VisionPAD: A Vision-Centric Pre-training Paradigm for Autonomous Driving
Supplementary Materials

1. Additional Implementation Details

In this section, we elaborate on the implementation details of our proposed approach.

1.1. Evaluation Metrics

3D Objection Detection. In accordance with the evaluation protocols established in previous works [1, 4, 7], we utilize the NuScenes Detection Score (NDS) and mean Average Precision (mAP) as the primary metrics for assessing 3D object detection performance. The mAP is computed using the center distance on the ground plane rather than the 3D Intersection over Union (IoU), to match the predicted results and ground truth. Furthermore, the nuScenes dataset incorporates five types of true positive metrics (TP metrics), including ATE, ASE, AOE, AVE, and AAE for measuring translation, scale, orientation, velocity, and attribute errors, respectively. These TP metrics are defined for each class, and their means are computed across classes to yield mATE, mASE, mAOE, mAVE, and mAAE. Based on these TP errors, we define the TP score as $\mathrm{TP}_{\mathrm{score}} = \max(1 - \mathrm{TP}_{\mathrm{error}}, 0)$. Subsequently, the nuScenes detection score (NDS) is computed as:

$$\mathrm{NDS} = \frac{1}{10}\left[5\mathrm{mAP} + \sum \mathrm{TP}_{\mathrm{score}}\right], \tag{1}$$

to encapsulate all aspects of the nuScenes detection comprehensively.

Semantic Occupancy Prediction. In the context of 3D semantic occupancy prediction, we adhere to the evaluation methodology employed by existing methods [2, 5, 6], utilizing mean intersection-over-union (mIoU) across all classes and intersection-over-union (IoU) as the evaluation metrics:

$$\mathrm{mIoU} = \frac{1}{|C'|}\sum_{i \in C'} \frac{TP_i}{TP_i + FP_i + FN_i}, \tag{2}$$

where $C' = 17$, c_0, TP, FP, FN denote the number of semantic classes in the Occ3D-nuScenes dataset [6], the empty class, the number of true positive, false positive and false negative predictions, respectively.

Algorithm 1 The pseudocode of voxel velocity estimation.

Input: $\mathcal{V}_t$, M
Output: $\mathcal{V}_{t+n}$
 # predict the absolute flow for each voxel
$\mathcal{F}_t \leftarrow$ **flow_decoder**$(\mathcal{V}_t)$
 # transform the flow into absolute grid displacement to the future
$\hat{\mathcal{F}}_{t+n} \leftarrow \Delta t \cdot n \cdot \mathcal{F}_f$
 # transform the absolute displacement to the future ego coordinate by using the current to future transformation
$\tilde{\mathcal{F}}_{t+n} \leftarrow M \cdot \hat{\mathcal{F}}_{t+n}$
 # warping the current frame volume feature to the future
$\mathcal{V}_{t+n} \leftarrow$ **grid_sample**$(\mathcal{V}_t,\ \tilde{\mathcal{F}}_{t+n})$
return $\mathcal{V}_{t+n}$

1.2. Details of Voxel Velocity Estimation Module

In the main paper, we have presented the effectiveness of the proposed self-supervised voxel velocity estimation module in capturing inherent temporal consistencies. Here we elaborate on more details of this module. The pseudocode for the voxel velocity estimation module is presented in Alg. 1. In this context, the inputs $\mathcal{V}_t$, M represent the extracted volume features at the current frame and the pose transformation matrix from the current frame to a future frame, respectively. The output is the predicted volume features $\mathcal{V}_{t+n}$ in the future frame at time $n+t$. The process begins by predicting the absolute flow $\mathcal{F}_t$ for each voxel relative to the current frame coordinates, utilizing an MLPs-based flow decoder. Subsequently, the voxel flow is transformed into the voxel grid displacement for future frames using the formula $\Delta t \cdot n \cdot \mathcal{F}_f$, where Δt and n are the time interval between adjacent frames, and n denotes the number of frames to be forecasted. In the nuScenes dataset, Δt is set to $\frac{1}{12}$ by default, corresponding to the synchronized camera frame rate of 12 frames per second. The next step involves transforming the voxel displacement from the current frame to the future frame, relative to the future ego coordinate, by employing the pose transformation matrix. Finally, the future volume feature is obtained by warping the current volume feature with the predicted voxel displacement.

