

A APPENDIX

In this appendix, we provide extended experiments, qualitative results, and implementation details of MAPLE. Section B reports a challenging zero-shot experiment on the *bicycle* class, designed to stress-test the framework’s ability to recover recognition ability for categories entirely absent from training. Section C presents extended results on rare but safety-critical categories—such as *animal* and *police vehicle*—that are not evaluated under the standard 10-class nuScenes protocol. Section D provides additional qualitative examples illustrating how MAPLE synthesizes diverse and contextually consistent instances. Finally, Section E and Section F include implementation details of the pipeline and training configurations for the 3D perception tasks.

B ZERO-SHOPT EXPERIMENTAL RESULTS

Table 9: **Zero-shot detection results for the bicycle category (0.5 m threshold).** All bicycle instances were removed from the training set. GT-Aug, despite reusing 8,185 real bicycles 1.69M times, completely fails. In contrast, MAPLE is trained without access to a single real bicycle instance, yet achieves the first successful detections, demonstrating its ability to recover previously unseen categories purely through generative augmentation.

Method	AP	P@R=0.1	P@R=0.2	P@R=0.3	P@R=0.5	P@R=0.7
GT-Aug	0.000	0.000	0.000	0.000	0.000	0.000
Ours	8.021	0.399	0.118	0.000	0.000	0.000

We design an adversarial *zero-shot* setting to stress-test MAPLE’s ability to recover categories that are entirely absent from training. Specifically, all bicycle instances—the rarest class in nuScenes—were removed from the training set. The GT-Aug baseline, despite reusing 8,185 real bicycle objects and inserting them 1.69M times across 22,461 scenes, collapses to 0 mAP. In contrast, MAPLE, without access to a single real bicycle instance, generates 514,852 synthetic bicycles and achieves the first non-trivial recognition ability on this category (8.0 mAP, 0.4 precision at 10% recall).

While the absolute AP is understandably modest under such an adversarial condition, the key result is that MAPLE transforms a complete failure case into a detectable signal. Even a small but consistent set of true positives is practically valuable: it provides a bootstrap for downstream active learning and annotation pipelines, and demonstrates that generative augmentation can go beyond database sampling by introducing novel, diverse instances. This highlights MAPLE’s unique potential to inject recognition capability in extreme scarcity regimes where conventional augmentation entirely fails.

C EXTENDED EXPERIMENTAL RESULTS

Table 10: **Class distribution of nuScenes annotations.** Each column group shows the number of cuboids and their ratio for a given class.

Category	# Cuboids	Ratio (%)	Category	# Cuboids	Ratio (%)	Category	# Cuboids	Ratio (%)
animal	787	0.07	personal mobility	395	0.03	barrier	152,087	13.04
adult	208,240	17.86	police officer	727	0.06	debris	3,016	0.26
child	2,066	0.18	stroller	1,072	0.09	pushable	24,605	2.11
construction worker	9,161	0.79	wheelchair	503	0.04	traffic cone	97,959	8.40
bike rack	2,713	0.23	bicycle	11,859	1.02	bus.bendy	1,820	0.16
bus.rigid	14,501	1.24	car	493,322	42.30	construction	14,671	1.26
ambulance	49	0.00	police	638	0.05	motorcycle	12,617	1.08
trailer	24,860	2.13	truck	88,519	7.59			

The nuScenes dataset (Caesar et al., 2020) defines 23 object categories for 3D bounding box annotations, covering a broad spectrum of urban traffic objects. However, as shown in Table 10, the distribution of annotated cuboids is highly imbalanced. To mitigate this long-tail skew, the official benchmark consolidates 23 categories into 10 major classes for 3D detection. While this improves label balance, it also prevents explicit evaluation of rare but safety-critical categories. To address

Table 12: **Per-class detection performance on extended rare categories under multimodal training (0.5 m threshold).** MAPLE substantially improves recognition across all rare categories, including *animal* and *police vehicle*, which are excluded from the official nuScenes protocol. Whereas baselines often fail to detect such ultra-rare classes, MAPLE introduces meaningful recognition ability by injecting diverse synthetic instances, directly increasing true positives (TPs) and demonstrating scalability beyond the standard benchmark taxonomy.

Category	Method	mAP	Prec@r=0.5	Prec@r=0.7
Motorcycle	BEVFusion	0.819	0.953	0.864
	+ Ours	0.871	0.973	0.942
Bicycle	BEVFusion	0.723	0.915	0.587
	+ Ours	0.826	0.964	0.897
Constr. Veh.	BEVFusion	0.671	0.435	0.000
	+ Ours	0.815	0.815	0.000
Police	BEVFusion	0.809	0.928	0.828
	+ Ours	0.937	0.987	0.987
Animal	BEVFusion	0.416	0.296	0.135
	+ Ours	0.612	0.666	0.411

this gap, we extend the evaluation set to include two underrepresented classes—*animal* and *police vehicle*—that are excluded from the standard protocol.

Table 11: **Multimodal 3D object detection performance on the extended train/val splits.** We report class-wise AP with a minimum precision and recall threshold of 0. Our method augments **objects** to address class imbalance and supports arbitrary categories without manual annotations.

