# SparseOcc: Rethinking Sparse Latent Representation for Vision-Based Semantic Occupancy Prediction

Pin Tang[1]    Zhongdao Wang[2]    Guoqing Wang[1]    Jilai Zheng[1]
Xiangxuan Ren[1]    Bailan Feng[2]    Chao Ma[1*]

[1] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[2] Huawei Noah's Ark Lab

{pin.tang,guoqing.wang,zhengjilai,bunny_renxiangxuan,chaoma}@sjtu.edu.cn

{wangzhongdao,fengbailan}@huawei.com

Project page: https://pintang1999.github.io/sparseocc.html

## Abstract

*Vision-based perception for autonomous driving requires an explicit modeling of a 3D space, where 2D latent representations are mapped and subsequent 3D operators are applied. However, operating on dense latent spaces introduces a cubic time and space complexity, which limits scalability in terms of perception range or spatial resolution. Existing approaches compress the dense representation using projections like Bird's Eye View (BEV) or Tri-Perspective View (TPV). Although efficient, these projections result in information loss, especially for tasks like semantic occupancy prediction. To address this, we propose SparseOcc, an efficient occupancy network inspired by sparse point cloud processing. It utilizes a lossless sparse latent representation with three key innovations. Firstly, a 3D sparse diffuser performs latent completion using spatially decomposed 3D sparse convolutional kernels. Secondly, a feature pyramid and sparse interpolation enhance scales with information from others. Finally, the transformer head is redesigned as a sparse variant. SparseOcc achieves a remarkable 74.9% reduction on FLOPs over the dense baseline. Interestingly, it also improves accuracy, from 12.8% to 14.1% mIOU, which in part can be attributed to the sparse representation's ability to avoid hallucinations on empty voxels.*

## 1. Introduction

Accurate perception of the surrounding environment is crucial for autonomous driving systems [11, 14]. In recent years, vision-based 3D perception algorithms have gained significant attention and advancement due to their cost-

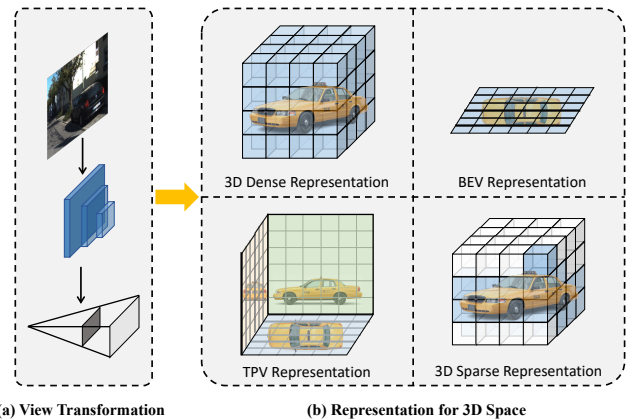---

*  Corresponding author.



Figure 1. (a) Vision-based perception methods for autonomous driving typically first extract image features by a 2D latent encoder and then map them to 3D using view transformation. (b) For the 3D latent space, existing methods mostly employ the dense, BEV, or TPV representation, while we rethink the possibility of using sparse representation to achieve superior efficiency and accuracy.

effectiveness. The typical workflow involves employing a 2D encoder to extract latent representations from images. A view transformation method, such as lift-splat-shoot (LSS) [25], is then applied to lift the perspective 2D latent features to a 3D voxel space, utilizing predicted depth information. This 3D scene representation serves as the foundation for deriving geometry and semantic information to describe the driving environment, supporting various 3D perception tasks including object detection [15, 17, 40, 43], semantic segmentation [11, 30, 36], and semantic occupancy prediction [13, 33, 34]. In this study, we specifically focus on the challenging task of semantic occupancy prediction [31–34], which entails predicting both static and dynamic elements within the scene.

Several alternatives are available for representing 3D

spatial latent information. The dense representation [33] is the most straightforward approach, storing features in continuous memory and enabling direct application of dense 3D operations like convolutions. However, this representation is redundant and inefficient, as approximately 67% of the considered 3D space is empty[1]. Bird Eye's View (BEV) [12, 19, 22] has gained popularity as a recent prevalent representation. It involves projecting the 3D space onto a BEV plane, significantly reducing computational costs by leveraging efficient 2D building blocks in a BEV encoder. However, this projection introduces the loss of geometry information, thereby limiting the fine-grained capacity of the BEV representation to comprehend the 3D scene structure. A Tri-Perspective View (TPV) [13] representation is proposed to mitigate the information loss, but it still suffers from degraded perception accuracy.

In this paper, we seek a latent representation that encodes the 3D scene structure in a lossless manner while minimizing computational costs. Drawing inspiration from the similarities between sparse 3D vision features and point clouds, we rethink the feasibility of employing a pure sparse representation for the 3D latent space, a common practice in point cloud processing [7]. Specifically, we utilize the coordinate (COO) format to store sparse tensors and introduce a series of sparse building blocks tailored for the sparse representation. Our proposed method, SparseOcc, is an occupancy network where all 3D layers operate on sparse tensors. The key designs of SparseOcc are as follows:

- **Sparse Latent Diffuser:** This component enables the propagation of non-empty features to adjacent empty regions, facilitating scene completion. To ensure efficiency, a 3D diffusion kernel is spatially decomposed into a combination of three orthogonal convolutional kernels.
- **Sparse Feature Pyramid:** We build a feature pyramid that incorporates sparse interpolation operations to enhance scales with information from other scales. This pyramid design expands the reception fields, reducing the need for excessive diffusers within each scale, which helps preserve sparsity.
- **Sparse Transformer Head:** The final component of SparseOcc is a 3D sparse transformer head responsible for generating semantic occupancy predictions. By segmenting only occupied voxels rather than the entire 3D volume, we achieve a remarkable reduction in computational costs.

Based on the sparse latent representation, SparseOcc achieves a significant reduction in computation overhead. In the nuScenes-Occupancy benchmark [33], it reduces the FLOPs of existing approaches using dense or TPV representations by **59.8% to 74.9%**, and memory usage by **31.6% to 40.9%**. Remarkably, the semantic occupancy

---

[1]We conduct statics using the ground-truth occupancy labels of the first 10 sequences in the SemanticKITTI dataset.

accuracy not only remains intact but improves, surpassing the state-of-the-art C-CONet. Specifically, the semantic occupancy accuracy increases from 12.8% to 14.1% mIoU. This improvement highlights the superiority of the sparse representation over the dense one in terms of accuracy, as it naturally avoids hallucinations on empty voxels. Considering the effectiveness and efficiency demonstrated by SparseOcc, we propose that it can serve as a new baseline for occupancy networks.