Primitives	Variants	Value	NDS	mAP
Mean	Absolute	-	26.8	25.9
	Offset	0.25	27.3	26.5
		0.50	27.2	26.5
Scale	Fixed	0.4	26.9	25.3
	Learnable	[0.1, 0.5]	27.3	26.5
		[0.2, 0.8]	26.8	26.1
Rotation	Fixed	[1, 0, 0, 0]	27.2	26.7
	Learnable	-	27.3	26.5

Table 1. Ablation studies on various learning objectives for predicting Gaussian primitives.

2. Additional Experimental Results

Due to the space constraint of the main paper, we provide more experimental results here to facilitate a more comprehensive understanding of our proposed method.

Effects of Gaussian Primitives. In VisionPAD, we present autonomous driving scenes as a set of Gaussian primitives $\{g_k = (\mu_k, \Sigma_k, \alpha_k, c_k)\}_{k=1}^{K}$, where each Gaussian g_k is parameterized by its 3D position or mean $\mu_k \in \mathbb{R}^3$, a covariance Σ_k, an opacity $\alpha_k \in [0, 1]$, and spherical harmonics (SH) coefficients $c_k \in \mathbb{R}^k$. The covariance Σ_k is further formulated using a scaling matrix and rotation matrix to ensure its positive semi-definite. Therefore, it's worth exploring various potential learning objectives for predicting these Gaussian primitives. To mitigate computational demands during training, this ablation study is confined to the 3D object detection task, employing 50% of the training data during the pre-training phase and 25% during the fine-tuning stage. Results are reported on the entire nuScenes validation set. The model used in this ablation study is consistent with the one employed in the main paper.

The results are presented in Tab. 1. We designed various experimental variants to explore the effects of different learning objectives. For each variant, we maintained the predictions of other primitives as consistent with those used in the main paper. We initially exploit the manners for the prediction of Gaussian mean parameters. By default, we transformed the Gaussian mean prediction problem into an offset prediction task using voxel centers as anchors. We also introduced an offset scaling factor to constrain the range of offset values. Consequently, a Gaussian mean can be obtained by:

$$\mu_k = \mathbf{x}_v + (\text{sigmoid}(\mathcal{O}_k) - 0.5) \cdot s, \tag{3}$$

where $\mathbf{x}_v$ is the coordinate of an anchor point, $\mathcal{O}_k \in \mathbb{R}^{1\times 3}$ is the learnable offset, and s is the scaling factor. We empirically set the s to 0.25, as the same in our main paper, achieving a performance of 27.3 NDS and 26.5 mAP. When the scaling factor was increased to 0.5, no obvious performance changes were observed. Furthermore, instead of predicting position offset relative to the centers of volume anchors, we directly regressed the absolute μ_k via MLPs (second row in Tab. 1). This resulted in a performance decline in both NDS and mAP, from 27.3 NDS and 26.5 mAP to 26.8 NDS and 25.9 mAP. This indicates that it's beneficial to predict Gaussian means by predicting the offsets relative to the anchor centers.

Methods	Decoder	Memory	Latency
UniPAD-C	NeRF	1973MB	900ms
VisionPAD	3D-GS	134MB	70ms

Table 2. The speed analysis of our method with the UniPAD.

In relation to the Gaussian scale and rotation, our primary focus is on exploring the effects of their learnability. By default, the scale and rotation parameters are set to be learnable, with the scale constrained to the range of $[0.1, 0.5]$. The final scale of Gaussian k is calculated as:

$$S_k = s_l + (s_u - s_l) \cdot \text{sigmoid}(\mathcal{S}_k), \tag{4}$$

where s_l and s_u represent the lower and upper bounds of the scale, respectively, and $\mathcal{S}_k$ is the learnable scale parameter. We also experimented with a larger range of $[0.2, 0.8]$, which resulted in a slight decrease in performance. When the scale and rotation is fixed by setting them as a constant of 0.4 (*i.e.* half of the voxel size) and $[1, 0, 0, 0]$, *i.e.* the Gaussian degenerates into a spherical form. Fixing these parameters leads to varying degrees of performance decline, particularly notable when the scale is constant, resulting in a decrease in mAP from 26.5 to 25.3. This indicates that a fixed scale is inadequate for effectively modeling the scene.

