspective, ensuring that objects

are accurately localized on the ground and further enhancing voxel consistency.

In practice, placing a camera involves specifying its extrinsic and intrinsic parameters, respectively defining the viewing transformation $W \in \mathbb{R}^{4 \times 4}$ and the projective transformation $K \in \mathbb{R}^{3 \times 3}$. Our method keeps the intrinsic parameters fixed and only update the extrinsic parameters for each training batch.

3.3. Gaussian rendering

For each camera, once its position is chosen and its intrinsics and extrinsics are set, we render the 3D occupancy into the corresponding view. Our goal is fast, fully differentiable rendering that supports efficient gradient backpropagation. To do it, we adopt a Gaussian splatting approach [23], which is significantly faster than traditional ray-casting while preserving differentiability. Consequently, we represent each voxel as a Gaussian primitive.

Voxel ‘Gaussianization’. To derive a Gaussian from a voxel, we emphasize simplicity to ease the learning process. In practice, we: (1) *use spherical Gaussians*: represent each voxel as a sphere, which removes the need for orientation parameters; (2) *fix the center*: place each Gaussian at the center of its corresponding voxel, eliminating the need to learn an offset; (3) *fix the scale*: set the scale of each Gaussian based on the voxel dimensions.

Concretely, as illustrated in Fig. 2 (bottom right), for each voxel at position $\mu = (x, y, z)$, we create a simple Gaussian primitive with:

- position μ : inherited from voxel grid coordinates.
- scale $S = \text{Diag}(s)$: a diagonal matrix defined by a scalar factor $s \in \mathbb{R}$.
- semantic ‘color’ logits c : taken from the model’s final prediction.
- opacity o : learned from voxel features (or derived from the logit of the empty semantic class when features are absent).
- rotation $R = I$: set to the identity matrix, as spheres require no orientation. With this choice, the Gaussian covariance matrix becomes $\Sigma_{3D} = S^2$.

Gaussian rendering. To relate the 3D Gaussian representation to the 2D image, we project the 3D covariance matrix Σ_{3D} into the image plane using the camera parameters. The projected covariance is given by:

$$\Sigma_{2D} = J \cdot W \cdot \Sigma_{3D} \cdot W^T \cdot J^T, \quad (2)$$

where J is the Jacobian of the affine approximation of the projective transformation K related to the intrinsic parameters (see [23] for details). This 2D covariance defines the shape and spread of each Gaussian in the image, which

directly influences the computation of opacity and transmittance in the following rendering step.

For each pixel p in the 2D projection, the rendering computes its semantic value by aggregating contributions from all projected Gaussians. Specifically, at p , the rendered semantic color value $C_p \in [0, 1]^C$ (C is the number of semantic classes) and the rendered depth $D_p \in \mathbb{R}^+$ are given by:

$$C_p = \sum_{i=1}^N T_i \alpha_i \mathbf{c}_i, \quad D_p = \sum_{i=1}^N T_i \alpha_i \mathbf{d}_i, \quad (3)$$

where:

- N is the total number of Gaussians.
- $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ is the opacity (or alpha blending factor) for the i th Gaussian, with σ_i representing its density and δ_i the distance traversed along the ray.
- $T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$ is the accumulated transmittance from all Gaussians closer to the camera, which accounts for occlusions.
- $\mathbf{c}_i \in [0, 1]^C$ is the ‘color’ probability of the i th Gaussian.
- $\mathbf{d}_i \in \mathbb{R}^+$ is the distance of the i th Gaussian to the projected camera.

We apply the same rendering process to both the predicted semantic occupancy and the 3D ground truth. For ground-truth rendering, we render only occupied voxels by assigning them an opacity of 1 (and 0 for empty voxels), while for predictions we use the learned opacities.

This differentiable and efficient rendering pipeline thus translates the 3D occupancy into 2D views, enabling robust pixel-wise supervision to enforce spatial consistency.

3.4. L_{2D} rendering loss computation

For each virtual camera, note $**$, and the associated pixel set P^* , we render semantic and depth images $I_{\text{sem}}^* = \{C_p, p \in P^*\}$ and $I_{\text{depth}}^* = \{D_p, p \in P^*\}$, from the predicted semantic occupancy, following Eq. 3. Similarly, we obtain \tilde{I}_{sem}^* and $\tilde{I}_{\text{depth}}^*$ from the ground-truth voxels.