	Car	Truck	Constr. Veh.	Bus	Trailer	Barrier	Motorcycle	Bicycle	Pedestrian	Traffic Cone	Animal	Police	mAP
<i>Val Set</i>													
BEVFusion	0.890	0.602	0.380	0.724	0.450	0.759	0.739	0.560	0.883	0.806	0.0003	0.016	0.567
+ Ours	0.897	0.627	0.371	0.752	0.461	0.754	0.751	0.635	0.895	0.814	0.0014	0.071	0.586
<i>Train Set</i>													
BEVFusion	0.912	0.703	0.671	0.832	0.670	0.884	0.819	0.723	0.920	0.893	0.416	0.809	0.771
+ Ours	0.932	0.821	0.815	0.904	0.777	0.925	0.871	0.826	0.949	0.924	0.612	0.937	0.868

We further evaluate MAPLE on an extended nuScenes benchmark that explicitly includes rare but safety-critical categories—*animal* and *police vehicle*—which are excluded from the official protocol. As shown in Table 10, these categories account for only 0.07% and 0.05% of annotations, respectively, making standard validation metrics unstable. To mitigate this, we additionally report class-wise results on the training set, where such categories appear more frequently. While training-set performance may partly reflect memorization, it still reveals whether models can recognize novel categories introduced by augmentation.

As summarized in Table 12, MAPLE consistently improves recognition across all long-tail categories, including construction vehicles, bicycles, motorcycles, animals, and police vehicles. Notably, even for ultra-rare categories, MAPLE substantially increases both precision and mAP, with gains reflected directly in the number of true positives (TPs) recovered. Whereas baselines often yield zero detections under extreme scarcity, MAPLE introduces meaningful recognition ability by injecting diverse synthetic instances with varied shapes, scales, and poses. This demonstrates that our framework scales beyond the official taxonomy and promotes more robust decision boundaries even in severely low-annotation regimes.

D ADDITIONAL QUALITATIVE RESULTS

This section presents additional qualitative results generated by MAPLE across underrepresented object categories. MAPLE easily supports new categories because it can flexibly extend to arbitrary target classes without requiring additional manual annotations. Here, we showcase examples involving animals (Fig. 11) and police vehicles (Fig. 12), extending construction vehicles (Fig. 13), motorcycles (Fig. 14), and bicycles (Fig. 15). Our visualizations highlight the effectiveness of context-aware instance placement, which ensures that synthesized objects are seamlessly integrated into the scene with realistic scale and spatial alignment.

We show: (a) the inpainted RGB image, where the inserted object is synthesized at a context-aware scale and position, ensuring visual and geometric consistency with the surrounding scene. (b) a sequence of video frames synthesized for 10-sweep simulation, providing temporally coherent motion. (c) per-frame depth estimation and semantic segmentation results, used to construct pseudo-LiDAR point clouds and verify object identity and boundaries. (d) the resulting 1-sweep pseudo-LiDAR, showing the point cloud of the inserted instance from a single frame. (e) the accumulated 10-sweep pseudo-LiDAR, capturing temporally propagated objects via our motion simulation module.

Specifically, in (e), we visualize pseudo-LiDARs under static and dynamic settings. The static setting assumes the object remains fixed at its $t = 0$ location, with only ego-motion applied. The dynamic setting simulates both ego and object motion, resulting in a realistic object trajectory across frames. Note that for *motorcycle* and *bicycle* instances without riders, our motion simulation module is not applied. In these cases, we generate multi-sweep pseudo-LiDARs only under the static setting.

E IMPLEMENTATION DETAILS

E.1 PRETRAINED FOUNDATION MODELS USED IN OUR PIPELINE

Our pipeline leverages a set of pretrained 2D foundation models, each specialized for a different sub-task in the multimodal data generation process.

Models for One-Sweep Generation. We utilize several foundation models to generate high-quality multimodal data in a single-frame setting:

- **ChatGPT-4 (OpenAI, 2023):** We use the GPT-4 model via the ChatGPT API to generate natural language descriptions for synthesized objects.
- **PowerPaint (Zhuang et al., 2024):** PowerPaint V2 is used for context-aware inpainting. It is based on a UNet-like architecture with multi-resolution feature blending and supports both geometric and text-prompted editing. In particular, we adopt the BrushNet (Ju et al., 2024) pipeline from the official PowerPaint implementation, incorporating structure-aware refinement for improved object boundary reconstruction.
- **MoGE (Wang et al., 2024b):** MoGE uses a ViT-L backbone to infer 3D structure from single RGB images. It is pretrained on large-scale image-depth pairs and is used to lift inpainted RGB objects into 3D point cloud space.
- **Grounded-SAM (Ren et al., 2024):** This model integrates Grounding DINO (Swin-T variant) for open-vocabulary object detection and SAM for pixel-accurate segmentation. The combination enables text-driven, high-quality instance mask extraction.
- **InvSR (Yue et al., 2024):** InvSR is a diffusion-inversion-based super-resolution model. It leverages backward sampling from pretrained latent diffusion models to reconstruct high-resolution images from low-resolution observations.

Models for Multi-Sweep Generation. To simulate temporally coherent sequences resembling multi-frame LiDAR scans, we employ additional models:

- **MOFA (Niu et al., 2024):** MOFA animates static object images into temporally coherent video sequences. It uses user-defined 2D tracking points to control motion generation.