Speed Analysis. The NeRF-based methods often suffer from substantial GPU memory consumption and slow rendering speeds. Furthermore, the resource consumption in NeRF-based methods notably increases with larger rendering sizes. In contrast, our 3D-GS-based approach maintains faster rendering speeds irrespective of rendering size. As presented in Tab. 2, we compare the efficiency and memory consumption between UniPAD-C with ours when rendering the same 360×640 images. We conduct this experiment in a single NVIDIA A100 GPU. Our approach distinctly results in an approximate 93.2% reduction in memory usage, alongside an approximate 92.2% decrease in rendering latency. This validates the potential of the 3D-GS-based pre-training paradigm.

UniPAD with V.V. Est. and P.C. To further validate the effectiveness of the proposed photometric consistency (P.C.) and the self-supervised voxel velocity estimation (V.V. Est.), we apply these two strategies in the UniPAD. The results are shown in Tab. 3, our proposed modules achieve

significant improvements on UniPAD (+3.3%/+5.0%). It is worth noting that 3D-GS renders the **entire image** with lower computational cost. Compared to **ray-sampling** approach in UniPAD, 3D-GS increases the effective pixels for photometric consistency supervision, thereby achieving better results.

Model	UniPAD	w/ V.V. Est. and P.C.	VisionPAD
NDS/mAP	22.3/18.3	25.6/23.3	27.3/26.5

Table 3. The effectiveness and generalizability of proposed proposed photometric consistency (P.C.) and the self-supervised voxel velocity estimation (V.V. Est.) strategies for pre-training.

Computation Cost of Gaussian Filtering. As illustrated in Tab. 4, we demonstrate that using the Gaussian filter can effectively reduce memory usage and improve speed.

Model	Memory	Latency	NDS
w/o filter	186MB	90ms	26.8
VisionPAD	134MB	70ms	27.3

Table 4. The computation cost analysis of Gaussian filtering operation.

3. Additional Visualization Results

As illustrated in Fig. 1, we visualize an additional scene to qualitatively assess the efficacy of our method. To facilitate an intuitive comparison of the perception results, we simultaneously demonstrate the 3D object detection and 3D semantic occupancy prediction within the same scene. The 3D object detection results are predicted by the UVTR [3], while the 3D semantic occupancy prediction results are obtained by the BEVDet [1]. Both of them are pre-trained by our method.

From the results, we can not only discern the desirable performance of both 3D object detection and occupancy prediction but also highlight the respective strengths and limitations inherent in these two representations. The dense occupancy representation, is capable of depicting arbitrary irregular objects and background obstacles in a detailed voxel format, but tends to demand higher computational resources. Conversely, 3D bounding boxes primarily represent foreground objects in a sparse format. In this scene, despite the accurate detections of most objects, some false positive bounding boxes are observed, and the orientation is also not accurate enough to describe the excavator at the front right. In comparison, the occupancy predictions offer a more satisfactory representation of these critical objects. Encouragingly, as the visualization results indicate, our pre-training paradigm enhances performance across both tasks. It further underscores the efficacy and adaptability of our proposed method.

References

[1] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *CoRR*, abs/2112.11790, 2021. 1, 3

[2] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9223–9232, 2023. 1

[3] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. In *NeurIPS*, 2022. 3

[4] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022. 1

[5] Zhiqi Li, Zhiding Yu, David Austin, Mingsheng Fang, Shiyi Lan, Jan Kautz, and Jose M Alvarez. Fb-occ: 3d occupancy prediction based on forward-backward view transformation. *CVPRW*, 2023. 1

[6] Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[7] Honghui Yang, Sha Zhang, Di Huang, Xiaoyang Wu, Haoyi Zhu, Tong He, Shixiang Tang, Hengshuang Zhao, Qibo Qiu, Binbin Lin, et al. Unipad: A universal pre-training paradigm for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15238–15250, 2024. 1

