Seamie Hayes[1], Reenu Mohandas[1], Tim Brophy[1], Ganesh Sistu[1], and Ciaran Eising[1]

[1]Affiliation not available

October 22, 2025

## Abstract

Gaussian splatting and semantic occupancy prediction for autonomous perception share the common goal of achieving accurate 3D scene understanding. Emerging within a similar time frame, the use of 3D Gaussians in occupancy models has gained considerable attention. These models have demonstrated benefits such as reduced latency and a more memory-efficient scene representation compared to dense voxels, with results comparable to previous state-of-the-art approaches. This holds despite the differences between continuous and discrete scene representations when compared with voxelbased methods. This growing area of research builds upon work in Bird's Eye View perception and voxel-driven semantic occupancy prediction, both of which have been the subject of major surveys that outline their development to the present day. In this survey, Gaussian-driven semantic occupancy prediction models are analysed in detail, with particular emphasis on their contributions to the field, design choices, multi-modal research directions, and supervision strategies. In addition, a detailed results table for these models across three notable datasets is compiled to better understand the influence of specific design decisions. We also investigate the distribution of Gaussian parameters at run time to determine whether redundancy exists within these models. Finally, the limitations and future directions of this line of research are explored, providing a clearer view of the strengths and weaknesses of 3D Gaussians in occupancy prediction. In particular, we conclude that the use of Gaussians in occupancy prediction models is positioned to complement voxel-driven methods, namely through Gaussianbased rendering loss for the enforcement of temporal and view consistency across the surround-view cameras.

# 3D Gaussian Representations in Semantic Occupancy Prediction: A Comprehensive Survey and Analysis

Seamie Hayes, Reenu Mohandas, Tim Brophy, Ganesh Sistu, Ciaran Eising

*Abstract*— **Gaussian splatting and semantic occupancy prediction for autonomous perception share the common goal of achieving accurate 3D scene understanding. Emerging within a similar time frame, the use of 3D Gaussians in occupancy models has gained considerable attention. These models have demonstrated benefits such as reduced latency and a more memory-efficient scene representation compared to dense voxels, with results comparable to previous state-of-the-art approaches. This holds despite the differences between continuous and discrete scene representations when compared with voxel-based methods. This growing area of research builds upon work in Bird's Eye View perception and voxel-driven semantic occupancy prediction, both of which have been the subject of major surveys that outline their development to the present day. In this survey, Gaussian-driven semantic occupancy prediction models are analysed in detail, with particular emphasis on their contributions to the field, design choices, multi-modal research directions, and supervision strategies. In addition, a detailed results table for these models across three notable datasets is compiled to better understand the influence of specific design decisions. We also investigate the distribution of Gaussian parameters at run time to determine whether redundancy exists within these models. Finally, the limitations and future directions of this line of research are explored, providing a clearer view of the strengths and weaknesses of 3D Gaussians in occupancy prediction. In particular, we conclude that the use of Gaussians in occupancy prediction models is positioned to complement voxel-driven methods, namely through Gaussian-based rendering loss for the enforcement of temporal and view consistency across the surround-view cameras.**

## I. Introduction

Perception for automated vehicles has advanced significantly over the past decade in both academia and industry. The overarching objective remains the efficient and accurate detection of entities within the scene to ensure safer road usage for all, with the potential to remove the human driver from behind the steering wheel. Achieving this vision requires establishing trust in the system's ability to make safety-critical decisions. To that end, the system must not only detect and track relevant entities but also interpret the broader context of the environment to make informed decisions. This understanding is embodied in what is known as scene representation: a structured and comprehensive abstraction of the surrounding environment that includes static and dynamic elements, their relationships, and predicted behaviours [1]. Scene representation serves as the foundation for downstream tasks such as motion planning and decision-making, and it must be both machine-interpretable and, when necessary, human-interpretable for validation and debugging [2], [3]. In addition, the testing of robustness of sensor modality inputs is critical, as downstream planning and control rely on the accuracy of these modalities; uncertainty in raw sensor pipelines directly undermines driving performance [4]. Concerning the input modalities, cameras provide high-quality semantic information, while LiDAR and radar offer accurate depth measurements, and Doppler radar further contributes velocity information [5]. In academic research, camera–LiDAR fusion remains the dominant choice for peak performance [6], [7], [8], as LiDAR's dense point clouds effectively complement the semantic richness of camera data, surpassing the sparser returns of radar [5]. However, camera–radar fusion has also attracted research attention, particularly with the integration of 4D radar [9], [7], [10]. Referring back to scene representation, three methods have dominated recent research: 3D object detection, Bird's Eye View (BEV) perception, and semantic occupancy prediction.

3D object detection emerged as a method for precise bounding box localisation of objects in 3D space, where even millimetre-level deviations can affect evaluation metrics such as $AP_{3D}$ and $AP_{BEV}$ [11], [12]. This approach excels in detecting large, rigid objects such as vehicles due to their well-defined geometry and size. However, it often struggles with vulnerable road users, such as pedestrians and cyclists, whose smaller and more deformable footprints make accurate bounding box regression more challenging [11]. Furthermore, the discrete, object-centric nature of 3D bounding boxes limits the system's ability to reason about the scene holistically.

To address these limitations, researchers turned toward BEV representations. While early BEV methods lacked the fine-grained spatial precision of 3D bounding boxes, they offered a unified, dense, top-down view of the environment. BEV presents several advantages, including reduced computational cost, easier real-time deployment [13], [14], and greater compatibility with downstream tasks such as

This publication has emanated from research conducted with the financial support of Taighde Éireann – Research Ireland under Grant number 18/CRT/6049. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

Thanks to Deepika-Rani Kaliappan-Mahalingam and Shubham Sonarghare for their feedback and advice regarding Gaussian parameter analysis.

Seamie Hayes and Ciarán Eising are with the Department of Electronic and Computer Engineering, the Research Ireland Centre for Research Training in Foundations in Data Science, and the Data Driven Computer Engineering (D²iCE) Research Centre, all hosted in the University of Limerick, Limerick, V94 T9PX Ireland.

Reenu Mohandas, Tim Brophy, and Ganesh Sistu are with the Department of Electronic and Computer Engineering, and the Data Driven Computer Engineering (D²iCE) Research Centre, University of Limerick, Limerick, V94 T9PX, Ireland.

Corresponding author: Seamie Hayes (e-mail: seamie.hayes@ul.ie)

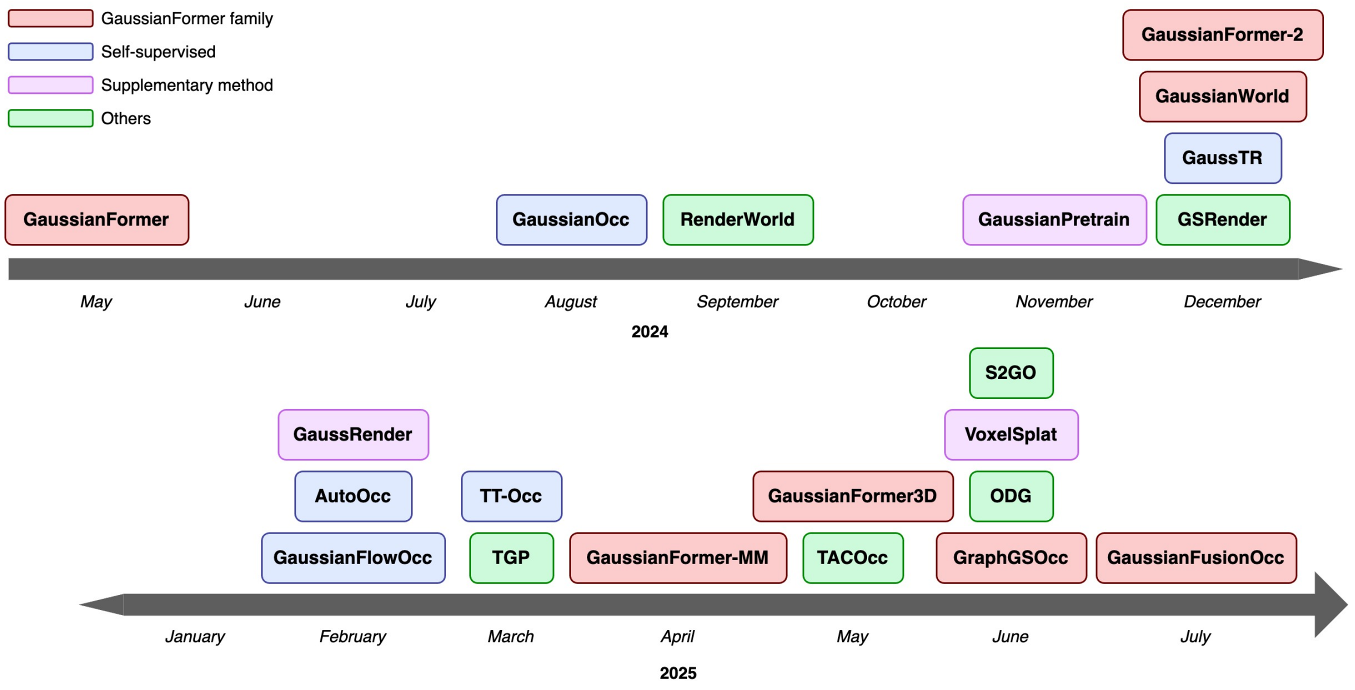

Fig. 1: **Timeline of 3D Gaussian-driven semantic occupancy models publications:** For months with multiple releases, models are ordered top-to-bottom by earliest release date. The release date is defined by the first paper publication in a conference proceedings or on arXiv. Model details are listed in full in Table II.

trajectory prediction [15], [16]. A notable drawback, however, is the absence of height information, which can be crucial for interpreting overpasses, traffic lights, and other elevated elements. As such, BEV alone cannot provide a complete representation of the inherently 3D physical world. Following closely after the rise of BEV, semantic occupancy prediction began to gain momentum.

Semantic occupancy prediction can be considered an extension of BEV into the height dimension, and often incorporates a broader range of scene elements. This approach addresses BEV's primary limitation, the absence of a $z$ axis [17]. It enables dense and detailed scene modelling through three degrees of freedom and supports diverse class predictions owing to the large number of voxels typically involved, with many datasets using approximately 640,000 voxels for prediction [18], [19]. Despite its advantages over BEV, two main challenges remain. The first is limited fine-grained object reconstruction, as conventional datasets often use voxel resolutions of 0.4m or 0.2m per axis. This limitation has been partially addressed by the introduction of the nuCraft dataset, which provides a finer voxel resolution of 0.1m per axis [20]. However, increasing resolution significantly raises memory requirements. For instance, in a typical driving scene, a 0.4m resolution may require 640k voxels, whereas a 0.1m resolution would require over 83 million voxels in the nuCraft dataset [20]. To mitigate these issues, sparse scene representations, such as those based on 3D Gaussians, have been proposed as a more memory-efficient alternative, enabling compact and continuous scene modelling. In particular, they avoid the need to model empty space and can represent large objects more compactly using a single primitive.

A continuous 3D Gaussian scene representation offers an efficient modelling approach that extends beyond automated perception. This concept was first popularised in novel view synthesis, where 3D Gaussian splatting [21] surpassed Neural Radiance Fields (NeRF) [22] for rendering photorealistic scenes. 3D Gaussians enabled real-time rendering while providing more accurate scene representation. Although automated driving applications do not require photorealistic output and instead rely on discrete modelling, continuous 3D Gaussian representations have proven advantageous for reducing both memory consumption and latency. Furthermore, Gaussian splatting for 2D view rendering has been extended to 3D, enabling conversion of the continuous 3D Gaussian scene representation into a voxel representation [23], which allows direct comparison with voxel-driven methods. Beyond scene representation, 3D Gaussians can also be integrated into voxel-driven models by incorporating the 2D view rasterisation, improving perspective and temporal consistency [24], [25]. This demonstrates that 3D Gaussian-driven semantic occupancy prediction models are not an isolated category, like BEV, but rather form part of the broader research landscape in occupancy prediction. Moreover, multi-modal deployment has been widely explored for all scene representation methods; however, due to the point-based nature of 3D Gaussians, LiDAR and radar integration may be particularly beneficial in this context.

Existing surveys in autonomous perception focus on earlier scene representation approaches, including 3D object detection [11] and BEV perception [13], [14], as well as related topics such as 3D Gaussian splatting [26], [27],

[28] and semantic occupancy prediction [17]. None of these surveys addresses the specific developments in 3D Gaussian-driven semantic occupancy prediction, with Xu *et al.* [17] predating the introduction of GaussianFormer [23]. GaussianFormer laid the groundwork for a succession of models that have significantly advanced this field, warranting a detailed examination.

In this survey, we examine the development of 3D Gaussian-driven semantic occupancy prediction models. Figure 1 presents a timeline of publications in this field since the release of the first such model in May 2024. A method is considered Gaussian-driven if it uses 3D Gaussians for scene representation or retains a voxel scene representation but computes loss functions using Gaussians. Models that use voxels for both scene representation and loss computation are referred to as voxel-driven. The GaussianFormer family of models refers to methods sharing the same underlying architecture as GaussianFormer [23], with only minor modifications. In summary, the contributions of this survey are as follows:

- A detailed review of 3D Gaussian-driven semantic occupancy prediction methods, including an analysis of each model's architectural components.
- Evaluation of these methods on three notable datasets, considering aspects such as sensor modalities, supervision type, and scene representation method.
- Investigation of Gaussian parameter distributions to assess redundancy of specific parameters.
- Discussion of the limitations of Gaussian-driven models and exploration of potential future research directions.

This paper is organised as follows. Section II presents the problem statement and reviews related work to provide a comprehensive foundation for the discussion that follows. Section III discusses the datasets and evaluation metrics used. Section IV describes the methodologies of the examined models. Section V compares these models across evaluation metrics, inference time, and qualitative analysis. Section VI analyses Gaussian parameter distributions to identify potential redundancy. Section VII addresses current limitations and outlines future directions for this line of research. Finally, Section VIII provides concluding remarks.

## II. Background

Subsection II-A presents the problem statement for Gaussian-driven semantic occupancy prediction. This is followed in Subsection II-B by a review of works related to Gaussian-driven semantic occupancy prediction, beginning with the preceding BEV perception, then the closely related semantic occupancy prediction, and concluding with 3D Gaussian splatting.

### A. Problem Statement

The use of a Gaussian scene representation is particularly advantageous for occupancy modelling. Unlike voxels, which require explicit modelling of empty space and often represent a single object with many voxels, Gaussians avoid this overhead and can represent objects more compactly. The task of 3D Gaussian-driven semantic occupancy prediction can be formulated as a special case of occupancy prediction. The task of occupancy prediction is formulated as follows, as presented in Xu *et al.*: Let the continuous 3D world be represented by $W \subset \mathbb{R}^3$. This space is discretised into a regular voxel grid indexed by $\mathbb{Z}^3$, where each voxel is assigned a semantic label from a fixed set of classes $\mathcal{C} = \{c_0, c_1, \ldots, c_n\}$:

$$V : \mathbb{Z}^3 \to \mathcal{C}. \tag{1}$$

The input at time $t$ consists of $N$ calibrated multi-view camera images $I_t = \{I_t^1, I_t^2, \ldots, I_t^N\}$ and point cloud data $P_t$ obtained from LiDAR and/or radar sensors. We denote the fused multimodal observation at time $t$ as

$$D_t = \{I_t, P_t\}. \tag{2}$$

Given a temporal sequence of $k + 1$ observations $(D_t, D_{t-1}, \ldots, D_{t-k})$, the occupancy model $F$ produces a semantic voxel map:

$$V_t = F(D_t, D_{t-1}, \ldots, D_{t-k}), \quad k \geq 0. \tag{3}$$

An occupancy model $F$ is classified as *3D Gaussian-driven* if it incorporates 3D Gaussian primitives in at least one of the following capacities:

1) **Scene representation:** The 3D environment is parameterised using continuous Gaussian distributions in place of, or in conjunction with, voxel grids.
2) **Loss computation:** Gaussian rasterisation is employed to compute supervision signals, even if the underlying representation remains voxel-based.

### B. Related Works

*1) Birds Eye View Perception:* BEV perception can be divided into three main categories: BEV Camera, BEV LiDAR, and BEV Fusion, as described by Li *et al.* [13]. BEV Camera focuses on camera-only perception, where the most significant challenge is the depth ambiguity inherent to 2D images, an issue that is particularly pronounced in self-supervised occupancy models [29], [30], [31]. Orthographic Feature Transform addresses this issue by orthographically projecting image features into 3D space [32]. Subsequent developments include depth-based splatting [15], attention-based projection [33], [34], and bilinear sampling, which strikes a balance between performance and computational efficiency [7]. In the context of Gaussian scene representations, attention-based projection of features into 3D is analogous to deformable attention between image features and Gaussian queries. Additionally, the application of Gaussians in BEV perception has been explored with GaussianBeV [35], which employs a Gaussian scene representation, demonstrating superior performance compared to voxel-driven methods.