To enforce consistency, we compare each predicted rendering against its ground-truth counterpart from the same viewpoint using the L1 distance:

$$L_{\text{depth}}^* = \frac{1}{d_{\text{range}}^*} \|I_{\text{depth}}^* - \tilde{I}_{\text{depth}}^*\|_1, \quad L_{\text{sem}}^* = \|I_{\text{sem}}^* - \tilde{I}_{\text{sem}}^*\|_1, \quad (4)$$

where $d_{\text{range}}^* \in \mathbb{R}^+$ is a normalization factor based on the maximum depth, ensuring scale consistency across different scenes. The per-camera loss is then obtained as the sum of these two terms:

$$L_{2D}^* = L_{\text{depth}}^* + L_{\text{sem}}^*. \quad (5)$$

Thus, the overall 2D rendering loss for our module is:

$$L_{2D} = L_{2D}^{\text{bev}} + L_{2D}^{\text{cam}}. \quad (6)$$

where ‘bev’ is the orthographic BeV camera (top-down perspective) crucial for global scene understanding and ‘cam’ is the dynamically generated camera generated following the strategy outlined in [Sec. 3.2](#), which ensures diverse viewpoints and improves generalization.

This loss ensures that the learned 3D occupancy aligns with both depth and semantic projections, improving the consistency between 3D and 2D representations.

4. Experiments

We evaluate GaussRender on several models and datasets to demonstrate its versatility. We present experimental details in [Sec. 4.1](#), compare our results against state-of-the-art models and datasets in [Sec. 4.2](#), show how GaussRender enhances 3D semantic occupancy predictions from multiple views in [Sec. 4.3](#), and present ablations in [Sec. 4.4](#).

4.1. Data and models

Data. The trainings and evaluations are conducted on three datasets: SurroundOcc-nuScenes [54], Occ3D-nuScenes [50], and SSCBench-Kitti360 [31]. We briefly outline below the specific characteristics of each dataset. More details can be found in [App. A](#).

SurroundOcc-nuScenes is derived from the nuScenes dataset [3], acquired in Boston and Singapore. It aggregates the lidar annotations of nuScenes to create 3D semantic occupancy grids of range $[-50, 50] \times [-50, 50] \times [-5, 3]$ meters with 50cm voxel resolution, with labels corresponding to 17 lidar semantic segmentation classes. This dataset takes into account both visible and occluded voxels. The occluded voxels are obtained by accumulating lidar data over the frames of the whole sequence consequently introducing temporal artifacts over dynamic objects.

Occ3D-nuScenes is also based on the nuScenes dataset. It contains 18 semantic classes and has a 40cm voxel grid of range $[-40, 40] \times [-40, 40] \times [-1, 5.4]$ meters. One major difference with SurroundOcc-nuScenes, is that it only evaluates the voxels visible from the cameras at the current time frame. Thus, it focuses on the geometric and semantic understanding of the visible objects, rather than extrapolating to occluded regions, leading to a simpler task.

SSCBench-Kitti360 [31] is derived from the Kitti360 dataset [34], acquired in Germany. It contains 19 semantic classes and has a 20cm voxel grid of range $[0, 51.2] \times [-25.6, 25.6] \times [-2, 4.4]$ meters, resulting in a very precise semantic of urban scenes. It evaluates both visible and occluded voxels, making the dataset particularly challenging due to its voxel resolution and the presence of occlusions.

Models and training details. We integrate GaussRender into three different models that use different intermediate representations: **SurroundOcc** [54] (multi-scale voxel-

