

PrITTI: Primitive-based Generation of Controllable and Editable 3D Semantic Urban Scenes

Christina Ourania Tze¹ Daniel Dauner^{1,4} Yiyi Liao² Dzmitry Tsishkou³ Andreas Geiger^{1,4}

¹University of Tübingen, Tübingen AI Center ²Zhejiang University ³Noah's Ark Lab, Huawei

⁴KE:SAI - Kyutai ELLIS Scalable Autonomous Intelligence

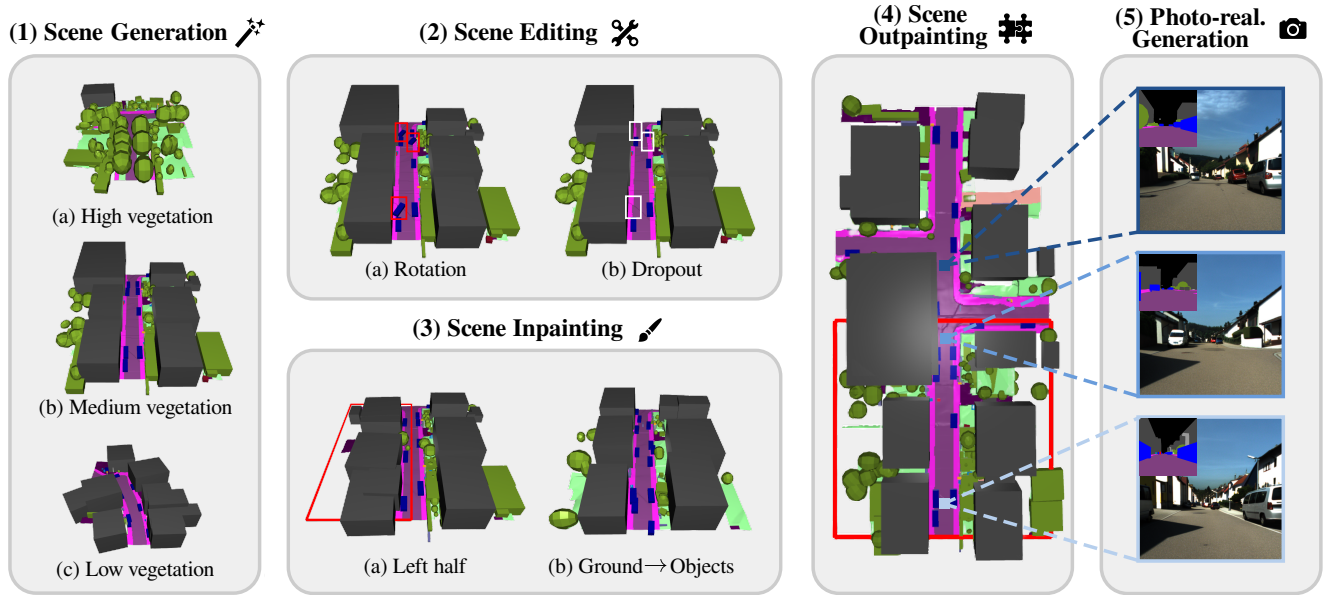


Figure 1. **PrITTI** generates (1) high-quality, controllable 3D semantic urban scenes in a compact primitive-based representation using a latent diffusion model. Starting from a generated scene (e.g. middle sample), we demonstrate downstream applications including (2) scene editing, (3) inpainting, (4) outpainting, and (5) photo-realistic street view synthesis.

Abstract

Existing approaches to 3D semantic urban scene generation predominantly rely on voxel-based representations, which are bound by fixed resolution, challenging to edit, and memory-intensive in their dense form. In contrast, we advocate for a primitive-based paradigm where urban scenes are represented using compact, semantically meaningful 3D elements that are easy to manipulate and compose. To this end, we introduce *PrITTI*, a latent diffusion model that leverages vectorized object primitives and rasterized ground surfaces for generating diverse, controllable, and editable 3D semantic urban scenes. This hybrid representation yields a structured latent space that facilitates object- and ground-level manipulation. Experiments on KITTI-360 show that primitive-based representations unlock the full

capabilities of diffusion transformers, achieving state-of-the-art 3D scene generation quality with lower memory requirements, faster inference, and greater editability than voxel-based methods. Beyond generation, *PrITTI* supports a range of downstream applications, including scene editing, inpainting, outpainting, and photo-realistic street-view synthesis. The source code and more results can be found at <https://raniatze.github.io/pritti/>.

1. Introduction

Generating large-scale, semantically structured 3D urban scenes is essential for digital world modeling and autonomous driving simulation. Such virtual worlds enable safety validation, rare-scenario testing, and sensor data synthesis for perception and planning. However, constructing them typically requires expert knowledge and extensive man-

ual effort, making the process costly and time-consuming. Although some structured 3D datasets exist, their scarcity, particularly for outdoor settings, restricts diversity and yields repetitive scene patterns. These limitations motivate generative approaches capable of automatically producing diverse and realistic urban environments. To be practically useful, such methods should additionally support controllability and flexible editing. The extent to which these capabilities can be achieved largely depends on the chosen underlying representation. An ideal representation should be memory-efficient, compositional, and support localized, intuitive edits. Most existing approaches model 3D urban environments using voxel grids [38, 105] or hierarchical voxel structures [44, 64], which suffer from three fundamental limitations. First, due to cubic scaling with resolution, they are *memory-inefficient* and prohibitively expensive for large environments. Second, their fixed spatial resolution leads to *limited scene detail*. For instance, tall elements of a scene (e.g. buildings or trees) often appear truncated or distorted. Third, voxel grids are *difficult to manipulate*. Even simple edits, like translating a vehicle, require identifying and updating all related voxels and then filling the vacated space, a process which is complicated by ambiguous object-background boundaries (e.g. car vs. road). These challenges highlight the need for a more compact, flexible, and object-centric scene representation.

In this work, we advocate for primitives as a compelling alternative for scalable 3D semantic scene synthesis, owing to their compositional and parsimonious nature. Despite their extensive use in shape abstraction [39, 52, 78, 107] and scene parsing [22], their potential in generative settings, especially for synthesizing 3D semantic urban scenes, remains largely underexplored. Motivated by this gap, we present PrITTI, the first framework to use coarse 3D bounding primitives for generating controllable and editable 3D semantic scenes in urban environments. While recent approaches [3, 13, 45, 62, 93, 95, 96] employ abstract semantic layouts, these are typically hand-crafted or pre-defined inputs rather than generated. Our method adopts the common primitive-based scene representations [10, 11, 41, 86], modeling objects as 3D cuboids (e.g. buildings, vehicles) or ellipsoids (e.g. trees) and ground surfaces as extruded 2D polygons of varying shapes (e.g. road, sidewalk). Although cuboids and ellipsoids allow consistent parameterization, the arbitrary geometry of polygons makes uniform encoding impractical. We address this via a hybrid representation: ground elements are rasterized, while objects are modeled as parameterized primitives (cuboids and ellipsoids). Training proceeds in two stages: a variational autoencoder (VAE) [34] first encodes 3D layouts into compact 2D latents, followed by a latent diffusion model (LDM) [68] trained on discrete primitive density labels for controllable scene generation. Our LDM supports downstream tasks such as scene inpainting and outpainting without fine-tuning (see Fig. 1). This

is achieved via a latent manipulation mechanism [47] that allows localized edits while preserving global structure. A key component of our approach is its disentangled latent structure, where ground and object elements are encoded in two separate channel groups. This design enables further applications (e.g. ground-conditioned object generation) without extra supervision. Additionally, our instance-level primitive representation supports intuitive editing operations (e.g. translation, scaling, rotation). Finally, we introduce a Cholesky-based [15] parameterization of 3D object orientation and size, offering greater numerical stability over quaternion encodings, as evidenced by our experiments.

To summarize, our method introduces: (i) the first primitive-based framework for large-scale urban 3D scene generation at an abstract, semantic level, (ii) a disentangled latent representation enabling controllable manipulation, and (iii) a stable encoding of 3D object orientation and size based on Cholesky decomposition. Experiments demonstrate that PrITTI achieves competitive reconstruction with lower memory requirements and delivers superior generation quality, diversity, and editability compared to voxel-based baselines.

2. Related Work

Primitive-based Representation. Primitives have been explored since the early days of computer vision [5, 6, 9, 60, 67] as compact, compositional building blocks for objects and scenes. With deep learning, these ideas re-emerged for decomposing 3D shapes into semantically meaningful parts or recovering object structure from RGB or RGB-D inputs. Most works represent objects using simple cuboids [39, 52, 53, 78, 107], while others adopt more expressive primitive types such as superquadrics [55, 56], convexes [16], axis-aligned Gaussians [21], and spheres [24]. Recent part-aware generative approaches like SPAGHETTI [25] and SALAD [35] decompose objects into primitive-based subparts (e.g. legs, seat of a chair) for part-level manipulation. Beyond individual objects, few recent methods decompose indoor [80–82] or outdoor scenes [22] into simple geometric shapes. While [25, 35, 40, 51, 107] demonstrate primitive generation at the object level, the majority of prior approaches instead use primitives for shape abstraction or scene parsing. In contrast, we leverage parameterized primitives, each corresponding to a distinct object instance, for abstract 3D semantic layout generation, enabling direct scene-level operations (e.g. object arrangement), which remain beyond the scope of prior object-centric methods.

3D Scene Generation. Recent works tackle 3D scene generation from different perspectives, aiming to synthesize either detailed geometry or realistic appearances. Geometry-focused methods [7, 49, 88] produce high-fidelity meshes but lack semantic and compositional structure. Early appearance-oriented approaches [20, 29, 73] update explicit meshes to generate immersive 3D scenes from text. Other meth-

ods adopt volumetric scene representations, optimizing NeRFs [103] or 3D Gaussians [54, 72, 97] for global consistency and photorealism. More recent frameworks [65, 98–100] introduce 3D-aware caches that guide view generation toward spatio-temporal coherence. Finally, two-stage panoramic methods [30, 71, 92, 94] generate 3D scenes from text or single images by first synthesizing a 2D panorama and then lifting it to 3D to optimize Gaussian splats. Despite impressive realism, these appearance-driven approaches represent scenes holistically, without explicit, object-level structure. In contrast, we emphasize a controllable semantic layout over appearance or geometric fidelity, enabling direct instance-level control and high-level manipulation.

Semantic Layout Generation. Prior work in this area can be divided into indoor and outdoor approaches. Early indoor methods generate objects sequentially using autoregressive models [57, 66, 84], whereas recent approaches [77, 85] predict all elements jointly using transformers or diffusion models. Conditioning signals include floor plans [57], partial layouts [57, 66, 84], text [84], and scene graphs [18, 42, 48, 101, 102]. While effective at room scale, these techniques have not been demonstrated in outdoor settings, which involve higher object counts and greater geometric diversity. Early methods in this direction [17, 43, 89–91] produce 2.5D layouts by generating pairs of top-down semantic and height maps, then lifting objects into 3D via vertical extrusions of 2D footprints. However, they are limited by the generated maps’ resolution and cannot capture complex scene geometry. Recently, voxel grids have been applied for 3D semantic scene generation. [38] and [105] compress voxels into latent triplanes to improve efficiency and train unconditional or mask-conditional latent diffusion models, respectively. Hierarchical voxel models [44, 64] further aim to improve scalability by iteratively refining scenes across multiple resolutions. However, this design requires training multiple VAEs and diffusion models per hierarchy level. Moreover, although sparse voxel representations can significantly reduce memory requirements, e.g. as implemented in XCube [64] via custom CUDA operations, voxel-based approaches still inherit the fundamental limitations of voxel grids, including limited editability and fixed spatial resolution. Semantic scene generation has also been applied in generative driving simulators to synthesize traffic layouts, using agent bounding boxes [31, 46, 76, 106], lane graphs [50], or both [14, 69, 75], with the closest to ours being SLEDGE [14]. Unlike these 2D abstraction-based methods, our approach models 3D scenes with richer primitive types, semantic classes, and flexible ground geometry beyond 2D polylines.