BEV LiDAR methods typically outperform camera-only approaches due to the dense and accurate nature of point cloud data [13]. However, widespread deployment has been limited by the high cost of LiDAR compared to camera sensors [36]. LiDAR-only models can be categorised into pre-BEV and post-BEV feature extraction methods, with pre-BEV approaches [37] shown to outperform post-BEV methods [38], though at the expense of higher inference time. LiDAR-only methods are rare in Gaussian-driven semantic occupancy prediction, with GaussianFusionOcc [39] being the only paper to investigate a LiDAR-only model.

BEV Fusion methods aim to combine the complementary strengths of camera and LiDAR/radar data, integrating rich semantic cues with precise depth information. Fusion strategies are generally divided into early and late fusion. Early fusion combines BEV features from both modalities [7], [6], [40], while late fusion merges the predictions from each modality. Early fusion is typically preferred for its performance advantages [13] and is also favoured in Gaussian-driven occupancy methods [41], [39], [42].

BEV perception laid the groundwork for semantic occupancy prediction, sharing core techniques such as image feature extraction, 2D-to-3D lifting, and multi-modal fusion. However, BEV's lack of vertical structure motivated the development of semantic occupancy prediction, which extends into the height dimension for full 3D scene modelling. Many of these concepts also translate to Gaussian-driven approaches.

*2) Semantic Occupancy Prediction:* First introduced for automated perception by MonoScene [43], semantic occupancy prediction has grown into a significant research area in 3D perception [17]. It has also influenced the industry, as demonstrated by the large-scale deployment of Tesla's occupancy networks for camera-only systems. Semantic occupancy prediction follows many of the same paradigms as BEV perception, with sensor modality being the primary factor distinguishing models. According to Xu *et al.* [17], these can be classified into vision-centric networks, LiDAR-centric networks, and multi-modal networks.

Vision-centric networks generally perform feature extraction followed by 2D-to-3D lifting and, in some cases, temporal feature fusion, mirroring the pipeline of BEV methods. These networks again face the inherent challenge of depth ambiguity, resulting in lower performance than LiDAR-only approaches, though still achieving competitive results [44], [45], [46]. Comparable performance trends are observed for camera-only Gaussian-driven methods [23], [47], [48].

LiDAR-based networks typically extract features in BEV space, full 3D space, or both, followed by optional fusion prior to further encoding. Methods that avoid 3D feature extraction [49], [50] benefit from lower computational cost but generally underperform compared to those using 3D features [51], [52] or a combination of both approaches [53]. The Gaussian-driven GaussianFusionOcc's LiDAR-only model [39] adopts BEV feature extraction for LiDAR data, prioritising computational efficiency.

Multi-modal networks combine features from cameras and LiDAR/radar, often with additional refinement stages, before producing the final occupancy prediction. These approaches [54], [55], [56] generally outperform single-modal methods. Similar to Gaussian-driven research, limited attention has been given to camera–radar fusion despite its potential to provide both depth perception and velocity information for flow prediction.

A key limitation of occupancy scene representations is their high memory consumption due to dense voxel storage, a problem partially alleviated by decoding from BEV features. Sparse scene representations based on 3D Gaussians offer a potential solution, enabling more memory-efficient modelling without sacrificing scene coverage, which shall be explored in the later results Subsection V-C.

*3) 3D Gaussian Splatting:* 3D Gaussian splatting is a novel view synthesis method that enables more efficient rendering compared to earlier approaches. Novel view synthesis refers to the task of generating a scene from a viewing angle not present in the training images. Early approaches achieved this without requiring explicit 3D geometry, instead relying on dense image sampling or geometric constraints [57], [58]. Later methods incorporated geometric information to improve rendering accuracy [59], [60], and, following the deep learning revolution, neural-based approaches emerged [61], with continuous volumetric scene representation methods dominating the field [22], [62], [63].

Building on these advancements, 3D Gaussian splatting was introduced as an explicit scene representation technique, where the scene is modelled using anisotropic 3D Gaussians defined by a mean $m$, covariance matrix $\Sigma$, RGB colour, and opacity $a$ [21]. The mean defines the Gaussian center in 3D space, while opacity regulates transparency during alpha blending in rendering. The covariance is parameterized by scale, which determines axis lengths, and a quaternion, which specifies rotation. Finally, color is typically modeled with RGB values represented through spherical harmonics coefficients, allowing the appearance to vary with viewing direction [21]. A Gaussian evaluated at a point $x$ is given by the following Equation (4):

$$G(x) = e^{-\frac{1}{2}(x-m)^\top \Sigma^{-1}(x-m)} \tag{4}$$

The term inside the exponential is the negative half of the squared Mahalanobis distance [64], which generalises Euclidean distance to account for anisotropic scaling and correlation through the covariance matrix $\Sigma$. The covariance is parameterised as $\Sigma = RSS^TR^T$, where $S$ is a diagonal scaling matrix and $R$ is a rotation matrix [65]. During both training and inference, scale is represented as a 3D vector and rotation as a normalized 4D quaternion. This parametrization guarantees the positive definiteness of $\Sigma$, which would otherwise be challenging to enforce if predicted directly.

For 2D view rendering, the 3D Gaussian must be projected into image space, resulting in a 2D elliptical Gaussian, and so the covariance $\Sigma$ must be transformed into a screen-space covariance $\Sigma'$. This is computed by applying a linear transformation $W$ (e.g., scaling, rotation) followed by the local Jacobian $J$, which captures the effects of the non-linear projection. The screen-space covariance is formulated by (5), following the formulation of [65]:

$$\Sigma' = JW\Sigma W^TJ^T \tag{5}$$

TABLE I: **Semantic occupancy dataset summary**

| Dataset | Venue | Basis Dataset | Datset Details | | | | GitHub |
|---|---|---|---|---|---|---|---|
| | | | Total Scenes | Bounds | Resolution | Classes | |
| SemanticKITTI [71] | ICCV2019 | KITTI [12] | 22 | [0m, -25.6m, -2m, 51.2m, 25.6m, 4.4m] | $0.2m^3$ | 28 | Link |
| KITTI-360 [72], [73] | TPAMI2022 | - | 11 | [0m, -25.6m, -2m, 51.2m, 25.6m, 4.4m] | $0.2m^3$ | 19 | Link |
| OpenOccupancy [51] | ICCV2023 | nuScenes [5] | 850 | [-51.2m, -51.2m, -5m, 51.2m, 51.2m, 3m] | $0.2m^3$ | 16 | Link |
| SurroundOcc [19] | ICCV2023 | nuScenes [5] | 850 | [-50m, -50m, -5m, 50m, 50m, 3m] | $0.5m^3$ | 16 | Link |
| Occ3D-nuScenes [18] | NeurIPS2023 | nuScenes [5] | 900 | [-40m, -40m, -1m, 40m, 40m, 5.4m] | $0.4m^3$ | 16 | Link |
| Occ3D-Waymo [18] | NeurIPS2023 | Waymo [74] | 1000 | [-40m, -40m, -1m, 40m, 40m, 5.4m] | $0.4m^3$ | 16 | Link |
| WildOcc [75] | arXiv2024 | RELLIS-3D [76] | 5 | [0m, -10m, -2m, 20m, 10m, 6m] | $0.2m^3$ | 7 | Link |
| nuCraft [20] | ECCV2024 | nuScenes [5] | 850 | [-51.2m, -51.2m, -5m, 51.2m, 51.2m, 3m] | $0.2m^3$, $0.1m^3$ | 16 | Link |

**Bounds**: [x min, y min, z min, x max, y max, z max]
**Classes**: 'empty' and 'others' classes are not included in total

After projection into image space, the pixel intensity is obtained by aggregating the contributions of all Gaussians splatted onto it, as defined in Equation (6).

$$C = \sum_{i=1}^{N} T_i \, \alpha_i \, \mathbf{c}_i \tag{6}$$

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \qquad \alpha_i = 1 - e^{-\sigma_i \, \delta_i} \tag{7}$$

Here, $T$ denotes the transmittance of all previously sampled Gaussians, which models occlusion effects; $\alpha_i$ is the alpha blending weight, defined by the density $\sigma$ of the Gaussian and distance $\delta$ from the camera; and $\mathbf{c}$ represents the colour. This formulation enables smooth blending of all contributing Gaussians, with opacity and colour weighted accordingly. The resulting render is then used to optimise Gaussian parameters via gradient descent with a rendering loss [21]. The same formulation is incorporated into several Gaussian-driven occupancy models that utilise Gaussian rasterisation for loss computation.

Since its introduction, 3D Gaussian splatting has been extended to dynamic scene reconstruction [66], [67], [68] and semantic Gaussian splatting [69], [70]. Both of these directions have influenced the development of Gaussian-driven semantic occupancy prediction models, with dynamic scene reconstruction in particular sharing conceptual similarities with occupancy world models.

## III. Datasets and Evaluation Metrics

In Subsections III-A and III-B, the datasets and evaluation metrics used by the occupancy models discussed in the following sections are described.

### A. Datasets

*1) Basis Datasets:* We define a basis dataset as the underlying dataset upon which an occupancy dataset is constructed. The primary basis dataset used is the automotive perception dataset, nuScenes, which is described first, followed by other datasets that have been used as the basis for other dense occupancy annotation datasets.

**nuScenes** [5]**:** The nuScenes dataset is a widely used large-scale benchmark covering two cities, Boston and Singapore, providing diversity in traffic orientation (left- and right-hand driving) and weather conditions. The ego-vehicle is equipped with six RGB cameras, one LiDAR sensor, and five radar sensors, enabling multi-modal fusion. It contains 850 annotated driving scenes, 700 for training and 150 for validation, along with 150 unannotated scenes for testing. In total, the dataset includes 40k frames, each representing 20 seconds of urban driving.

**Others:** Other widely used datasets for autonomous perception include KITTI, Waymo, and RELLIS-3D. KITTI features two RGB and two grayscale stereo cameras, along with a roof-mounted LiDAR sensor [12]. It contains 22 scenes with a total of 15k frames. The Waymo dataset includes five RGB cameras, one mid-range LiDAR, and four short-range LiDARs [74], with 798 training scenes, 202 validation scenes, and 80 test scenes, totalling 230k frames [74], [13]. RELLIS-3D is an off-road dataset featuring one RGB camera, two stereo depth cameras, and two LiDAR systems [76], comprising five driving scenes and 6.2k frames.

*2) Occupancy Datasets:* Three main occupancy datasets used for evaluating models are outlined below, followed by other datasets of relevance. A summary of their key characteristics is provided in Table I.

**KITTI-360** [72], [73]**:** This dataset is a successor to the KITTI [12] and SemanticKITTI [71] datasets, providing labels for both camera and LiDAR data. Occupancy labels are not provided directly and instead are sourced from SSCBench [73]. The voxel grid origin is defined at the ego-vehicle's LiDAR sensor and follows the LiDAR coordinate frame.

**SurroundOcc** [19]**:** Based on the nuScenes dataset [5], SurroundOcc provides an evaluation range larger than the commonly used Occ3D dataset by 20m along both the $x$ and $y$ axes, though at a slightly coarser resolution of $0.5m^3$. The dataset defines its origin at the LiDAR sensor, expressed in the LiDAR coordinate system.

**Occ3D-nuScenes** [18]**:** Among the most widely adopted datasets for semantic occupancy prediction, Occ3D augments semantic labels with a camera mask. This mask is generated via ray-casting from the camera to each voxel to determine occlusion. Since self-supervised models cannot infer content beyond the visible region, this mask is particularly advantageous. The dataset defines its origin at the rear axle of the ego-vehicle, expressed in the ego-vehicle's coordinate system. Occ3D-nuScenes contains 600 training scenes compared to 750 in nuScenes; the reason for this reduction is not documented.

**Others:** Additional datasets include SemanticKITTI, OpenOccupancy, Occ3D-Waymo, WildOcc, and nuCraft. SemanticKITTI was the first large-scale dataset to provide semantic occupancy labels, focusing primarily on LiDAR data, though camera inputs can also be used via KITTI [71], [12]. OpenOccupancy introduced the first occupancy dataset for nuScenes, at a resolution of $0.2m^3$ [51]. Occ3D-Waymo follows the same design as its nuScenes counterpart but is based on the Waymo dataset [18]. WildOcc provides dense occupancy labels for the off-road RELLIS-3D dataset, which differs substantially from urban driving [76], [75]. Finally, nuCraft offers high-resolution annotations of $0.1m^3$, incorporating improvements such as KISS-ICP LiDAR scan alignment for enhanced label accuracy [20], [77].

*3) Comparison:* Occupancy prediction datasets have significantly improved in annotation quality and resolution, with nuCraft offering a 125× higher resolution compared to SurroundOcc. Other advances include the introduction of camera masks for more effective evaluation of self-supervised models [18], [20], and more complete scene coverage [20]. Despite this progress, challenges remain: LiDAR scan misalignment can still produce faulty occupancy labels [20], and incomplete geometries persist in several datasets [51], [18], [19]. In academic research, Occ3D [18] is widely favored for its inclusion of a camera mask, enabling fairer evaluation restricted to regions visible from the camera view.

## B. Metrics

Three primary metrics are commonly used in the semantic occupancy prediction literature: Intersection over Union (IoU), mean Intersection over Union (mIoU), and Ray Intersection over Union (RayIoU).

*1) IoU:* IoU measures the overlap between binary ground truth occupancy and predicted occupancy. Originally introduced for image segmentation [90], it can be computed either directly from binary occupancy predictions or inferred from semantic occupancy predictions by masking non-empty labels as occupied. The formula for IoU is given in (8):

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{8}$$

where $TP$, $FP$, and $FN$ represent true positives, false positives, and false negatives, respectively. IoU evaluates a model's ability to capture occupancy irrespective of semantic class. It is typically reported on datasets evaluated over the full voxel grid [19], [51], [72]. For datasets such as Occ3D [18] that provide a camera mask, supervised models do not report IoU, while self-supervised models do. This metric is included in our evaluation in Section V.

*2) mIoU:* Beyond binary occupancy, IoU can be computed per semantic class and averaged to obtain the mIoU. This metric is preferred for semantic occupancy prediction as it penalises misclassification of occupied voxels. The formula is shown in (9):

$$\text{mIoU} = \frac{1}{C}\sum_{c=1}^{C} \frac{TP_c}{TP_c + FP_c + FN_c} \tag{9}$$

where $C$ is the total number of semantic classes, excluding the empty class (assumed to be class index zero), all models reviewed in this survey report mIoU, and it is therefore included in our evaluation in Section V.

*3) RayIoU:* RayIoU, introduced in SparseOcc [91], addresses occlusion by considering only voxels visible from the LiDAR sensor. It improves fairness in evaluating small objects and addresses uncertainty in unoccupied voxels that may be unseen by LiDAR. Rays are cast through the ground-truth voxel grid, and only those intersecting occupied voxels are evaluated. The same ray is then cast in the predicted voxel grid. A true positive is counted if the predicted intersection is within a depth threshold (e.g., 2 m) and the semantic class matches. The score follows the same formulation as mIoU, per (10).

$$\text{RayIoU} = \frac{1}{C}\sum_{c=1}^{C} \frac{TP_c}{TP_c + FP_c + FN_c} \tag{10}$$

As only 8 of 21 surveyed models report RayIoU [30], [81], [24], [83], [85], [25], [88], [89], it is excluded from the evaluation in Section V.

## IV. Methodologies

In this section, we present the methodologies employed in Gaussian-driven semantic occupancy prediction models. A comprehensive list of these models is provided in Table II, which includes details such as supervision type, sensor modalities, and key contributions. Subsection IV-A briefly describes methods that use a voxel-based scene representation. Subsections IV-B to IV-E discuss the methodologies of models employing a Gaussian scene representation. Finally, Subsection IV-F outlines the loss functions and supervision strategies used by both voxel-based and Gaussian-based scene representation methods.

## A. Voxel Scene Representation Methods

In this subsection, we briefly discuss methods that do not employ a Gaussian scene representation and instead use a voxel-based representation. Supplementary methods, those designed to enhance existing models rather than operate independently, are excluded [80], [25], [24]. The traditional voxel-based pipeline, including optional Gaussianization and rendering, is illustrated in Figure 2.

**GaussianOcc** extracts image features using a ResNet-101 backbone, followed by bilinear sampling for 2D-to-3D projection. To improve photometric consistency, the model employs a pre-training strategy for pose estimation between adjacent frames, which outperforms the use of nuScenes GPS and IMU data [30].