	Surround-Occ nusc. [54]		Occ3D nusc [50]	SSCBench KITT360 [31]	
<i>Evaluation split</i>	<i>val.</i>		<i>val.</i>	<i>test</i>	
Visible voxels only	✗		✓	✗	
Model	IoU	mIoU	mIoU	IoU	mIoU
BEVDet [14]	-	-	19.38	-	-
BEVStereo [28]	-	-	24.51	-	-
RenderOcc [43]	-	-	26.11	-	-
CTF-Occ [50]	-	-	28.53	-	-
GSRender [49]	-	-	29.56	-	-
TPVFormer-lidar [15]	11.51	11.66	-	-	-
MonoScene [4]	23.96	7.31	6.06	37.87	12.31
Atlas [42]	28.66	15.00	-	-	-
GaussianFormer [18]	29.83	19.10	-	-	-
BEVFormer [32]	30.50	16.75	26.88	-	-
GaussianFormerv2 [16]	30.56	20.02	-	-	-
VoxFormer [30]	-	-	-	38.76	11.91
OccFormer [61]	31.39	19.03	21.93	40.27	13.81
TPVFormer [15]	30.86	17.10	27.83	-	-
w/ GaussRender	32.05	20.85	30.48	-	-
	+1.19	+3.75	+2.65	-	-
SurroundOcc [54]	31.49	20.30	29.21	38.51	13.08
w/ GaussRender	32.61	20.82	30.38	38.62	13.34
	+1.12	+0.52	+1.17	+0.11	+0.26
Symphonies [19]	-	-	-	43.40	17.82
w/ GaussRender	-	-	-	44.08	18.11
	-	-	-	+0.68	+0.29

Table 1. **Performance Comparison on Multiple 3D Occupancy Benchmarks.** We report IoU (\uparrow) and mIoU (\uparrow) metrics on SurroundOcc-nuScenes [54], Occ3D-nuScenes [50], and SSCBench-KITT360 [31]. The best results are highlighted in bold. Our module, GaussRender, consistently improves performance when integrated with standard models, achieving state-of-the-art results across all benchmarks. Performance gains introduced by GaussRender are shown in green.

based approach), **TPVFormer** [15] (tri-plane-based approach), and **Symphonies** [19] (voxel-with-instance query-based approach). By doing so we validate the claim that our proposed approach is compatible with any type of architecture. For each combination of models and datasets, we follow the same procedure. By default, we evaluate the original model checkpoints when available; otherwise, we report scores from previous papers if provided or re-train the models for the target dataset. Each model uses the same training settings, following the optimization parameters of [54]. Camera strategy parameters (elevation and translation are detailed in [Sec. 4.4](#)). More technical details can be found in [App. B](#).

4.2. 3D semantic occupancy results

We evaluate GaussRender across multiple models [15, 19, 54] and datasets [31, 50, 54] for 3D semantic occupancy prediction. Our method consistently improves performance

Methods	Labels	2D GT Img.	Train Mem. Overhead	Single Frame	mIoU (↑)
SelfOcc	Pseudo-lbs.	Dense	High	✓	9.3
OccNerf	Pseudo-lbs.	Dense	High	✗	9.5
GaussianOcc	Pseudo-lbs.	Dense	Low	✓	9.9
RenderOcc	Lidar	Sparse	High	✗	19.3 (S) / 23.9 (T)
GaussRender	Voxels	Dense	Low	✓	25.3

Table 2. **Comparison of rendering-based methods on Occ3d-nuScenes under 2D-only supervision.** We report the mIoU on fully trained models. Labels include pseudo-labels, sparse Lidar, or dense voxel supervision. GaussRender achieves the best performance with low memory and using a dense image supervision signal.

without relying on other sensors such as the lidar used in other works [43, 49]. Results are summarized in Tab. 1 and detailed scores by class can be found in App. F.

SurroundOcc-nuScenes [54]. On this dataset, considering visible and occluded voxels, GaussRender brings significant gains to all tested models. As shown in Tab. 1, TPVFormer [15] and SurroundOcc [54] reach the top two ranks. They achieve higher IoU (**+1.2** and **+1.1**) and mIoU (**+3.8** and **+0.5**) compared to their original implementations. Remarkably, GaussRender enables these models to surpass more recent approaches like GaussianFormer and GaussianFormerV2 [16] in both IoU and mIoU, proving that older architectures can achieve state-of-the-art results when their training is enhanced with GaussRender.