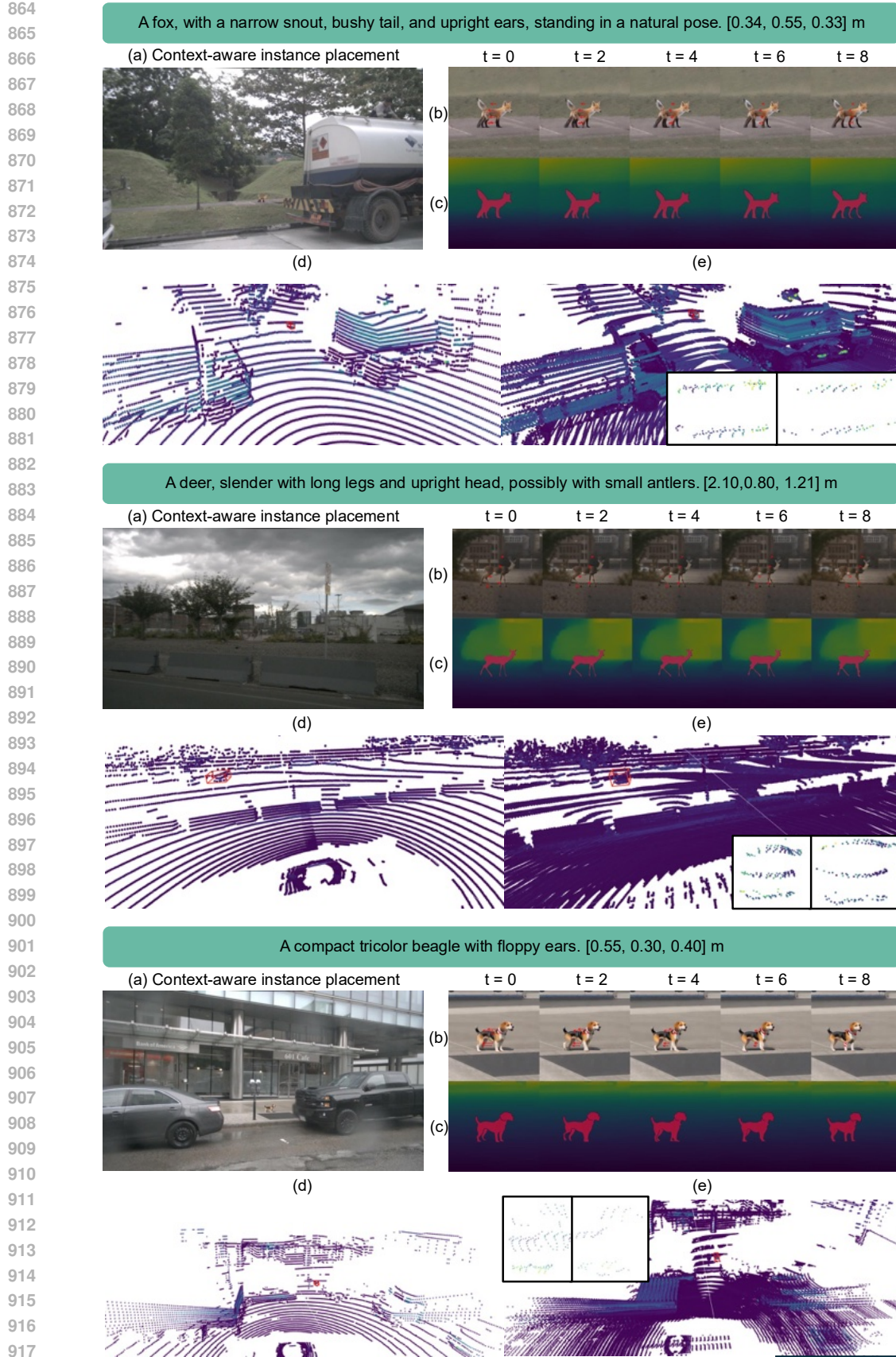


Figure 11: Examples of animals.