**RenderWorld** [79] introduces an Air Mask Variational AutoEncoder (AM-VAE) that encodes empty and non-empty

TABLE II: **Summary of Gaussian-driven semantic occupancy prediction models.**

| Method | Venue | Model Design | | Gaussian | | Evaluation Datasets | | | | Contribution | Open Source | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Supervision | Modality | Representation | Rendering | Occ3D [18] | SurroundOcc [19] | KITTI-360 [72], [73] | Others | | GitHub | Code | Weights |
| GaussianFormer [23] | ECCV2024 | Strong | C | ✓ | | | ✓ | ✓ | | Models scene using gaussian primitives | Link | ✓ | ✓ |
| GaussianOcc [30] | ICCV2025 | Self | C | | ✓ | ✓ | | | DDAD [78] | Gaussian rasterisation for self-supervision | Link | ✓ | ✓ |
| RenderWorld [79] | arXiv2024 | Weak | C | | ✓ | ✓ | | | | Gaussian rasterisation for weak supervision | | | |
| GaussianPretrain [80] | arXiv2024 | Strong* | C | | ✓ | ✓ | | | | Gaussian rasterisation for pretraining models | Link | ✓ | ✓ |
| GaussianFormer-2 [47] | CVPR2025 | Strong | C | ✓ | | | ✓ | ✓ | | Gaussian superposition for occupancy modelling | Link | ✓ | ✓ |
| GaussianWorld [48] | CVPR2025 | Strong | C | ✓ | | | ✓ | | | Previous gaussian primitives as basis for world model | Link | ✓ | ✓ |
| GaussTR [31] | CVPR2025 | Self | C | ✓ | ✓ | ✓ | | | | Open-vocabulary semantic labels | Link | ✓ | ✓ |
| GSRender [81] | arXiv2024 | Weak | C | | ✓ | ✓ | | | | Ray compensation to overcome object duplication | | | |
| GaussRender [24] | ICCV2025 | Strong* | C | | ✓ | ✓ | ✓ | ✓ | | Supplementary loss using gaussian rasterization | Link | ✓ | ✓* |
| AutoOcc [82] | arXiv2025 | Self | C, C+L | ✓ | ✓ | ✓ | | | SemanticKITTI [71] | Test-time method which uses foundation models | | | |
| GaussianFlowOcc [83] | ICCV2025 | Self | C | ✓ | ✓ | ✓ | | | | Rendering of adjacent frames for temporal consistency | Link | ✓ | |
| TT-Occ [84] | arXiv2025 | Self | C, C+L | ✓ | ✓ | ✓ | | | nuCraft [20] | Test-time method | Link | ✓ | |
| TGP [85] | arXiv2025 | Strong | C | ✓ | | ✓ | | | | Combines voxel and Gaussian anchor predictions | | | |
| GaussianFormer-MM [86] | OJVT2025 | Strong | C+L, C+R | ✓ | | | ✓ | | | Implements radar data | | | |
| GaussianFormer3D [42] | arXiv2025 | Strong | C+L | ✓ | | ✓ | ✓ | | WildOcc [75] | 3D deformable attention with gaussians | Link | | |
| TACOcc [41] | arXiv2025 | Strong | C+L | | ✓ | | ✓ | | SemanticKITTI [71] | RGB Gaussian rendering for fine detail reconstruction | | | |
| S2GO [87] | arXiv2025 | Strong | C | ✓ | ✓ | | ✓ | ✓ | | Anchored Gaussians with multi-offset representation | | | |
| VoxelSplat [25] | CVPR2025 | Strong* | C | | ✓ | ✓ | | | OpenOccupancy [51] | Supplementary loss using scene flow of gaussians | Link | ✓ | |
| ODG [88] | arXiv2025 | Strong | C | ✓ | ✓ | ✓ | | | Occ3D Waymo [18] | Dual-Gaussian modeling of dynamic and static elements | | | |
| GraphGSOcc [89] | arXiv2025 | Strong | C | ✓ | | ✓ | ✓ | ✓ | OpenOccupancy [51] | Graph-based dual-attention on gaussians | | | |
| GaussianFusionOcc [39] | arXiv2025 | Strong | All | ✓ | | | ✓ | | | Multi-modal Gaussian fusion | | | |

**Supervision**: Strongly supervised models use dense voxel labels obtained through manual annotation. Weakly supervised models utilise LiDAR segmentation data projected to the camera view for loss. Self-supervised models are trained without any manually annotated nuScenes data.
**Modality**: C - Camera; L - LiDAR; R - Radar
**Gaussian**: Representation means the method opts for a 3D Gaussian scene representation. Rendering denotes that the model incorporates Gaussian rasterisation for loss computation.
**Notes**
Strong* denotes the fact that these are not independent models and hence they could be supplemented into weakly supervised models
The absence of a checkmark indicates that it is not implemented or supplied
✓*: For GaussRender weights for TPVFormer for the Occ3D and SurroundOcc datasets are the only supplied.
All denotes using the following modalities in their method: C, L, C+R, C+L, C+L+R.

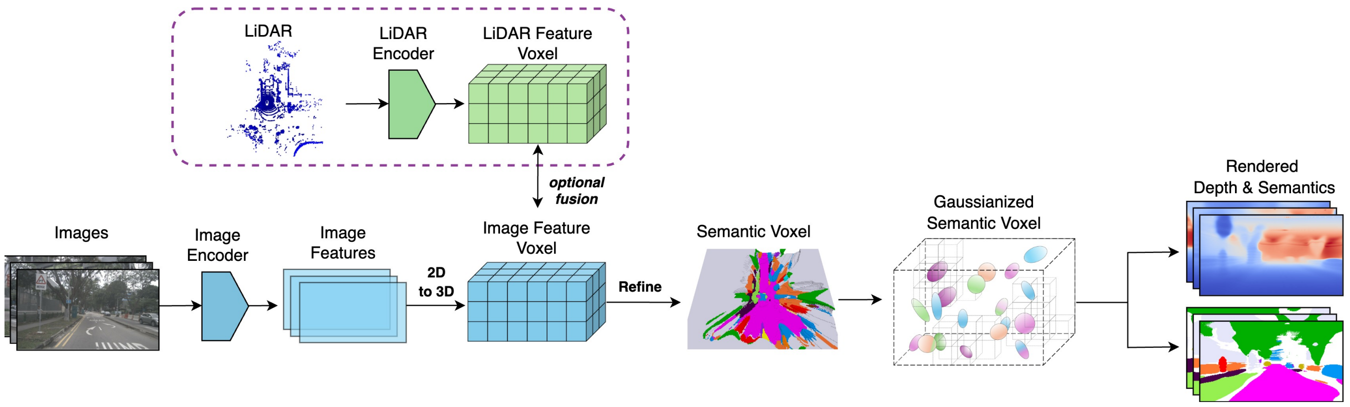

Fig. 2: **Example flow of a model with a voxel scene representation:** The model extracts image features, which are projected into 3D with optional fusion of LiDAR features. The voxel grid is then refined and optionally Gaussianized to enable Gaussian rasterisation.

regions separately to address class imbalance often encountered in standard VAEs [92]. The two categories are encoded with separate codebooks for richer representation and then decoded into a combined occupancy prediction. Temporal information is also incorporated to enhance scene understanding and enable occupancy forecasting.

**GSRender** [81] adopts a pipeline similar to the volume-based rendering method RenderOcc [93], extracting image features using a Swin Transformer [94] followed by 2D-to-3D projection.

**TACOcc** combines LiDAR and image data through an adaptive fusion module [41]. The model incorporates the Gumbel-Softmax [95] and Straight-Through Estimator [96] to enable bidirectional feature retrieval between LiDAR and image feature voxels, improving spatial understanding for both small and large objects.

As stated previously, the major disadvantage of these methods compared to Gaussian-driven models is their dense

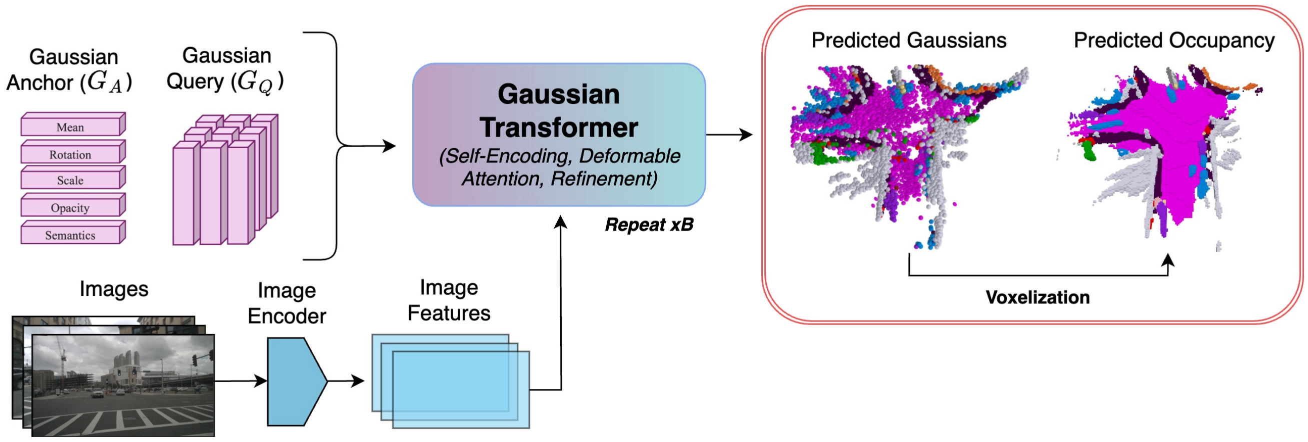

Fig. 3: **Example flow of a model with Gaussian scene representation:** Image features, Gaussian anchors, and queries are passed to the Gaussian transformer, which refines the Gaussians. The specifics of the Gaussian transformer are described in Subsection IV-C. B denotes the number of transformer blocks.

voxel representation, which dramatically increases memory consumption. Moreover, when employing a Gaussian-based loss, each voxel must first be Gaussianized (Figure 2), leading to tens of thousands of Gaussians for rendering supervision. In contrast, Gaussian scene representation methods natively model the scene with far fewer Gaussians, yielding both a lower memory footprint and a reduced rendering cost during training.

These methods, detailed in this section, rely on a discrete voxel representation and will be revisited in the supervision Subsection IV-F. The following subsections explore methods that adopt continuous Gaussian scene representations, offering potential advantages in memory efficiency and rendering flexibility.

## B. Gaussian Initialization

The technical details of models employing a 3D Gaussian scene representation are presented next, with a general pipeline shown in Figure 3.

Gaussian-based scene modelling can be broadly categorised into two approaches: *anchor-based* and *query-based*. Anchor-based methods instantiate a learnable Gaussian anchor, $G_A$, along with a Gaussian query vector, $G_Q$. Query-based methods instantiate $G_Q$ and a Gaussian mean, $G_m$, which is used for spatial encoding. The specifics of each approach are described below.

The Gaussian anchor $G_A$ is typically a set of learnable $(11 + D)$-dimensional vectors representing Gaussian properties: mean, scale, rotation, opacity, and semantics. Opacity can be optional [23], and $D$ denotes the number of semantic classes that are modelled. This formulation was first introduced in GaussianFormer [23] and later used in other models [86], [42], [48], [39], [89]. All Gaussian properties of $G_A$ are randomly initialised at the start of training.

GaussianFormer-2 [47] introduces a slightly different initialisation strategy. The model first predicts occupancy distributions along points sampled from camera-view rays, supervised using binary occupancy labels from ground truth annotations. This allows the network to identify likely occupied regions and initialise Gaussian means at those locations, while other Gaussian properties of $G_A$ remain randomly initialised. This approach increases performance metrics relative to random mean initialisation, with the tradeoff being higher latency [47].

In anchor-based models, the final Gaussian properties are obtained by embedding $G_A$ through an Multi-Layer Perceptron (MLP) and combining the result with $G_Q$, which is then used to predict the final properties. The architectures in [82], [84] explicitly refine $G_A$ without embedding it, removing the need for a query vector $G_Q$. In certain multi-modal setups, the mean of $G_A$ is fixed and initialised from LiDAR or radar point cloud data [47], [86], [39]. GaussianFormer-MM [86] initialises a subset of the Gaussians with LiDAR data while the rest are to be learned from camera data, resulting in a more balanced scene representation.

The Gaussian query $G_Q$ is a learnable vector designed to capture features for each Gaussian, refined by subsequent network components such as deformable attention and self-encoding. Anchor-based methods refine the embedded $G_A$ and $G_Q$ jointly. Query-based methods refine $G_m$ and $G_Q$ through similar processes, encoding only $G_Q$ into the final Gaussian properties using an MLP [31], [83], [87], [88], [85]. Modelling $G_m$ is essential for operations such as deformable attention, spatial encoding, and temporal Gaussian propagation [83].

Some models extend this framework with additional functionality. Two-modal Occupancy Prediction (TGP) [85] initialises $G_m$ in the first decoder block as both the Gaussian mean and a sparse point $P$. As training progresses, $G_m$ and $P$ diverge into separate entities, with $P$ serving specifically for image feature sampling. Occupancy Dual Gaussian (ODG) [88] models dynamic and static Gaussians separately, adding bounding box attributes for dynamic objects. Furthermore, GaussTR [31] does not initialise $G_m$ directly, instead learning it from deformable attention reference points and sampling pseudo-depth maps provided by Metric3Dv2 [97].

During training, the learnable parameters $G_A$, $G_Q$, and $G_m$ evolve to represent the average scene structure in the dataset, and are subsequently refined to capture scene-specific details through a transformer-based architecture. Recent methods adopt query-based modelling due to its greater flexibility during training, with the only models adopting the anchor-based scene modelling being the GaussianFormer family of models. This is likely because these models only modify a specific aspect of the architecture, for example, GaussianWorld adding a world element [48], so any differences observed relative to the original model can be attributed to that aspect rather than altering the anchor-based scene modelling element.

The following subsections describe how these representations are further processed and supervised across different model architectures.

## C. Gaussian Transformer

Following Gaussian initialisation, the relevant learnable Gaussian components (e.g., $G_A$, $G_Q$, $G_m$) are refined, with each model using a subset of these learnable components depending on its design, to accurately model the scene. This is typically achieved through query refinement blocks, as seen in DETR [98], where the learnable components are iteratively processed by repeated architectural blocks for progressively richer scene understanding. An example of such a block is shown in Figure 4.

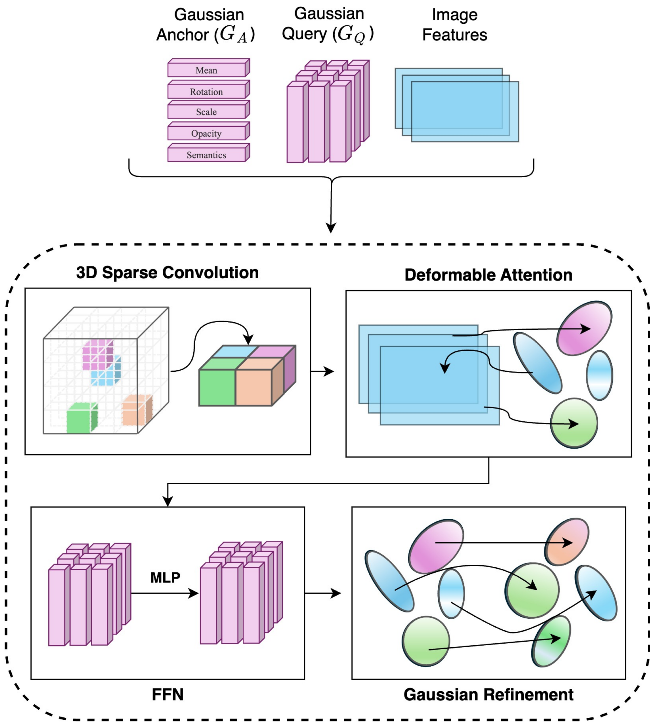

Fig. 4: **Gaussian Transformer block in GaussianFormer [23]:** Includes sparse convolution for self-encoding, deformable attention, a feed-forward network, and Gaussian refinement.

This block-based refinement approach was first introduced in GaussianFormer [23], where Gaussian queries $G_Q$ are processed via sparse convolution to enable efficient interaction among Gaussians. Deformable cross-attention is then performed against image features extracted from a ResNet-101-DCN backbone [99], [100], enriching $G_Q$ with high-quality features. Finally, $G_Q$ is passed through a feed-forward network before $G_Q$, and the embedded $G_A$ are used for predicting the final Gaussian properties in the Gaussian refinement module. Subsequent models adopt this general structure [47], [48], [86], [42], [89], [39], with certain variations. GaussianFormer3D [42] replaces 2D cross-attention with 3D deformable attention on LiDAR data enhanced by image features. GraphGSOcc [89] introduces Dual Gaussian Graph Attention (DGGA) and Dynamic–Static Decoupled Gaussian Attention (DSDGA). DGGA builds geometric and semantic graphs to aggregate local and global context for each Gaussian, while DSDGA applies cross-attention to static and dynamic Gaussians to improve predictions for both object types. GaussianFusionOcc [39] concatenates multi-modal $G_Q$ vectors, and downsamples the feature dimension for efficiency. Additionally, the model and extracts BEV features using VoxelNet [101] and a Feature Pyramid Network (FPN) [102] for LiDAR, with PointPillars being used [103] for radar.

GaussianWorld [48] extends GaussianFormer with a world model framework. Historical Gaussians from previous timesteps are first transformed into the current frame via ego-motion, with dynamic ones further adjusted using encoded features and an MLP. New Gaussians are then initialised and combined with the updated old Gaussians as input to the transformer. The model performs three evolution layers to update historical Gaussian means, followed by two refinement layers to update the new Gaussians.