Occ3D-nuScenes. On Occ3D-nuScenes [50], GaussRender leads to the top two results: TPVFormer with GaussRender achieves 30.48 mIoU (**+2.65**), ranking first, and SurroundOcc with GaussRender reaches 30.38 mIoU (**+1.17**), ranking second. Notably, our approach outperforms other rendering methods such as RenderOcc [43] and GSRender [49] without requiring lidar inputs for loss computation. In addition, we also evaluate our model using only the rendering losses and compare to other supervised and unsupervised methods, see Tab. 2. While being static (single frame), it outperforms both static (**+6.0** mIoU) and temporal (**+1.4** mIoU) versions of RenderOcc, the latter casting rays from other frames achieving a new state-of-the-art score on 2D supervision for 3D Occupancy on Occ3d-nuScenes. This demonstrates that methods augmented with GaussRender can learn strong 3D representations using only cameras.

SSCBench-Kitti360. On the challenging SSCBench-Kitti360 [34] dataset, integrating GaussRender to SurroundOcc [54] and Symphonies [19] boosts IoU (**+0.11** and **+0.68**) and mIoU (**+0.26** and **+0.29**). Absolute gains are smaller because models in this benchmark achieve tightly clustered performance due to the smaller voxel size making the segmentation task more difficult. However, when we look only at the visible voxels or in the sensor image metrics, we better see the overall improvements as figured

	3D all mIoU / IoU	3D visible mIoU / IoU	Image mIoU / IoU	Depth L1
SurroundOcc [54]	13.08 / 38.51	25.83 / 69.40	34.20 / 90.72	1.394
w/ GaussRender	13.34 / 38.62	27.06 / 71.39	35.86 / 94.55	1.173
	+0.26 / +0.11	+1.23 / +1.99	+1.66 / +3.83	-0.221

Table 3. **Performance improvement of SurroundOcc using GaussRender on KITTI-360.** GaussRender leads to clear gains even on challenging datasets, especially in visible 3D regions.

Models	RayIoU	RayIoU _{1m, 2m, 4m}
RenderOcc [43]	19.5	13.4, 19.6, 25.5
BEVDet-Occ [13]	29.6	23.6, 30.0, 35.1
BEVFormer [32]	32.4	26.1, 32.9, 38.0
BEVDet-Occ-Long [13]	32.6	26.6, 33.1, 38.2
SparseOcc[36] (8f)	34.0	28.0, 34.7, 39.4
FB-Occ [33]	33.5	26.7, 34.1, 39.7
SparseOcc [36] (16f)	36.1	30.2, 36.8, 41.2
TPVFormer [15]	37.2	31.5, 38.1, 41.9
w/ GaussRender	38.3 +1.1	32.3, 39.3, 43.4
SurroundOcc [54]	35.5	29.9, 36.3, 40.1
w/ GaussRender	37.5 +2.0	31.4, 38.5, 42.6

Table 4. **Impact of GaussRender on RayIoU (↑) metrics on the Occ3D-nuScenes validation dataset.** Best results are in bold. Previous results are reported from [36]. RayIoU_{1m, 2m, 4m} refers to the RayIoU with a depth tolerance of 1, 2, or 4 meters.

in Tab. 3. SurroundOcc with GaussRender achieves higher mIoU (**+1.23**) and IoU (**+1.99**) on visible voxels. It clearly appears that GaussRender has strong impact on visible voxels meaning it has removed visibility artefacts.

Our results show that GaussRender consistently enhances various architectures leading to state-of-the-art results, reaches top results without requiring projected lidar annotations, and remains effective across different dataset scales and annotation densities. This demonstrates the significant advantages of GaussRender for 3D semantic occupancy learning.

4.3. Finer-grained multi-view metric analysis

RayIoU. Classical 3D occupancy metrics, such as voxel-wise IoU, treat all voxels equally, often failing to capture inconsistencies in object surfaces and depth localization. This can lead to misleading evaluations, as models may artificially inflate scores by predicting thick or duplicated surfaces [36] rather than accurately reconstructing scene geometry. To address this, we use RayIoU [36], a metric designed to assess 3D occupancy predictions in a depth-aware manner. Instead of evaluating individual voxels, RayIoU casts rays through the predicted 3D volume and determines correctness based on the first occupied voxel the ray intersects. A prediction is considered correct if both the class






Cameras	Sensor	Elevated	Elevated + Around	Fully Rand.	Dynamic
					
2D+3D	25.9	<u>26.0</u>	26.3	25.4	25.8
2D-only	16.1	11.7	<u>12.9</u>	6.9	4.8

Table 5. **Comparison of mIoU with different camera sampling strategies** trained and evaluated on 20% of Occ3D. We evaluate five sampling strategies under 2D-only and 2D+3D supervision. Images illustrate the impact of camera positioning, black dots represent camera centers while red dots represent visible points casted in their frustum.

label and depth fall within a given tolerance. This approach mitigates issues with voxel-level IoU, ensuring that models are rewarded for precise surface localization rather than over-segmentation.