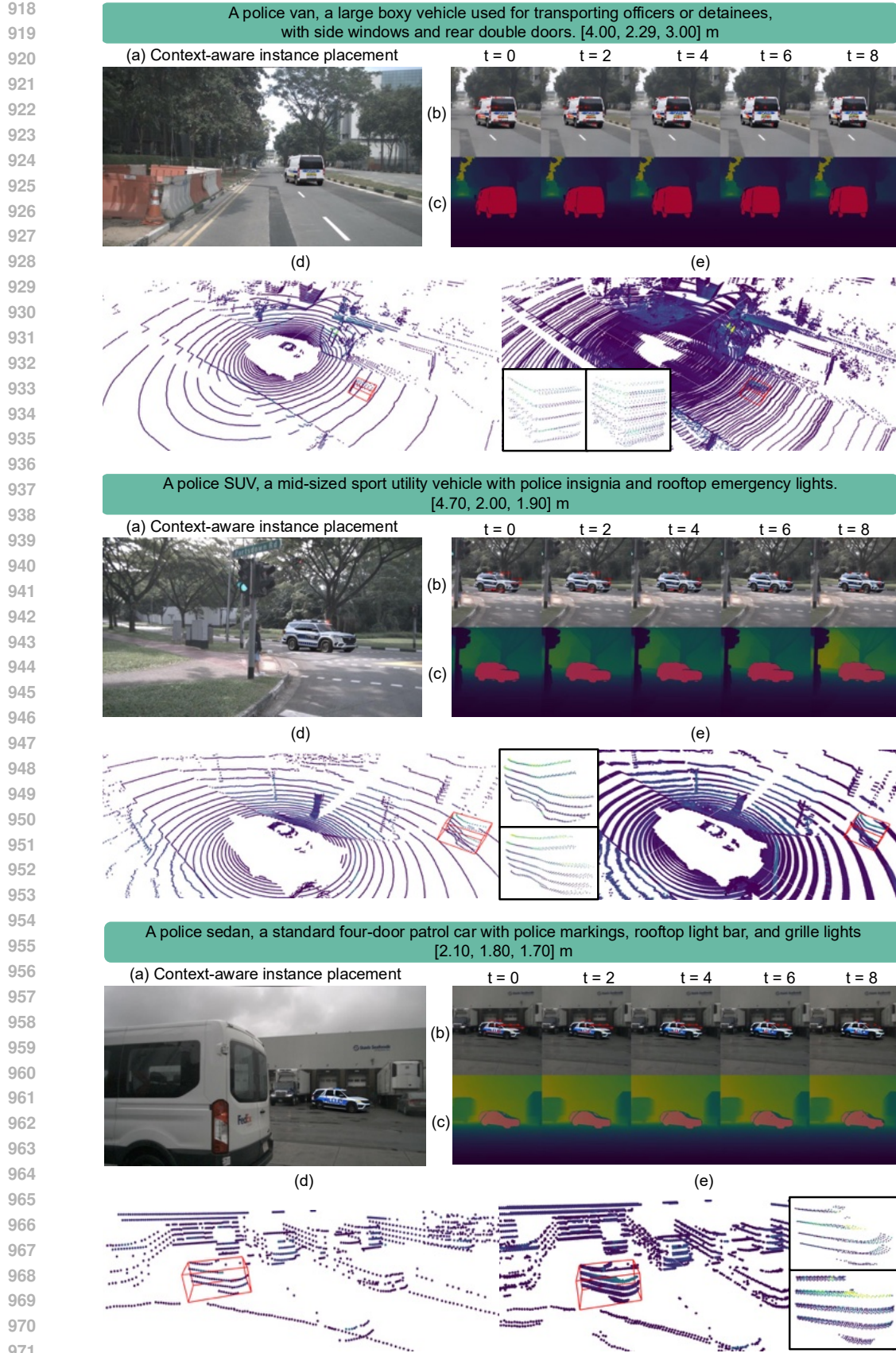


Figure 12: Examples of police vehicle.