TGP [85] refines Gaussians using a six-layer transformer decoder, following a paradigm similar to Occupancy Prediction Using a Sparse Set (OPUS) [104]. The model first performs consistent point sampling to sample image features from a ResNet-50 backbone, which are adaptively fused with $G_Q$. The resulting queries are further processed through self-attention, residual connections, and feed-forward layers. Finally, $G_Q$ is passed to an MLP to update point positions via an offset $\Delta p$ and to predict a semantic class $c_i$ for each point $P$.

Streaming Sparse Gaussian Occupancy Prediction (S2GO) [87] models the scene using $G_Q$ to implicitly represent multiple Gaussians via offsets. Queries are first pretrained by rendering depth and RGB from Gaussians, supervised with LiDAR segmentation and camera images, providing a strong scene prior despite Gaussian sparsity. Occupancy prediction is then trained with deformable attention using a ResNet-50 backbone, Flash Attention [105] for efficient self-attention, and temporal attention via a Temporal Transformer operating on a queue of past queries. Unlike block-based models, S2GO performs refinement in a single pass.

ODG [88] employs a coarse-to-fine refinement strategy, increasing the number of Gaussians in subsequent blocks. It also models motion using historical Gaussians for denser predictions and applies cross-attention between dynamic and static Gaussians, similar to GraphGSOcc.

Among self-supervised models, GaussTR [31] follows a GaussianFormer-like pipeline but replaces sparse convolution with self-attention between Gaussians, and uses DINOv2 [106], [107] for image feature extraction. GaussianFlowOcc [83] builds on this by incorporating temporal attention through induced attention [108] and using a ResNet-50 backbone.

AutoOcc [82] and Test-Time-Occ (TT-Occ) [84] employ test-time optimisation without conventional training, leveraging Vision–Language Model (VLM) and Visual Foundation Model (VFM) for Gaussian refinement. AutoOcc queries object categories and attention maps [109], [110], [111] to generate semantic and depth maps [112], [113], [114], [115], which are used for flow estimation and Gaussian Splatting–based smoothing. TT-Occ initialises Gaussian semantics with the VLM OpenSeed [116], estimates Gaussian depth using the VFM VGGT [117], and integrates optical flow via RAFT [118]. It further optimises Gaussians using a Trilateral Radial Basis Function (TRBF) for semantic smoothing and point cloud denoising.

Certain design choices in the Gaussian transformer appear clearly advantageous, such as the use of historic data via a world model [48], the avoidance of modelling empty space using Gaussians [47], iterative Gaussian refinement, and separate treatment of static and dynamic elements [88], [89], as evidenced by direct comparisons with models that omit these features. Other choices, such as whether to employ sparse convolution or self-attention for self-encoding, remain open questions due to the absence of independent tests.

### D. Final Gaussian Prediction

Following refinement, the final Gaussian properties are predicted for loss computation or model evaluation.

In anchor-based methods, such as the GaussianFormer family of models [23], [86], [42], [89], [39], the embedded anchors $G_A$ are summed with the refined queries $G_Q$ and passed through a single MLP to predict all Gaussian properties. The predicted means are treated as deltas, which are added to the original means rather than directly replacing them. In contrast, GaussianWorld [48] concatenates $G_A$ with $G_Q$, and the MLP outputs deltas that update all anchor properties, including the mean.

In query-based methods, including GaussTR [31], S2GO [87], GaussianFlowOcc [83], and ODG [88], the refined queries $G_Q$ are input to separate MLPs to predict Gaussian properties and Gaussian mean offsets, which are then added to $G_m$. Notably, ODG uses distinct MLPs for dynamic and static objects, allowing the model to better capture differences in their behaviour [88]. TGP employs a single MLP to predict all Gaussian properties and mean offsets [85].

AutoOcc [82] and TT-Occ [84] differ in that they do not employ a dedicated Gaussian property prediction module; instead, properties are explicitly produced during the Gaussian refinement stage.

For Gaussian property prediction, two main strategies exist: using a single MLP to jointly predict all properties, or assigning separate MLPs for each property. A joint MLP can exploit correlations between properties; for instance, semantics and Gaussian size are often related, providing greater modelling flexibility. However, this flexibility can also be a drawback, since it entangles different factors and may hinder specialisation. In contrast, separate MLPs enforce disentanglement, allowing each network to specialise in predicting its assigned property. Finally, one may argue that the benefit of shared correlations in a joint MLP is limited, as much of this information is already encoded implicitly within $G_Q$ and $G_A$.

### E. Gaussian Voxelization

For models that use a 3D Gaussian scene representation, voxelization is required for evaluation and, in some cases, during training when voxel-based loss functions are used. A key consideration is ensuring that the Gaussians surrounding a voxel contribute proportionally to its value. All but one model, ODG [88], explicitly describes this process.

*1) GaussianFormer-Style Methods:* GaussianFormer [23] computes each Gaussian's influence on a voxel based on the Mahalanobis distance from the voxel centre to the Gaussian mean. To limit computation, a radius of influence is applied so only nearby Gaussians are considered for this. The Gaussian value at voxel centre $x$ is:

$$\alpha(x, \mathcal{G}) = \exp\left(-\frac{1}{2}(x - m)^\top \Sigma^{-1}(x - m)\right) \tag{11}$$

The semantic occupancy is then the weighted sum over all Gaussians in the neighbourhood $N(x)$ of the voxel for their semantic logits $c$, opacity $a$, and Gaussian value $\alpha$:

$$\mathcal{O}_{GF}(x, c) = \sum_{i \in N(x)} \alpha(x, \mathcal{G}_i) \cdot a_i \cdot c_i \tag{12}$$

A visualisation of the Gaussians and the resulting semantic occupancy is shown in Figure 5. GaussianFormer-MM [86] and GaussianFormer3D [42] utilise this exact approach also. This equation reflects this version of the GaussianFormer model, where the empty class was removed from the Gaussian logits and a single large Gaussian spanning the entire scene is used to represent empty space, reducing the representation size by more than fivefold. The original version, by contrast, included empty logits, allowing individual Gaussians to represent empty space, and opacity was not represented in the Gaussians or in the occupancy Equation (12). However, both approaches share the same limitation: modelling emptiness with Gaussians undermines the intended sparsity advantage of Gaussian scene representations.

TGP [85] extends this framework by cross-gating the voxel-wise predictions from (12) with the point-wise semantic logits $c_i$ of $P$, using a fully connected layer (FCN) to align feature dimensions.

GaussianFlowOcc [83] derives the semantic voxel from class logits and the occupancy voxel from opacity, both computed using (12). A voxel is classified as empty if its occupancy falls below a threshold $\tau$; otherwise, it is assigned the most likely semantic label. GaussTR [31] adopts a similar strategy; however, prior to applying a threshold, it weights the semantic voxel by the occupancy voxel.

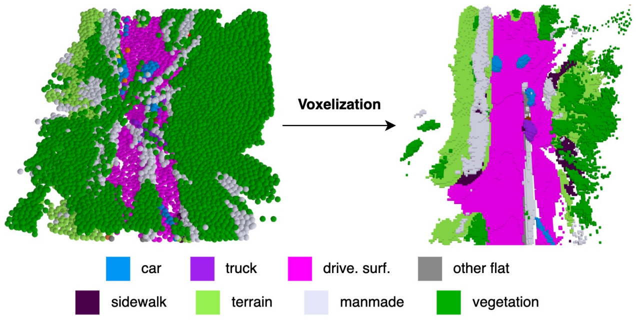

Fig. 5: **GaussianFormer [23] voxelization module visualization:** Example for the 25,600 Gaussian model. Gaussians attend to voxels in a neighbourhood, and some Gaussians do not contribute to the final prediction due to insufficient opacity. Gaussians below 0.75 opacity are filtered out for visualisation purposes.

*2) Probabilistic Methods:* GaussianFormer-2 [47] and related work [48], [89], [39], [87] avoid empty Gaussians by jointly modelling occupancy and semantics. Occupancy probability is computed as:

$$o(x) = 1 - \prod_{i \in N(x)} (1 - \alpha(x, \mathcal{G}_i)) \tag{13}$$

Semantic probability is derived via a Bayesian weighting of Gaussian semantics by opacity:

$$e(x; G) = \frac{\sum_i p(x \mid G_i)\, a_i\, \tilde{c}_i}{\sum_j p(x \mid G_j)\, a_j} \tag{14}$$

The final prediction is:

$$\mathcal{O}_{GF2}(x; G) = [1 - o(x);\, o(x) \cdot e(x; G)] \tag{15}$$

S2GO [87] further improves this with opacity-weighted occupancy, forced locality, and CUDA optimisations for faster training.

*3) Distance-Weighted Methods:* AutoOcc [82] weights Gaussian semantics by the distance from the voxel centre to the Gaussian ellipsoid surface, Gaussian influence $\alpha$, and opacity $a$. First, the distance from the voxel centre $x$ to the ellipsoid surface along the ray from $x$ to the Gaussian mean $m$ is computed as:

$$d = m_z - \frac{\eta^{-1}\Sigma_{0,2}^{-1}(m_x - x_x) + \eta^{-1}\Sigma_{1,2}^{-1}(m_y - x_y)}{\Sigma_{2,2}^{-1}}, \tag{16}$$

where subscripts on $m$ and $x$ denote coordinates, $\eta = m - x$, and $\Sigma_{a,b}^{-1}$ is the $(a, b)$ element of $\Sigma^{-1}$. The final occupancy prediction is:

$$\mathcal{O}_{AO}(x) = \sum_{i \in N(x)} d_i \cdot \alpha(x, \mathcal{G}_i) \cdot a_i \cdot \mathrm{softmax}(c_i). \tag{17}$$

*4) Kernel-Based Methods:* TT-Occ [84] uses a spatial kernel:

$$K(x, \mathcal{G}) = \exp\left(-\frac{\|x - m\|^2}{2\sigma^2}\right) \tag{18}$$

This treats Gaussians as isotropic spheres, ignoring scale and rotation, which may reduce fidelity. Semantic logits are weighted by $K$ and normalised over the grid:

$$\mathcal{O}_{TT}(x) = \frac{1}{Z} \sum_{i \in N(x)} c_i \cdot K(x, \mathcal{G}_i) \tag{19}$$

*5) Discussion:* For voxelization, it is difficult to determine which strategy is superior, as no paper has systematically compared different approaches. However, one critical factor is the alignment between the voxelization and the chosen loss function. Models using voxel-based loss functions are less affected, since the supervision is consistent with the voxel grid. In contrast, methods relying solely on 2D supervision must ensure that voxelization faithfully reflects the underlying Gaussian representation. For example, TT-Occ [84] models Gaussians as anisotropic ellipsoids but voxelizes them as isotropic spheres, introducing misalignment that may limit performance. The loss computation of these methods is explored in the next section.

## *F. Gaussian Supervision*

A discussion of the supervision strategies employed in Gaussian-driven models is provided, with emphasis on their associated loss functions. Models are first grouped into those that adopt a voxel-based loss, and then into those that rely on Gaussian rasterisation for supervision. Within the latter, methods are further categorised as supervised, self-supervised, or employing Gaussian rendering as a supplementary method for additional loss or pretraining. As a preliminary, we begin with the Gaussian rendering loss formulation.

*1) Gaussian Rendering:* For a given sample, semantics and depth can be rendered using Gaussian splatting, with a formulation analogous to Equation (6), with the new formulation given in (20):

$$C = \sum_{i=1}^{N} T_i\, \alpha_i\, \mathbf{c}_i, \;\; D = \sum_{i=1}^{N} T_i\, \alpha_i\, \delta_i, \tag{20}$$

Here, $C$ and $D$ denote the rendered colour and depth, respectively. The depth formulation replaces the colour term $\mathbf{c}_i$ with $\delta_i$, the distance from the Gaussian to the camera. As in the original Gaussian Splatting work [21], these rendered outputs are compared against ground-truth or pseudo ground-truth labels to compute the supervision loss and guide model training. Now we begin our exploration of the losses employed in Gaussian-driven models.

*2) Voxel Loss:* The primary models employing voxel-based supervision are the GaussianFormer family [23], [47], [48], [86], [89], [42], [39]. After voxelizing the 3D Gaussians, they apply Lovász-softmax loss [119] and cross-entropy (CE) loss against ground-truth occupancy labels, enabling the network to model the complete scene; important for benchmarks such as SurroundOcc, which evaluate the full voxel grid. TGP [85] similarly applies a 3D loss using Focal loss, but also supervises Gaussian means with the Chamfer Distance to ensure they align with occupied regions in the ground truth.

*3) Supervised:* RenderWorld and GSRender [79], [81] decompose the semantic voxel grid into 3D Gaussians by setting the mean to the voxel centre, semantics from voxel labels, and learning scale (see Fig. 6). RenderWorld opts to learn rotation, whereas GSRender leaves it static for all voxels. These are splatted into the camera view for depth and semantic supervision against LiDAR segmentation labels in a weakly supervised manner. RenderWorld additionally supervises its AM-VAE codebook, while GSRender incorporates a ray compensation module to enforce temporal consistency and prevent object duplication.

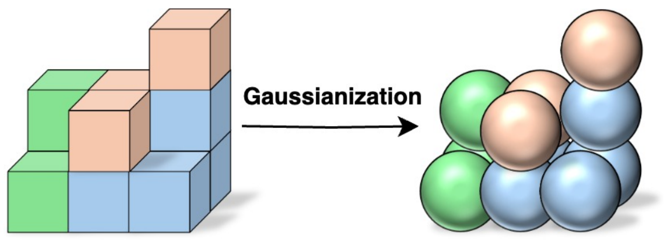

Fig. 6: **Gaussianization of a semantic voxel grid:** Illustration case for non learned parameters.

Target-Adaptive Cross-Modal Fusion Occupancy (TACOcc) [41] predicts Gaussian parameters from a semantic voxel grid using a 3D CNN, and supplements them with LiDAR data to address blind spots. The Gaussians are rendered and compared to RGB images using L1 and D-SSIM losses. Parameter refinement is performed via a progressive diffusion mechanism, with additional losses applied to both the original and refined parameters. A 3D semantic voxel loss is also incorporated to supervise the final voxel predictions. S2GO [87] performs two-stage training: Stage 1 renders depth and RGB for supervision, while Stage 2 switches to voxel-based semantic occupancy loss. ODG [88] supervises each refinement block by rendering depth and semantics from Gaussians, and also predicts bounding boxes and semantic voxels.

*4) Self-Supervised:* GaussianOcc [30] employs an approach similar to the previously described GSRender; however, it adds multi-frame photometric consistency loss for depth supervision and pseudo-semantic labels from Grounded-SAM [113], [120], [121], as in OccNerf [122]. GaussTR [31] supervises depth with Metric3Dv2 [97] pseudo-labels and semantics via Talk2DINO text embeddings [123], [124]. GaussianFlowOcc [83] also uses Metric3Dv2 and Grounded-SAM labels, and adds temporal Gaussian propagation for view consistency, which greatly increases performance [83]. TT-Occ [84] enforces RGB consistency with camera images, while AutoOcc [82] employs Chamfer Distance for dynamic Gaussian consistency and optional geometric consistency with LiDAR.

*5) Supplementary Methods:* Supplementary methods are defined as those that do not operate independently but are designed to enhance existing models, typically through pretraining strategies or additional loss functions. GaussianPretrain [80] serves as a pretraining stage for RGB, depth, and occupancy, masking image patches in an MAE-like fashion [125] and initialising Gaussian anchors from ray-projected 2D features. Predictions are rendered, with opacity used to infer occupancy. GaussRender [24] converts predicted and ground-truth voxels into Gaussians and renders them from various viewpoints (camera, elevated, BEV) to enforce multi-view consistency. VoxelSplat [25] samples occupied regions and associated features, semantics, and scene flow to form 4D Gaussians, which are rendered for both current and next frames. Static Gaussians are handled separately to improve temporal consistency.

*6) Discussion:* For supervised models, combining a 2D rendering loss with a voxel-based loss has shown clear benefits [41], [87], [88], [25], [80], [24], as it enforces both view consistency and, when coupled with flow modelling, temporal consistency [25] (see Subsection V-B). For self-supervised models, using pseudo-depth labels [31], [83] has proven more effective for learning depth than relying on multi-frame photometric consistency [30], a trend further confirmed in the results section.

## V. Model Performance

Following the discussion of model architectures, this section evaluates the surveyed methods across multiple dimensions. Section V-A compares quantitative results on benchmark datasets. Section V-B analyses the benefits of augmenting voxel-based methods with supplementary Gaussian loss or pretraining, and Section V-C examines inference time and memory consumption. Finally, Section V-D performs a qualitative analysis on select models.

### A. Model Comparison

Tables III, IV, and V summarize performance on the Occ3D [18], SurroundOcc [19], and KITTI-360 [72], [73] benchmarks. Results are analysed along four axes: (1) the effect of multi-modal fusion, (2) differences in supervision strategies, (3) the impact of Gaussian versus voxel scene representations, and (4) loss computation techniques.