Using GaussRender consistently improves RayIoU across tested architectures, as reported in Tab. 4, highlighting its ability to enhance spatial consistency. Notably, models enhanced with GaussRender achieve state-of-the-art performances with a single frame, outperforming prior works by a significant margin.

4.4. Ablation studies

We conduct our ablations on fixed subsets of the datasets, each representing 20% of the training and validation sets.

4.4.1. Impact of supervising with virtual viewpoints

A key advantage of GaussRender is its ability to render 3D occupancy from arbitrary viewpoints, offering a flexible way to supervise 3D occupancy. To investigate the influence of virtual camera configurations, we evaluate five different camera placement strategies: Sensor, Elevated, Elevated + Around, Fully Random, Dynamic. More details in Appendix E.4. Quantitatively Tab. 5 quantifies the effect of each strategy under both 2D-only and 2D+3D supervision using mIoU. To do this we train a TPVFormer model with GaussRender during 20% of the Occ3d-nuSc training set. Several conclusions can be drawn. Under 2D-only supervision, the **Sensor strategy** leads by a large margin (16.1 mIoU), suggesting that when only 2D supervision is available, aligning virtual views with actual camera positions maximizes consistency and learning efficiency. Both **Fully Random** and **Dynamic** strategies suffer heavily, with performance dropping to 6.9 and 4.8 mIoU, respectively. These results highlight the risk of unstructured camera placement: random viewpoints often observe unoccupied or irrelevant parts of the scene, weakening the supervision signal while dynamic positioning may introduce instability and bad signals during training leading to poor results. Under 2D+3D supervision, we clearly see that what matters is to have complementary viewpoints since the Sen-

Loss Components				SurroundOcc-nuSc [50]	
Cam	Depth	BEV	Bev Depth	IoU (\uparrow)	mIoU (\uparrow)
✓				26.3	14.3
✓	✓			26.8	15.1
✓	✓	✓		27.2	15.6
✓	✓	✓	✓	27.5	16.4

Table 6. **Impact of adding different loss components on 3D semantic occupancy performances.** The architecture used is TPVFormer [15]. Models are trained with different combinations of losses and evaluated on 3D IoU and mIoU. We train the models on 20% of SurroundOcc-nuScenes [50].

sor strategy is beaten by **Elevated + Around** achieving the best result (26.3 mIoU). This indicates that providing diverse and informative viewpoints enhances learning, even when 3D supervision is available.

Overall, these findings emphasize that virtual camera placement is important. If we consider a 2D only training, positioning cameras at sensor location ensures consistency with sensors during training, while when training 2D+3D, we need additional viewpoints to enhance supervision.

4.4.2. Loss increment

Finally, we analyze the contribution of each individual loss component to the final metrics by gradually introducing each term. The results, in Tab. 6, show that each successive addition of a loss component leads to a gradual improvement in performance, justifying the inclusion of each term.

5. Conclusion

GaussRender introduces a novel paradigm for enhancing 3D occupancy prediction through differentiable Gaussian rendering, bridging the gap between voxel-based supervision and projective consistency. By enforcing projective geometric and semantic coherence, it significantly reduces spatial artifacts while maintaining compatibility with diverse architectures. The module’s efficiency and flexibility enable state-of-the-art performance across benchmarks.

Looking ahead, several exciting directions emerge. First, integrating dynamic viewpoint synthesis with temporal sequences could further improve occlusion reasoning. Second, extending Gaussian rendering to open-vocabulary 3D understanding (leveraging vision-language models) might unlock semantic reasoning beyond predefined classes.

6. Acknowledgments

This work was partially supported by the ANR MultiTrans project (ANR-21-CE23-0032). This work was granted access to the HPC resources of IDRIS under the allocation AD011014252R1 made by GENCI. We also thank Yihong Xu for his feedback on the paper and his valuable discussions.