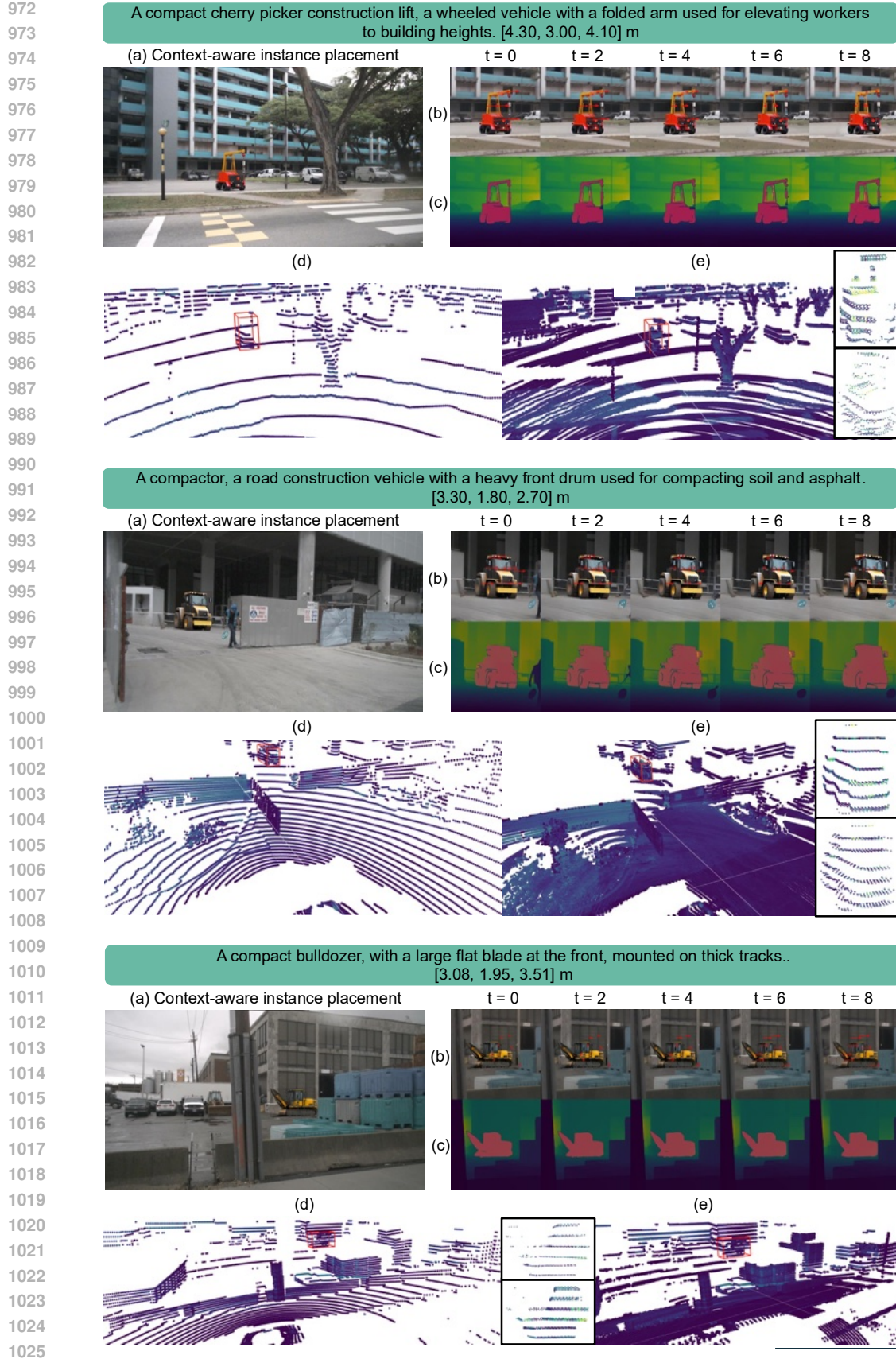


Figure 13: Examples of construction vehicle.



Figure 14: Examples of motorcycle.

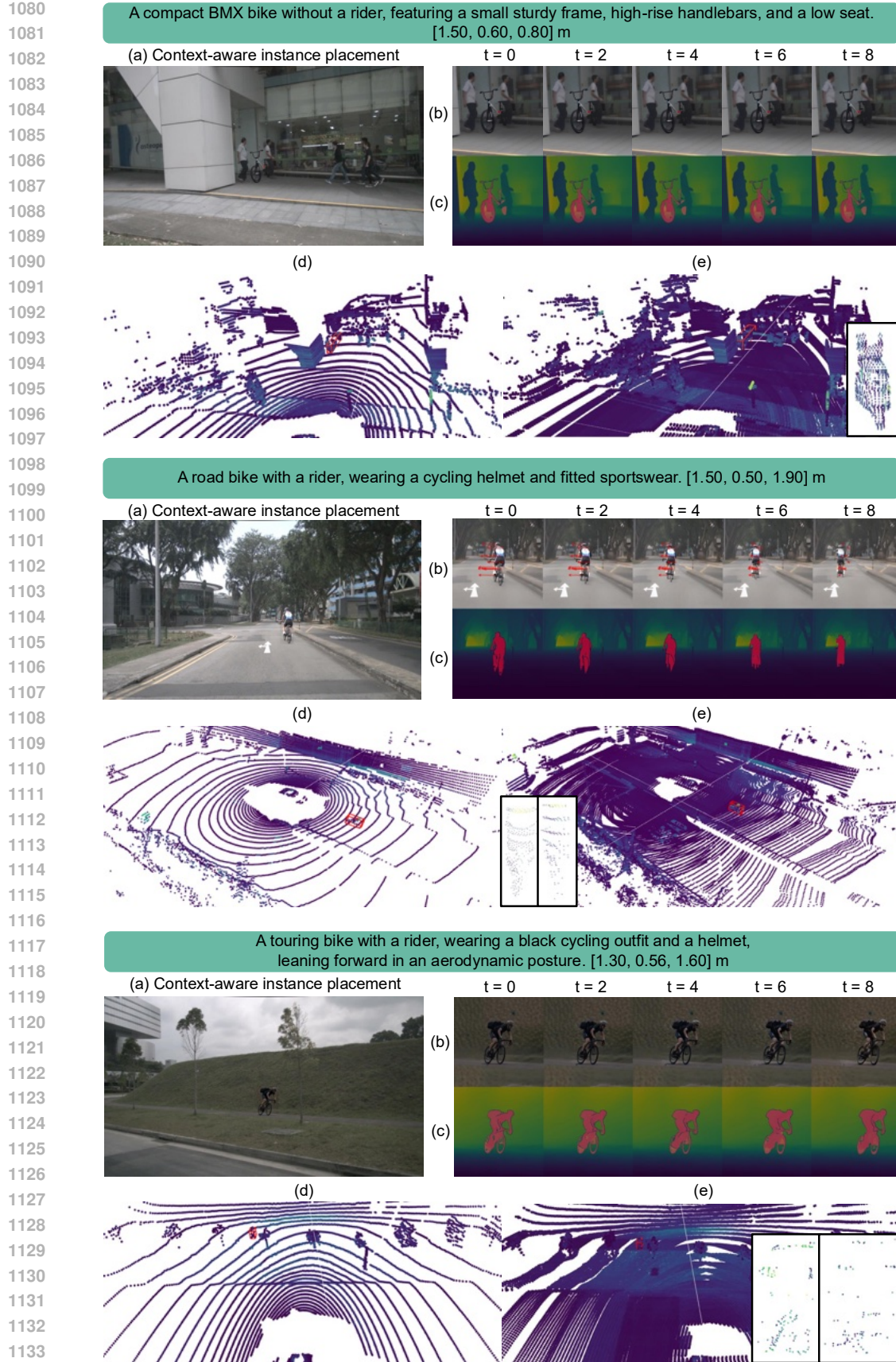


Figure 15: Examples of bicycle.