3. Method

We introduce PrITTI, a latent diffusion model for controllable and editable generation of 3D primitive-based semantic scene layouts. Fig. 2 shows an overview of our approach.

3.1. 3D Semantic Scene Layout Representation

An input 3D semantic layout $\mathcal{S}$ contains 3D cuboids and ellipsoids for object instances and upwards-extruded polygons for ground surfaces. Given their structural differences, we employ distinct representations for each element type:

Ground Raster Maps. Ground annotations include five semantic classes (road, sidewalk, parking, terrain, ground) and are defined as 2D polygons extruded into 3D (see Fig. 2). Their varying shapes and vertex counts make consistent parameterization difficult. We therefore perform ray casting along the gravity direction to generate bird’s eye view (BEV) ground raster maps that are well-suited for convolutional networks. These consist of height maps $\mathbf{H} \in \mathbb{R}^{H \times W \times 5}$ and binary occupancy masks $\mathbf{B} \in \{0, 1\}^{H \times W \times 5}$.

Object primitives. We represent objects using 11 primitive categories: vegetation cuboid ($\mathcal{VC}$), vegetation ellipsoid ($\mathcal{VE}$), vehicle big ($\mathcal{VB}$), vehicle small ($\mathcal{VS}$), two-wheelers ($\mathcal{TW}$), human ($\mathcal{H}$), construction big ($\mathcal{CB}$), construction small ($\mathcal{CS}$), pole ($\mathcal{P}$), traffic control ($\mathcal{TC}$), and object ($\mathcal{O}$). All are annotated with 3D cuboids except $\mathcal{VE}$, which uses 3D ellipsoids. Each primitive is parameterized by a 9D feature vector $\mathbf{f}_i \in [0, 1]^9$, comprising a normalized 3D center location $\mathbf{t}_i \in \mathbb{R}^3$ and 6D Cholesky parameters $\mathbf{c}_i \in \mathbb{R}^6$. The latter jointly encode orientation and size via the Cholesky decomposition [15] of a positive-definite scatter matrix representing the object’s 3D extent. Unlike eigen- or quaternion-based encodings, this unique parameterization eliminates orientation ambiguities caused by rotational symmetries and improves training stability. Further details, including semantic class-to-primitive mappings and Cholesky derivations, are provided in the supplementary material.

Overall, we represent $\mathcal{S}$ as $\mathcal{L} = \{\mathbf{H}, \mathbf{B}, \mathbf{F}\}$, where $\mathbf{F} \in \mathbb{R}^{M \times 9}$ stores the feature vectors of all M object primitives.

3.2. Layout Variational Autoencoder

To enable LDM-based generation, we design a layout VAE (LVAE) with jointly trained raster and vector branches that encode 3D scene layouts into a compact 2D latent space.

Encoding. Our architecture employs two encoders for distinct input modalities. The convolutional **ground encoder** $\mathcal{E}_G$ encodes normalized ground raster maps to a latent code $\mathbf{z}_G \in \mathbb{R}^{h \times w \times c}$, where $h = H/2^d$, $w = W/2^d$, and c is the latent channel dimension. The **object encoder** $\mathcal{E}_O$ processes object primitives represented by feature vectors $\mathbf{f}_i$. To support batching during training and fixed-size set prediction at decoding, we pad each layout to a constant number of primitives per category, determined as a conservative upper bound based on dataset statistics (see supplementary material). Each primitive is thus augmented with a 10D learnable category embedding and a binary padding flag $p_i \in \{0, 1\}$, yielding features $\mathcal{F} \in \mathbb{R}^{N \times 20}$ for a fixed number N of objects across all categories. These features are projected to a higher-dimensional space and processed

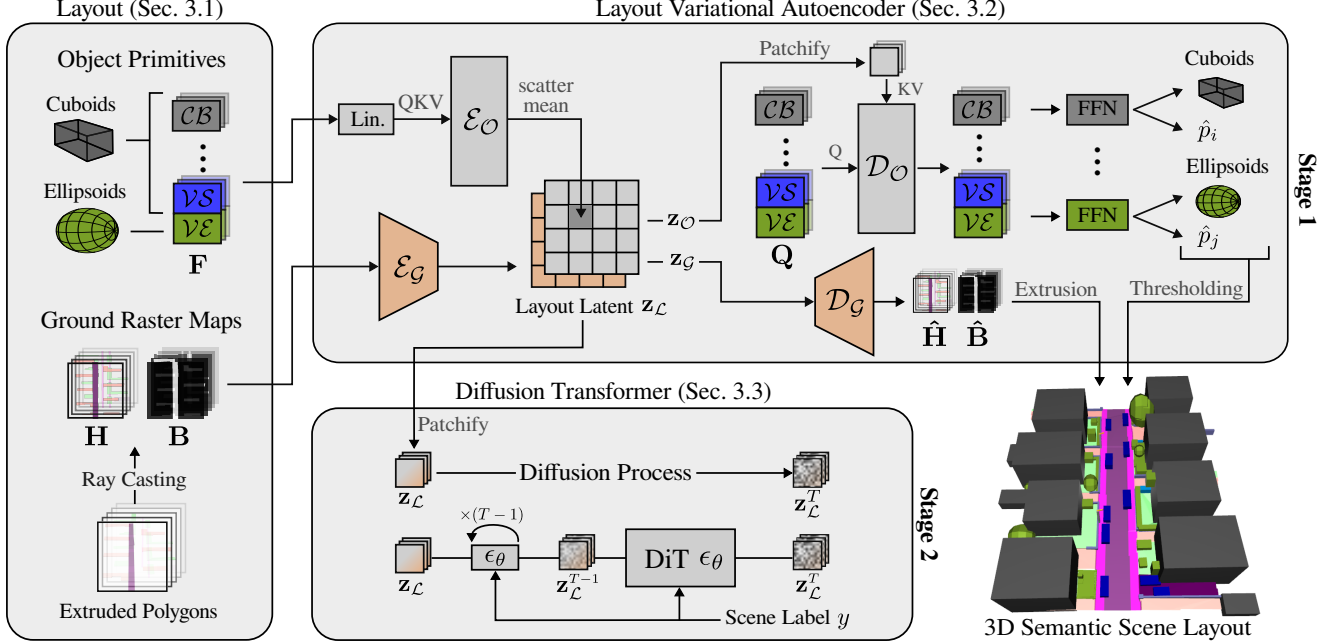


Figure 2. **Training Overview.** An input 3D semantic layout $\mathcal{S}$ comprises object primitives, encoded as feature vectors $\mathbf{F}$, and extruded ground polygons, rasterized into height maps $\mathbf{H}$ and binary occupancy masks $\mathbf{B}$ (Sec. 3.1). A layout VAE with separate encoder-decoder pairs for objects ($\mathcal{E}_O/\mathcal{D}_O$) and ground ($\mathcal{E}_G/\mathcal{D}_G$) first compresses $\mathcal{S}$ into a structured latent representation $\mathbf{z}_L$ (Sec. 3.2). In the second stage, a diffusion model is trained over this latent space for controllable scene generation (Sec. 3.3). At inference, the diffusion model generates latent codes either unconditionally or conditioned on the scene label y , which are then decoded by the VAE into novel 3D layouts.

by a Transformer encoder [79] to model inter-primitive relations. The resulting embeddings are mapped onto a learnable 2D latent grid matching the ground latent resolution via a scatter-mean operation. Specifically, each feature is assigned to a grid cell based on the normalized 2D center location of its corresponding primitive, and overlapping entries are averaged. This design preserves permutation invariance, maintains spatial structure in the latent space, and empirically worked best in our initial experiments. The final grid serves as the object latent $\mathbf{z}_O \in \mathbb{R}^{h \times w \times c}$, which is channel-wise concatenated with $\mathbf{z}_G$ to form the joint layout latent $\mathbf{z}_L = [\mathbf{z}_G; \mathbf{z}_O] \in \mathbb{R}^{h \times w \times 2c}$ for diffusion modeling.

Decoding. During decoding, a latent code is sampled from the joint distribution and split along the channel dimension into ground and object components. The **ground decoder** $\mathcal{D}_G$ reconstructs raster maps from $\mathbf{z}_G$ using a convolutional architecture [68]. For object primitives, the **object decoder** $\mathcal{D}_O$ is based on the DETR [12] Transformer. The latent $\mathbf{z}_O$ is patchified into $h' \times w'$ tokens, which are linearly projected and augmented with positional embeddings. These serve as keys and values in the decoder’s cross-attention layers. For each category, $\mathcal{D}_O$ uses a fixed number of learnable *object queries*, yielding $\mathbf{Q} \in \mathbb{R}^{N \times d_L}$, which are transformed into output embeddings. These are grouped by category and passed through category-specific Feed Forward Networks (FFNs) to predict 3D center locations $\hat{\mathbf{t}}_i$, Cholesky parameters $\hat{\mathbf{c}}_i$, and existence probabilities $\hat{p}_i$ for all primitives.

Training. Our first-stage training objective combines raster and vector losses with a KL divergence term over the joint latent (see supplementary material for a detailed formulation). For **ground raster maps**, we apply an L1 loss on height maps, averaged over occupied pixels, and a binary cross-entropy (BCE) loss on occupancy masks. For **object primitives**, predicted and ground-truth instances are first matched within each category via the Hungarian algorithm [36], minimizing a pairwise cost consisting of (i) BCE loss on existence probabilities, (ii) L1 loss on center locations, and (iii) L1 loss on Cholesky parameters. After matching, the corresponding losses are computed per category, normalized by the number of real ground-truth instances in the batch, and then averaged across the batch.

Layout Reconstruction. We recover the 3D layout by fusing per-class height maps into a single height field, which is extruded into a triangle mesh. Object primitives with existence probability above a threshold are placed using their predicted centers and Cholesky parameters. This enables handling both sparse and dense scenes while maintaining a fixed-size prediction set. As with the ground layout, object primitives are meshed for visualization. Additional implementation details are provided in the supplementary material.

3.3. Diffusion Transformer and Applications

The second stage trains a LDM [68] on the joint LVAE latent space. A RePaint-inspired [47] latent manipulation strategy

enables further downstream applications without fine-tuning. **Training.** Joint layout latent codes are extracted from the frozen LVAE and used to train a diffusion model $\epsilon_\theta(\mathbf{z}_L^t, t, y)$, where y is a scene label controlling vegetation density (low, medium, or high). Although we focus on vegetation, this conditioning generalizes to other object categories and their combinations (see supplementary material). Following [59], we adopt a Transformer backbone and incorporate y via adaLN-Zero blocks [61]. The model is trained to predict Gaussian noise using a mean squared error (MSE) loss.