References

- [1] Florent Bartoccioni, Eloi Zablocki, Andrei Bursuc, Patrick Pérez, Matthieu Cord, and Karteek Alahari. Lara: Latents and rays for multi-camera bird's-eye-view semantic segmentation. In *CoRL*, 2022. 1
- [2] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The iovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018. 1
- [3] 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 multi-modal dataset for autonomous driving. In *CVPR*, 2020. 6, 1
- [4] Anh-Quan Cao and Raoul de Charette. Monoscene: Monocular 3d semantic scene completion. In *CVPR*, 2022. 6, 4
- [5] Loïck Chambon, Éloi Zablocki, Mickaël Chen, Florent Bartoccioni, Patrick Pérez, and Matthieu Cord. Pointbev: A sparse approach to bev predictions. In *CVPR*, 2024. 1
- [6] Lan Feng, Mohammadhossein Bahari, Kaouther Mes-saoud Ben Amor, Éloi Zablocki, Matthieu Cord, and Alexandre Alahi. Unitraj: A unified framework for scalable vehicle trajectory prediction. In *ECCV*, 2024. 1
- [7] Wanshui Gan, Fang Liu, Hongbin Xu, Ning kai Mo, and Naoto Yokoya. Gaussianocc: Fully self-supervised and efficient 3d occupancy estimation with gaussian splatting. *arXiv preprint arXiv:2408.11447*, 2024. 2, 4
- [8] Junru Gu, Chenxu Hu, Tianyuan Zhang, Xuanyao Chen, Yilun Wang, Yue Wang, and Hang Zhao. ViP3D: End-to-end visual trajectory prediction via 3d agent queries. In *CVPR*, 2023. 1
- [9] Xingtai Gui, Teng teng Huang, Haonan Shao, Haotian Yao, and Chi Zhang. Fiptr: A simple yet effective transformer framework for future instance prediction in autonomous driving. In *ECCV*, 2024. 1
- [10] Adam W. Harley, Zhaoyuan Fang, Jie Li, Rares Ambrus, and Katerina Fragkiadaki. Simple-BEV: What really matters for multi-sensor bev perception? In *ICRA*, 2023.
- [11] Anthony Hu, Zak Murez, Nikhil Mohan, Sofia Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. FIERY: Future instance segmentation in bird's-eye view from surround monocular cameras. In *ICCV*, 2021.
- [12] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. ST-P3: end-to-end vision-based autonomous driving via spatial-temporal feature learning. In *ECCV*, 2022. 1
- [13] Junjie Huang and Guan Huang. BEVDet4D: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022. 7
- [14] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *CoRR*, 2021. 6
- [15] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *CVPR*, 2023. 1, 2, 4, 6, 7, 8, 3
- [16] Yuanhui Huang, Amonnut Thammatadatrakoon, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, and Jiwen Lu. Gaussianformer-2: Probabilistic gaussian superposition for efficient 3d occupancy prediction. *CoRR*, 2024. 1, 2, 6, 7, 4
- [17] Yuanhui Huang, Wenzhao Zheng, Borui Zhang, Jie Zhou, and Jiwen Lu. Selfocc: Self-supervised vision-based 3d occupancy prediction. In *CVPR*, 2024. 2, 3, 4
- [18] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Gaussianformer: Scene as gaussians for vision-based 3d semantic occupancy prediction. In *ECCV*, 2024. 2, 6, 4
- [19] Haoyi Jiang, Tianheng Cheng, Naiyu Gao, Haoyang Zhang, Tianwei Lin, Wenyu Liu, and Xinggang Wang. Symphonize 3d semantic scene completion with contextual instance queries. In *CVPR*, 2024. 1, 2, 4, 6, 7
- [20] Haoyi Jiang, Liu Liu, Tianheng Cheng, Xinjie Wang, Tianwei Lin, Zhizhong Su, Wenyu Liu, and Xinggang Wang. Gausstr: Foundation model-aligned gaussian transformer for self-supervised 3d spatial understanding. *CoRR*, 2024. 4
- [21] Xiaohui Jiang, Shuailin Li, Yingfei Liu, Shihao Wang, Fan Jia, Tiancai Wang, Lijin Han, and Xiangyu Zhang. Far3d: Expanding the horizon for surround-view 3d object detection. In *EAAI*, 2024. 1
- [22] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018. 3
- [23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 2, 3, 5
- [24] Tarasha Khurana, Peiyun Hu, David Held, and Deva Ramanan. Point cloud forecasting as a proxy for 4d occupancy forecasting. In *CPVR*, 2023. 2
- [25] ByeoungDo Kim, Seong Hyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsu Kim, Jeong Soo Kim, and Jun Won Choi. Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *CVPR*, 2021. 1
- [26] Bohan Li, Jiajun Deng, Wenyao Zhang, Zhuji Liang, Dalong Du, Xin Jin, and Wenjun Zeng. Hierarchical temporal context learning for camera-based semantic scene completion. In *ECCV*, 2024. 2
- [27] Jinke Li, Xiao He, Chonghua Zhou, Xiaoqiang Cheng, Yang Wen, and Dan Zhang. Viewformer: Exploring spatiotemporal modeling for multi-view 3d occupancy perception via view-guided transformers. In *ECCV*, 2024. 2
- [28] Yin hao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. Bevestereo: Enhancing depth estimation in multi-view 3d object detection with temporal stereo. In *EAAI*, 2023. 6
- [29] Yin hao 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 *EAAI*, 2023. 1
- [30] Yiming Li, Zhiding Yu, Christopher B. Choy, Chaowei Xiao, José M. Álvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion. In *CVPR*, 2023. 1, 6