- **Video Depth Anything** (Chen et al., 2025): This model extends Depth Anything V2 to videos, providing temporally consistent monocular depth estimation. We use the ViT-L variant trained on long-sequence consistency objectives. It enables the generation of pseudo-LiDAR sequences from synthetic videos.
- **SAM2** (Ravi et al., 2024) **HQ** (Ke et al., 2023): We adopt sam2.1-hiera-large, a hierarchical high-resolution segmentation model from Meta AI. We use this model for image and video segmentation to obtain consistent instance masks across frames.

Computational Cost. To estimate the cost of semantic verification, we consider the full nuScenes training set comprising 168,780 images. Each cropped region yields approximately 85 visual tokens, concatenated with a prompt of ~ 30 tokens and an expected output of ~ 10 tokens. Assuming GPT-4o pricing, the total verification cost is estimated at \$48.52. Although MAPLE integrates multiple foundation models, the entire pipeline remains cost-effective, requiring less than \$50 for semantic verification of the entire nuScenes dataset. This demonstrates the practicality of scaling multimodal augmentation to millions of samples without additional training.

E.2 RGB GENERATION WITH SEMANTIC VERIFICATION

Object Description Generation To generate realistic subclass-level object descriptions, we employ a large language model (LLM) with carefully designed prompts. Each description contains three key attributes: subclass identity, visual appearance, and approximate physical size (s_x, s_y, s_z). For each target label (e.g., construction vehicle, bicycle, motorcycle), the LLM is instructed to produce one subclass commonly observed in urban environments, together with its visual traits and typical dimensions. Whenever possible, physical sizes are grounded in official manufacturer specifications of a real-world product model. If no exact specification is available, a closely related model is used with the `approximate` flag set to `true`, accompanied by a web search query for traceability. These descriptions not only serve as conditioning for diffusion-based inpainting but also provide priors for subsequent geometric verification.

For bicycles and motorcycles, we explicitly request two sets of bounding box dimensions: one excluding the rider (vehicle only) and one including a seated rider, which significantly changes the aspect ratio and bounding box height. This distinction is crucial for realistic pseudo-LiDAR reconstruction and for training detectors that must handle both rider-free and rider-occupied instances.

The following script shows the exact instruction given to the LLM. The output is required to be valid JSON conforming.

LLM Prompt: Subclass Description Generator

You are generating ONE subclass description for data augmentation.

GOAL

- Provide one concrete subclass of {TARGET_LABEL} commonly found in urban environments.
- Output MUST be valid JSON matching the schema below.
- Use meters for all sizes.

INSTRUCTIONS

- 1) Choose a realistic subclass (e.g., "bulldozer", "box truck", "commuter bicycle", "naked motorcycle").
- 2) Give a concise visual description (colors, shape, notable parts; 1--2 sentences).
- 3) For size, identify a specific real-world product model from the web (official/manufacturer spec preferred).
 - Normalize length/width/height to meters.
 - If exact spec is unavailable, pick a close model, set "approximate": true, and include a "web_search_query".
- 4) If TARGET_LABEL \in {bicycle, motorcycle}, ALSO provide a second bounding box that includes a seated rider with typical posture. Keep it realistic (do NOT under/over-estimate).

OUTPUT SCHEMA (JSON only, no extra text):

```

{
  "target_label": "string",
  "subclass_identity": "string",
  "product_model": "string",
  "visual_description": "string",
  "size_m": { "length": float, "width": float, "height": float },
  "with_rider_bbox_m": { "length": float, "width": float, "height": float } |
  ↪ null,
  "approximate": boolean,
  "source_note": "string",
  "web_search_query": "string"
}

CONSTRAINTS
- JSON must parse. No comments, no trailing commas, no units in numbers.
- Keep "visual_description" factual; avoid brand slogans.
- If bicycle/motorcycle: "with_rider_bbox_m" MUST NOT be null. Else: set null.

Begin now for TARGET_LABEL = {TARGET_LABEL}.

```

Representative outputs are shown in Fig. 3, which illustrate how the LLM generates diverse subclasses within the same category, each grounded by explicit product dimensions.

Object Inpainting After obtaining textual descriptions \mathcal{T}_c and spatial masks \mathcal{M} from projected 3D bounding boxes, we perform diffusion-based inpainting to insert novel objects into the RGB image. To preserve scene fidelity, we first extract a local image crop $\mathcal{I}_{\text{crop}}$ centered on the inpainting region $\mathcal{R}_{\text{inpaint}}$. A common challenge is that small or distant objects yield low-resolution crops, which degrade generation quality. To address this, we employ a super-resolution model f_{SR} (Yue et al., 2024) whenever the shorter side of the crop is below a resolution threshold R . This ensures that the subsequent diffusion model operates on high-quality inputs regardless of the native crop size.