Scene Inpainting. RePaint [47] introduces a modified diffusion sampling scheme for inpainting that synchronizes the denoising of unknown pixels with the noised version of known pixels. We apply this strategy in the 2D latent space of our pre-trained DiT to enable 3D layout editing. A binary latent mask specifies editable regions, allowing diverse edits such as inpainting the left half of a scene while keeping the right side intact (see Fig. 1 (3a)). Our disentangled latent structure naturally supports ground-conditioned object generation, where only object channels are inpainted while ground channels remain fixed (see Fig. 1 (3b)).

Scene Outpainting. We extend a 3D layout block beyond its original spatial extent using a sliding-window approach that leverages the same manipulation strategy as inpainting. To preserve spatial consistency, each newly generated block is designed to partially overlap by 50% with its adjacent block. The scene is first expanded in parallel along the four cardinal directions, followed by a second pass to fill the corner regions. This process can be iteratively applied to produce arbitrarily large 3D scenes.

Additional details on the diffusion objective and its applications are provided in the supplementary material.

4. Experiments

Dataset. Although our method can be trained on any dataset with 3D primitive annotations for arbitrary semantic categories, we focus on the underexplored urban setting and conduct our main experiments on KITTI-360 [41]. KITTI-360 annotations cover entire urban environments rather than only selected object classes, resulting in many primitives per scene and a demanding evaluation setting. We remove rare labels and group the remaining ones by semantic and scale similarity into 16 categories. Each layout covers a $64m \times 64m$ area centered on the ego vehicle. To minimize spatial overlap between training and test data, we thoroughly split the dataset and discard frames with incomplete annotations, yielding 61,913 training and 1,233 test poses. Split details and additional results on Argoverse 2 [86] are provided in the supplementary material.

Baselines. As the first primitive-based method for urban 3D semantic scene generation, we compare against recent voxel-based baselines: SemCity [38], PDD [44], and XCube [64]. SemCity employs a two-stage pipeline, training a triplane

Method	Resolution	Size (MB) ↓	IoU↑	mIoU↑
SemCity [38]	$64^2 \times 8$	0.12	82.19	50.52
	$128^2 \times 16$	1.00	90.38	68.98
	$256^2 \times 32$	8.00	81.75	70.52
SemCity-1M	$256^2 \times 32$	8.00	97.69	93.81
XCube [64]	$64^2 \times 8$	0.051	82.76	35.52
	$256^2 \times 32$	3.05	99.97	79.47
PrITTI (voxelized)	$256^2 \times 32$	2.52	90.58	70.27

Table 1. **Stage 1: Voxel reconstruction results** comparing IoU, mIoU, and mean per-sample memory for voxel-based baselines and voxelized PrITTI. Despite being natively primitive-based (and hence at a disadvantage), PrITTI remains competitive on these metrics. **Best**, **second-best**, and **third-best** results refer to the same (finest) resolution, where native size comparisons are meaningful.

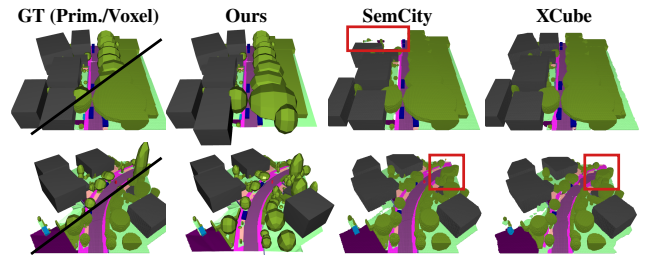


Figure 3. **Stage 1: Qualitative reconstruction results** on the same test scenes shown in each method’s native representation: primitives for PrITTI and voxel grids for SemCity and XCube. Voxel-based methods sometimes yield incomplete geometry and grid-induced distortions, such as vertical clipping at tall primitives.

autoencoder (AE) followed by a triplane diffusion model. We adopt its setup and voxelize our primitive scenes into a $256^2 \times 32$ grid. PDD [44] and XCube [64] are hierarchical voxel methods that refine scenes across scales. PDD applies discrete diffusion over a three-level pyramid ($32^2 \times 4$, $64^2 \times 8$, $256^2 \times 16$), while XCube uses a VAE-diffusion paradigm across two hierarchy levels ($64^2 \times 8$, $256^2 \times 32$).

Implementation Details. We implement the raster AE and LDM using standard VAE [68] and DiT [59] backbones from Diffusers [83]. Ground raster maps have resolution $H = W = 256$, and during encoding, we use a down-sampling factor $d = 8$ and latent channel dimension $c = 32$. Both the object encoder and decoder use six Transformer layers, the latter processing $N = 514$ object queries. The diffusion model is based on the DiT-B backbone with DDPM [28] noise scheduling. Additional details are provided in the supplementary material. Next, we evaluate scene reconstruction (Sec. 4.1), followed by the full generative model (Sec. 4.2).

4.1. Stage 1: 3D Semantic Scene Reconstruction

Metrics. To evaluate **3D scene reconstruction**, we compute standard voxel-based metrics [38], including Intersection-over-Union (IoU) and mean IoU (mIoU). IoU is evaluated

Method	Raster ($\times 10^{-2}$)		Primitive	
	MSE $\downarrow$	IoU $\uparrow$	AP3D $\uparrow$	AP3D@50 $\uparrow$
Default	0.75	99.96	62.12	46.96
w/o latent split	3.55	99.70	53.78	39.09
w/o objects	0.59	99.96	—	—
w/o ground	—	—	60.28	46.37

Table 2. **LVAE ablations** for latent split and joint training.

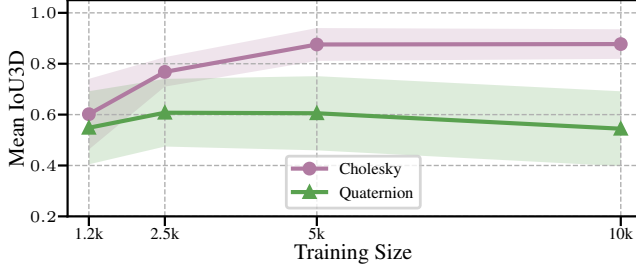


Figure 4. **Cholesky vs. quaternion** encodings across training sizes.

on a binary occupancy grid (foreground vs. background), while mIoU is averaged over all classes excluding the empty one. As PDD lacks a VAE component, we report results for SemCity, XCube, and our voxelized primitive reconstructions. Note that PrITTI is inherently primitive-based and hence at a disadvantage in this evaluation. However, it also supports standard 3D object detection metrics [8]. Specifically, we report the mean AP_{3D} across all object categories (excluding $\mathcal{VE}$) and over a range of IoU3D [63] thresholds $\tau \in \{0.05, \dots, 0.50\}$, along with AP_{3D}^{50} at $\tau = 0.50$. For ground raster maps, we measure IoU for occupancy masks and MSE for height maps reconstruction.

Primitives vs. Voxels. Our method combines ground raster maps with primitives, yielding an edit-friendly but also more compact and memory-efficient representation than voxel-based baselines. As shown in Tab. 1, it requires only 2.52 MB per scene (2.50 MB for rasters, 0.02 MB for primitives) versus 8.00 MB for SemCity and 3.05 MB for XCube at their finest resolution. By default, during AE training, SemCity’s decoder is evaluated on 400K sampled query points. We found that this limits performance on our voxelized dataset that preserves full volumetric geometry (e.g. entire building volumes instead of outlines). We hence also evaluated an improved version with 1M query points, which greatly improved reconstruction (mIoU 70.52 $\rightarrow$ 93.81). We refer to this configuration as SemCity-1M. To reduce memory usage, we also train our baselines at lower resolutions and upsample the outputs to $256^2 \times 32$. This reduces per-scene memory requirements but causes a sharp drop in reconstruction quality (see Tab. 1). While upsampled models are more memory-efficient and sometimes comparable to ours (e.g. SemCity at $128^2 \times 16$), they cannot scale to higher resolutions, where finer grids demand prohibitive memory or incur severe upsampling artifacts. This highlights a clear trade-off between memory requirements and reconstruction

quality in voxel-based approaches. In contrast, our representation is resolution-independent and achieves comparable reconstruction quality with constant, low memory, despite voxelization of our scenes introducing additional errors not present in voxel-native baselines. Importantly, as will be demonstrated in Sec. 4.2, our primitive-based model exhibits superior generation capabilities, benefiting from its compact and semantically-structured 3D scene representation.

Ablations. We analyze autoencoder design and object parameterization variants in Tab. 2 and Fig. 4: **(i) Cholesky vs. quaternions:** We compare our Cholesky-based encoding to a quaternion-based alternative on synthetic samples, each containing one zero-centered vehicle primitive with random scale and yaw. We vary the training set size while keeping the test set fixed across all settings (see supplementary material for details). As shown in Fig. 4, our parameterization consistently achieves higher mean IoU3D and greater stability, while the quaternion baseline plateaus and even degrades with more data, likely due to sign ambiguity causing loss discontinuities. **(ii) Latent split:** In this ablation (see Tab. 2 “w/o latent split”), both decoders operate on the joint latent $\mathbf{z}_L$ instead of their respective components $\mathbf{z}_G$ and $\mathbf{z}_O$. This modification degrades performance across all metrics (e.g. AP3D: 62.12 $\rightarrow$ 53.78; MSE: 0.0075 $\rightarrow$ 0.0355), suggesting each decoder benefits from domain-specific features. **(iii) Joint vs. separate training:** We compare joint training of ground and object branches in a shared latent space against training them independently. In the default setup, a single latent is sampled from the joint distribution and split during decoding, whereas the ablated models (see Tab. 2 “w/o objects” and “w/o ground”) use no shared latent or joint optimization. Although separate training slightly improves height map MSE (0.0059 vs. 0.0075), joint training achieves higher AP3D (62.12 vs. 60.28) by promoting better semantic alignment between branches, which enables context-aware object placement (e.g. cars on road).