*1) Modality Comparison:* Across all benchmarks, camera+LiDAR (C+L) models consistently outperform camera-only counterparts. On Occ3D (Table III), the gap is particularly pronounced for self-supervised methods: TT-Occ [84] improves by over 75%, and AutoOcc [82] by more than 21% with LiDAR input. This reflects the reliance of self-supervised models on pseudo-labels from VFMs (e.g., Metric3Dv2), which LiDAR complements with accurate geometric priors. Among supervised approaches, GaussianFormer3D [42] is the only multi-modal entry, surpassing the next best model by 8.5%, a smaller margin due to the already strong depth and semantic priors provided by dense occupancy labels.

TABLE III: **Comparison of models trained on the Occ3D-nuScenes [18] dataset:** Gauss. denotes if a model employs a Gaussian scene representation. Sup. denotes the supervision method for training the model. Mod. denotes the modalities used in the inference of the model. Notation of modality: C - Camera; L - LiDAR. IoU denotes the evaluation metric Intersection over Union, and mIoU denotes mean Intersection over Union. Supervised methods do not report IoU. GP denotes models trained with the GaussianPretrain pertaining strategy. GSR denotes models that deploy GaussRender's supplementary loss function. VS denotes models that deploy VoxelSplat's supplementary loss function. The best-performing models for each supervision type for each evaluation metric are highlighted in **bold**.

| **Model** | **Gauss.** | **Sup.** | **Mod.** | **IoU** | **mIoU** | others ■ | barrier ■ | bicycle ■ | bus ■ | car ■ | const. veh. ■ | motorcycle ■ | pedestrian ■ | traffic cone ■ | trailer ■ | truck ■ | drive. surf. ■ | other flat ■ | sidewalk ■ | terrain ■ | manmade ■ | vegetation ■ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GaussianOcc [30] | | Self | C | 42.91 | 9.94 | 0.00 | 1.79 | 5.82 | 14.58 | 13.55 | 1.30 | 2.82 | 7.95 | 9.76 | 0.56 | 9.61 | 44.59 | 0.00 | 20.10 | 17.58 | 8.61 | 10.29 |
| TT-Occ [84] | ✓ | Self | C | - | 11.84 | 0.00 | 0.00 | 5.90 | 8.94 | 12.58 | 2.75 | 9.67 | 4.71 | 4.04 | 0.00 | 8.77 | 55.65 | 0.00 | 26.49 | 30.20 | 15.13 | 16.57 |
| GaussTR [31] | ✓ | Self | C | 44.54 | 12.27 | 0.00 | 6.50 | 8.54 | 21.77 | 24.27 | **6.26** | 15.48 | 7.94 | 1.86 | **6.10** | 17.16 | 36.98 | 0.00 | 17.21 | 7.16 | 21.18 | 9.99 |
| GaussianFlowOcc [83] | ✓ | Self | C | 46.91 | 15.07 | 0.00 | **7.23** | 9.33 | 17.55 | 17.94 | 4.50 | 9.32 | 8.51 | 10.66 | 2.00 | 11.80 | **63.89** | 0.00 | 31.11 | 35.12 | 14.64 | 12.59 |
| AutoOcc [82] | ✓ | Self | C | 83.01 | 17.87 | 0.00 | 2.70 | 10.45 | 7.81 | 20.42 | 5.79 | 17.58 | 18.50 | 24.25 | 4.23 | 12.88 | 55.54 | 0.00 | 24.23 | 27.14 | 35.62 | 36.61 |
| TT-Occ [84] | ✓ | Self | C+L | - | 20.83 | 0.00 | 0.00 | 15.99 | **23.01** | 25.42 | 5.61 | **20.50** | 20.68 | 7.36 | 0.00 | **24.32** | 51.89 | 0.00 | 31.06 | **37.15** | 43.87 | **47.20** |
| AutoOcc [82] | ✓ | Self | C+L | **88.62** | **21.66** | 0.00 | 1.19 | **16.08** | 18.42 | **25.90** | 4.32 | 14.58 | **25.62** | **27.18** | 3.51 | 20.93 | 58.38 | 0.00 | **32.03** | 29.80 | **46.15** | 43.59 |
| GSRender [81] | | Weak | C | - | 21.36 | 2.44 | 20.13 | **13.56** | 19.54 | 18.38 | 8.35 | **18.77** | 10.67 | 14.81 | 16.63 | 14.92 | 61.74 | 29.77 | 35.95 | 40.56 | 19.83 | 17.06 |
| RenderWorld [79] | | Weak | C | - | **27.87** | **6.83** | **32.54** | 7.44 | **21.15** | **29.92** | **16.68** | 11.43 | **17.45** | **16.48** | **24.02** | **27.86** | **75.05** | **36.82** | **50.12** | **53.04** | **22.75** | **24.23** |
| BEVFormer (w/ GP) [34], [80] | | Strong | C | - | 24.21 | 5.53 | 34.88 | 6.95 | 36.55 | 39.18 | 11.88 | 16.62 | 16.93 | 17.10 | 13.83 | 27.03 | 54.09 | 32.36 | 33.02 | 27.05 | 20.39 | 18.16 |
| SurroundOcc (w/ GSR) [19], [24] | | Strong | C | - | 30.38 | 8.87 | 40.98 | 23.25 | 43.76 | 46.37 | 19.49 | 25.20 | 23.96 | 19.08 | 25.56 | 33.65 | 58.37 | 33.28 | 36.41 | 33.21 | 22.76 | 22.19 |
| TPVFormer (w/ GSR) [126], [24] | | Strong | C | - | 30.48 | 9.84 | 42.30 | 24.09 | 41.79 | 46.49 | 18.22 | 25.85 | 25.06 | 22.53 | 22.90 | 33.34 | 58.86 | 33.19 | 36.57 | 31.84 | 23.55 | 21.80 |
| S2GO [87] | ✓ | Strong | C | - | 31.20 | | | | | | | | | | | | | | | | | |
| TGP [85] | ✓ | Strong | C | - | 34.50 | | | | | | | | | | | | | | | | | |
| ODG [88] | ✓ | Strong | C | - | 38.18 | **14.11** | 46.62 | 27.09 | 48.77 | 52.09 | 26.79 | 28.05 | 23.21 | 27.92 | 30.86 | 38.17 | 77.13 | 40.35 | 46.94 | 47.37 | 40.01 | 33.52 |
| FB-Occ (w/ VS) [127], [25] | | Strong | C | - | 42.30 | | | | | | | | | | | | | | | | | |
| PanoOcc (w/ GP) [128], [80] | | Strong | C | - | 42.42 | 11.58 | 49.30 | 28.81 | 49.88 | 55.21 | 22.27 | 31.30 | 29.42 | **30.37** | 34.29 | 42.05 | **84.06** | **47.76** | **55.90** | **58.13** | 48.20 | 42.54 |
| GraphGSOcc [89] | ✓ | Strong | C | - | 42.46 | | | | | | | | | | | | | | | | | |
| GaussianFormer3D [42] | ✓ | Strong | C+L | - | **46.40** | 9.80 | **50.00** | **31.30** | **54.00** | **59.40** | **28.10** | **36.20** | **46.20** | 26.70 | **40.20** | **49.70** | 79.10 | 37.30 | 49.00 | 55.00 | **69.10** | **67.60** |

TABLE IV: **Comparison of models trained on the SurroundOcc [19] dataset:** Gauss. denotes if a model employs a Gaussian scene representation. Mod. denotes the modalities used in the inference of the model. Notation of modality: C - Camera; L - LiDAR; R - Radar. mIoU denotes the evaluation metric, mean Intersection over Union. GSR denotes models that deploy GaussRender's supplementary loss function. The best-performing models for C and C+L models for each evaluation metric are highlighted in **bold** (L, C+R, C+L+R are omitted as they each have one model).

| **Model** | **Gauss.** | **Mod.** | **IoU** | **mIoU** | barrier ■ | bicycle ■ | bus ■ | car ■ | const. veh. ■ | motorcycle ■ | pedestrian ■ | traffic cone ■ | trailer ■ | truck ■ | drive. surf. ■ | other flat ■ | sidewalk ■ | terrain ■ | manmade ■ | vegetation ■ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GaussianFusionOcc [39] | ✓ | L | 45.32 | 29.75 | 30.02 | 16.46 | 35.02 | 38.93 | 22.25 | 24.65 | 29.64 | 18.41 | 24.64 | 30.93 | 43.31 | 26.30 | 28.95 | 29.29 | 34.33 | 42.92 |
| GaussianFormer [23] | ✓ | C | 29.93 | 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 |
| GaussianFormer-2 [47] | ✓ | C | 31.74 | 20.82 | 21.39 | 13.44 | 28.49 | 30.82 | 10.92 | 15.84 | 13.55 | 10.53 | 14.04 | 22.92 | 40.61 | 24.36 | 26.08 | 24.27 | 13.83 | 21.98 |
| SurroundOcc (w/ GSR) [19], [24] | | C | 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 |
| TPVFormer (w/ GSR) [126], [24] | | C | 32.05 | 20.85 | 20.20 | 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 |
| GaussianWorld [48] | ✓ | C | 33.40 | 22.13 | 21.38 | **14.12** | 27.71 | 31.84 | 13.66 | **17.43** | **13.66** | 11.46 | **15.09** | 23.94 | 42.98 | 24.86 | 28.84 | 26.74 | **15.69** | 24.74 |
| S2GO [87] | ✓ | C | 35.46 | 22.72 | **21.93** | 13.36 | 27.47 | **32.08** | **14.86** | 15.31 | 12.91 | **11.79** | 13.42 | **23.98** | **46.85** | **29.14** | **30.30** | **29.05** | 14.69 | **26.40** |
| GraphGSOcc [89] | ✓ | C | **36.11** | **25.20** | | | | | | | | | | | | | | | | |
| GaussianFormer3D [42] | ✓ | C+L | 43.30 | 27.10 | 26.90 | 15.80 | 32.70 | 36.10 | 18.60 | 21.70 | 24.10 | 13.00 | 21.30 | 29.00 | 40.60 | 23.70 | 27.30 | 28.20 | 32.60 | 42.30 |
| TACOcc [41] | | C+L | 41.80 | 28.40 | 30.10 | 18.50 | 31.20 | 38.60 | 18.60 | 23.10 | 20.90 | **23.80** | **25.60** | **39.30** | 40.30 | **26.70** | **29.80** | 26.60 | 24.10 | 37.60 |
| GaussianFormer-MM [86] | ✓ | C+L | 43.54 | 28.71 | 29.12 | **19.45** | 33.92 | 38.21 | 19.84 | 25.57 | 27.60 | 18.91 | 21.99 | 30.86 | 40.92 | 24.10 | 27.49 | 27.48 | 32.50 | 41.41 |
| GaussianFusionOcc [39] | ✓ | C+L | **45.16** | **30.21** | **30.22** | 18.70 | **35.91** | **39.57** | **22.67** | **27.36** | **30.10** | 18.59 | 24.45 | 31.25 | **43.06** | 25.76 | 29.12 | **29.33** | **34.65** | **42.70** |
| GaussianFormer-MM [86] | ✓ | C+R | 32.35 | 21.00 | 20.02 | 14.38 | 27.60 | 31.27 | 11.92 | 18.08 | 14.99 | 10.18 | 14.07 | 23.79 | 39.54 | 22.79 | 25.11 | 22.75 | 15.90 | 23.58 |
| GaussianFusionOcc [39] | ✓ | C+L+R | 45.20 | 30.37 | 30.43 | 18.54 | 36.23 | 39.66 | 22.57 | 27.35 | 30.30 | 19.14 | 24.56 | 31.95 | 42.60 | 25.82 | 29.48 | 29.70 | 34.78 | 42.95 |

TABLE V: **Comparison of models trained on the KITTI-360 [72], [73] dataset:** Gauss. denotes if a model employs a Gaussian scene representation. Mod. denotes the modalities used in the inference of the model. Notation of modality: C - Camera. IoU denotes the evaluation metric Intersection over Union, and mIoU denotes mean Intersection over Union. GSR denotes models that deploy GaussRender's supplementary loss function. The best-performing models for each evaluation metric are highlighted in **bold**.

| **Model** | **Gauss.** | **Mod.** | **IoU** | **mIoU** | car ■ | bicycle ■ | motorcycle ■ | truck ■ | other-veh. ■ | person ■ | road ■ | parking ■ | sidewalk ■ | other-grnd ■ | building ■ | fence ■ | vegetation ■ | terrain ■ | pole ■ | traf.-sign ■ | other-struct. ■ | other-object ■ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GaussianFormer [23] | ✓ | C | 35.38 | 12.92 | 18.93 | 1.02 | 4.62 | **18.07** | 7.59 | 3.35 | 45.47 | 10.89 | 25.03 | **5.32** | 28.44 | 5.68 | 29.54 | 8.62 | 2.99 | 2.32 | **9.51** | 5.14 |
| SurroundOcc (w/ GSR) [19], [24] | | C | 38.62 | 13.34 | 21.61 | 0.00 | 0.00 | 6.75 | 4.50 | 0.00 | 53.64 | 11.93 | 30.24 | 2.67 | 35.01 | 4.55 | 29.81 | 17.32 | 6.19 | 8.49 | 4.80 | 2.59 |
| GaussianFormer-2 [47] | ✓ | C | 38.37 | 13.90 | 21.08 | 2.55 | 4.21 | 12.41 | 5.73 | 1.59 | 54.12 | 11.04 | 32.31 | 3.34 | 32.01 | 4.98 | 28.94 | 17.33 | 3.57 | 5.48 | 5.88 | 3.54 |
| S2GO [87] | ✓ | C | 40.80 | 15.05 | 22.72 | 1.28 | 1.66 | 15.87 | 5.13 | 2.07 | 53.77 | 13.31 | 33.40 | 3.83 | 35.30 | 7.17 | 31.20 | **21.11** | 6.36 | 6.54 | 6.03 | 4.22 |
| GraphGSOcc [89] | ✓ | C | 40.92 | 15.58 | | | | | | | | | | | | | | | | | | |
| Symphonies (w/ GSR) [129], [24] | | C | **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** |

On SurroundOcc (Table IV), more multi-modal Gaussian-driven models are present, and C+L methods also outperform camera-only methods. GaussianFusionOcc achieves the highest mIoU (30.21), outperforming GaussianFormer-MM [86] by 5.2%, despite the latter using only a simple LiDAR-based initialisation of Gaussian means. Radar remains largely underexplored, primarily due to the limited vertical resolution of nuScenes radar [5]. Nevertheless, the radar-enabled rendition of GaussianFormer-MM outperforms their camera-only baseline [23], suggesting radar can still contribute useful depth cues. Note, GaussianFusionOcc [39] details a camera+radar (C+R) model; however, this particular model uses LiDAR initialisation and is hence not considered a C+R model in this survey.

GaussianFusionOcc is the only Gaussian-driven model to incorporate a LiDAR-only model and camera+LiDAR+radar (C+L+R) fusion. The LiDAR variant outperforms all camera-only models and even some C+L models through efficient encoding, while the C+L+R variant achieves the highest mIoU overall, highlighting the benefits of fusing all available sensor modalities.

*2) Supervision Comparison:* Supervision type is compared using Occ3D (Table III), the only benchmark containing self- and weakly-supervised models. The dataset's camera mask favours such comparisons; on other datasets, RayIoU [91] offers a partial alternative.

Results reveal clear performance stratification. The best self-supervised model, AutoOcc [82], matches only the weakly-supervised GSRender [81], despite leveraging LiDAR, underscoring that even sparse ground-truth labels outperform dense pseudo-labels. However, the paper does not address the feasibility of real-time deployment, as its approach relies on computationally heavy foundation models. A similar gap appears between weak and strongly supervised methods: RenderWorld [79] surpasses the lowest-ranked strong model, BEVFormer+GP [34], [80], despite the latter's Gaussian pretraining. GraphGSOcc [89], the top camera-only strong model, outperforms BEVFormer+GP by over 75%, reflecting advances in temporal modelling and graph-based Gaussian reasoning.

Within the weakly-supervised category, RenderWorld leads by over 6 mIoU, likely due to its separate encoding of empty and non-empty voxels, yielding finer scene understanding. Overall, strong supervision remains dominant by directly supervising in the same space as the evaluation space. Hybrid strategies can still provide gains: supplementing GSRender with strong 3D supervision boosts performance by more than 38%.

*3) Voxel versus Gaussian Representation:* In Table III, for the Occ3D dataset, all but one self-supervised model uses Gaussians; the sole voxel-based method, GaussianOcc [30], ranks last. Many top Gaussian-driven models benefit from design choices tailored to a Gaussian scene representation, such as adjacent-frame rendering with flow [83] and semantic smoothing [84]. Among strongly-supervised models, the three lowest performers [34], [19], [126] are voxel-based, suggesting a potential disadvantage, yet two of the top three camera-only models [128], [127] also use voxels. This points to a nuanced trade-off: with dense occupancy labels, voxel grids can remain competitive; without them, Gaussians appear more effective.