Formally, the inpainting process is defined as:

$$\hat{\mathcal{I}} = f_{\text{in}}(\mathcal{I}_{\text{sr}}, \mathcal{T}_c, \mathcal{M}), \quad \mathcal{I}_{\text{sr}} = \begin{cases} f_{\text{SR}}(\mathcal{I}_{\text{crop}}), & \text{if } \text{Res}(\mathcal{I}_{\text{crop}}) < R, \\ \mathcal{I}_{\text{crop}}, & \text{otherwise,} \end{cases} \quad (4)$$

where $\text{Res}(\mathcal{I}_{\text{crop}})$ denotes the shorter side of the crop. The diffusion inpainting model f_{in} is conditioned on both the description \mathcal{T}_c and the binary mask \mathcal{M} , which specifies the exact region to be synthesized. The result is a high-fidelity augmented image $\hat{\mathcal{I}}$ where the inserted object is visually diverse, consistent with the surrounding context, and aligned with the size priors from the description stage. Examples of inpainted results are shown in Fig. 4, demonstrating how MAPLE produces context-aware augmentations with plausible scale, heading, and occlusions.

E.3 PSEUDO-LiDAR GENERATION WITH GEOMETRIC VERIFICATION

LiDAR Intensity Simulation To simulate reflectance, we compute a per-point intensity map $\mathcal{I}_{\text{intensity}}$ based on the grayscale inpainted image $\mathcal{I}_{\text{gray}}$, modulated by surface normal and range-based attenuation (Viswanath et al., 2024; Marcus et al., 2025):

$$\mathcal{I}_{\text{intensity}} = \text{clip} \left(\mathcal{I}_{\text{gray}} \cdot |\mathbf{n}|^{\gamma} \cdot e^{-\alpha|\mathbf{p}|}, 0, 255 \right), \quad (5)$$

where \mathbf{n} is the surface normal aligned with the sensor axis, γ controls orientation influence, $|\mathbf{p}|$ is the range to point \mathbf{p} , and α is the attenuation coefficient. As shown in Fig. 5, the final pseudo-LiDAR is obtained by back-projecting the scaled depth map and semantic mask into 3D space, incorporating both geometry and synthesized intensity.

Motion Simulation Module for Virtual Object Sweeps From the initial object point cloud \mathcal{P} , we uniformly sample K anchor points using voxel-wise binning along the x , y , and z axes. These anchor points are temporally propagated according to the object’s simulated motion, which models plausible 3D trajectories informed by class-specific velocities. Here, object-specific velocity statistics is sampled from the training set.

1. For bicycle and motorcycle, if the description does not include a rider, we set $(v_x, v_y) = 0$.
2. For animals and police, we query an LLM for typical urban speeds.

For each sweep, we compute the displacement vector δ_t and apply it uniformly to all sampled anchor points:

$$\delta_t = \mathbf{c}_t - \mathbf{c}_0, \quad \mathcal{P}_t = \mathcal{P} + \delta_t, \quad (6)$$

resulting in a set of temporally displaced point clouds $\{\mathcal{P}_t\}_{t=0}^T$, where \mathbf{c}_0 denote the initial center of the object and \mathbf{c}_t be its center at sweep t obtained from motion simulation. These propagated points define coarse motion trajectories that are projected onto the image plane via the projection matrice π :

$$\mathbf{u}_t^{(k)} = \pi(\mathbf{x}_t^{(k)}), \quad \text{for } k = 1, \dots, K, \quad (7)$$

producing smooth 2D trajectories $\{\mathbf{u}_t^{(k)}\}$ that condition the generation of temporally coherent video frames via the image animation diffusion model.

Ego-Motion Compensation and LiDAR Formatting To simulate temporally consistent LiDAR sequences, we apply ego-motion compensation to each sweep and unify them into a common coordinate frame.

Let \mathbf{T}_t denote the ego-pose at time t , and \mathbf{T}_t^{-1} its inverse. For each sweep \mathcal{P}_t , we perform the following steps:

1. **Ego-frame alignment.** Each point cloud \mathcal{P}_t is transformed from the camera coordinate frame into the ego-centric frame using homogeneous coordinates:

$$\mathcal{P}_t^{\text{ref}} = (\mathbf{T}_t^{-1} \cdot [\mathbf{x}_t, 1]^T)^T, \quad (8)$$

where a homogeneous coordinate 1 is appended to each point \mathbf{x}_t to support affine transformation. This yields $\mathcal{P}_t^{\text{ref}}$, a point cloud expressed in the unified ego-frame.

2. **LiDAR formatting.** The points are projected onto a simulated 32-channel LiDAR range-view, then unprojected back to 3D space.
3. **Sweep-specific re-transformation.** To simulate the original acquisition conditions, the formatted points are re-transformed back to the original sweep frame using \mathbf{T}_t .

The re-aligned point clouds from all sweeps are concatenated to form a temporally coherent pseudo-LiDAR sequence. Additionally, we estimate planar velocity by comparing the mean position across the first and last sweeps for dynamic instances:

$$\mathbf{v} = \frac{\mathbb{E}[\mathcal{P}_T^{(x,y)}] - \mathbb{E}[\mathcal{P}_0^{(x,y)}]}{\Delta t}. \quad (9)$$

This process ensures that our pseudo-LiDAR reflects both temporal motion and ego-vehicle dynamics, providing realistic training signals for motion-aware 3D perception models.