4.2. Stage 2: 3D Semantic Scene Generation

Metrics. We evaluate **3D scene generation** using Precision (Prec.) [37], Recall (Rec.) [37], Fréchet Inception Distance (FID) [26], and Inception Score (IS) [70], measuring fidelity and diversity. Following [57, 77], both the reference and generated 3D scenes are rendered into 256^2 top-down semantic maps. As standard practice [19, 28, 32, 33], we use the training set for reference and generate an equal number of outputs using 250 DDPM sampling steps. For PrITTI, we balance the conditioning labels by sampling an equal number of scenes per class (low, medium, and high) without classifier-free guidance, while all baseline methods are unconditional. We found Prec. and Rec. to be sensitive to factors such as the evaluation sample size, the diversity of reference scenes, and the neighborhood size used for manifold estimation. In practice, we select 1K reference samples

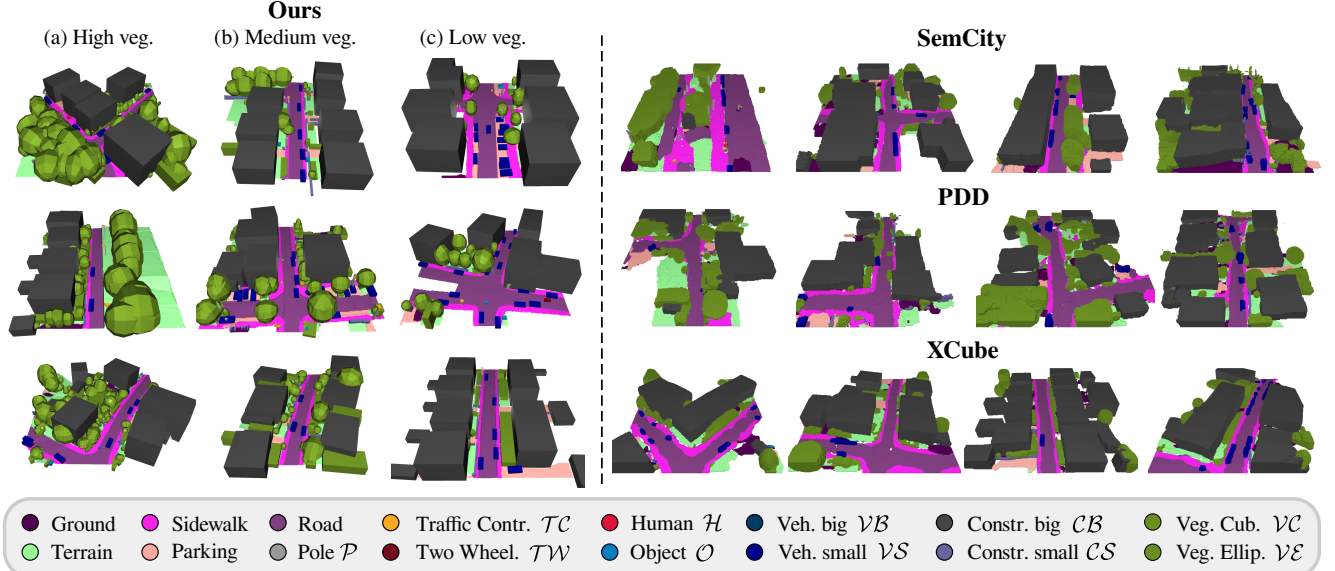


Figure 5. **3D Semantic Scene Generation.** Comparison of 3D semantic layouts generated by PrITTI (Ours, left) and voxel-based baselines (SemCity, PDD, XCube, right). PrITTI enables controllable generation, here conditioned on vegetation density (low, medium, or high), and produces more realistic, well-shaped scenes with clearer object boundaries. All baseline samples are generated unconditionally.

Method	Variant	Prec. $\uparrow$	Rec. $\uparrow$	FID $\downarrow$	IS $\uparrow$	Time (s) $\downarrow$
SemCity [38]	1M	0.133	0.063	177.623	3.454	2.35
PDD [44]	Level 1	0.000	0.000	302.256	1.558	0.94
	Level 2	0.182	0.000	166.086	1.896	1.89
	Level 3	0.156	0.021	159.878	2.711	12.84
XCube [64]	Level 1	0.243	0.008	140.455	2.397	1.41
	Level 2	0.482	0.230	94.822	3.480	3.50
PrITTI (Ours)	DiT-B	0.712	0.491	73.952	3.856	0.58
	DiT-L	0.807	0.545	71.028	4.210	1.78
	DiT-XL	0.812	0.616	71.735	3.964	2.11

Table 3. **Generation Results.** Comparison of PrITTI and baselines across generative metrics and mean generation time per scene.

that are maximally spaced apart and use a neighborhood size of $k = 3$. Additional evaluation settings and results are provided in the supplementary material.

Results. For fair comparison, we compute all generative metrics using the same reference samples across methods. For hierarchical baselines, each level is evaluated on the same set of generated scenes, where outputs from coarser levels are upsampled to match the finest resolution. As shown in Tab. 3, PrITTI significantly outperforms all baselines. We further evaluate larger diffusion model variants, DiT-L and DiT-XL, trained with the same configuration and total compute as DiT-B. Scaling to DiT-L yields notable improvements, while DiT-XL offers marginal gains. Regarding SemCity, we observe a large gap between reconstruction and generation quality, partly due to class-biased query point sampling during training and reconstruction versus uniform sampling during generation. This leads to overprediction of frequent classes seen in training, thereby reducing precision and recall. Moreover, its triplane-based diffusion design poses challenges in maintaining semantic consistency across planes, whereas

our unified 2D latent space effectively mitigates these issues. Similarly, PDD shows low generative metrics across hierarchy levels. Coarser stages produce overly simplified outputs lacking fine structural detail, while the final level introduces additional noise and semantic inconsistencies, even after two weeks of training the corresponding diffusion model, as in the original setup. Furthermore, the split-scene generation strategy used at this level, where four $128^2 \times 16$ blocks are synthesized to form a $256^2 \times 16$ scene, may further complicate learning global consistency. Among the baselines, XCube attains the best generative performance. Qualitative results further highlight PrITTI’s superior generation quality over baselines. As shown in Fig. 5, PrITTI produces diverse and coherent scenes with well-shaped, clearly separated objects, while also supporting controllable generation. In contrast, baseline methods often yield irregular shapes, indistinct object boundaries, and semantically inconsistent regions. Their scenes generally lack fine structural detail and occasionally appear fragmented, with disjoint areas or missing surface regions. Additional 3D and BEV-rendered results for all methods are included in the supplementary material, along with representative examples of low-quality generations that help contextualize the quantitative metrics.

Tab. 3 also reports the mean generation time per scene, averaged over 100 samples for all methods using the same number of diffusion timesteps. For hierarchical approaches, the reported time is cumulative across levels. Consequently, PDD and XCube exhibit longer generation times (12.84 seconds and 3.5 seconds, respectively) due to their multi-stage processing. Overall, PrITTI achieves the fastest generation (0.58), outperforming all baselines, including SemCity

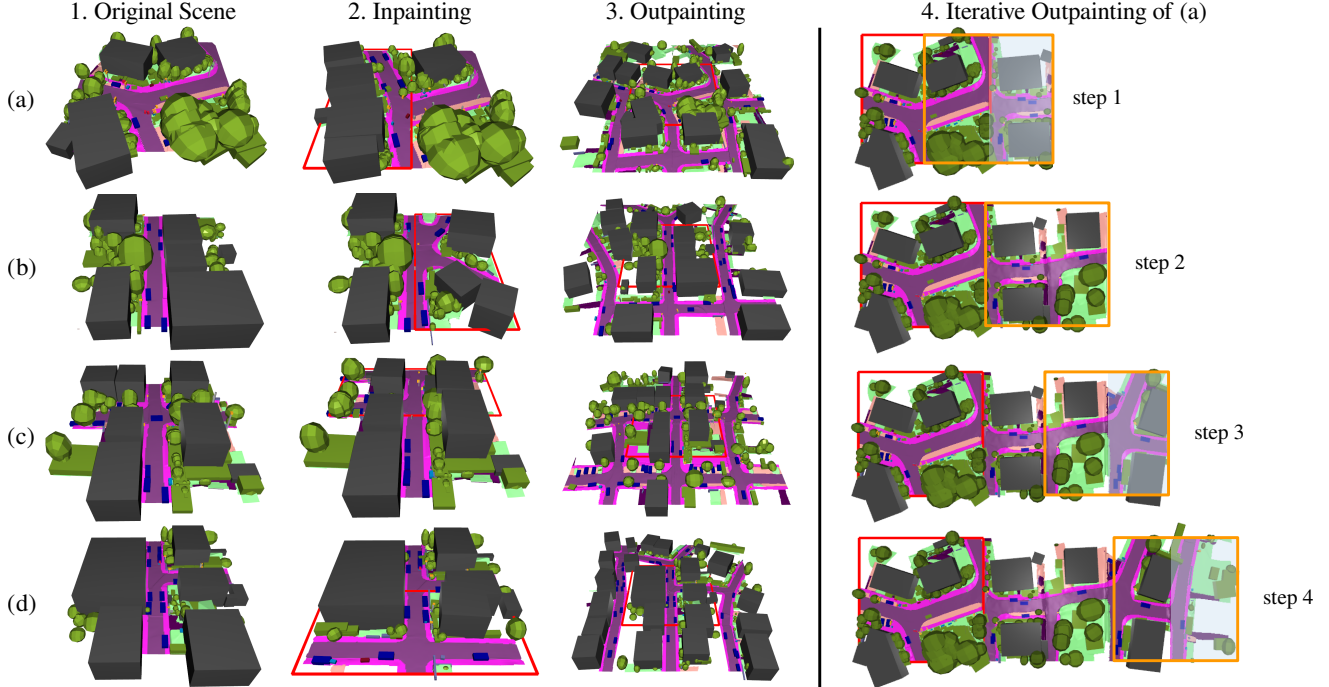


Figure 6. **Scene-level Editing.** (Left) Inpainting and outpainting results. Red outlines indicate the inpainted region (second column) or the original scene area (third column). (Right) An iterative outpainting example where the scene is progressively extended to the right. The red outline highlights the original scene boundary, while the orange denotes the current sliding-window region being generated at each step.

(2.35), which may be attributed to its more compact latent representation compared to SemCity’s triplane-based design.

4.3. Downstream Applications

Scene-level Editing. As illustrated in Fig. 6, our method supports both inpainting and outpainting of 3D layouts. Inpainting results show that localized edits are achieved while preserving global structure, whereas outpainting results demonstrate that scenes are extended with diverse road layouts and coherently placed primitives. Our sliding-window mechanism (Fig. 6, right) progressively expands the scene by overlapping with the previously generated block, maintaining overlapping regions while synthesizing new content in the non-overlapping space. By iteratively applying this process, our model can generate large-scale, diverse, and semantically coherent urban layouts without additional training.

Object-level Editing. Our instance-level primitive representation enables intuitive object-wise editing of 3D scenes through direct geometric operations. As shown in Fig. 1, primitives can be easily modified, such as being rotated or removed. Unlike voxel grids, which lack explicit object instances and require costly re-generation for simple edits, our approach supports direct manipulation by updating the corresponding primitive parameters (e.g. center for translation).

Photo-realistic Street-view Generation. Given a generated 3D scene, we extract the road centerline from the binary road mask, sample camera viewpoints along it, and render 2D semantic maps. These maps are then used to condition a

ControlNet [104] fine-tuned on KITTI-360 [45]. As shown in Fig. 1, this enables realistic view synthesis consistent with the input semantics. Despite the coarse primitive geometry, our layouts effectively guide image generation, producing diverse object shapes beyond simple cuboids and ellipsoids. Additional results for all applications, including large-scale scene extrapolation, are provided in the supplementary.

5. Conclusion

We introduced PrITTI, the first framework for large-scale 3D semantic urban scene generation using vectorized object primitives and rasterized ground surfaces. Our latent diffusion model generates high-quality, diverse, and controllable layouts while further supporting downstream tasks without fine-tuning. Compared to voxel-based baselines, PrITTI achieves similar reconstruction quality with lower memory requirements and stronger generative performance. In addition, unlike voxel grids, its instance-level primitive representation enables intuitive object-level edits.