## 2. Related Work

### 2.1. 3D Scene Representation

**3D Dense Representation.** Representing the surrounding environment with spatial latent information is an indispensable procedure for autonomous driving perception algorithms. A straightforward solution is to split the scene into voxels and describe the scene with 3D volume representation where 3D operators are applied [4, 28, 33, 34, 41]. For example, OpenOccupancy [33] first uses a 2D encoder to extract image features and further uses LSS to lift the features to 3D space. Then, ResNet3D [9] and FPN3D [20] are used to diffuse non-empty features to adjacent empty areas. SurroudOcc [34] uses successive deformable cross attention layers [45] to transform the multi-scale image features to multi-scale 3D dense volume. However, these 3D operators are often in cubic time and spatial time complexity, which is not affordable in practice.

**BEV Representation.** The past several years have witnessed the prosperous development of BEV representation in tasks such as 3D object detection [8, 17, 19, 22], BEV semantic segmentation [24, 26, 42], and instance prediction [10]. They either forward project the image features to 3D spaces with estimated depth, then compress the 3D feature volume to BEV map, or update the BEV queries via backward deformable attention. Despite efficiency, the geometry lossy projection results in the relatively coarse representation of the 3D scene, hindering its generalization to the fine-grained semantic occupancy prediction task.

**TPV Representation.** To make a trade-off between efficacy and efficiency, TPV representation is proposed [13, 47]. It first constructs three cross-planes perpendicular to each other to represent the 3D scene. Then, sets of queries are initialized on these planes to aggregate features from images and exchange features across views via the attention mechanism. Then, they efficiently reconstruct each voxel in the 3D space by summing its projected features on the three planes for downstream tasks. Although the loss of geometry is mitigated, it is still hard to capture the complex driving scene, leading to degraded performance.

In this paper, we rethink the 3D scene representation from a sparse view and seek an effective but also efficient method for semantic occupancy prediction.

## 2.2. Scene Completion in Occupancy Prediction

Both LiDAR points and lifted 3D representation only cover the initial intersection surface between the ray from the sensor and objects. Hence, a few methods have been proposed to diffuse non-empty features to surrounding empty regions [18, 35]. These methods essentially are the same as the 3D dense representation based ones since they also stack 3D dense operators to diffuse features. In contrast, we explore the potential of using a pure sparse representation and propose a sparse latent diffuser for scene completion. This approach rethinks the conventional method of feature diffusion in 3D space, offering a more efficient way to fill in the gaps in the scene representation.

## 3. Proposed Approach

### 3.1. Preliminary: Sparse Representation

Given a monocular image or a set of images $\mathbf{I} = \{I_i\}_{i=1}^{N_{\text{view}}}$ captured by surrounding cameras, the goal of vision-based 3D semantic occupancy prediction is to predict a semantic label for every voxel in a predefined 3D volume $\{0, 1, 2, ..., C\}^{H \times W \times D}$, where $H, W, D$ denotes the spatial resolution, the class label 0 indicates empty voxel, $C$ is the number of semantic classes for non-empty voxels.

In the initial stage of our model, we follow the Lift-Splat-Shoot (LSS) [25] framework. The images are first passed through an image encoder, such as ResNet [9] augmented with FPN [20], This encoder generates latent features on the 2D perspective plane. Subsequently, the 2D latent features are lifted to the 3D space using predicted depth maps, resulting in dense cubic features denoted as $\mathbf{V} \in \mathbb{R}^{H \times W \times D \times C}$. Notably, we observe that approximately 80% of the voxels in $\mathbf{V}$ are empty. This sparsity arises due to the nature of LSS, where 2D features are cast through ray casting and become sparser in distant regions.

We then convert the dense feature $\mathbf{V}$ to a sparse representation by gathering non-empty voxels. The sparse tensor is stored in a commonly used coordinate (COO) format:

$$\mathbb{V} = \{(\mathbf{p}_i = [x_i, y_i, z_i] \in \mathbb{R}^3, \mathbf{f}_i \in \mathbb{R}^C) | i = 1, 2, ...N\}.$$

In the above equation, $N$ represents the number of non-empty voxels, while $\mathbf{p}_i$ and $\mathbf{f}_i$ denote the coordinates and features of the $i$-th voxel, respectively. All subsequent operations are performed on this sparse representation, eliminating redundant computations on empty voxels. The following sections elaborate on the proposed sparse building blocks, namely the sparse latent diffuser, feature pyramid, and transformer head. An overview is shown in Fig. 2.

### 3.2. Sparse Latent Diffuser

The sparse representation $\mathbb{V}$ is derived through a ray casting manner, resulting in a predominantly sparse depiction

limited to the initial intersection face between a ray and an object. Consequently, the majority of observations are inherently incomplete. Contrarily, the objective of an occupancy network is to predict complete occupancy rather than solely the visible parts. Traditional approaches address this by incorporating 3D dense convolutions (such as 3D ResNet and 3D FPN) or attention layers (such as deformable self-attention) to diffuse non-empty features to adjacent empty regions, thereby completing the scene. In this work, we aim to design a sparse variant of the latent diffuser. However, a notable challenge arises as the objective of the diffuser appears to conflict with the sparse design: By stacking more completion blocks, the scene is better completed, but the spatial sparsity also decreases, hindering efficiency. To strike a balance between scene completion and sparsity, we build our sparse latent diffuser with two key components: A sparse completion block, which executes only *necessary* latent diffusion; and a contextual aggregation block, which aggregates valid features *without* engaging in completion.

**Sparse Completion Block.** We opt for the 3D sparse convolution implemented by [7] to build the sparse completion block. A sparse convolution performs the computation in a local window in which at least one non-empty voxel resides, allowing the diffusion of features from non-empty voxels to their neighbors. The range of diffusion can be expanded by stacking multiple layers of sparse convolution. To maintain the spatial sparsity, we only use one 3D convolution layer in a sparse completion block.

