Drive-Cascade: Autoregressive Occupancy to LiDAR and Video Synthesis

Supplementary Material

A. Implementation Details

A.1. Training Configuration

We implement Drive-Cascade in PyTorch and train on 8× NVIDIA A100 GPUs (80GB each). The three modules—Occ Dreamer, LiDAR Dreamer, and Video Dreamer—are trained independently, each requiring approximately one week. We use the AdamW optimizer (β_1=0.9, β_2=0.999, weight decay 0.01) with an initial learning rate of 1×10^{-4}, a cosine annealing scheduler after warm-up, gradient clipping at 1.0, and Automatic Mixed Precision (AMP) throughout.

A.2. Data Preprocessing

All data modalities are upsampled to 12Hz. Input video frames are resized to 448×800. Occupancy voxels use a 0.4m resolution. LiDAR point clouds are voxelized within $[\text{-}50\text{m}, 50\text{m}]$ along X and Y.

A.3. Inference and Sampling

The Occ Dreamer uses an Euler ODE solver with 50 integration steps. The Video and LiDAR Dreamers use the DDIM sampler with 50 denoising steps. Classifier-Free Guidance (CFG) is applied to all three modules: during training, conditioning signals are dropped with 10% probability; at inference, CFG scales are set to 4.5, 3.0, and 2.5 for the Video, LiDAR, and Occ Dreamers, respectively. The autoregressive history window k is set to 3 frames.

A.4. Conditioning Robustness

Since HD maps and 3D bounding boxes may originate from noisy perception pipelines in practice, we augment training with controlled perturbations: random translational ($\pm$0.5m) and rotational ($\pm5^\circ$) noise is applied to 3D boxes with 30% probability, and individual HD map elements are randomly dropped with 10% probability. This prevents overfitting to clean annotations and improves robustness during extended autoregressive rollouts.

B. System Complexity and Efficiency

Table 10 reports parameter count, peak GPU memory, and per-frame inference speed for each module, measured on a single A100 GPU. Drive-Cascade targets offline data engine use rather than real-time deployment; the decoupled modular design keeps each component's memory footprint manageable.

Table 10. System complexity, memory footprint, and inference speed on a single A100 GPU.

Module	Params (B)	Memory (GB)	Inference (FPS)
Occ Dreamer	0.12	52	0.67
LiDAR Dreamer	1.29	62	0.22
Video Dreamer	2.48	65	0.18

C. Detailed Network Architectures

C.1. Occupancy Dreamer

The Occ Dreamer generates 4D semantic occupancy sequences using a HexPlane VAE combined with a Flow Matching DiT backbone. Directly modeling the full 4D voxel volume ($T \times Z \times X \times Y$) incurs $\mathcal{O}(N^4)$ complexity. The HexPlane VAE resolves this by factorizing the volume into six orthogonal 2D feature planes (*e.g.*, XY, XT, YT), reducing complexity to $\mathcal{O}(N^2)$ while preserving both spatial layout and temporal dynamics. In the resulting compact latent space, we train a DiT with the optimal-transport Flow Matching objective, which regresses the velocity field of a straight-line path from noise to data. This produces straighter generation trajectories than DDPM/DDIM, accelerating sampling and improving stability when modeling complex dynamic scene structures.

C.2. Video Dreamer

The Video Dreamer synthesizes multi-view driving videos strictly aligned with the 3D geometry of Occ_t. Built on a spatiotemporal Video DiT that processes video as flattened patch tokens, we inject occupancy guidance via a ControlNet-based mechanism to avoid disrupting the pretrained visual priors. The 4D semantic occupancy is projected into per-camera perspective-view feature maps (Occ-MPI) using the MPI Projector (see Section 3.2 of the main paper), then passed through zero-convolution layers into the main DiT branch. This ensures that generated visual elements—vehicles, lanes, obstacles—are spatially consistent with the occupancy layout while maintaining photorealism.