- [31] Yiming Li, Sihang Li, Xinhao Liu, Moonjun Gong, Kenan Li, Nuo Chen, Zijun Wang, Zhiheng Li, Tao Jiang, Fisher Yu, Yue Wang, Hang Zhao, Zhiding Yu, and Chen Feng. Ss-bench: A large-scale 3d semantic scene completion benchmark for autonomous driving. In *IROS*, 2024. 2, 4, 6, 1, 3
- [32] Zhiqi Li, Wenhao Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird’s-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022. 1, 6, 7, 4
- [33] Zhiqi Li, Zhiding Yu, David Austin, Mingsheng Fang, Shiyi Lan, Jan Kautz, and José M. Álvarez. FB-OCC: 3d occupancy prediction based on forward-backward view transformation. *CoRR*, 2023. 1, 7
- [34] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *PAMI*, 2022. 6, 7, 1
- [35] Xuewu Lin, Tianwei Lin, Zixiang Pei, Lichao Huang, and Zhizhong Su. Sparse4d: Multi-view 3d object detection with sparse spatial-temporal fusion. *CoRR*, 2022. 1
- [36] Haisong Liu, Haiguang Wang, Yang Chen, Zetong Yang, Jia Zeng, Li Chen, and Limin Wang. Fully sparse 3d occupancy prediction. In *ECCV*, 2024. 1, 2, 4, 7
- [37] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM*, 2018. 3
- [38] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETR: position embedding transformation for multi-view 3d object detection. In *ECCV*, 2022. 1
- [39] Yuhang Lu, Xinge Zhu, Tai Wang, and Yuexin Ma. Octreeocc: Efficient and multi-granularity occupancy prediction using octree queries. *CoRR*, 2023. 2
- [40] Qihang Ma, Xin Tan, Yanyun Qu, Lizhuang Ma, Zhizhong Zhang, and Yuan Xie. COTR: compact occupancy transformer for vision-based 3d occupancy prediction. In *CVPR*, 2024. 2
- [41] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3
- [42] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*, 2020. 6, 4
- [43] Mingjie Pan, Jiaming Liu, Renrui Zhang, Peixiang Huang, Xiaoqi Li, Hongwei Xie, Bing Wang, Li Liu, and Shanghang Zhang. Renderocc: Vision-centric 3d occupancy prediction with 2d rendering supervision. In *ICRA*, 2024. 2, 3, 4, 6, 7
- [44] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *NeurIPS*, 2022. 1
- [45] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. MTR++: multi-agent motion prediction with symmetric scene modeling and guided intention querying. *TPAMI*, 2024. 1
- [46] Sathira Silva, Savindu Bhashitha Wannigama, Gihan Jayatilaka, Muhammad Haris Khan, and Roshan Ragel. Unified spatio-temporal tri-perspective view representation for 3d semantic occupancy prediction, 2024. 2
- [47] Sophia Sirko-Galouchenko, Alexandre Boulch, Spyros Gi-daris, Andrei Bursuc, Antonín Vobecký, Patrick Pérez, and Renaud Marlet. Occfeat: Self-supervised occupancy feature prediction for pretraining BEV segmentation networks. In *CVPR Workshop*, 2024. 1
- [48] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *DLMIA*, 2017. 1