**Contextual Aggregation Block.** After completion, we introduce the contextual aggregation block to effectively utilize geometry and semantic features from the local context. For constructing this block, we choose sparse submanifold convolution [7] over regular sparse convolution. Submanifold convolution ensures that an output location is active only if the corresponding input location is active, thereby maintaining sparsity even when stacking multiple layers.

**Kernel Decomposition.** Foreground objects and background elements in driving scenes often exhibit specific shape distributions. For instance, roadways and sidewalks typically have a thin, flat shape located at the bottom of the 3D volume, making them amenable to completion through convolutions in the horizontal direction. Conversely, structures like buildings or car-like objects have a rectangular shape, necessitating feature diffusion in the vertical direction. To fully leverage these distinct shape distributions, we decompose a conventional $k \times k \times k$ kernel into orthogonal kernels [46]. Specifically, for the sparse completion block, we replace the sparse convolution with three consecutive layers with $k \times k \times 1$, $k \times 1 \times k$, and $1 \times k \times k$ kernels, respectively. For the contextual aggregation block, we follow Cylinder3D [46] and replace a $k \times k \times k$ submanifold convolution with two parallel but asymmetrical branches of decomposed layers. One branch consists of two consecu-
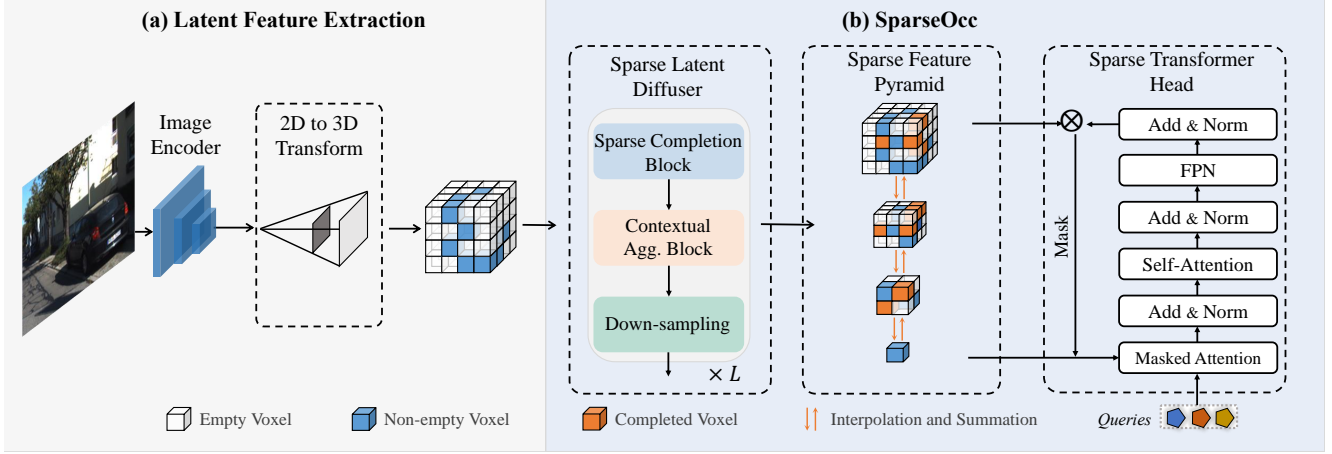
Figure 2. **Overview of the proposed approach.** (a) Images captured by monocular or surrounding cameras are first passed to a 2D encoder, yielding 2D latent features. Then the latent features are mapped to 3D using the predicted depth map following the LSS [25]. (b) SparseOcc adopts a sparse representation for the latent space. Upon this representation, we introduce three key building blocks: a latent diffuser that performs completion, a feature pyramid that enhances receptive filed, and a transformer head that predicts semantic occupancy.



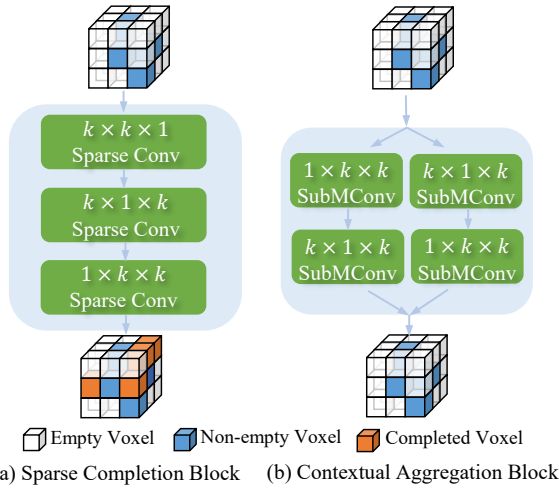(a) Sparse Completion Block    (b) Contextual Aggregation Block

Figure 3. **Two building blocks of the sparse latent diffuser.** (a) The sparse completion block diffuses non-empty features to empty neighbors, and (b) The contextual aggregation block aggregates geometry and semantic features without engaging in completion.

tive layers with $1 \times k \times k$ ad $k \times 1 \times k$ kernels, and the other branch with $k \times 1 \times k$ ad $1 \times k \times k$ kernels. Note that the complexity is reduced from $\mathcal{O}(k^3)$ to $\mathcal{O}(3k^2)$ or $\mathcal{O}(4k^2)$ after the decomposition. Though the actual cost is not reduced when we use a small kernel with $k = 3$, the expressive capacity of decomposed kernels outperforms a single full kernel, thus we can still achieve efficiency improvements by stacking fewer layers.

## 3.3. Sparse Feature Pyramid

A straightforward approach to complete the scene is to stack the proposed sparse diffuser multiple times, akin to SCP-

Net [35]. However, this necessitates a substantial number of sparse diffusers to ensure an adequately large receptive field, which is particularly important for recognizing large objects like "truck" or static elements such as "road". The computation cost is obviously expensive. To address this issue, we note that down-sample layers, implemented with sparse convolution with a stride greater than one, not only reduce the spatial resolution but also increase the relative sparsity in the new scale. By building a multi-scale sparse feature pyramid with down-sample layers, we are readily to obtain a coarse-to-fine representation of the scene. This ensures that querying any spatial location can be addressed by at least one feature scale, simultaneously reducing computation costs. Formally, we stack the sparse diffuser for $L$ times, each is followed by a down-sample layer, and the feature pyramid is collected as $\{\mathbb{V}_l\}_{l=1}^L$. Additionally, the spatial size along the height dimension of the last two scales is too small, so we simply omit the $D$ dimension. For the completion blocks, the 3D convolution is replaced with a 2D version with $3 \times 3$ kernel. For the contextual aggregation block, we replace the asymmetrical branches with two parallel 2D submanifold convolutions with $5 \times 5$ kernels.