C.3. LiDAR Dreamer

The LiDAR Dreamer autoregressively generates LiDAR point cloud sequences using a DiT conditioned via a Feature Alignment Adapter. Because the occupancy input is a dense volumetric representation while the LiDAR target is sparse and unstructured, direct concatenation is suboptimal. The Adapter bridges this gap through a series of convolutional blocks and MLPs that extract hierarchical geometric

features from the occupancy BEV projection and map them into the DiT's latent space. The DiT then denoises random latents into precise 3D point coordinates and intensity values, producing sensor-realistic sweeps that faithfully reflect the scene geometry predicted by the Occ Dreamer.

D. Data Processing

D.1. nuScenes Dataset

All experiments use the nuScenes dataset [3], comprising 1000 driving scenes (700 train / 150 val / 150 test), each 20 seconds long. Each scene includes six surround-view cameras, one 32-beam LiDAR, and HD maps.

D.2. Occupancy Upsampling

Semantic occupancy annotations from nuScenes-Occupancy [30] are provided at 2Hz, insufficient for high-frequency dynamic generation. We upsample to 12Hz using Neural Kernel Surface Reconstruction (NKSR) [17] via three stages: (1) *sparsification* — convert dense voxel grids of keyframes into sparse point clouds while retaining semantic labels; (2) *surface reconstruction* — fit a continuous implicit surface with NKSR to interpolate geometry between keyframes; (3) *resampling* — query the implicit surface at intermediate timestamps to produce dense occupancy grids at 12Hz. This yields a high-frequency, temporally consistent 4D occupancy dataset used as ground truth for training the Occ Dreamer.

E. Evaluation Metrics

E.1. Occupancy

We use three complementary metrics. **mIoU** measures per-class semantic accuracy (excluding the empty class), averaged across all semantic categories. **F3D** (Fréchet 3D Distance) adapts the FID paradigm to 3D occupancy by measuring the distance between feature distributions of generated and real sequences; lower values indicate higher fidelity. **MMD** (Maximum Mean Discrepancy) quantifies distributional distance between generated and real sequences, capturing temporal dynamics and physical plausibility.

E.2. Video Generation

We evaluate from two perspectives. **FVD** (Fréchet Video Distance) [12] measures both visual realism and temporal coherence using an I3D network pre-trained on Kinetics-400; lower is better. For perception utility, we use a pre-trained BEVFormer [22] as an oracle to compute **mAP** for 3D object detection and **mIoU** for BEV map segmentation on the generated videos. High scores on these metrics confirm that the generated scenes preserve accurate semantic layouts recognizable by state-of-the-art detectors.

E.3. LiDAR Generation

Following [19, 23], we evaluate scene-level fidelity with two metrics. **JSD** (Jensen-Shannon Divergence) measures how well the range-image distribution of generated point clouds matches that of the real validation set. **MMD** quantifies geometric similarity between generated and real LiDAR frames in feature space, penalizing local shape inaccuracies.

F. Long-Horizon Stability

We evaluate long-horizon stability by measuring FVD over 5, 10, and 20-second clips. As shown in Table 11, FVD increases gradually (87.47 $\rightarrow$ 101.17 $\rightarrow$ 113.74) without abrupt collapse, confirming that the shared 4D occupancy scaffold effectively suppresses error accumulation over extended autoregressive rollouts.

Table 11. Long-horizon stability: FVD over different clip lengths.

Method	$\text{FVD}_5 \downarrow$	$\text{FVD}_{10} \downarrow$	$\text{FVD}_{20} \downarrow$
Ours	**87.47**	**101.17**	**113.74**

G. Additional Qualitative Results

Figure 7 presents multi-frame generation results across a representative driving scene. The visualization demonstrates strong cross-modal consistency: geometric structures in the generated LiDAR and occupancy are tightly aligned with the visual semantics in the corresponding video frames. The sequence also exhibits stable long-horizon temporal coherence, with smooth object motion and consistent background details throughout, free from flickering or structural drift.

H. Limitations and Broader Impact

Limitations. The iterative diffusion process in all three modules incurs substantial inference latency, which currently limits applicability to offline data engine scenarios rather than real-time simulation.

Broader Impact. Drive-Cascade can reduce reliance on costly real-world data collection for autonomous driving. Synthesized data should nonetheless be carefully validated before use in training safety-critical perception systems, as domain gaps between generated and real data may affect downstream model reliability.