Bounding Box Refinement. For each accepted instance, we refine the initial estimate \mathbf{b}_{3D} into a tight bounding box around the synthesized point cloud $\hat{\mathcal{P}}_{\text{lidar}}$, written as $\hat{\mathbf{b}}_{3D} = [\hat{c}_x, \hat{c}_y, \hat{c}_z, \hat{s}_x, \hat{s}_y, \hat{s}_z, \hat{\theta}]$. Here, the center $(\hat{c}_x, \hat{c}_y, \hat{c}_z)$ is taken from the centroid of $\hat{\mathcal{P}}_{\text{lidar}}$, the heading $\hat{\theta}$ is estimated from the dominant eigenvector of the XY -plane covariance, and the dimensions $(\hat{s}_x, \hat{s}_y, \hat{s}_z)$ are derived from axis-aligned bounds in the rotated frame and from the vertical extent.

F TRAINING SETTING

F.1 AUGMENTATION STRATEGIES

We consider three instance-level augmentation strategies: GT-Aug (Yan et al., 2018), Text3DAug (Reichardt et al., 2024), and PGT-Aug (Chang et al., 2024). GT-Aug inserts objects sampled from a ground-truth database into training scenes. Text3DAug and PGT-Aug generate synthetic samples by rendering 3D meshes at 13 discrete heading angles (from -180° to 180° in 30° increments) and

distributing them across a 2D spatial grid covering the full perception range ($[-50\text{ m}, 50\text{ m}]$ in both x and y , sampled every 5 m). The resulting synthetic instances are stored in a custom database and inserted into training scenes in a 1:1 ratio with ground-truth instances; that is, for each rare class, half of the inserted instances are drawn from the ground-truth database, and the other half from generated samples.

F.2 3D OBJECT DETECTION

Instance Augmentation for LiDAR-only Setting. For the seven frequent object categories (car, truck, bus, trailer, barrier, pedestrian, traffic cone), we apply GT-Aug (Yan et al., 2018). We insert 7, 6, and 6 instances per scene for the three long-tail categories—construction vehicle, motorcycle, and bicycle. The synthetic augmentation strategies and sampling configurations follow the protocols described in Section F.1. We prioritize inserting our synthesized instances via instance composition to construct each augmented scene. If the number of available synthetic instances for a given category falls short of the per-scene target, we supplement the remainder with ground-truth objects, maintaining consistency with the GT-Aug.

Implementation Details. Following the nuScenes 3D detection benchmark (Caesar et al., 2020), we conduct experiments using the 10-sweep setting. We evaluate our method in the multimodal setting using BevFusion (Liang et al., 2022), which integrates LiDAR and multi-view camera features via a Swin Transformer backbone and a voxel-based encoder, followed by ConvFuser and a TransFusionHead. We follow the original training configuration (Liang et al., 2022), using an AdamW optimizer with a cosine annealing schedule, a base learning rate of 0.0001, and train the model for 6 epochs with a batch size of 3 per GPU. To assess the impact of pseudo-LiDAR augmentation in LiDAR-only scenarios, we additionally report results using CP-Voxel (Yin et al., 2021) and PointPillar (Lang et al., 2019). CP-Voxel is a CenterPoint-based detector configured with a fine voxel size of $[0.075, 0.075, 0.2]$, while PointPillar adopts a lightweight PillarVFE encoder with a coarser voxel size of $[0.2, 0.2, 8.0]$. Both models are trained using a one-cycle learning rate schedule, with an initial learning rate of 0.001 and a weight decay of 0.01. The total number of epochs is 20, with a batch size of 8 per GPU. Following common practice in OpenPCDet, we apply a curriculum-style augmentation strategy by disabling instance sampling during the final five epochs to mitigate overreliance on synthetic data and improve generalization to real-world distributions.

F.3 3D SEMANTIC SEGMENTATION

Instance Augmentation. For the 14 frequent classes, we do not apply any instance-level augmentation. To ensure a controlled comparison, we insert the same number of instances per class in GT-Aug and Text3DAug, PGT-Aug variants as in our method for each training scene, following the sampling and insertion strategy described in Section F.1.

Implementation Details. We follow the nuScenes 3D semantic segmentation benchmark (Caesar et al., 2020) and conduct experiments using the 1-sweep setting. We evaluate our method for 3D semantic segmentation using two architectures: MinkowskiNet (Choy et al., 2019) and SPVCNN (Tang et al., 2020). Both models operate directly on voxelized point clouds with a voxel resolution of 0.05 m. The segmentation head is trained to predict 17 semantic classes following the nuScenes benchmark. MinkowskiNet is a sparse convolutional network based on the MinkowskiEngine framework. SPVCNN follows a similar design but integrates sparse point-voxel convolution layers to enhance efficiency while maintaining high-resolution feature propagation. Both models are trained for 80 epochs using the SGD optimizer with a learning rate of 0.24, momentum of 0.9, and a weight decay of 1×10^{-4} . We use cosine annealing learning rate scheduling and enable standard augmentation techniques during training, including random rotation, flipping, scaling, transformation, and point dropout. The batch size is 32, with eight workers per GPU for training and validation.