**Sparse Voxel Decoder.** Former methods [6, 41] uses multi-scale deformable attention transformer (MSDeformAttn) [44] in the pixel/voxel decoder responsible for aggregating intra-scale and inter-scale features. Targeting for saving GPU memory and time, we simplify this process by using interpolation to fuse multi-scale sparse features output by the 3D sparse encoder. Specifically, for a given scale $\mathbb{V}_l$ from the feature pyramid, we augment it by fusing all the other scales,

$$\hat{\mathbb{V}}_l = \sum_{j \neq l} W_j \cdot \text{Interp}(\mathbb{V}_j, \mathbb{V}_l), 2 \qquad (1)$$

where $W_j$ is a learned weight for the $j$-th scale, and $\text{Interp}(\mathbb{X}, \mathbb{Y})$ indicates linear interpolation from a sparse tensor $\mathbb{X}$ to another $\mathbb{Y}$.

Leveraging this lightweight feature fusion approach, the feature pyramid is enriched with semantic information from different scales. Moreover, the high-resolution features benefit from additional completion provided by the low-resolution features, as the denser low-resolution features resist dilution by interpolation operators.

## 3.4. Sparse Transformer Head

We frame the semantic occupancy prediction as a 3D segmentation problem and employ a transformer head, inspired by the design of Mask2Former [6]. This head iteratively updates a set of learnable queries through masked attention and subsequently decodes these queries into 3D masks for all semantic classes. However, a challenge arises due to the need for dense binary masks to represent semantic classes. Additionally, as we perform predictions in a 3D space [41], the computational and memory costs associated with applying successive transformer decoder layers to dense masks become impractical.

To overcome this challenge, we adapt the dense transformer head to a sparse variant. Specifically, we assert that accurate segmentation is crucial for occupied voxels, while non-occupied voxels need not be considered during this phase. In the ensuing discussion, we outline several steps to delineate the sparse transformer decoder.

**Preprocessing and Notations.** Given the sparse feature pyramid $\hat{\mathbb{V}}_l$, we first employ a linear binary classifier for coarse segmentation. The binary classifier is trained to label a voxel as empty if the semantic ground truth is 0 and non-empty otherwise. We only preserve voxels that are classified as non-empty, resulting in a filtered, sparser tenser $\hat{\mathbb{V}}_l = \{(\mathbf{p}_i, \mathbf{f}_i)|i = 1, ..., N_l\}$, where we use $N_l$ to denote the number of remaining voxels for the $l$-th scale. Storing variant features for the empty voxels is unnecessary, and instead, we find it suffices to use only a single learnable token $\mathbf{p}_\phi$ to represent all the empty voxels.

**Query Decoding.** The former method [41] performs outer product between queries $Q \in \mathbb{R}^{N_q \times C}$ and 3D dense features $F \in \mathbb{R}^{C \times H \times W \times D}$ to decode the queries to 3D mask with shape $(N_q, H, W, D)$, which has a time complexity of $\mathcal{O}(N_q HWDC)$. To facilitate inference speed, we only perform outer product between $Q$ and $\mathbf{p}_\phi \cup \{\mathbf{f}_i|(\mathbf{p}_i, \mathbf{f}_i) \in \hat{\mathbb{V}}_l\}$ to obtain a series of occupied masks $M^{\text{occ}} \in \mathbb{R}^{N_q \times N_l}$ and a single mask $M^\phi \in \mathbb{R}^{N_q \times 1}$ that represents empty voxels. With the saved coordinates $\mathbf{p}$ of occupied voxels, we can easily reconstruct the dense 3D mask $M \in \mathbb{R}^{Q \times H \times W \times Z}$ from the predicted sparse tensor, using a scatter operation that does not break the gradient flow. This way, the complexity is reduced to $\mathcal{O}(N_l N_q C + HWZ)$ for mask predic-

tion and reconstruction. Note that

$$
\begin{aligned}
\mathcal{O}(N_l N_q C &+ HWZ) \\
&= \mathcal{O}(HWZN_q C(\frac{N_l}{HWZ} + \frac{1}{N_q C})) \\
&< \mathcal{O}(HWZN_q C),
\end{aligned}
\tag{2}
$$

because $N_l/HWZ$ is rather smaller that 1 in our sparse case. Moreover, the $N_q$ queries are also input to a $C$-way linear classifier to classify the corresponding 3D mask into predefined $C$ semantic categories.

**Query Updating.** Given the input $L$ layers of multi-scale sparse features, we iteratively alternate between query decoding and query updating in each transformer layer. With the predicted 3D masks $M_{l-1}$ in the $(l-1)$-th transformer layer, we update the queries via

$$
Q_l = \text{softmax}\left[\mathcal{M}_{l-1} + W_q Q_{l-1}(W_k \hat{\mathbf{V}}_l)^T\right] W_v \hat{\mathbf{V}}_l + Q_{l-1},
\tag{3}
$$

where $W_q, W_k, W_v$ are linear layers, $\hat{\mathbf{V}}_l$ is the dense version reconstrcuted from $\hat{\mathbb{V}}_l$, and the attention mask $\mathcal{M}_{l-1}$ at location $(x, y, z)$ is obtained by

$$
\mathcal{M}_{l-1}(x, y, z) = \begin{cases} 0 & \text{if } \sigma\left(M_{l-1}'(x, y, z)\right) \geq 0.5 \\ -\infty & \text{otherwise} \end{cases}
\tag{4}
$$

where $\sigma$ is the sigmoid function, $M_{l-1}' = \text{maxpooling}(M_{l-1})$ which resizes the 3D mask to the same resolution of $\hat{\mathbf{V}}_l$, similar to implementation in [41].