Limitations. In future work, we plan to address appearance synthesis with task-specific rendering pipelines tailored to our layouts [45]. Learning more expressive primitives or deformable shapes promises to enhance geometric and visual fidelity [2, 58]. Finally, extending our approach to dynamic environments will pave the way toward autonomous driving simulations and diverse adversarial scenario generation [23].

Acknowledgments

Andreas Geiger was supported by the ERC Starting Grant LEGO-3D (850533) and the DFG EXC number 2064/1 - project number 390727645. Christina Ourania Tze and Daniel Dauner were supported by the German Federal Ministry for Economic Affairs and Energy within the project NXT GEN AI METHODS (19A23014S). We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Daniel Dauner. Christina Ourania Tze acknowledges her membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program.

References

- [1] 3d karton city model. <https://www.turbosquid.com/3d-models/3d-karton-city-2-model-1196110>. 20
- [2] Mehmet Aygun and Oisín Mac Aodha. Saor: Single-view articulated object reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 8
- [3] Sherwin Bahmani, Jeong Joon Park, Despoina Paschalidou, Xingguang Yan, Gordon Wetzstein, Leonidas J. Guibas, and Andrea Tagliasacchi. CC3D: layout-conditioned generation of compositional 3d scenes. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2
- [4] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 20
- [5] Irving Biederman. Human image understanding: Recent research and a theory. *Computer vision, graphics, and image processing*, 1985. 2
- [6] Thomas Binford. Visual perception by computer. In *Proc. IEEE Conf. on Systems and Control*, 1975, 1975. 2
- [7] Alexey Bokhovkin, Quan Meng, Shubham Tulsiani, and Angela Dai. Scenefactor: Factored latent 3d diffusion for controllable 3d scene generation. *arXiv.org*, 2412.01801, 2024. 2
- [8] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3d: A large benchmark and model for 3d object detection in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 6
- [9] Rodney A Brooks, Russell Creiner, and Thomas O Binford. The acronym model-based vision system. In *Proc. of the International Joint Conf. on Artificial Intelligence (IJCAI)*, 1979. 2
- [10] 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 multi-modal dataset for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 23, 24
- [11] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric M. Wolff, Alex H. Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021. 2, 23
- [12] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 4, 16
- [13] Minglin Chen, Longguang Wang, Sheng Ao, Ye Zhang, Kai Xu, and Yulan Guo. Layout2scene: 3d semantic layout-guided scene generation via geometry and appearance diffusion priors. *arXiv.org*, 2501.02519, 2025. 2
- [14] Kashyap Chitta, Daniel Dauner, and Andreas Geiger. Sledge: Synthesizing driving environments with generative models and rule-based traffic. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 3
- [15] A. Cholesky. Sur la résolution numérique des systèmes d'équations linéaires. *Bulletin Géodésique*, 1924. 2, 3, 15
- [16] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [17] Jie Deng, Wenhao Chai, Jianshu Guo, Qixuan Huang, Wenhao Hu, Jenq-Neng Hwang, and Gaoang Wang. Citygen: Infinite and controllable 3d city layout generation. *arXiv.org*, 2312.01508, 2023. 3
- [18] Helisa Dharmo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3
- [19] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 6, 22
- [20] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2
- [21] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2
- [22] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010. 2
- [23] Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 8
- [24] Zekun Hao, Hadar Averbuch-Elor, Noah Snaveley, and Serge J. Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [25] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing implicit shapes

through part aware generation. *ACM Trans. on Graphics*, 2022. 2

- [26] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 6
- [27] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv.org*, 2207.12598, 2022. 20
- [28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 5, 6, 18
- [29] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2
- [30] Zilong Huang, Jun He, Junyan Ye, Lihan Jiang, Weijia Li, Yiping Chen, and Ting Han. Scene4u: Hierarchical layered 3d scene reconstruction from single panoramic image for your immerse exploration. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3
- [31] Max Jiang, Yijing Bai, Andre Cornman, Christopher Davis, Xiukun Huang, Hong Jeon, Sakshum Kulshrestha, John Lambert, Shuangyu Li, Xuanyu Zhou, et al. Scenediffuser: Efficient and controllable driving simulation initialization and rollout. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3
- [32] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6
- [33] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 6
- [34] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014. 2, 17
- [35] Juil Koo, Seungwoo Yoo, Minh Hieu Nguyen, and Minhyuk Sung. Salad: Part-level latent diffusion for 3d shape generation and manipulation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2
- [36] Harold W Kuhn. The hungarian method for the assignment problem. 1955. 4, 16
- [37] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 6, 22
- [38] Jumin Lee, Sebin Lee, Changho Jo, Woobin Im, Juhyeong Seon, and Sung-Eui Yoon. Semcity: Semantic scene generation with triplane diffusion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 5, 7, 20, 21, 24, 31, 39
- [39] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Trans. on Graphics*, 2017. 2
- [40] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao (Richard) Zhang, and Leonidas J. Guibas. GRASS: generative recursive autoencoders for shape structures. *ACM Trans. on Graphics*, 36(4), 2017. 2
- [41] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2022. 2, 5, 14, 24
- [42] Chenguo Lin and Yadong Mu. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2024. 3
- [43] Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. Infinicity: Infinite-scale city synthesis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 3
- [44] Yuheng Liu, Xinke Li, Xueting Li, Lu Qi, Chongshou Li, and Ming-Hsuan Yang. Pyramid diffusion for fine 3d large scene generation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 2, 3, 5, 7, 19, 20, 21, 24, 33
- [45] Fan Lu, Kwan-Yee Lin, Yan Xu, Hongsheng Li, Guang Chen, and Changjun Jiang. Urban architect: Steerable 3d urban scene generation with layout prior. *arXiv.org*, 2404.06780, 2024. 2, 8, 22
- [46] Jack Lu, Kelvin Wong, Chris Zhang, Simon Suo, and Raquel Urtasun. Scenecontrol: Diffusion for controllable traffic scene generation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2024. 3, 24
- [47] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 4, 5, 18
- [48] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B. Tenenbaum. End-to-end optimization of scene layout. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3
- [49] Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. Lt3sd: Latent trees for 3d scene diffusion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2
- [50] Lu Mi, Hang Zhao, Charlie Nash, Xiaohan Jin, Jiyang Gao, Chen Sun, Cordelia Schmid, Nir Shavit, Yuning Chai, and Dragomir Anguelov. Hdmapgen: A hierarchical graph generative model of high definition maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4227–4236, 2021. 3
- [51] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenets: Hierarchical graph networks for 3d shape generation. In *ACM Trans. on Graphics*, 2019. 2
- [52] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenets: Hierarchical graph networks for 3d shape generation. *ACM Trans. on Graphics*, 2019. 2
- [53] Chengjie Niu, Jun Li, and Kai Xu. Im2struct: Recovering 3d shape structure from a single RGB image. In *Proc. IEEE*

Conf. on Computer Vision and Pattern Recognition (CVPR), 2018. 2

- [54] Hao Ouyang, Tiancheng Sun, Stephen Lombardi, and Kathryn Heal. Text2immersion: Generative immersive scene with 3d gaussians. *arXiv.org*, 2312.09242, 2023. 3
- [55] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [56] Despoina Paschalidou, Luc van Gool, and Andreas Geiger. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [57] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3, 6
- [58] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8
- [59] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 5, 18, 22
- [60] Alex Pentland. Parts: Structured descriptions of shape. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 1986. 2
- [61] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2018. 5
- [62] Ryan Po and Gordon Wetzstein. Compositional 3d scene generation using locally conditioned diffusion. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2024. 2
- [63] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv.org*, 2007.08501, 2020. 6
- [64] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 5, 7, 20, 21, 24, 35
- [65] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3
- [66] Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3
- [67] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963. 2
- [68] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4, 5, 18
- [69] Luke Rowe, Roger Girgis, Anthony Gosselin, Liam Paull, Christopher Pal, and Felix Heide. Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. *arXiv.org*, 2503.22496, 2025. 3
- [70] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 6
- [71] Katja Schwarz, Denis Rozumny, Samuel Rota Buló, Lorenzo Porzi, and Peter Kotschieder. A recipe for generating 3d worlds from a single image. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2025. 3
- [72] Jaidev Shriram, Alex Trevithick, Lingjie Liu, and Ravi Ramamoorthi. Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2025. 3
- [73] Liangchen Song, Liangliang Cao, Hongyu Xu, Kai Kang, Feng Tang, Junsong, Yuan, and Yang Zhao. Roomdreamer: Text-driven 3d indoor scene synthesis with coherent geometry and texture. *arXiv.org*, 2305.11337, 2023. 2
- [74] Pei Sun, Henrik Kretschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 20
- [75] Shuo Sun, Zekai Gu, Tianchen Sun, Jiawei Sun, Chengran Yuan, Yuhang Han, Dongen Li, and Marcelo H Ang. Drivescenenet: Generating diverse and realistic driving scenarios from scratch. *IEEE Robotics and Automation Letters (RA-L)*, 2024. 3
- [76] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3
- [77] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 6
- [78] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2
- [79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017. 4

- [80] Vaibhav Vavilala and David Forsyth. Convex decomposition of indoor scenes. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2
- [81] Vaibhav Vavilala, Seemantdar Jain, Rahul Vasanth, Anand Bhattad, and David Forsyth. Blocks2world: Controlling realistic scenes with editable primitives. *arXiv.org*, 2307.03847, 2023.
- [82] Vaibhav Vavilala, Florian Kluger, Seemantdar Jain, Bodo Rosenhahn, and David Forsyth. Improved convex decomposition with ensembling and boolean primitives. *arXiv.org*, 2405.19569, 2024. 2
- [83] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, Steven Liu, William Berman, Yiyi Xu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. 5
- [84] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2021. 3
- [85] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. Lego-net: Learning regular rearrangements of objects in rooms. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3
- [86] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 2, 5, 23, 24, 26
- [87] Joey Wilson, Jingyu Song, Yuewei Fu, Arthur Zhang, Andrew Capodieci, Paramsothy Jayakumar, Kira Barton, and Maani Ghaffari. Motionsc: Data set and network for real-time semantic mapping in dynamic environments. *IEEE Robotics and Automation Letters (RA-L)*, 2022. 20
- [88] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, and Pan Ji. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. In *ACM Transactions on Graphics*, 2024. 2
- [89] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. GaussianCity: Generative gaussian splatting for unbounded 3d city generation. *arXiv.org*, 2406.06526, 2024. 3
- [90] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. CityDreamer: Compositional generative model of unbounded 3D cities. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [91] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. CityDreamer4D: Compositional generative model of unbounded 4D cities. *arXiv.org*, 2501.08983, 2025. 3
- [92] Ziyang Xie. Worldgen: Generate any 3d scene in seconds. <https://github.com/ZiYang-xie/WorldGen>, 2025. 3
- [93] Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skokhodov, Aliaksandr Siarohin, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, et al. Discoscene: Spatially disentangled generative radiance fields for controllable 3d-aware scene synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [94] Shuai Yang, Jing Tan, Mengchen Zhang, Tong Wu, Yixuan Li, Gordon Wetzstein, Ziwei Liu, and Dahua Lin. Layerpano3d: Layered 3d panorama for hyper-immersive scene generation. *ACM Trans. on Graphics*, 2025. 3
- [95] Xiuyu Yang, Yunze Man, Junkun Chen, and Yu-Xiong Wang. Scenecraft: Layout-guided 3d scene generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 2
- [96] Yuanbo Yang, Yifei Yang, Hanlei Guo, Rong Xiong, Yue Wang, and Yiyi Liao. Urbangiraffe: Representing urban scenes as compositional generative neural feature fields. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 2
- [97] Yuanbo Yang, Jiahao Shao, Xinyang Li, Yujun Shen, Andreas Geiger, and Yiyi Liao. Prometheus: 3d-aware latent diffusion models for feed-forward text-to-3d scene generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3
- [98] Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T Freeman, Forrester Cole, Deqing Sun, Noah Snaveley, Jiajun Wu, et al. Wonderjourney: Going from anywhere to everywhere. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3
- [99] Mark YU, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2025.
- [100] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv.org*, 2409.02048, 2024. 3
- [101] Guangyao Zhai, Evin Pinar Örneke, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. Commonsenses: Generating commonsense 3d indoor scenes with scene graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3
- [102] Guangyao Zhai, Evin Pinar Örneke, Dave Zhenyu Chen, Ruotong Liao, Yan Di, Nassir Navab, Federico Tombari, and Benjamin Busam. Echoscene: Indoor scene generation via information echo over scene graph diffusion. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 3
- [103] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *arXiv.org*, 2305.11588, 2023. 3
- [104] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. 8, 22, 44
- [105] Haowen Zheng and Yanyan Liang. Sseditor: Controllable mask-to-scene generation with diffusion model. *arXiv.org*, 2411.12290, 2024. 2, 3, 24