- [49] Qianpu Sun, Changyong Shu, Sifan Zhou, Zichen Yu, Yan Chen, Dawei Yang, and Yuan Chun. Gsrender: Deduplicated occupancy prediction via weakly supervised 3d gaussian splatting. *CoRR*, 2024. 2, 3, 4, 6, 7
- [50] Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving. In *NeurIPS*, 2023. 2, 4, 6, 7, 8, 1, 3
- [51] Yonglin Tian, Songlin Bai, Zhiyao Luo, Yutong Wang, Yisheng Lv, and Fei-Yue Wang. Mambaocc: Visual state space model for bev-based occupancy prediction with local adaptive reordering. *CoRR*, 2024. 2
- [52] Antonin Vobecky, Oriane Siméoni, David Hurych, Spyridon Gidaris, Andrei Bursuc, Patrick Pérez, and Josef Sivic. Pop-3d: Open-vocabulary 3d occupancy prediction from images. In *NeurIPS*, 2023. 1
- [53] Lening Wang, Wenzhao Zheng, Yilong Ren, Han Jiang, Zhiyong Cui, Haiyang Yu, and Jiwen Lu. Occsora: 4d occupancy generation models as world simulators for autonomous driving. *CoRR*, 2024. 2
- [54] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. *ICCV*, 2023. 1, 2, 4, 6, 7, 3, 5
- [55] Guixing Xu, Wei Liu, Zuotao Ning, Qixi Zhao, Shuai Cheng, and Jiwei Nie. 3d semantic scene completion and occupancy prediction for autonomous driving: A survey. In *CAIT*, 2023. 2
- [56] Yihong Xu, Loïck Chambon, Éloi Zablocki, Mickaël Chen, Alexandre Alahi, Matthieu Cord, and Patrick Pérez. Towards motion forecasting with real-world perception inputs: Are end-to-end approaches competitive? In *ICRA*, 2024. 1
- [57] Yihong Xu, Yuan Yin, Tuan-Hung Vu, Alexandre Boulch, Éloi Zablocki, and Matthieu Cord. PPT: pre-training with pseudo-labeled trajectories for motion forecasting. *CoRR*, 2024. 1
- [58] Yu Yang, Jianbiao Mei, Yukai Ma, Siliang Du, Wenqing Chen, Yijie Qian, Yuxiang Feng, and Yong Liu. Driving in the occupancy world: Vision-centric 4d occupancy forecasting and planning via world models for autonomous driving. *CoRR*, 2024. 2
- [59] Zhangchen Ye, Tao Jiang, Chenfeng Xu, Yiming Li, and Hang Zhao. Cvt-occ: Cost volume temporal fusion for 3d occupancy prediction. In *ECCV*, 2024. 2

- [60] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *ICCV*, 2021. [1](#)
- [61] Yunpeng Zhang, Zheng Zhu, and Dalong Du. Occformer: Dual-path transformer for vision-based 3d semantic occupancy prediction. In *ICCV*, 2023. [6](#), [4](#)
- [62] Linqing Zhao, Xiuwei Xu, Ziwei Wang, Yunpeng Zhang, Borui Zhang, Wenzhao Zheng, Dalong Du, Jie Zhou, and Jiwen Lu. Lowrankocc: Tensor decomposition and low-rank recovery for vision-based 3d semantic occupancy prediction. In *CVPR*, 2024. [2](#)
- [63] Jian Zou, Tianyu Huang, Guanglei Yang, Zhenhua Guo, Tao Luo, Chun-Mei Feng, and Wangmeng Zuo. Unim²ae: Multi-modal masked autoencoders with unified 3d representation for 3d perception in autonomous driving. In *ECCV*, 2024. [1](#)