## 3.5. Objective Function

Considering the sparse transformer head formulates semantic occupancy as a mask set prediction task [6, 41], bipartite matching with Hungarian solver is used to assign binary mask labels and corresponding semantic class labels to the predicted masks. Based on the assignment, we calculate the mask loss $\mathcal{L}_{\text{mask}}$ and classification loss $\mathcal{L}_{\text{cls}}$. Additionally, $\mathcal{L}_{\text{depth}}$ is calculated between the predicted depth map and ground-truth projected by point clouds, for supervision of the LSS component. Moreover, the coarse binary classification on non-empty voxels is also supervised by a segmentation loss $\mathcal{L}_{\text{seg}}$. Finally, the overall objective function is a simple summation of these loss terms

$$
\mathcal{L} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{seg}}.
\tag{5}
$$

# 4. Experiments

## 4.1. Experimental Setup

**Datasets.** We evaluate our proposed SparseOcc on nuScenes-Occupancy [33] and SemanticKITTI [2]. **nuScenes-Occupancy** [33] extends the famous large-scale

| Method | Input | IoU | mIoU | barrier | bicycle | bus | car | const. veh. | motorcycle | pedestrian | traffic cone | trailer | truck | drive. suf. | other flat | sidewalk | terrain | manmade | vegetation | FLOPs | Memory | 3D/Overall Latency |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MonoScene [4] | C | 18.4 | 6.9 | 7.1 | 3.9 | 9.3 | 7.2 | 5.6 | 3.0 | 5.9 | 4.4 | 4.9 | 4.2 | 14.9 | 6.3 | 7.9 | 7.4 | **10.0** | 7.6 | 1132G | 20G | 0.57/0.73s |
| TPVFormer [13] | C | 15.3 | 7.8 | 9.3 | 4.1 | 11.3 | 10.1 | 5.2 | 4.3 | 5.9 | 5.3 | 6.8 | 6.5 | 13.6 | 9.0 | 8.3 | 8.0 | 9.2 | 8.2 | 1716G | 19G | 0.84/1.22s |
| OpenOccupancy [33] | C | 19.3 | 10.3 | 9.9 | 6.8 | 11.2 | 11.5 | 6.3 | 8.4 | 8.6 | 4.3 | 4.2 | 9.9 | 22.0 | 15.8 | 14.1 | 13.5 | 7.3 | 10.2 | 1916G | 19G | 0.84/1.22s |
| C-CONet [33] | C | 20.1 | 12.8 | 13.2 | 8.1 | **15.4** | 17.2 | 6.3 | **11.2** | 10.0 | 8.3 | 4.7 | 12.1 | 31.4 | 18.8 | 18.7 | 16.3 | 4.8 | 8.2 | 1810G | 21G | 2.18/2.58s |
| SparseOcc (ours) | C | **21.8** | **14.1** | **16.1** | **9.3** | 15.1 | **18.6** | **7.3** | 9.4 | **11.2** | **9.4** | **7.2** | **13.0** | **31.8** | **21.7** | **20.7** | **18.8** | 6.1 | **10.6** | 455G | 13G | 0.19/0.25s |

Table 1. **Semantic occpancy prediction results on nuScenes-Occupancy [33] validation set.** For accuracy evaluation, We report the geometric metric IoU, semantic metric mIoU, and the IoU for each semantic class. For efficiency evaluation, we report the FLOPs, training GPU memory, and 3D/overall inference latency. The C denotes camera and the **bold** numbers indicate the best results.

| Method | Input | IoU | mIoU | road (15.30%) | sidewalk (11.13%) | parking (1.12%) | other-ground (0.56%) | building (14.1%) | car (3.92%) | truck (0.16%) | bicycle (0.03%) | motorcycle (0.03%) | other-vehicle (0.20%) | vegetation (39.3%) | trunk (0.51%) | terrain (9.17%) | person (0.07%) | bicyclist (0.07%) | motorcyclist (0.05%) | fence (3.90%) | pole (0.29%) | traffic-sign (0.08%) | FLOPs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LMSCNet* [27] | C | 28.61 | 6.70 | 40.68 | 18.22 | 4.38 | 0.00 | 10.31 | 18.33 | 0.00 | 0.00 | 0.00 | 0.00 | 13.66 | 0.02 | 20.54 | 0.00 | 0.00 | 0.00 | 1.21 | 0.00 | 0.00 | - |
| 3DSketch* [5] | C | 33.30 | 7.50 | 41.32 | 21.63 | 0.00 | 0.00 | 14.81 | 18.59 | 0.00 | 0.00 | 0.00 | 0.00 | 19.09 | 0.00 | 26.40 | 0.00 | 0.00 | 0.00 | 0.73 | 0.00 | 0.00 | - |
| AICNet* [16] | C | 29.59 | 8.31 | 43.55 | 20.55 | 11.97 | 0.07 | 12.94 | 14.71 | 4.53 | 0.00 | 0.00 | 0.00 | 15.37 | 2.90 | 28.71 | 0.00 | 0.00 | 0.00 | 2.52 | 0.06 | 0.00 | - |
| JS3C-Net* [37] | C | **38.98** | 10.31 | 50.49 | 23.74 | 11.94 | 0.07 | 15.03 | 24.65 | 4.41 | 0.00 | 0.00 | 6.15 | 18.11 | **4.33** | 26.86 | 0.67 | 0.27 | 0.00 | 3.94 | 3.77 | 1.45 | - |
| MonoScene† [4] | C | 36.86 | 11.08 | 56.52 | 26.72 | 14.27 | 0.46 | 14.09 | 23.26 | 6.98 | 0.61 | 0.45 | 1.48 | 17.89 | 2.81 | 29.64 | 1.86 | 1.20 | 0.00 | 5.84 | 4.14 | 2.25 | - |
| TPVFormer [13] | C | 35.61 | 11.36 | 56.50 | 25.87 | **20.60** | **0.85** | 13.88 | 23.81 | 8.08 | 0.36 | 0.05 | 4.35 | 16.92 | 2.26 | 30.38 | 0.51 | 0.89 | 0.00 | 5.94 | 3.14 | 1.52 | 946G |
| OccFormer [41] | C | 36.50 | **13.46** | 58.85 | 26.88 | 19.61 | 0.31 | 14.40 | **25.09** | **25.53** | **0.81** | **1.19** | 8.52 | **19.63** | 3.93 | **32.62** | 2.78 | **2.82** | 0.00 | 5.61 | **4.26** | **2.86** | 889G |
| SparseOcc | C | 36.48 | 13.12 | **59.59** | **29.68** | 20.44 | 0.47 | **15.41** | 24.03 | 18.07 | 0.78 | 0.89 | **8.94** | 18.89 | 3.46 | 31.06 | **3.68** | 0.62 | 0.00 | **6.73** | 3.89 | 2.60 | 393G |