- [106] Yunsong Zhou, Naisheng Ye, William Ljungbergh, Tianyu Li, Jiazhi Yang, Zetong Yang, Hongzi Zhu, Christoffer Petersson, and Hongyang Li. Decoupled diffusion sparks adaptive scene generation. *arXiv.org*, 2504.10485, 2025. [3](#)
- [107] Chuhan Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 900–909, 2017. [2](#)

PrITTI: Primitive-based Generation of Controllable and Editable 3D Semantic Urban Scenes

Supplementary Material

In this supplementary material, we provide additional details on dataset preprocessing, implementation, and experimental setup. We also present further quantitative results evaluating generative performance and additional qualitative examples for all applications discussed in the main paper. Our [supplementary video](#) offers a brief motivation and overview of our method as well as extended visual results.

A. Dataset Details

In this section, we describe our dataset construction and the key procedures used to analyze it.

Dataset Construction. KITTI-360 [41] defines 37 labels, which we merge into 16 semantic categories, covering all major urban scene components. As an initial step, we analyze the dataset statistics and remove a few extremely rare labels including tunnel, bridge, train, stop, and caravan. Among the remaining ones, five correspond to ground-related classes (road, terrain, sidewalk, parking, and ground), which we treat as distinct ground categories. The rest are grouped into 11 object categories based on both semantic and physical scale similarity, the latter being important for accurate primitive reconstruction. A complete mapping of semantic classes to object categories is provided in Tab. 4. For detailed definitions of individual classes, we refer readers to [41].

For each ego-vehicle pose, we adopt its IMU coordinate system as the local reference frame and define a $64\text{ m} \times 64\text{ m}$ 3D layout centered on the vehicle. The field of view (FOV) extends 64 m in the forward direction and 32 m laterally on each side. To avoid including distant or marginally visible objects, we retain only primitives whose 3D centers lie within this FOV and ground polygons with at least one vertex contained in it. Poses without any primitive annotations (typically at sequence boundaries) are excluded. The provided KITTI-360 test poses are sequentially distributed over limited areas of each sequence, covering only a narrow spatial extent. In short sequences, they typically span one continuous region, while in longer sequences, they may occupy two disjoint areas. Consequently, the original split insufficiently represents the full spatial diversity of the scenes.

To address this, we construct a custom train-test split. We first compute the total driving distance across all KITTI-360 sequences and divide the full trajectory into 100 equal-distance segments. Within each segment, the majority of poses are assigned to the training set, while a 15 m region near the end is reserved for testing. To minimize spatial overlap with training poses, we exclude 5 m margins on either side of the test regions. Furthermore, we remove any

test pose located within 5 m in the Bird’s Eye View (BEV) of a training pose, and vice versa. This prevents overlap caused by the ego-vehicle revisiting the same location from opposite directions. The final split comprises 61,913 training poses and 1,233 test poses, ensuring broad and non-overlapping spatial coverage across the entire dataset.

Primitive Count Selection. We adopt a fixed-size set prediction strategy for object primitives rather than an autoregressive decoding scheme, as this offers clear advantages for large-scale 3D scenes. Predicting all primitives in parallel enables the model to handle hundreds of objects in a single forward pass, whereas an autoregressive decoder would require sequential sampling, leading to higher inference time and cumulative prediction errors. In addition, generating the entire scene concurrently avoids imposing an arbitrary ordering on objects, which is inconsistent with the inherently unordered nature of scene layouts. Accordingly, we design the object decoder $\mathcal{D}_O$ (see Sec. B.2) to infer a fixed-size set of N^c primitives for each category c . We determine N^c by analyzing the distribution of ground-truth primitive counts and selecting the 95th percentile as the cutoff. The resulting values for each category are listed in Tab. 4. During preprocessing, if the count of instances in a category exceeds its predefined threshold in a given scene, we retain only the N^c closest objects to the ego-vehicle and discard the farthest ones. This ensures that all scenes, regardless of their density, are processed. Importantly, not all decoded objects are necessarily retained in the final scene. Each predicted primitive is associated with an existence probability, and only those exceeding a predefined threshold are kept. This probabilistic filtering allows the model to naturally handle both sparse and dense scenes while maintaining a fixed-size set.

Scene Labeling. To train our class-conditional latent diffusion model (see Sec. B.3), each scene is assigned a discrete label reflecting its vegetation density. Since such annotations are not provided in the KITTI-360 dataset, we derive them automatically from dataset statistics. Specifically, for each scene, we compute the total number and volume of vegetation primitives, considering both cuboids and ellipsoids. A scene is then labeled as *low* if both quantities fall below their respective 25th percentile thresholds, *high* if both exceed the 75th percentile, and *medium* otherwise.

B. Methodological Details

This section describes the representations, architectural components, and implementation details of our method.

Object Category	Semantic Classes	Primitive Type	Predicted Count
vegetation cuboid ($\mathcal{VC}$)	vegetation	3D cuboid	178
vegetation ellipsoid ($\mathcal{VE}$)	vegetation	3D ellipsoid	159
vehicle big ($\mathcal{VB}$)	truck, bus	3D cuboid	2
vehicle small ($\mathcal{VS}$)	car, trailer, unknown vehicle	3D cuboid	18
two-wheelers ($\mathcal{TW}$)	motorcycle, bicycle	3D cuboid	6
human ($\mathcal{H}$)	rider, pedestrian	3D cuboid	5
construction big ($\mathcal{CB}$)	building, garage, unknown construction	3D cuboid	16
construction small ($\mathcal{CS}$)	wall, fence, guardrail, gate	3D cuboid	77
pole ($\mathcal{P}$)	small pole, big pole	3D cuboid	19
traffic control ($\mathcal{TC}$)	traffic light, traffic sign, lamp	3D cuboid	17
object ($\mathcal{O}$)	trashbin, box, unknown object	3D cuboid	17

Table 4. List of object categories defined in our method along with their corresponding semantic classes and annotated primitive types. The last column indicates the fixed number of primitives predicted per category, determined based on dataset statistics.

B.1. Cholesky Parameters

Standard quaternion-based encodings of 3D orientation suffer from sign ambiguity: both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation but differ numerically. This causes discontinuities in the loss landscape when optimized with standard regression objectives. Similarly, eigendecomposition-based encodings are ambiguous because eigenvectors are only defined up to sign (i.e. both $\mathbf{v}$ and $-\mathbf{v}$ are valid but numerically distinct), leading to comparable instability during gradient-based training. To overcome these issues, we propose a Cholesky-based parameterization that jointly encodes orientation and size in a continuous 6D representation.

Scatter Matrix Construction. To derive this representation, we first define a scatter matrix $\mathbf{S} \in \mathbb{R}^{3 \times 3}$ that describes the spatial extent of an object in 3D space. Let $\lambda_1, \lambda_2, \lambda_3 > 0$ denote the object’s scaling factors along its principal axes, and let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ be the orthonormal column vectors of its rotation matrix. Defining the orthonormal basis matrix $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3]$ and the diagonal scale matrix $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$, the scatter matrix is given by:

$$\mathbf{S} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top = \sum_{j=1}^3 \lambda_j \mathbf{v}_j \mathbf{v}_j^\top \quad (1)$$

Cholesky Decomposition. We apply the Cholesky decomposition [15] to the scatter matrix $\mathbf{S}$, yielding

$$\mathbf{S} = \mathbf{L} \mathbf{L}^\top \quad (2)$$

where $\mathbf{L} \in \mathbb{R}^{3 \times 3}$ is a lower-triangular matrix. The Cholesky parameters $\mathbf{c} \in \mathbb{R}^6$ are defined as the six non-zero entries of $\mathbf{L}$. This representation removes orientation ambiguities by operating on the sign-invariant scatter matrix $\mathbf{S}$: since each outer product $\mathbf{v}_j \mathbf{v}_j^\top$ is unchanged under sign flips $\mathbf{v}_j \mapsto -\mathbf{v}_j$, $\mathbf{S}$ remains invariant.

Scatter Matrix Properties. Additionally, we can show that $\mathbf{S}$ is symmetric and positive-definite:

Symmetric:

$$\mathbf{S}^\top = (\mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top)^\top = (\mathbf{V}^\top)^\top \mathbf{\Lambda}^\top \mathbf{V}^\top = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top = \mathbf{S} \quad (3)$$

since $\mathbf{\Lambda}$ is diagonal and therefore symmetric.

Positive-definite: Because $\mathbf{V}$ is orthonormal (i.e. $\mathbf{V}^\top = \mathbf{V}^{-1}$), $\mathbf{S}$ is similar to $\mathbf{\Lambda}$:

$$\mathbf{S} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^{-1} \quad (4)$$

and thus shares the same eigenvalues $\lambda_1, \lambda_2, \lambda_3 > 0$.

These properties guarantee a unique Cholesky decomposition [15] with a lower-triangular matrix $\mathbf{L}$ having strictly positive diagonal entries. This in turn defines a *unique* 6D representation $\mathbf{c} \in \mathbb{R}^6$, well-suited for regression-based learning.

Reconstruction. Given predicted Cholesky parameters $\hat{\mathbf{c}}$, we reshape them into a lower-triangular matrix $\hat{\mathbf{L}} \in \mathbb{R}^{3 \times 3}$ and recover the scatter matrix as $\hat{\mathbf{S}} = \hat{\mathbf{L}} \hat{\mathbf{L}}^\top$. Performing eigendecomposition of $\hat{\mathbf{S}}$ yields its eigenvalues $\hat{\lambda}_1, \hat{\lambda}_2, \hat{\lambda}_3$ and eigenvectors $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \hat{\mathbf{v}}_3$. These define the object’s scale matrix $\hat{\mathbf{\Lambda}} = \text{diag}(\hat{\lambda}_1, \hat{\lambda}_2, \hat{\lambda}_3)$ and rotation matrix $\hat{\mathbf{R}} = [\hat{\mathbf{v}}_1 \ \hat{\mathbf{v}}_2 \ \hat{\mathbf{v}}_3]$.