For SurroundOcc (Table IV), Gaussian-based models dominate across camera-only, C+L, and C+R setups, with GraphGSOcc [89] leading as it did for the Occ3D dataset. Voxel-based methods [19], [126], [41] perform consistently worse.

KITTI-360 (Table V) shows the reverse trend. Here, the voxel-based Symphonies+GSR [129], [24] surpasses the best Gaussian model by over 16%, aided by sophisticated interactions between image features, instance queries, and voxel queries, indicating that scene representation choice may be dataset-specific.

*4) Loss Computation:* In Table III, all self- and weakly-supervised models adopt Gaussian losses, reflecting their reliance on 2D supervision. Strongly-supervised methods employ both voxel and Gaussian losses, with the top four performing camera-only models [42], [89], [128], [127], including three models that utilise a Gaussian-based loss.

On SurroundOcc (Table IV), nearly all camera-only models use Gaussian losses, except S2GO [87], which still remains competitive with the best Gaussian-loss method, GraphGSOcc. In the multi-modal setting, TACOcc is the only voxel-loss model and underperforms compared to the Gaussian-based counterparts.

KITTI-360 (Table V) presents a mixed outcome: S2GO is the only Gaussian-loss-only method, while the top-performing Symphonies+GSR [129], [24] again uses a hybrid approach.

Across datasets, voxel losses tend to offer stronger 3D spatial grounding, while Gaussian losses provide flexibility for 2D-projected supervision. The most competitive results often come from hybrid schemes that leverage the strengths of both, a direction explored further in Section V-B.

## B. Supplementary Methods

Table VI summarises the performance gains from augmenting voxel-based models with an additional Gaussian loss. All methods benefit, though the magnitude varies. The largest improvement is seen in FB-Occ [127], which gains $+7.91\%$ mIoU when combined with VoxelSplat [25]. This method enforces both multi-view consistency, through Gaussian rendering, and temporal consistency via flow modelling, the latter proving particularly important for boosting performance.

GaussRender [24] yields consistent but more modest gains, with the largest relative increase for TPVFormer [126], though the base performance of this model is not state-of-the-art. For PanoOcc [128] and BEVFormer [34], GaussianPretrain [80] provides only marginal improvements, suggesting that Gaussian losses may be most impactful during the main training phase, where they can enforce view consistency absent from traditional voxel losses.

Overall, the evidence supports coupling voxel and Gaussian losses to leverage their complementary strengths in 3D spatial grounding and multi-view consistency.

## C. Inference Time and Memory Consumption

Inference speed shows no strict divide between voxel and Gaussian representations. The fastest model, S2GO [87], uses a Gaussian representation with anchor-offset modelling to

TABLE VI: **Comparison of voxel-based semantic occupancy methods with and without Gaussian-based rendering losses:** Evaluated on Occ3D [18], SurroundOcc [19], and KITTI-360 [72], [73]. % denotes the percentage increase from the original mIoU to mIoU with Gaussian loss introduced. Methods are grouped by dataset and sorted by relative improvement. GP = GaussianPretrain, GSR = GaussRender loss, VS = VoxelSplat loss, * = value from GaussRender [24].

| **Model** | **mIoU** (Original) | **mIoU** (Gaussian) | % |
|---|---|---|---|
| PanoOcc (w/ GP) [128], [80] | 41.60 | 42.42 | 1.97 |
| BEVFormer (w/ GP) [34], [80] | 23.67 | 24.21 | 2.28 |
| SurroundOcc (w/ GSR) [19], [24] | 29.21 | 30.38 | 4.00 |
| FB-Occ (w/ VS) [127], [25] | 39.20 | 42.30 | 7.91 |
| TPVFormer (w/ GSR) [126], [24] | 27.83 | 30.48 | 9.52 |
| SurroundOcc (w/ GSR) [19], [24] | 20.30 | 20.82 | 2.56 |
| TPVFormer (w/ GSR) [126], [24] | 17.10 | 20.85 | 21.93 |
| Symphonies (w/ GSR) [129], [24] | 17.82* | 18.11 | 1.63 |
| SurroundOcc (w/ GSR) [19], [24] | 13.08 | 13.34 | 1.99 |

reduce redundant computation, achieving 48ms in its smaller variant. The next fastest, PanoOcc [128], is voxel-driven but employs a coarse-to-fine scheme, processing an eight-times sparser grid to cut both memory and runtime, while also improving accuracy.

TABLE VII: **Comparison of inference time and memory consumption of voxel and Gaussian-based supervised methods:** * means Occ3D, others are SurroundOcc. ** means the model is C+L. *** means the model is LiDAR only. Figures are sourced from their papers. Models are sorted by increasing latency, irrespective of dataset and GPU used.

| **Model** | **Gauss. Rep.** | **Latency** (ms) | **Memory** (GB) | **GPU** |
|---|---|---|---|---|
| RenderWorld [79] * | | - | 14.4 | A30 |
| PanoOcc [128] * | | 109 | 5.7 | A100 |
| SurroundOcc [19] | | 340 | 5.9 | 3090 |
| TACOCC [41] * | | 640 | 11.8 | A6000 |
| S2GO [87] * | ✓ | 69 | - | A100 |
| GaussianFusionOcc [39] *** | ✓ | 179 | 0.5 | A6000 |
| ODG [88] * | ✓ | 204 | - | A100 |
| GaussianFormer [23] | ✓ | 227 | 4.9 | 4090 |
| GaussianWorld [48] | ✓ | 228 | 7.0 | 4090 |
| GaussianFormer-2 [23] | ✓ | 357 | 3.1 | 4090 |
| GraphGSOcc [89] | ✓ | 368 | 6.1 | 4090 |
| GaussianFusionOcc [39] ** | ✓ | 460 | 2.6 | A6000 |
| GaussianFormer3D [42] ** | ✓ | 555 | 5.5 | A40 |

Other Gaussian models typically report 200+ms latency, still faster than most dense voxel approaches. Adding LiDAR to Gaussian models [39], [42] significantly increases runtime due to complex fusion steps, limiting real-time applicability. This can be mitigated by simple initialisation of Gaussian means with LiDAR point cloud data [86].

Memory consumption trends are clearer: Gaussian methods consistently require less memory, avoiding explicit modelling of empty space. GaussianFusionOcc [39] C+L runs at just 2.6GB, with most others around 5GB. Even the most efficient voxel model, PanoOcc, consumes 5.7GB, underscoring the high storage cost of voxel grids. For memory consumption, the LiDAR only GaussianFusionOcc achieves a tiny 0.5GB due to its efficient BEV encoded LiDAR data in conjunction with 3D Gaussians to model the scene, displaying that representation of input data is paramount in addition to scene representation.

## *D. Qualitative Analysis*

In this section, we analyse the predicted Gaussians and semantic occupancy outputs of models that adopt a Gaussian scene representation. Only open-source models with publicly available weights are considered, with the addition of GaussianFormer-MM [86]. Table VIII lists the analysed models along with relevant implementation details.

TABLE VIII: **Configuration of models that will be investigated:** Gaussian parameters shall be analysed for a random scene. In GaussTR, scale is restricted to [1,16]; however, it is then augmented with depth and camera intrinsic parameters, hence leading to potential unboundedness.

| **Model** | **Sup.** | **Total Gaussians** | **Scale Range** |
|---|---|---|---|
| GaussianFormer [23] | Strong | 25600 | [0.08 0.64] |
| GaussianFormer-2 [47] | Strong | 6400 | [0.01, 3.2] |
| GaussianWorld [48] | Strong | 25600 | [0.08, 0.64] |
| GaussTR [31] | Self | 1800 | [1e-6, $\infty$] |
| GaussianFormer-MM [86] | Strong | 102400 | [0.08, 0.64] |

As shown in Figure 7, the models on the left, GaussianFormer [23], GaussianWorld [48], and GaussianFormer-MM [86], contain significantly more Gaussians than the others. This results from their larger initialisation count and smaller scale range, which prevents individual Gaussians from covering wide areas. While this increases the number of Gaussians, the smaller scale range enables finer predictions, producing less "blobby" outputs. Despite the high number of Gaussians, many are not reflected in the voxelization, as they are effectively suppressed due to insufficient opacity.

GaussianWorld demonstrates more accurate drivable-surface prediction by leveraging temporal data, allowing it to correctly recover regions it had previously observed in past frames that other models fail to capture. GaussianFormer-MM provides a richer representation of the scene through the inclusion of high-quality LiDAR input, though it aggregates only three LiDAR sweeps and therefore incorporates less temporal context than GaussianWorld.

In contrast, GaussTR [31] initialises only 1600 Gaussians, far fewer than the 102,400 Gaussians of GaussianFormer-MM, resulting in much sparser scene representations. While the larger scale range helps compensate, the model still struggles to capture fine details, such as tree tops.

Finally, the visualisations also reveal insights into Gaussian property distributions. For instance, Gaussians from GaussianFormer and related models appear nearly spherical,

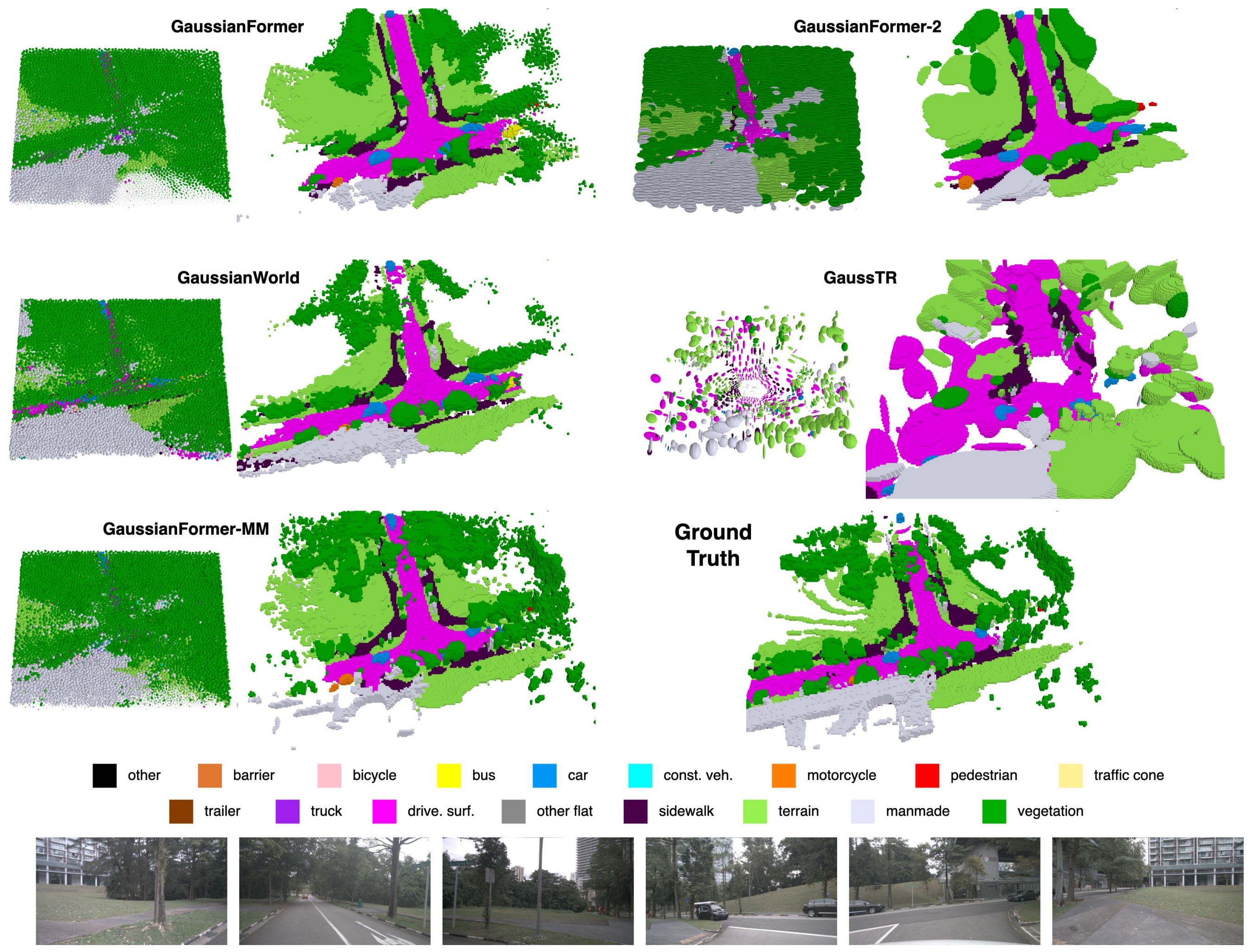

Fig. 7: **Qualitative analysis of several models:** Gaussians and predicted semantic occupancy are visualised alongside ground truth occupancy and RGB images evaluated on the SurroundOcc [19] dataset, with the exception of GaussTR, which uses the Occ3D [18] dataset. All Gaussians, regardless of their opacity, are rendered. We evaluate for sample token `edde57e6dfeb416e936d9056824b8253`, which occurs later in the scene sequence to allow for historical data to aggregate for GaussianWorld.

without adopting anisotropic ellipsoid shapes. Moreover, the Gaussians in GaussianFormer-2 [47] appear to share nearly identical rotations. These observations motivate a deeper exploration of Gaussian parameter distributions, which is presented in Section VI.

## VI. Gaussian Parameter Analysis

As discussed in Subsection V-C, methods with Gaussian scene representations generally achieve lower memory usage and reduced latency. Nonetheless, many models still employ more Gaussians than necessary, with some parameters contributing little to the final prediction. To assess efficiency, we analyse the Gaussian parameters of several models on the validation set, with full parameter distributions shown in Figure 8, following a methodology similar to Bagdasarian *et al.* [130].

### A. Mean

For the $x$ and $y$ components, Gaussian distributions are generally well spread, with the exceptions of GaussianFormer-MM [86] and GaussTR [31]. In GaussTR, this concentration arises from its 2D pseudo-ground-truth labels, which provide less reliable supervision for distant objects. To avoid heavy penalisation, the model tends not to allocate Gaussians far from the vehicle. Moreover, since initialisation occurs in the 2D camera view where near-field objects dominate, many Gaussians are placed close to the ego-vehicle. As also seen in Figure V-D, some Gaussians are redundantly initialised at the origin with very small scales, further contributing to this near-field density.

In GaussianFormer-MM, the clustering near the origin reflects the higher LiDAR point density close to the ego-vehicle, from which the Gaussians are initialised. Note that along the $y$ axis, the distribution shifts toward negative values, corresponding to regions behind the ego-vehicle. This

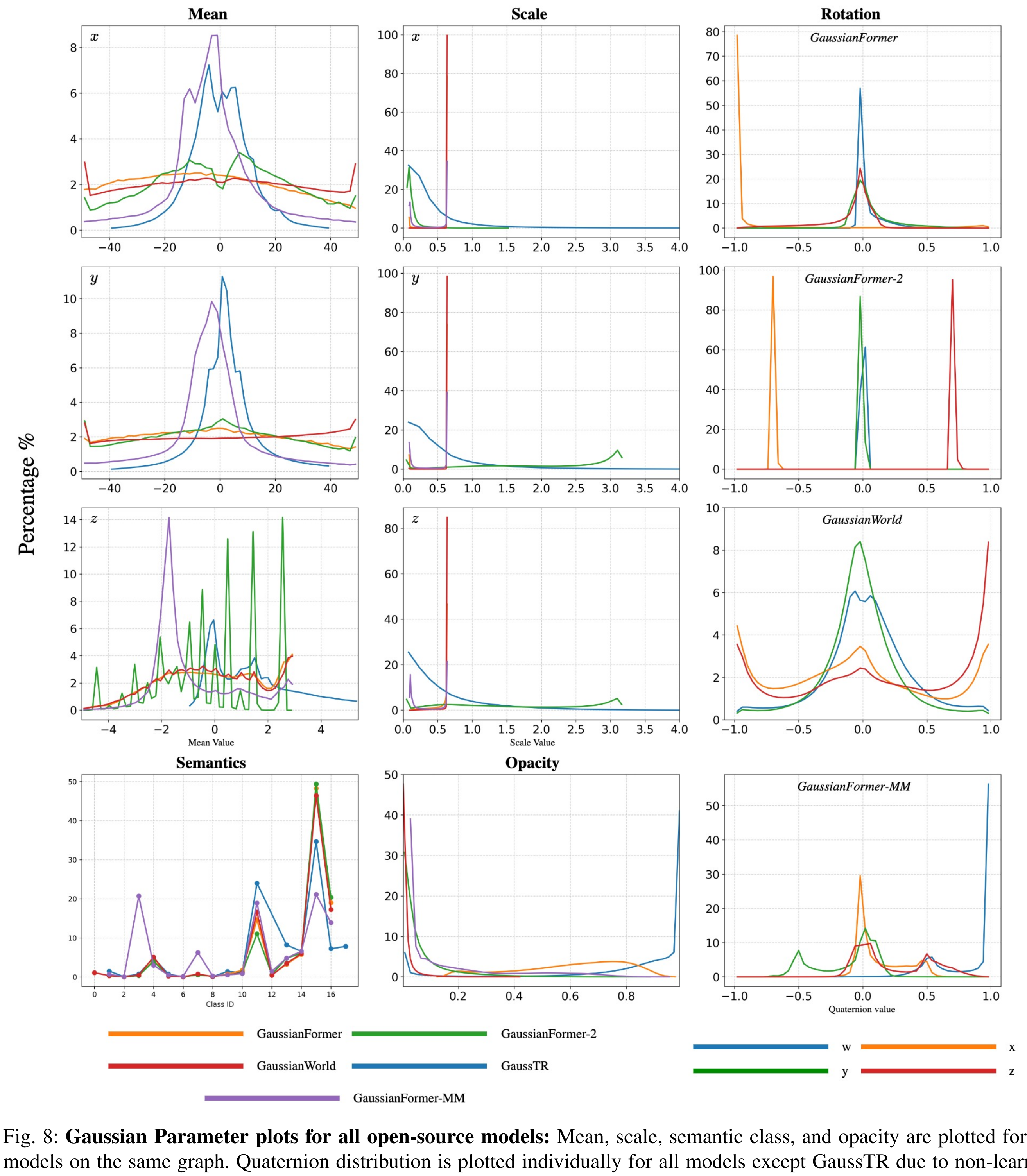

Fig. 8: **Gaussian Parameter plots for all open-source models:** Mean, scale, semantic class, and opacity are plotted for all models on the same graph. Quaternion distribution is plotted individually for all models except GaussTR due to non-learned rotation.

arises from aggregating historical LiDAR sweeps, which place past observations behind the current vehicle position. This shift in values is also seen for the $x$ axis, although the reason is unknown.