Table 2. **Semantic scene completion results on SemanticKITTI [1] validation set.** For accuracy evaluation, We report the geometric metric IoU, semantic metric mIoU, and the IoU for each semantic class. For efficiency evaluation, we report the FLOPs. The C denotes camera and the **bold** numbers indicate the best results. The methods with "*" are RGB-input variants reported by [4] for fair comparison.

autonomous driving datasets nuScenes [3] with 3D dense semantic occupancy annotations on key frames for 1 "empty" class and 16 semantic classes using Augmenting And Purifying pipeline. It covers 700 and 150 driving scenes in the training and validation set of nuScenes. The multi-view images and corresponding LiDAR points are provided by the original nuScenes dataset. **SemanticKITTI** [2] contains 22 sequences including monocular images, LiDAR points, point cloud segmentation labels and semantic scene completion annotations. The sequence 08 is officially split for validation, sequences 00-10 (excluding 08) are used for training, and sequences 11-21 are the test set. It annotates each 3D voxel with 1 "empty" class or 19 semantic classes.

**Evaluation Metric.** We follow the common practice and report the intersection over union (IoU) of occupied voxels regardless of their semantic labels to evaluate the reconstructed geometry shape. The semantic mIoU of all semantic classes is also reported to evaluate the semantic-aware perception ability. For efficiency analysis, FLOPs is also evaluated using the analysis tool provided by [22].

**Implementation Details.** For image encoder, we follow the previous work [33, 41] for fair comparison and use EfficientNetB7 [29] and SecondFPN [39] on SemanticKITTI and ResNet-50 [9] on nuScenes-Occupancy. The 2D to 3D view transformation is implemented as the same as [33, 41].

And it generates a 3D feature volume of size $128 \times 128 \times 16$ and $128 \times 128 \times 10$ with 128 channels for SemanticKITTI and nuScenes-Occupancy, which is converted to sparse representation. We stack the sparse diffuser for $L = 4$ times, each followed by a sparse convolution with stride 2 for downsampling. The kernel size $k$ of the orthogonal convolutions in sparse completion block and contextual aggregation block is set to 3. The sparse voxel decoder projects the multi-scale features to 192 channels and enhances them via interpolation and summation. The sparse transformer head consists of 9 layers for query updating and decoding and the number of queries $N_q$ is 100. Following [6, 41], we sample 50176 points for supervision according to the class frequency for fast training speed. During inference, the predicted masks are upsampled $2\times$ and $4\times$ to the size of ground-truth via trilinear interpolation for evaluation on SemanticKITTI and nuScenes-Occupancy, respectively.

**Optimization.** AdamW [23] optimizer with an initial learning rate of 1e-4 and a weight decay of 0.01 is leveraged during training. We train the model for 30 epochs on SemanticKITTI and 24 epochs on nuScenes-Occupancy with a batch size of 8 on 8 v100 GPUs.

## 4.2. Benchmark Performance

In this part, we compare our SparseOcc with the published state-of-the-art methods on nuScenes-Occupancy and Se-

manticKITTI. The results of the former methods are directly derived from their papers or code with their configurations. **NuScenes-Occupancy.** As shown in Tab. 1, our SparseOcc successfully outperforms the 3D dense representation based C-CONet [33] by 1.7% geometry IoU and 1.3% semantic mIoU. This accomplishment highlights SparseOcc's effectiveness. Besides, we can observe that the model FLOPs are reduced by 74.9% relatively, which further justifies the efficiency of our SparseOcc. More surprisingly, SparseOcc not only produces a remarkable 80.8% improvement in mIoU over TPVFormer [13] which uses efficient TPV view as 3D scene representation, but also reduces the FLOPs by 59.8%. We find that TPVFormer uses a relatively large image backbone ResNet-101 [9], and the multi-layer image cross-attention as well as cross-view attention between dense TPV maps also impose lots of computational burden. In contrast, our SparseOcc uses sparse operators in the 3D space, helping us achieve superior performance with a lightweight implementation. Besides, SparseOcc exhibits a noteworthy reduction in 3D inference latency, i.e., the time for 3D feature processing.

**SemanticKITTI.** We quantitatively compare the proposed SparseOcc with several previous works in Tab. 2. we can see that SparseOcc achieves comparable performance with OccFormer [41] and significantly better than other methods in terms of semantic mIoU. Please note that different from OccFormer which builds long-range and dynamic dependency in the 3D space using Swin Transformer [21], the 3D encoder of SparseOcc is built with spconv [7] only. Consequently, the model FLOPs of SparseOcc is just 44.2% of OccFormer [41] approximately.

### 4.3. Qualitative Evaluation

We visualize the 3D semantic occupancy prediction results of several challenging scenes in nuScenes-Occupancy validation set in Fig. 4. As can be seen, compared with the 3D dense counterpart C-CONet [33], our SparseOcc can better complete the large area flat-like road (first row), capture the complicated structure of vegetation (second row) and reconstruct car in the distance (third row). We can conclude that benefiting from the proposed sparse latent diffuser and learned sparse feature pyramid, our sparse occupancy transformer head can generate relatively accurate scene-level descriptions in an efficient way.