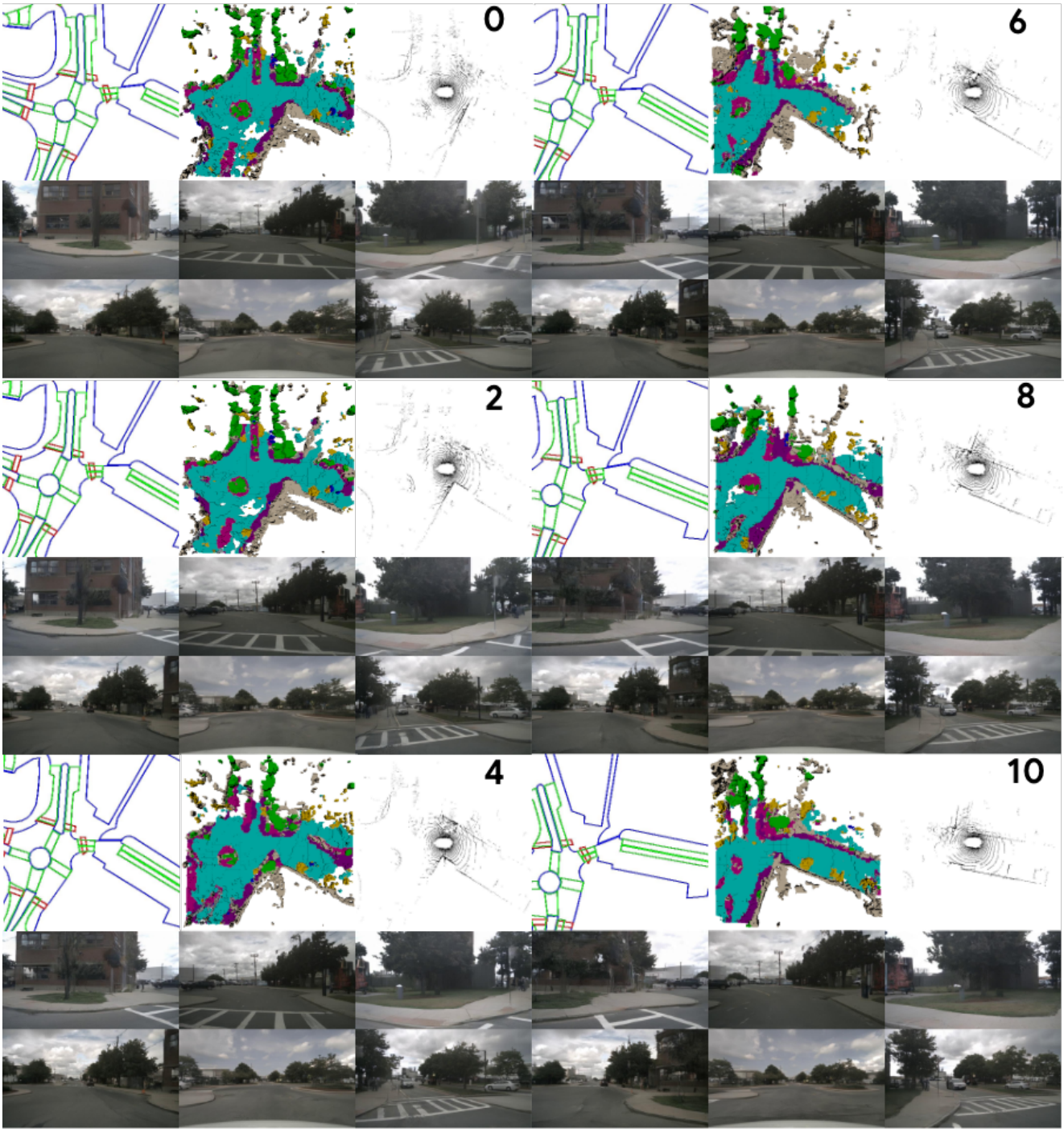

Figure 7. **Additional qualitative results of multimodal scene generation.** Conditioned on HD maps and 3D bounding boxes, DriveCascade jointly synthesizes three aligned modalities—occupancy, LiDAR point clouds, and multi-view videos—demonstrating strong cross-modal geometric consistency.