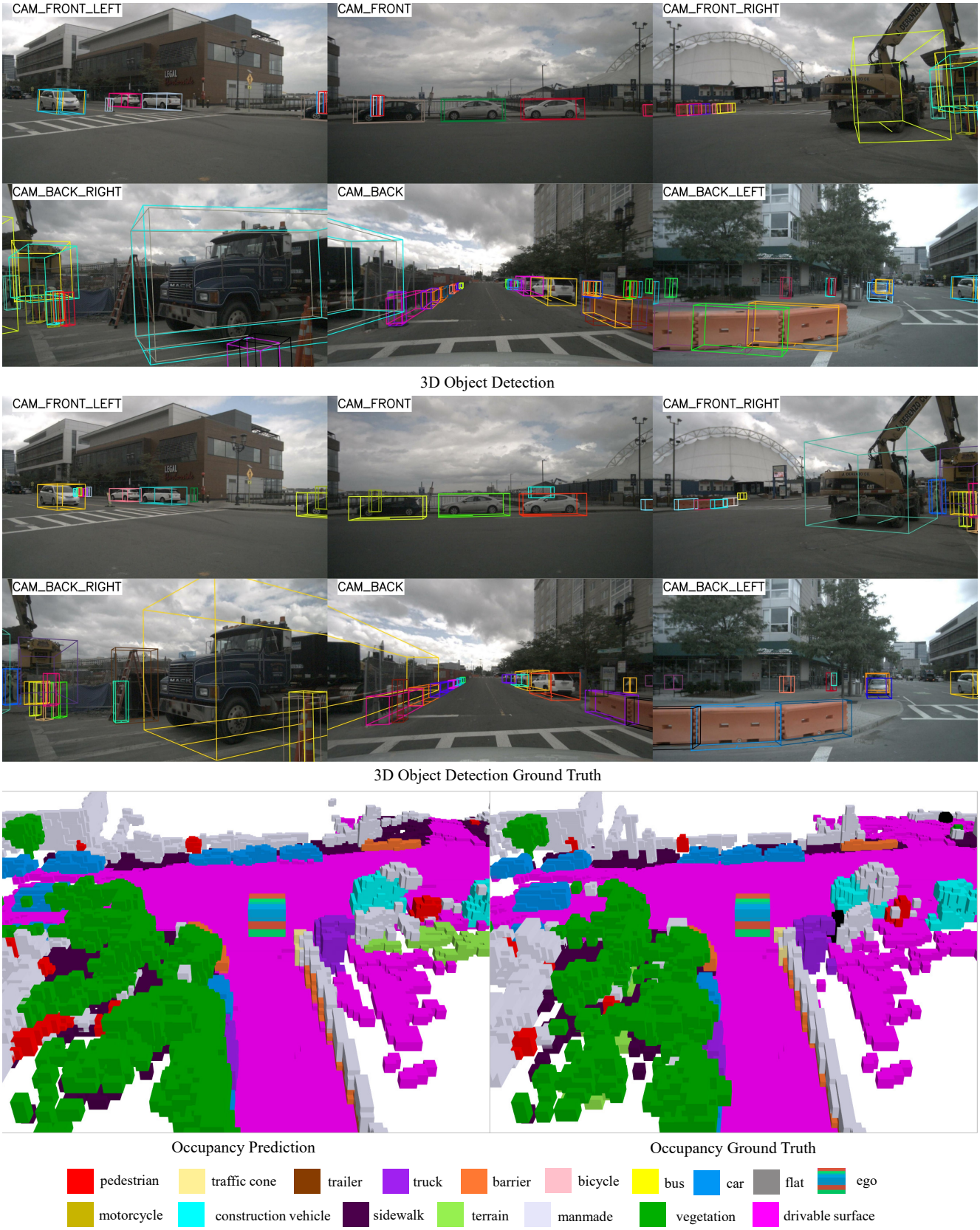

Figure 1. Visualization of 3D object detection and 3D semantic occupancy prediction results. Each detected object instance is depicted by a unique colored 3D bounding box. The legend at the bottom delineates the semantic classes of occupancy.