### 4.4. Ablation Studies

**Sparse Completion Blcok.** As illustrated in Sec. 3.2, a 3D diffusion kernel is spatially decomposed into a combination of three orthogonal convolutional kernels in the sparse completion block. Tab. 3 ablates the type and number of diffusion kernels. Surprisingly, we find that SparseOcc can perform satisfactorily even if no sparse completion block is applied. We think that it is because the decoder which

| Conv Type | Conv Num | Kernel Size | IoU | mIoU |
|---|---|---|---|---|
| - | 0 | - | 35.5 | 12.1 |
| Regular | 1 | 3×3×3 | 35.8 | 12.2 |
| | 2 | | 36.4 | 12.3 |
| | 3 | | 36.2 | 12.6 |
| Decomposed | 1 | 3×3×1 | 36.5 | **13.1** |
| | 2 | 3×1×3 | **36.6** | 12.8 |
| | 3 | 1×3×1 | 36.4 | 12.7 |

Table 3. Ablation on different designs of Sparse Completion Block on SemanticKITTI val set.

| Type | IoU | mIoU | Memory | FLOPs |
|---|---|---|---|---|
| FPN3D[20] | 34.4 | 9.8 | 13.2G | 307G |
| MSDeformAttn3D | **36.7** | **13.3** | 19.8G | 379G |
| Sparse Decoder | 36.5 | 13.1 | 13.3G | 279G |

Table 4. Ablation on voxel decoder on SemanticKITTI val set.

| Type | IoU | mIoU | Memory | FLOPs |
|---|---|---|---|---|
| Linear Head | **36.8** | 11.8 | 9.8G | 5.4G |
| Trans. Head | 36.2 | **13.2** | 19.9G | 19.0G |
| Sparse Trans. Head | 36.5 | 13.1 | 13.3G | 13.5G |

Table 5. Ablation on segmentation head on SemanticKITTI val set.

fuses sparse feature pyramid can also complete the scene to some extent. Compared with regular 3×3×3 kernel, the decomposed orthogonal achieves better scene completion and segmentation results. Besides, stacking more convolution blocks does not improve performance. Hence, we build our sparse completion block with only a group of decomposed orthogonal.

**Sparse Feature Pyramid.** As shown in Tab. 4, we observe that when the SparseOcc equipped with 6 layers of multi-scale deformable attention (MSDeformAttn), it performs better than the proposed sparse decoder. However, when the layer number of MSDeformAttn decreases, the IoU and mIoU both drop and lag behind the proposed simple sparse decoder. Moreover, the training GPU memory and FLOPs of the MSDeformAttn3D are much higher than the proposed interpolation and summation method. Compared with FPN3D [20], our sparse voxel decoder is 2.1% IoU and 3.3% mIoU higher with slightly more memory.

**Sparse Transformer Head.** We compare several different prediction heads, including simple linear head, transformer head proposed in [41], and our sparse transformer head in Tab. 5. As can be seen, the linear head achieves the best geometry IoU of 36.8. We postulate it may be because the output of the linear head is supervised by an explicit geometry loss, i.e., a cross-entropy loss on occupied and non-occupied voxels. On the contrary, the transformer decoder formulates occupancy prediction as mask generation for each semantic class exclusively. Consequently, no explicit supervision is or can be imposed on the occupied voxels. SparseOcc uses a linear layer for coarse segmentation to filter out the non-occupied voxels and then proposes

**Multi-View Images**      **C-CONet**      **SparseOcc**      **Ground-Truth**
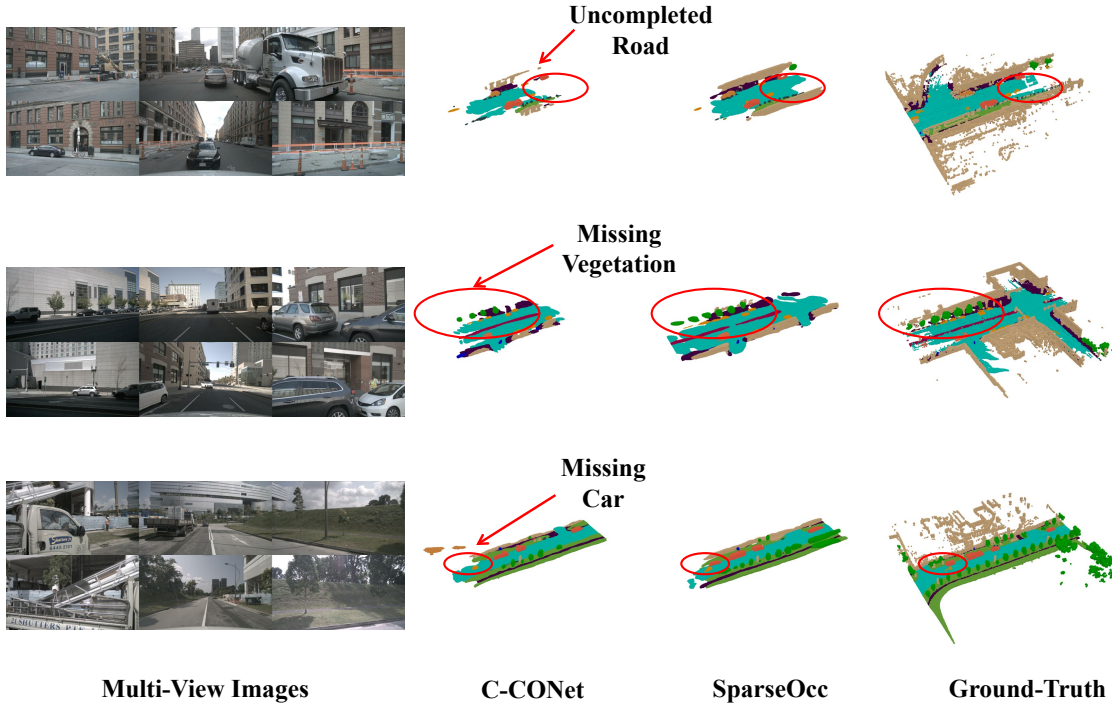
Figure 4. **Qualitative results of 3D semantic occupancy on nuScenes-Occupancy validation set.** The input multi-view images are shown on the leftmost and the occupancy predictions of C-CONet [33], our SparseOcc, and the ground-truth are then visualized sequentially. Compared to 3D dense representation based C-CONet [38], our SparseOcc achieves better completion and segmentation as highlighted by the red circles.

| Method | 2D Backbone | Input Size | IoU | mIoU |
|---|---|---|---|---|
| | R-50 | $704 \times 256$ | 16.6 | 8.6 |
| C-CONet [33] | R-50 | $1600 \times 900$ | 19.3 | 10.3 |
| | R-101 | $1600 \times 900$ | 20.2 | 11.4 |
| SparseOcc | R-50 | $704 \times 256$ | **21.8** | 14.1 |
| | R-50 | $1600 \times 900$ | 20.4 | **14.6** |

Table 6. Ablation on input size and 2D backbone on nuScenes-Occupancy.

a sparse transformer head for mask prediction. Therefore, it makes an appropriate trade-off between these two kinds of occupancy heads and achieves satisfactory performance with low training memory.