B.2. Layout Variational Autoencoder

In the first training stage, a Layout Variational Autoencoder (LVAE) learns to compress 3D semantic scene layouts into compact 2D latent representations suitable for subsequent latent diffusion modeling.

B.2.1. Architecture

The LVAE comprises separate encoder-decoder pairs for the ground and object modalities.

Ground Encoder. Our ground encoder $\mathcal{E}_g$ consists of stacked downsampling blocks that progressively reduce spatial resolution while increasing channel dimensionality. Each

block contains a residual unit followed by a strided convolution. Residual units include two convolutional layers with group normalization and SiLU activation. At the bottleneck, a self-attention layer captures global context, while additional residual blocks refine local features. The encoder takes as input the ground raster maps, $[\mathbf{H}; \mathbf{B}] \in \mathbb{R}^{256 \times 256 \times 10}$, and outputs a latent grid of shape $32 \times 32 \times 64$. This grid is split along the channel dimension into two chunks representing the mean $\mu_G \in \mathbb{R}^{32 \times 32 \times 32}$ and variance $\sigma_G \in \mathbb{R}^{32 \times 32 \times 32}$ of the ground latent representation.

Object Encoder. The object encoder $\mathcal{E}_O$ operates on a fixed-size set of object primitives, each represented by a 20D feature vector. These vectors are first projected to a 512D embedding space and then processed by a 6-layer Transformer encoder with 8 attention heads and a feedforward dimension of 1024. A subsequent linear projection layer reduces the output dimensionality to 64. The resulting per-object features are spatially arranged into a latent grid of shape $32 \times 32 \times 64$ using a scatter-mean operation. Similar to the ground branch, the grid is split along the channel dimension into the mean $\mu_O \in \mathbb{R}^{32 \times 32 \times 32}$ and variance $\sigma_O \in \mathbb{R}^{32 \times 32 \times 32}$ of the object latent representation.

Joint Layout Latent. We construct joint mean and variance tensors, $\mu_L, \sigma_L \in \mathbb{R}^{32 \times 32 \times 64}$, by concatenating the corresponding ground and object components along the channel axis. A joint layout latent representation $\mathbf{z}_L \in \mathbb{R}^{32 \times 32 \times 64}$ is then sampled via the reparameterization trick. This formulation maintains a disentangled latent structure, enabling straightforward separation into ground and object latents, $\mathbf{z}_G, \mathbf{z}_O \in \mathbb{R}^{32 \times 32 \times 32}$. We found this design beneficial for improving reconstruction fidelity and enabling further applications such as ground-conditioned primitive generation.

Ground Decoder. The ground decoder $\mathcal{D}_G$ processes the ground latent representation $\mathbf{z}_G$ through a self-attention layer followed by two residual units. The resulting features are progressively upsampled to the original spatial resolution of 256×256 using a cascade of upsampling blocks, each comprising two residual units and a transposed convolution. As the spatial resolution increases, the channel dimensionality is gradually reduced. A final convolutional layer maps the output to 10 channels, matching the dimensionality of the input ground raster maps, $[\hat{\mathbf{H}}; \hat{\mathbf{B}}] \in \mathbb{R}^{256 \times 256 \times 10}$.

Object Decoder. The object decoder $\mathcal{D}_O$ adopts a DETR [12] Transformer architecture with N learnable queries representing the total number of predicted primitives across all object categories. It consists of 6 Transformer decoder layers with 8 attention heads, a feedforward dimension of 1024, and no dropout. The queries interact with the object latent $\mathbf{z}_O$ through cross-attention. Specifically, $\mathbf{z}_O$ is first patchified with a patch size of 1, producing a sequence of 32×32 tokens augmented with sine positional embeddings. The resulting embeddings, each of dimension 512, are grouped by category and passed to category-specific

prediction heads.

Prediction Heads. Each object category employs a dedicated 3-layer feedforward network with a hidden size of 1024. For every query, the head outputs a 9D vector encoding the normalized 3D center location and 6D Cholesky parameters of its corresponding primitive. An additional linear layer predicts its probability of existence.

B.2.2. Training

The first-stage training loss consists of three components, described below.

Ground Loss. We supervise the reconstruction of input raster maps using: (i) a binary cross-entropy (BCE) loss for occupancy masks and (ii) an L1 loss for height maps, averaged over occupied pixels. Let $\mathbf{H} \in \mathbb{R}^{H \times W \times 5}$ and $\mathbf{B} \in \{0, 1\}^{H \times W \times 5}$ denote the ground-truth height maps and occupancy masks, and $\hat{\mathbf{H}}$ and $\hat{\mathbf{B}}$ their predicted counterparts. The ground loss is defined as:

$$\mathcal{L}_{\text{ground}} = \lambda_G^{\text{occ}} \cdot \mathcal{L}_{\text{BCE}}(\hat{\mathbf{B}}, \mathbf{B}) + \lambda_G^{\text{height}} \cdot \frac{1}{\sum \mathbf{B}} \|(\hat{\mathbf{H}} - \mathbf{H}) \odot \mathbf{B}\|_1 \quad (5)$$

where $\odot$ denotes element-wise multiplication. We set the loss weights $\lambda_G^{\text{occ}} = 1$ and $\lambda_G^{\text{height}} = 9$ to balance the relative contributions of the two terms.

Object Loss. The object decoder $\mathcal{D}_O$ predicts a fixed number of N^c primitives per object category c (see Tab. 4), leading to $N = \sum_{c=1}^{11} N^c$ total predictions per scene. Because the outputs are unordered, we establish one-to-one correspondences between predicted and ground-truth primitives using bipartite matching for each category independently. Specifically, let $\mathcal{O}^c = \{\mathbf{o}_i\}_{i=1}^{N^c}$ and $\hat{\mathcal{O}}^c = \{\hat{\mathbf{o}}_j\}_{j=1}^{N^c}$ denote the sets of ground-truth and predicted primitives for category c , respectively. Each primitive is parameterized as $\mathbf{o}_i = (\mathbf{t}_i, \mathbf{c}_i, p_i)$, where $\mathbf{t}_i \in \mathbb{R}^3$ is its 3D center location, $\mathbf{c}_i \in \mathbb{R}^6$ are the Cholesky parameters, and $p_i \in \{0, 1\}$ distinguishes valid primitives ($p_i = 1$) from padded placeholders ($p_i = 0$). We apply the Hungarian algorithm [36] to find the optimal assignment $\hat{\sigma}^c$ minimizing the total matching cost:

$$\hat{\sigma}^c = \arg \min_{\sigma^c \in \mathcal{P}_{N^c}} \sum_{i=1}^{N^c} \mathcal{L}_{\text{match}}(\mathbf{o}_i, \hat{\mathbf{o}}_{\sigma(i)}) \quad (6)$$

where $\mathcal{P}_{N^c}$ is the set of all permutations of N^c elements. The pairwise matching cost is defined as:

$$\begin{aligned} \mathcal{L}_{\text{match}}(\mathbf{o}_i, \hat{\mathbf{o}}_{\sigma(i)}) &= \lambda_O^{\text{prob}} \mathcal{L}_{\text{BCE}}(p_i, \hat{p}_{\sigma(i)}) \\ &\quad + \mathbb{1}_{[p_i > 0]} \lambda_O^{\text{center}} \|\mathbf{t}_i - \hat{\mathbf{t}}_{\sigma(i)}\|_1 \\ &\quad + \mathbb{1}_{[p_i > 0]} \lambda_O^{\text{chol}} \|\mathbf{c}_i - \hat{\mathbf{c}}_{\sigma(i)}\|_1 \end{aligned} \quad (7)$$

After obtaining the optimal assignments, we compute the object loss over all categories as:

$$\mathcal{L}_{\text{object}} = \sum_{c \in \mathcal{C}} \left(\sum_{i=1}^{N^c} \mathcal{L}_{\text{match}}(\mathbf{o}_i, \hat{\mathbf{o}}_{\hat{\sigma}^c(i)}) \right) \quad (8)$$

where $\mathcal{C}$ denotes the set of object categories. Each term in Eq. (8) is normalized by the number of real (i.e. non-padded) ground-truth primitives in the corresponding sample, and the normalized losses are averaged across the batch. The weights $\lambda_{\mathcal{O}}^{\text{prob.}}$, $\lambda_{\mathcal{O}}^{\text{center}}$, and $\lambda_{\mathcal{O}}^{\text{Chol.}}$ are configured differently for bipartite matching and for the final loss computation. For **matching**, we use category-agnostic constants: $\lambda_{\mathcal{O}}^{\text{prob.}} = 6$, $\lambda_{\mathcal{O}}^{\text{center}} = 3$, and $\lambda_{\mathcal{O}}^{\text{Chol.}} = 3$. This weighting emphasizes the BCE term to discourage pairing low-confidence predictions with true instances, while assigning equal importance to the center and Cholesky components for accurate geometric alignment. During **loss computation**, the weights vary across object categories according to their fixed number of predicted primitives. In particular, categories are grouped into low-, medium-, and high-count ranges, with higher weights assigned to higher-count groups. This adjustment compensates for normalization by the number of real primitives. Without it, high-count categories, typically containing more real instances per sample, would contribute less to the overall loss, leading to underfitting. Within each group, the center and Cholesky weights are balanced so that their corresponding losses have comparable magnitudes, ensuring that both geometric components contribute equally. The BCE term receives a smaller weight than the geometric terms, as existence confidence is already reinforced by the previous matching stage. Increasing its weight would disproportionately emphasize classification over geometry. The final group-specific weights are summarized in Tab. 5.

Group	Object Categories	$\lambda_{\mathcal{O}}^{\text{prob.}}$	$\lambda_{\mathcal{O}}^{\text{center}}$	$\lambda_{\mathcal{O}}^{\text{Chol.}}$
Low	$\mathcal{VB}, \mathcal{TW}, \mathcal{H}$	1	3	2
Medium	$\mathcal{VS}, \mathcal{CB}, \mathcal{P}, \mathcal{TC}, \mathcal{O}$	3	9	7
High	$\mathcal{VC}, \mathcal{VE}, \mathcal{CS}$	5	15	12

Table 5. Category-specific loss weights grouped by object count.

KL Loss. Following a standard VAE formulation [34], we apply a Kullback-Leibler (KL) divergence regularization on the joint layout latent. The loss is defined as:

$$\mathcal{L}_{\text{KL}} = \lambda^{\text{KL}} \cdot \left(-\frac{1}{2} \sum_{d=1}^{64} (1 + \log(\sigma_{\mathcal{L},d}^2) - \mu_{\mathcal{L},d}^2 - \sigma_{\mathcal{L},d}^2) \right) \quad (9)$$

where $\mu_{\mathcal{L},d}$ and $\sigma_{\mathcal{L},d}$ denote the mean and standard deviation of the d -th latent dimension. We set $\lambda^{\text{KL}} = 1e^{-6}$.