For the $z$ component, GaussianFormer-MM's Gaussians are concentrated on the road surface, again a consequence of LiDAR initialisation. GaussianFormer-2 [47] exhibits distinct peaks, likely originating from its initialisation strategy, which correctly identifies high-probability occupied regions such as roads, and tall structures such as trees.

Overall, these observations suggest that learnable 3D positions coupled with 3D supervision lead to a more evenly distributed set of Gaussians in space, producing better-balanced 3D predictions.

## B. Scale

In these models, scale is fixed to a given range, with specifics given in Table VIII. In GaussianFormer [23], GaussianWorld [120], and GaussianFormer-MM, Gaussians are approximately isotropic spheres with a fixed scale of $0.64$ along all axes. Thus, scale (and by extension rotation) is redundant for learning in these models. GaussTR learns scale more flexibly, showing a log-like distribution across axes; however, many scales collapse toward zero, effectively representing empty space, which is redundant. This could be mitigated with pruning strategies, as demonstrated in the superquadric-based model, QuadricFormer [131]. Additionally, a dynamic initialisation strategy could be beneficial to reducing redundant Gaussians, allowing the model to adapt the number of Gaussians to the scene, with denser scenes requiring more Gaussians and sparser scenes fewer.

GaussianFormer-2 also learns scale more robustly. The $y$ and $z$ axes trend toward larger values around $3.2$, though a subset still collapses toward zero. In contrast, the $x$ axis scale exhibits extreme flatness, with nearly all Gaussians below $0.25$. While this suggests compression along $x$, the qualitative analysis does not show such extreme thinness, highlighting the role of rotation in shaping final geometry.

It is evident that most models struggle to capture diverse scales, often resorting to fixed values to simplify learning. This limits their capacity to represent object size adaptively, for example, assigning a large, flat Gaussian to model a road surface or a small Gaussian to represent fine structures such as traffic cones. These issues highlight the need for more principled scale-learning strategies or adaptive pruning mechanisms to ensure Gaussians are allocated meaningfully across objects of varying sizes.

## C. Rotation

GaussianFormer and GaussianFormer-2 both exhibit nearly constant rotations. GaussianFormer converges to a quaternion of $(0, -1, 0, 0)$, corresponding to a 180° rotation about the $y$-axis, which is redundant given its isotropic spheres. GaussianFormer-2 stabilizes around $(0, 0.7, 0, 0.7)$, mapping $+X \rightarrow -Z$ and $Y \rightarrow -Y$. This explains the appearance of flattened Gaussians along the $z$-axis observed in the qualitative analysis.

GaussianWorld and GaussianFormer-MM produce more varied quaternion values; however, this remains inconsequential since their Gaussians are still constrained to isotropic spheres. GaussTR is excluded from quaternion analysis, as it does not learn rotation. Instead, its Gaussians are assigned rotations orthogonal to the originating camera view.

Similar to scale, rotation parameters also exhibit redundancy, as many models converge to learning a fixed rotation value. In dynamic scenes, incorporating an explicit rotation prediction module could be beneficial, for instance, to capture the orientation of vehicles and assign Gaussians aligned with their heading. Such a mechanism could further complement adaptive scale prediction, as discussed previously.

## D. Opacity

GaussianFormer shows a well-distributed opacity range, while GaussianFormer-MM and GaussianWorld exhibit values skewed toward zero. This is unexpected given that all three employ the same voxelization module. One possible explanation lies in the large empty Gaussian, which carries a learnable empty logit. In GaussianFormer, this logit stabilises at a higher value, forcing other Gaussians to adopt higher opacities to be voxelized. In contrast, GaussianWorld's empty logit remains lower, reducing the need for high opacity in individual Gaussians. For GaussianFormer-MM, the effect may stem from its very high count of Gaussians (102,400), where the cumulative sum of logits allows individual Gaussians to maintain lower opacity.

GaussianFormer-2's opacity distribution resembles that of GaussianFormer-MM but for different reasons: opacity here is normalised and treated as relative, used only for semantic weighting, while occupancy itself is computed independent of opacity, per (13).

GaussTR displays the most extreme behaviour, with opacity values concentrated near one. This is a direct consequence of its 2D view-based supervision: the model cannot account for occluded objects without 3D loss or temporal rendering. As a result, it drives most Gaussians to full opacity, ensuring foreground objects dominate the loss. While effective during training, this strategy can harm evaluation, since hidden or misinitialized Gaussians remain unseen in the render but later emerge during voxelization.

Opacity appears to be strongly influenced by the chosen loss function. For supervised methods that opt for a voxel-based loss, opacity effectively suppresses redundant Gaussians, fulfilling its intended role, though the heavy reliance on this mechanism suggests that pruning strategies could provide a more principled solution, as discussed previously. For self-supervised models, opacity largely governs foreground prioritisation during rasterisation; however, as previously discussed, this behaviour can be detrimental at evaluation.

## E. Semantics

Most models display broadly similar class distributions, with GaussianFormer-MM standing out as an exception. This model allocates noticeably more Gaussians to class 3 (bus) and class 7 (pedestrian). While this may partially reflect its much higher total Gaussian count (102,400), the effect appears significant and merits further investigation. Conversely, GaussianFormer-MM assigns fewer Gaussians to class 15 (manmade). This reduction is consistent with its use of LiDAR initialisation. As seen in the qualitative analysis (Figure 7), LiDAR improves accuracy for this class, reducing the need for redundant Gaussian assignments.

Semantic distributions generally mirror those of the ground-truth voxel grid. While this alignment is expected, it is not necessarily optimal for Gaussian scene representations. Large classes, such as the drivable surface (class ID 11), could be efficiently captured with only a few large Gaussians rather than many small ones, enabling a sparser and more memory-efficient representation.

## VII. Shortcomings and Future Directions

This section discusses the current limitations of Gaussian-driven semantic occupancy prediction models, both in isolation and in comparison to voxel-only approaches. It also outlines promising research directions that could address these limitations and advance the field. To avoid redundancy, we do not revisit the Gaussian parameter analysis already covered in Section VI.

### A. Multi-Modal Deployment

Multi-modal fusion has been extensively studied in voxel-driven semantic occupancy prediction, particularly for C+L [132], [133], [55], [51] and C+R [8], [134], [135], [136], [137] settings. In contrast, Gaussian-driven models are still in the early stages of multi-modal integration. For C+L, only three released models directly fuse LiDAR point cloud data with 3D Gaussians [86], [42], [39], while TACOcc [41] performs voxel-level fusion of multi-modal features. Given the inherently point-based nature of 3D Gaussians, this limited exploration is surprising. Even simple point-based initialisation techniques can yield substantial performance gains [86], [47], making LiDAR–Gaussian fusion a clear opportunity for further research.

Radar integration is even less explored, with only two Gaussian-driven models incorporating it [39], [86]. Beyond depth cues, radar offers rich velocity information that could enhance tasks such as Gaussian flow for scene forecasting. Additionally, it appears there is a trend towards reducing the number of Gaussians, e.g., S2GO-Small [87] uses just 900 multi-offset Gaussians, suggesting that radar-informed initialisation could provide strong priors for compact yet expressive scene representations.

### B. Fine Detail Modelling

Capturing fine-grained scene detail generally requires more Gaussians, which becomes a challenge at high voxel resolutions (e.g., $0.1\,\mathrm{m}$ in nuCraft [20]) due to increased memory and inference costs. Although Gaussians remain more compact than voxels in such settings, alternative primitive-based scene representations, such as superquadrics, may offer better efficiency. Superquadrics can flexibly represent cuboid or cylindrical shapes, and have been explored in QuadricFormer [131], which achieves performance comparable to a 6400-Gaussian model while using only 1600 superquadrics. Such representations may be better suited for structured objects such as buses or trucks, suggesting a potential hybrid Gaussian–superquadric approach.

### C. Scene Flow

Scene flow estimation has been actively explored in both voxel-driven methods [138], [139], [140] and Gaussian-driven approaches, particularly in self-supervised settings. Existing Gaussian models enforce temporal consistency [82], [84] and propagate Gaussians across adjacent timesteps [83], [81], [88] to improve stability. The VoxelSplat loss [25] further demonstrates how temporal consistency can benefit voxel methods through Gaussianization. Dedicated methods such as GaussianAD [141] show the promise of Gaussians for scene forecasting. Given their continuous, explicit nature, Gaussian representations are well-positioned for advanced scene flow modelling, including direct evaluation of multi-step forecasts for safety-critical tasks like collision avoidance [81].

### D. 2D Supervision

A key advantage of Gaussian-driven methods is that they can be trained without dense 3D occupancy labels, instead relying on 2D supervision from VFMs [30], [31], [83] or sparse LiDAR labels [79], [81], [88], [87]. However, for models with access to manually annotated 3D data, 2D-only supervision is a bottleneck, as peak performance on voxel grids typically requires full 3D labels. For example, GSRender [81], a weakly supervised voxel-based method, improves by 38.39% mIoU when 3D labels are added to its 2D supervision. Notably, there is currently no weakly supervised Gaussian-driven model in the literature. Developing such models could reduce annotation costs while preserving the efficiency advantages of Gaussian representations. It is important to note that relying solely on 2D supervision can introduce issues such as reduced confidence in predictions for distant objects and an imbalance in loss across 3D space, as discussed in Subsection VI-D.

## VIII. Conclusion

This survey has provided an in-depth examination of Gaussian-driven semantic occupancy prediction models, covering their architectural design choices, supervision strategies, and multi-modal fusion techniques, with a particular focus on the motivations and implications of adopting Gaussian-driven representations. Through comparative analysis, we evaluated these models against voxel-based counterparts, highlighting the respective strengths, limitations, and trade-offs of each approach. We observed that Gaussian scene representations inherently yield a reduced memory footprint, while incorporating Gaussian rendering into voxel-based methods consistently improves performance. This highlights the value of Gaussians as a complementary mechanism within semantic occupancy prediction. Nonetheless, certain shortcomings remain, such as the performance limitations of models trained with only 2D supervision. In addition, our analysis of Gaussian parameter distributions revealed the redundancy of scale and rotation in many models, along with the effect of 2D supervision on the variability of Gaussian mean, providing further insight into how these models allocate representational capacity across the scene.

The empirical results and discussions presented here aim to equip researchers with a clearer understanding of the current state of Gaussian-driven methods, their practical considerations, and the contexts in which they excel. We hope this work serves as both a comprehensive reference and a catalyst for further advancements in semantic occupancy prediction leveraging 3D Gaussians.

## References

[1] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *Conference on robot learning*. PMLR, 2018, pp. 947–956.

[2] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *European Conference on Computer Vision*. Springer, 2020, pp. 414–430.

[3] H. Wang, P. Cai, Y. Sun, L. Wang, and M. Liu, "Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 731–13 737.

[4] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 618–11 628.

[6] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," *arXiv preprint arXiv:2205.13542*, 2022.

[7] A. W. Harley, Z. Fang, J. Li, R. Ambrus, and K. Fragkiadaki, "Simple-bev: What really matters for multi-sensor bev perception?" in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2759–2765.

[8] Z. Ming, J. S. Berrio, M. Shan, and S. Worrall, "Occfusion: Multi-sensor fusion framework for 3d semantic occupancy prediction," *IEEE Transactions on Intelligent Vehicles*, 2024.

[9] Z. Meng, Y. Song, Y. Zhang, Y. Nan, and Z. Bai, "Traffic object detection for autonomous driving fusing lidar and pseudo 4d-radar under bird's-eye-view," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 18 185–18 195, 2024.

[10] X. Peng, H. Sun, K. Bierzynski, A. Fischbacher, L. Servadei, and R. Wille, "Mutualforce: Mutual-aware enhancement for 4d radar-lidar 3d object detection," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.

[11] J. Mao, S. Shi, X. Wang, and H. Li, "3d object detection for autonomous driving: A comprehensive survey," *International Journal of Computer Vision*, vol. 131, no. 8, pp. 1909–1963, 2023.

[12] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[13] H. Li, C. Sima, J. Dai, W. Wang, L. Lu, H. Wang, J. Zeng, Z. Li, J. Yang, H. Deng *et al.*, "Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 4, pp. 2151–2170, 2023.

[14] Y. Ma, T. Wang, X. Bai, H. Yang, Y. Hou, Y. Wang, Y. Qiao, R. Yang, and X. Zhu, "Vision-centric bev perception: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[15] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *European conference on computer vision*. Springer, 2020, pp. 194–210.

[16] R. Izquierdo, A. Quintanar, D. F. Llorca, I. G. Daza, N. Hernández, I. Parra, and M. Á. Sotelo, "Vehicle trajectory prediction on highways using bird eye view representations and deep learning," *Applied Intelligence*, vol. 53, no. 7, pp. 8370–8388, 2023.

[17] H. Xu, J. Chen, S. Meng, Y. Wang, and L.-P. Chau, "A survey on occupancy perception for autonomous driving: The information fusion perspective," *Information Fusion*, vol. 114, p. 102671, 2025.

[18] X. Tian, T. Jiang, L. Yun, Y. Mao, H. Yang, Y. Wang, Y. Wang, and H. Zhao, "Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving," *Advances in Neural Information Processing Systems*, vol. 36, pp. 64 318–64 330, 2023.

[19] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou, and J. Lu, "Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 21 729–21 740.

[20] B. Zhu, Z. Wang, and H. Li, "nucraft: Crafting high resolution 3d semantic occupancy for unified 3d scene understanding," in *European Conference on Computer Vision*. Springer, 2024, pp. 125–141.

[21] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[22] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[23] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu, "Gaussianformer: Scene as gaussians for vision-based 3d semantic occupancy prediction," in *European Conference on Computer Vision*. Springer, 2024, pp. 376–393.

[24] L. Chambon, E. Zablocki, A. Boulch, M. Chen, and M. Cord, "Gaussrender: Learning 3d occupancy with gaussian rendering," *arXiv preprint arXiv:2502.05040*, 2025.

[25] Z. Zhu, S. Wang, J. Xie, J.-j. Liu, J. Wang, and J. Yang, "Voxelsplat: Dynamic gaussian splatting as an effective loss for occupancy and flow prediction," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6761–6771.

[26] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3d gaussian splatting as new era: A survey," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[27] G. Chen and W. Wang, "A survey on 3d gaussian splatting," 2025. [Online]. Available: https://arxiv.org/abs/2401.03890

[28] S. Zhu, G. Wang, X. Kong, D. Kong, and H. Wang, "3d gaussian splatting in robotics: A survey," *arXiv preprint arXiv:2410.12262*, 2024.

[29] Y. Huang, W. Zheng, B. Zhang, J. Zhou, and J. Lu, "Selfocc: Self-supervised vision-based 3d occupancy prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 19 946–19 956.

[30] W. Gan, F. Liu, H. Xu, N. Mo, and N. o Yokoya, "Gaussianocc: Fully self-supervised and efficient 3d occupancy estimatio n with gaussian splatting," *CoRR*, vol. abs/2408.11447, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2408.11447

[31] H. Jiang, L. Liu, T. Cheng, X. Wang, T. Lin, Z. Su, W. Liu, and X. Wang, "Gausstr: Foundation model-aligned gaussian transformer for self-supervised 3d spatial understanding," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 11 960–11 970.