**Input Image Size.** A larger input image size has a larger resolution of image features, thus having a larger feature density in the 3D space, which further influences the performance. From Tab. 6, we can observe that SparseOcc still performs better than C-CONet, even though it is trained using a smaller input size ($704 \times 256$) and image backbone (R-50), which further demonstrates the effectiveness of our SparseOcc. Additionally, using a larger input image size ($1600 \times 900$) can improve mIoU for both C-CONet and the proposed SparseOcc, thanks to the density increase of semantic features. However, the IoU of SparseOcc drops when using higher image resolution. We blame it on the wrong hallucination on spatially empty voxels caused by over-dense 3D sparse features. When inactivating the sparse

completion block in the last sparse latent diffuser layer, this hallucination can be relieved.

## 5. Conclusion

In this paper, we explore the possibility of using pure sparse representation for 3D scene description and present SpaseOcc for 3D semantic occupancy prediction. Specifically, a sparse latent diffuser with decomposed orthogonal kernels is proposed to propagate the non-empty features to their adjacent empty area. To efficiently complete the scene, we stack layers of diffusers and downsampling layers to generate a coarse-to-fine sparse feature pyramid, which is further fused via simple interpolation and summation. In this way, the feature pyramid is enhanced by multi-scale semantic information. Finally, we use a sparse transformer head to query multi-scale sparse features and generate semantic occupancy predictions. Without bells and whistles, SparseOcc achieves state-of-the-art results on nuScenes-Occupancy and comparable performance on SemanticKITTI, fully demonstrating its superior effectiveness and efficiency.

# References

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. 6

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. 5, 6

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. Nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 6

[4] Anh-Quan Cao and Raoul de Charette. Monoscene: Monocular 3d semantic scene completion. In *CVPR*, 2022. 2, 6

[5] Xiaokang Chen, Kwan-Yee Lin, Chen Qian, Gang Zeng, and Hongsheng Li. 3d sketch-aware semantic scene completion via semi-supervised structure prior. In *CVPR*, 2020. 6

[6] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 4, 5, 6

[7] Spconv Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022. 2, 3, 7

[8] Chongjian Ge, Junsong Chen, Enze Xie, Zhongdao Wang, Lanqing Hong, Huchuan Lu, Zhenguo Li, and Ping Luo. Metabev: Solving sensor failures for 3d detection and map segmentation. In *ICCV*, 2023. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 6, 7

[10] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. Fiery: future instance prediction in bird's-eye view from surround monocular cameras. In *ICCV*, 2021. 2

[11] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *CVPR*, 2023. 1

[12] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv:2112.11790*, 2021. 2

[13] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. *arXiv:2302.07817*, 2023. 1, 2, 6, 7

[14] Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *CVPR*, 2023. 1

[15] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1

[16] Jie Li, Kai Han, Peng Wang, Yu Liu, and Xia Yuan. Anisotropic convolutional networks for 3d semantic scene completion. In *CVPR*, 2020. 6

[17] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. *arXiv:2206.10092*, 2022. 1, 2

[18] Yiming Li, Zhiding Yu, Christopher Choy, Chaowei Xiao, Jose M Alvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion. In *CVPR*, 2023. 3

[19] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv:2203.17270*, 2022. 2

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 3, 7

[21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 7

[22] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv:2205.13542*, 2022. 2, 6

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[24] Lang Peng, Zhirong Chen, Zhangjie Fu, Pengpeng Liang, and Erkang Cheng. Bevsegformer: Bird's eye view semantic segmentation from arbitrary camera rigs. *arXiv:2203.04050*, 2022. 2

[25] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*, 2020. 1, 3, 4

[26] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *CVPR*, 2020. 2

[27] Luis Roldao, Raoul de Charette, and Anne Verroust-Blondet. Lmscnet: Lightweight multiscale 3d semantic completion. In *3DV*, 2020. 6

[28] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 2

[29] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 6

[30] Pin Tang, Hai-Ming Xu, and Chao Ma. Prototransfer: Cross-modal prototype transfer for point cloud segmentation. In *ICCV*, 2023. 1

[31] Xiaoyu Tian, Tao Jiang, Longfei Yun, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving. *arXiv:2304.14365*, 2023. 1

[32] Wenwen Tong, Chonghao Sima, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, et al. Scene as occupancy. In *ICCV*, 2023.

[33] Xiaofeng Wang, Zheng Zhu, Wenbo Xu, Yunpeng Zhang, Yi Wei, Xu Chi, Yun Ye, Dalong Du, Jiwen Lu, and Xingang Wang. Openoccupancy: A large scale benchmark for surrounding semantic occupancy perception. In *ICCV*, 2023. 1, 2, 5, 6, 7, 8

[34] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *CVPR*, 2023. 1, 2

[35] Zhaoyang Xia, Youquan Liu, Xin Li, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, and Yu Qiao. Scpnet: Semantic scene completion on point cloud. In *CVPR*, 2023. 3, 4

[36] Shenghua Xu, Xinyue Cai, Bin Zhao, Li Zhang, Hang Xu, Yanwei Fu, and Xiangyang Xue. Rclane: Relay chain prediction for lane detection. In *ECCV*, 2022. 1

[37] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *AAAI*, 2021. 6

[38] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *ECCV*, 2022. 8

[39] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 6

[40] Weijia Zhang, Dongnan Liu, Chao Ma, and Weidong Cai. Alleviating foreground sparsity for semi-supervised monocular 3d object detection. In *WACV*, 2024. 1

[41] Yunpeng Zhang, Zheng Zhu, and Dalong Du. Occformer: Dual-path transformer for vision-based 3d semantic occupancy prediction. *arXiv:2304.05316*, 2023. 2, 4, 5, 6, 7

[42] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *CVPR*, 2022. 2

[43] Shengchao Zhou, Weizhou Liu, Chen Hu, Shuchang Zhou, and Chao Ma. Unidistill: A universal cross-modality knowledge distillation framework for 3d object detection in bird's-eye view. In *CVPR*, 2023. 1

[44] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020. 4

[45] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2

[46] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *CVPR*, 2021. 3

[47] Sicheng Zuo, Wenzhao Zheng, Yuanhui Huang, Jie Zhou, and Jiwen Lu. Pointocc: Cylindrical tri-perspective view for point-based 3d semantic occupancy prediction. *arXiv:2308.16896*, 2023. 2