[32] T. Roddick, A. Kendall, and R. Cipolla, "Orthographic feature transform for monocular 3d object detection," *arXiv preprint arXiv:1811.08188*, 2018.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[35] F. Chabot, N. Granger, and G. Lapouge, "Gaussianbev: 3d gaussian representation meets perception models for bev segmentation," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2025, pp. 2250–2259.

[36] P. Wang, "Research on comparison of lidar and camera in autonomous driving," in *Journal of Physics: Conference Series*, vol. 2093, no. 1. IOP Publishing, 2021, p. 012032.

[37] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 2, 2021, pp. 1201–1209.

[38] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[39] T. Pavković, M.-A. N. Mahani, J. Niedermayer, and J. Betz, "Gaussianfusionocc: A seamless sensor fusion approach for 3d occupancy prediction using 3d gaussians," 2025. [Online]. Available: https://arxiv.org/abs/2507.18522

[40] J. Schramm, N. Vödisch, K. Petek, B. R. Kiran, S. Yogamani, W. Burgard, and A. Valada, "Bevcar: Camera-radar fusion for bev map and object segmentation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 1435–1442.

[41] L. Lei, S. Xu, Y. Bai, and X. Wei, "Tacocc: Target-adaptive cross-modal fusion with volume rendering for 3d semantic occupancy," *arXiv preprint arXiv:2505.12693*, 2025.

[42] L. Zhao, S. Wei, J. Hays, and L. Gan, "Gaussianformer3d: Multi-modal gaussian-based semantic occupancy prediction with 3d deformable attention," *arXiv preprint arXiv:2505.10685*, 2025.

[43] A.-Q. Cao and R. De Charette, "Monoscene: Monocular 3d semantic scene completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3991–4001.

[44] Z. Yu, C. Shu, J. Deng, K. Lu, Z. Liu, J. Yu, D. Yang, H. Li, and Y. Chen, "Flashocc: Fast and memory-efficient occupancy prediction via channel-to-height plugin," *arXiv preprint arXiv:2311.12058*, 2023.

[45] Q. Ma, X. Tan, Y. Qu, L. Ma, Z. Zhang, and Y. Xie, "Cotr: Compact occupancy transformer for vision-based 3d occupancy prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 936–19 945.

[46] J. Hou, X. Li, W. Guan, G. Zhang, D. Feng, Y. Du, X. Xue, and J. Pu, "Fastocc: Accelerating 3d occupancy prediction by fusing the 2d bird's-eye view and perspective view," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 16 425–16 431.

[47] Y. Huang, A. Thammatadatrakoon, W. Zheng, Y. Zhang, D. Du, and J. Lu, "Gaussianformer-2: Probabilistic gaussian superposition for efficient 3d occupancy prediction," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 27 477–27 486.

[48] S. Zuo, W. Zheng, Y. Huang, J. Zhou, and J. Lu, "Gaussianworld: Gaussian world model for streaming 3d occupancy prediction," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6772–6781.

[49] ——, "Pointocc: Cylindrical tri-perspective view for point-based 3d semantic occupancy prediction," *arXiv preprint arXiv:2308.16896*, 2023.

[50] C. B. Rist, D. Emmerichs, M. Enzweiler, and D. M. Gavrila, "Semantic scene completion using local deep implicit functions on lidar data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 10, pp. 7205–7218, 2021.

[51] X. Wang, Z. Zhu, W. Xu, Y. Zhang, Y. Wei, X. Chi, Y. Ye, D. Du, J. Lu, and X. Wang, "Openoccupancy: A large scale benchmark for surrounding semantic occupancy perception," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 850–17 859.

[52] L. Roldao, R. De Charette, and A. Verroust-Blondet, "Lmscnet: Lightweight multiscale 3d semantic completion," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 111–119.

[53] R. Cheng, C. Agia, Y. Ren, X. Li, and L. Bingbing, "S3cnet: A sparse semantic scene completion network for lidar point clouds," in *Conference on Robot Learning*. PMLR, 2021, pp. 2148–2161.

[54] J. Zhang, Y. Ding, and Z. Liu, "Occfusion: Depth estimation free multi-sensor fusion for 3d occupancy prediction," in *Proceedings of the Asian Conference on Computer Vision*, 2024, pp. 3587–3604.

[55] J. Pan, Z. Wang, and L. Wang, "Co-occ: Coupling explicit feature fusion with volume rendering regularization for multi-modal 3d semantic occupancy prediction," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5687–5694, 2024.

[56] S. Sze and L. Kunze, "Real-time 3d semantic occupancy prediction for autonomous vehicles using memory-efficient sparse convolution," in *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2024, pp. 1286–1293.

[57] M. Levoy and P. Hanrahan, "Light field rendering," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 441–452.

[58] S. M. Seitz and C. R. Dyer, "View morphing," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 21–30.

[59] P. Debevec, Y. Yu, and G. Borshukov, "Efficient view-dependent image-based rendering with projective texture-mapping," in *Eurographics Workshop on Rendering Techniques*. Springer, 1998, pp. 105–116.

[60] J. Shade, S. Gortler, L.-w. He, and R. Szeliski, "Layered depth images," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 231–242.

[61] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5515–5524.

[62] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5855–5864.

[63] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.

[64] G. J. McLachlan, "Mahalanobis distance," *Resonance*, vol. 4, no. 6, pp. 20–26, 1999.

[65] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Ewa volume splatting," in *Proceedings Visualization, 2001. VIS'01.* IEEE, 2001, pp. 29–538.

[66] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4d gaussian splatting for real-time dynamic scene rendering," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 20 310–20 320.

[67] X. Zhang, Z. Liu, Y. Zhang, X. Ge, D. He, T. Xu, Y. Wang, Z. Lin, S. Yan, and J. Zhang, "Mega: Memory-efficient 4d gaussian splatting for dynamic scenes," *arXiv preprint arXiv:2410.13613*, 2024.

[68] Y. Yuan, Q. Shen, X. Yang, and X. Wang, "1000+ fps 4d gaussian splatting for dynamic scene rendering," *arXiv preprint arXiv:2503.16422*, 2025.

[69] J. Guo, X. Ma, Y. Fan, H. Liu, and Q. Li, "Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting," *arXiv preprint arXiv:2403.15624*, 2024.

[70] X. Wang, C. Lan, H. Zhu, Z. Chen, and Y. Lu, "Gsemsplat: Generalizable semantic 3d gaussian splatting from uncalibrated image pairs," *arXiv preprint arXiv:2412.16932*, 2024.

[71] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9297–9307.

[72] Y. Liao, J. Xie, and A. Geiger, "Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3292–3310, 2022.

[73] Y. Li, S. Li, X. Liu, M. Gong, K. Li, N. Chen, Z. Wang, Z. Li, T. Jiang, F. Yu *et al.*, "Sscbench: A large-scale 3d semantic scene completion benchmark for autonomous driving," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 333–13 340.

[74] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[75] H. Zhai, J. Mei, C. Min, L. Chen, F. Zhao, and Y. Hu, "Wildocc: A benchmark for off-road 3d semantic occupancy prediction," *arXiv preprint arXiv:2410.15792*, 2024.

[76] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 1110–1116.

[77] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023.

[78] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3d packing for self-supervised monocular depth estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[79] Z. Yan, W. Dong, Y. Shao, Y. Lu, L. Haiyang, J. Liu, H. Wang, Z. Wang, Y. Wang, F. Remondino, and Y. Ma, "Renderworld: World model with self-supervised 3d label," 2025. [Online]. Available: https://arxiv.org/abs/2409.11356

[80] S. Xu, F. Li, S. Jiang, Z. Song, L. Liu, and Z. xin Yang, "Gaussianpretrain: A simple unified 3d gaussian representation for visual pre-training in autonomous driving," 2024. [Online]. Available: https://arxiv.org/abs/2411.12452

[81] Q. Sun, C. Shu, S. Zhou, Z. Yu, Y. Chen, D. Yang, and Y. Chun, "Gsrender: Deduplicated occupancy prediction via weakly supervised 3d gaussian splatting," *arXiv preprint arXiv:2412.14579*, 2024.

[82] X. Zhou, J. Wang, Y. Wang, Y. Wei, N. Dong, and M.-H. Yang, "Autoocc: Automatic open-ended semantic occupancy annotation via vision-language guided gaussian splatting," *arXiv preprint arXiv:2502.04981*, 2025.

[83] S. Boeder, F. Gigengack, and B. Risse, "Gaussianflowocc: Sparse and weakly supervised occupancy estimation using gaussian splatting and temporal flow," *arXiv preprint arXiv:2502.17288*, 2025.

[84] F. Zhang, H. Yang, Z. Zhang, Z. Huang, and Y. Luo, "Tt-occ: Test-time compute for self-supervised occupancy via spatio-temporal gaussian splatting," *arXiv preprint arXiv:2503.08485*, 2025.

[85] M. Chen, W. Chen, M. Yang, Y. Zhang, T. Han, X. Li, Y. Li, and H. Zhao, "Tgp: Two-modal occupancy prediction with 3d gaussian and sparse points for 3d environment awareness," *arXiv preprint arXiv:2503.09941*, 2025.

[86] S. Hayes, G. Sistu, and C. Eising, "Leveraging frozen foundation models and multimodal fusion for bev segmentation and occupancy prediction," *IEEE Open Journal of Vehicular Technology*, 2025.

[87] J. Park, Y. Hu, C. Peng, W. Zheng, K. Kitani, and W. Zhan, "S2go: Streaming sparse gaussian occupancy prediction," *arXiv preprint arXiv:2506.05473*, 2025.

[88] Y. Shi, Y. Zhu, S. Han, J. Jeong, A. Ansari, H. Cai, and F. Porikli, "Odg: Occupancy prediction using dual gaussians," *arXiv preprint arXiv:2506.09417*, 2025.

[89] K. Song, Y. Wu, C. Siu, and H. Xiong, "Graphgsocc: Semantic and geometric graph transformer for 3d gaussian splating-based occupancy prediction," *arXiv preprint arXiv:2506.14825*, 2025.

[90] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[91] H. Liu, Y. Chen, H. Wang, Z. Yang, T. Li, J. Zeng, L. Chen, H. Li, and L. Wang, "Fully sparse 3d occupancy prediction," in *European Conference on Computer Vision*. Springer, 2024, pp. 54–71.

[92] D. P. Kingma, M. Welling *et al.*, "Auto-encoding variational bayes," 2013.

[93] M. Pan, J. Liu, R. Zhang, P. Huang, X. Li, H. Xie, B. Wang, L. Liu, and S. Zhang, "Renderocc: Vision-centric 3d occupancy prediction with 2d rendering supervision," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 404–12 411.

[94] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[95] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[96] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," *arXiv preprint arXiv:1903.05662*, 2019.

[97] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, and S. Shen, "Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[98] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[99] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[100] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[101] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[102] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[103] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[104] J. Wang, Z. Liu, Q. Meng, L. Yan, K. Wang, J. Yang, W. Liu, Q. Hou, and M.-M. Cheng, "Opus: occupancy prediction using a sparse set," *Advances in Neural Information Processing Systems*, vol. 37, pp. 119 861–119 885, 2024.

[105] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in neural information processing systems*, vol. 35, pp. 16 344–16 359, 2022.

[106] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[107] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[108] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 3744–3753.

[109] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu *et al.*, "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 24 185–24 198.

[110] Z. Lin, Y. Wang, and Z. Tang, "Training-free open-ended object detection and segmentation via attention as prompts," *Advances in Neural Information Processing Systems*, vol. 37, pp. 69 588–69 606, 2024.

[111] S. Abnar and W. Zuidema, "Quantifying attention flow in transformers," *arXiv preprint arXiv:2005.00928*, 2020.

[112] L. Piccinelli, Y.-H. Yang, C. Sakaridis, M. Segu, S. Li, L. Van Gool, and F. Yu, "Unidepth: Universal monocular metric depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 106–10 116.

[113] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.

[114] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv preprint arXiv:2306.12156*, 2023.

[115] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv preprint arXiv:2306.14289*, 2023.

[116] H. Zhang, F. Li, X. Zou, S. Liu, C. Li, J. Yang, and L. Zhang, "A simple framework for open-vocabulary segmentation and detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 1020–1031.

[117] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.

[118] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*. Springer, 2020, pp. 402–419.

[119] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proceedings of the IEEE*

*conference on computer vision and pattern recognition*, 2018, pp. 4413–4421.

[120] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.

[121] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European conference on computer vision*. Springer, 2024, pp. 38–55.

[122] C. Zhang, J. Yan, Y. Wei, J. Li, L. Liu, Y. Tang, Y. Duan, and J. Lu, "Occnerf: Self-supervised multi-camera occupancy prediction with neural radiance fields," *CoRR*, 2023.

[123] L. Barsellotti, L. Bianchi, N. Messina, F. Carrara, M. Cornia, L. Baraldi, F. Falchi, and R. Cucchiara, "Talking to dino: Bridging self-supervised vision backbones with language for open-vocabulary segmentation," *arXiv preprint arXiv:2411.19331*, 2024.

[124] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[125] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[126] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu, "Tri-perspective view for vision-based 3d semantic occupancy prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 9223–9232.

[127] Z. Li, Z. Yu, D. Austin, M. Fang, S. Lan, J. Kautz, and J. M. Alvarez, "Fb-occ: 3d occupancy prediction based on forward-backward view transformation," *arXiv preprint arXiv:2307.01492*, 2023.

[128] Y. Wang, Y. Chen, X. Liao, L. Fan, and Z. Zhang, "Panoocc: Unified occupancy representation for camera-based 3d panoptic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 17 158–17 168.

[129] H. Jiang, T. Cheng, N. Gao, H. Zhang, T. Lin, W. Liu, and X. Wang, "Symphonize 3d semantic scene completion with contextual instance queries," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 258–20 267.

[130] M. T. Bagdasarian, P. Knoll, Y. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern, "3dgs. zip: A survey on 3d gaussian splatting compression methods," in *Computer Graphics Forum*, vol. 44, no. 2. Wiley Online Library, 2025, p. e70078.

[131] S. Zuo, W. Zheng, X. Han, L. Yang, Y. Pan, and J. Lu, "Quadricformer: Scene as superquadrics for 3d semantic occupancy prediction," *arXiv preprint arXiv:2506.10977*, 2025.

[132] Y. Shi, K. Jiang, K. Wang, K. Qian, Y. Wang, J. Li, T. Wen, M. Yang, Y. Xu, and D. Yang, "Effocc: A minimal baseline for efficient fusion-based 3d occupancy network," *arXiv e-prints*, pp. arXiv–2406, 2024.

[133] Z. Yang, Y. Dong, H. Wang, L. Ma, Z. Cui, Q. Liu, and H. Pei, "Daocc: 3d object detection assisted multi-sensor fusion for 3d occupancy prediction," *arXiv preprint arXiv:2409.19972*, 2024.

[134] Y. Kim, J. Shin, S. Kim, I.-J. Lee, J. W. Choi, and D. Kum, "Crn: Camera radar net for accurate, robust, efficient 3d perception," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 615–17 626.

[135] P. Wolters, J. Gilg, T. Teepe, F. Herzog, A. Laouichi, M. Hofmann, and G. Rigoll, "Unleashing hydra: Hybrid fusion, depth consistency and radar for unified 3d perception," *arXiv preprint arXiv:2403.07746*, 2024.

[136] Y. Ma, J. Mei, X. Yang, L. Wen, W. Xu, J. Zhang, X. Zuo, B. Shi, and Y. Liu, "Licrocc: Teach radar for accurate semantic occupancy prediction using lidar and camera," *IEEE Robotics and Automation Letters*, 2024.

[137] Z. Lin, H. Jin, Y. Wang, Y. Wei, and N. Dong, "Teocc: Radar-camera multi-modal occupancy prediction via temporal enhancement," in *ECAI 2024*. IOS Press, 2024, pp. 129–136.

[138] Y. Yang, J. Mei, Y. Ma, S. Du, W. Chen, Y. Qian, Y. Feng, and Y. Liu, "Driving in the occupancy world: Vision-centric 4d occupancy forecasting and planning via world models for autonomous driving," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 9, 2025, pp. 9327–9335.

[139] D. Chen, J. Fang, W. Han, X. Cheng, J. Yin, C. Xu, F. S. Khan, and J. Shen, "Alocc: adaptive lifting-based 3d semantic occupancy and cost volume-based flow prediction," *arXiv preprint arXiv:2411.07725*, 2024.

[140] H. Zhang, X. Yan, Y. Xue, Z. Guo, S. Cui, Z. Li, and B. Liu, "D2-world: An efficient world model through decoupled dynamic flow," *arXiv preprint arXiv:2411.17027*, 2024.

[141] W. Zheng, J. Wu, Y. Zheng, S. Zuo, Z. Xie, L. Yang, Y. Pan, Z. Hao, P. Jia, X. Lang *et al.*, "Gaussianad: Gaussian-centric end-to-end autonomous driving," *arXiv preprint arXiv:2412.10371*, 2024.