Total Loss. The overall training objective for the first stage combines the three components as:

$$\mathcal{L}_{\text{LVAE}} = \mathcal{L}_{\text{ground}} + \mathcal{L}_{\text{object}} + \mathcal{L}_{\text{KL}} \quad (10)$$

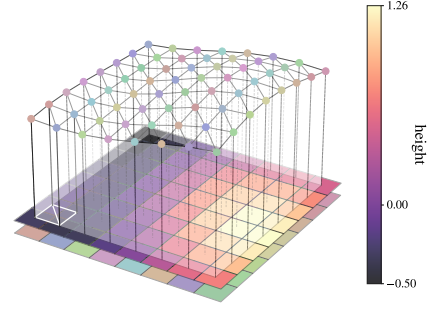


Figure 7. **Extrusion Process.** An illustrative extrusion example showing the composite color map (bottom), height field (middle), and lifted triangle mesh (top).

B.2.3. Layout Reconstruction

To reconstruct a 3D scene layout, we follow separate procedures for ground surfaces and objects, reflecting their distinct representations. For **ground surfaces**, we first fuse the per-class height maps into a composite height field by selecting, at each pixel, the highest elevation and recording the class that contributed it. Using a fixed label-to-color mapping, this process simultaneously yields a composite color map aligned with this height field. The resulting raster representation is then converted into a triangle mesh (see Fig. 7). Each occupied pixel (u, v) is lifted to a 3D vertex (x, y, z) using its height value and the vertex color is assigned from the composite color map at the same location. To obtain a continuous surface, we triangulate each 2×2 pixel block at indices (i, j) , $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$ using a consistent diagonal, forming two adjacent triangles per quad. For **objects**, we first derive the rotation matrix $\hat{\mathbf{R}}$ and scale matrix $\hat{\mathbf{A}}$ from the predicted Cholesky parameters $\hat{\mathbf{c}} \in \mathbb{R}^6$, as described in Sec. B.1. We then construct the full transformation matrix $\hat{\mathbf{T}} \in \mathbb{R}^{4 \times 4}$ by combining these with the predicted 3D center location $\hat{\mathbf{t}} \in \mathbb{R}^3$:

$$\hat{\mathbf{T}} = \begin{bmatrix} \hat{\mathbf{R}}\hat{\mathbf{A}} & \hat{\mathbf{t}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (11)$$

This transformation is applied to a unit-sized, zero-centered 3D cuboid or ellipsoid (depending on the object category) to position the primitive within the reconstructed scene layout.

B.2.4. Implementation Details

The LVAE comprises 93.1M parameters and is trained using PyTorch Lightning¹ for 3 days on a single A100 GPU with a batch size of 32. Optimization is performed using AdamW with a weight decay of $5e^{-4}$, combined with a OneCycleLR scheduler. The learning rate follows a cosine annealing schedule, starting at $1e^{-4}$ and decaying to $1e^{-5}$,

¹<https://lightning.ai/docs/pytorch/stable/>

without a warm-up phase. During inference, we discard object primitives whose predicted existence probability falls below a threshold of $\hat{p} = 0.3$. We use a single global threshold across all semantic categories. This value controls the trade-off between retaining more primitives and suppressing low-confident ones: lower values lead to denser but potentially noisier layouts, while higher values yield cleaner but sparser reconstructions. In practice, we found that reconstruction quality is not highly sensitive to this parameter, and values within $[0.25, 0.75]$ produce visually similar results.

B.3. Diffusion Transformer

In the second stage of our method, we train a latent diffusion model (LDM) [68] on the joint layout latent space of our pre-trained LVAE.

B.3.1. Architecture

We adopt the standard DiT architectures from [59] with a patch size of 2, yielding 16×16 tokens from the joint layout latent representation $\mathbf{z}_{\mathcal{L}} \in \mathbb{R}^{32 \times 32 \times 64}$. Tab. 6 summarizes the configurations of the DiT Base (B), Large (L), and XLarge (XL) variants used in our experiments.

Model	Layers	Hidden Size	Heads	Batch Size	Params (M)
DiT-B	12	768	12	256	130
DiT-L	24	1024	16	256	458
DiT-XL	28	1152	16	256	675

Table 6. Details of DiT-B, -L, and -XL model variants.

B.3.2. Training

Given an initial latent $\mathbf{z}_{\mathcal{L}}^0$, we define a *forward diffusion process* that gradually corrupts it by adding Gaussian noise over time, yielding a sequence of increasingly noisy latents $\mathbf{z}_{\mathcal{L}}^1, \dots, \mathbf{z}_{\mathcal{L}}^T$. The noise magnitude at each step is controlled by a variance schedule $\{\beta_t \in (0, 1)\}_{t=1}^T$. The joint distribution over all timesteps is expressed as:

$$q(\mathbf{z}_{\mathcal{L}}^{1:T} | \mathbf{z}_{\mathcal{L}}^0) = \prod_{t=1}^T q(\mathbf{z}_{\mathcal{L}}^t | \mathbf{z}_{\mathcal{L}}^{t-1}) \quad (12)$$

$$q(\mathbf{z}_{\mathcal{L}}^t | \mathbf{z}_{\mathcal{L}}^{t-1}) = \mathcal{N}(\mathbf{z}_{\mathcal{L}}^t; \sqrt{1 - \beta_t} \mathbf{z}_{\mathcal{L}}^{t-1}, \beta_t \mathbf{I}) \quad (13)$$

A convenient property of the above process is that the noisy latent $\mathbf{z}_{\mathcal{L}}^t$ at any timestep t can be sampled in a single step from the following distribution:

$$q(\mathbf{z}_{\mathcal{L}}^t | \mathbf{z}_{\mathcal{L}}^0) = \mathcal{N}(\mathbf{z}_{\mathcal{L}}^t; \sqrt{\bar{\alpha}_t} \mathbf{z}_{\mathcal{L}}^0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad (14)$$

where $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$.

To synthesize new latents, we learn a *reverse process* that denoises from pure Gaussian noise $\mathbf{z}_{\mathcal{L}}^T \sim \mathcal{N}(0, \mathbf{I})$ back to a

clean latent $\mathbf{z}_{\mathcal{L}}^0$. The joint distribution of this process is:

$$p_{\theta}(\mathbf{z}_{\mathcal{L}}^{0:T}) = p(\mathbf{z}_{\mathcal{L}}^T) \prod_{t=1}^T p_{\theta}(\mathbf{z}_{\mathcal{L}}^{t-1} | \mathbf{z}_{\mathcal{L}}^t) \quad (15)$$

$$p_{\theta}(\mathbf{z}_{\mathcal{L}}^{t-1} | \mathbf{z}_{\mathcal{L}}^t) = \mathcal{N}(\mathbf{z}_{\mathcal{L}}^{t-1}; \mu_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t), \Sigma_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t)) \quad (16)$$

where each transition is modeled as a Gaussian with learnable mean μ_{θ} and covariance Σ_{θ} , estimated by a denoising network parameterized by θ . In practice, we follow [28] and set $\Sigma_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t) = \sigma_t^2 \mathbf{I} = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \mathbf{I}$, treating it as an untrained, time-dependent constant. Moreover, we represent the mean $\mu_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t)$ as:

$$\mu_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{z}_{\mathcal{L}}^t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t) \right) \quad (17)$$

where ϵ_{θ} predicts the noise term ϵ in the reparameterization $\mathbf{z}_{\mathcal{L}}^t = \sqrt{\alpha_t} \mathbf{z}_{\mathcal{L}}^0 + \sqrt{1 - \alpha_t} \epsilon$.

For *training*, we adopt the simplified denoising objective of [28]:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{z}_{\mathcal{L}}^0, \epsilon, t} \left[\left\| \epsilon - \epsilon_{\theta}(\sqrt{\alpha_t} \mathbf{z}_{\mathcal{L}}^0 + \sqrt{1 - \alpha_t} \epsilon, t) \right\|^2 \right] \quad (18)$$

In our method, we further condition the denoising network on a scene label y , controlling the density of specific scene primitives (e.g. vegetation). The resulting conditional network $\epsilon_{\theta}(\mathbf{z}_{\mathcal{L}}^t, t, y)$ is trained with the modified objective:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathbf{z}_{\mathcal{L}}^0, \epsilon, t, y} \left[\left\| \epsilon - \epsilon_{\theta}(\sqrt{\alpha_t} \mathbf{z}_{\mathcal{L}}^0 + \sqrt{1 - \alpha_t} \epsilon, t, y) \right\|^2 \right] \quad (19)$$

B.3.3. Applications

Leveraging a latent manipulation strategy inspired by RePaint [47], our diffusion model enables practical downstream tasks without the need for additional fine-tuning.

Scene Inpainting. RePaint introduced a modified diffusion-based sampling strategy for image inpainting that employs a binary mask to distinguish known and unknown regions. The key idea is to leverage a pre-trained denoising diffusion model as a generative prior and to synchronize the denoising of unknown pixels with the noised version of known pixels. We adapt this approach to operate in the 2D latent space of a pre-trained LDM, enabling localized edits of 3D scene layouts. For simplicity, we omit the subscript $\mathcal{L}$ and denote the noisy layout latent at timestep t as $\mathbf{z}_t$. Given an initial latent $\mathbf{z}_0$, the modified denoising update is formulated as:

$$\mathbf{z}_{t-1}^{\text{known}} \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{z}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad (20)$$

$$\mathbf{z}_{t-1}^{\text{unknown}} \sim \mathcal{N}(\mu_{\theta}(\mathbf{z}_t, t, y), \sigma_t^2 \mathbf{I}) \quad (21)$$

$$\mathbf{z}_{t-1} = (1 - \mathbf{m}) \odot \mathbf{z}_{t-1}^{\text{known}} + \mathbf{m} \odot \mathbf{z}_{t-1}^{\text{unknown}} \quad (22)$$

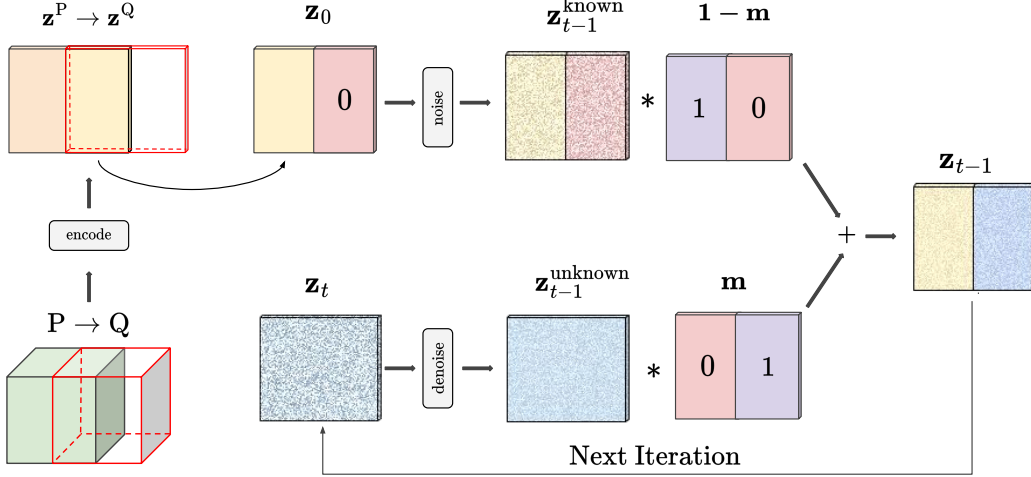


Figure 8. **Latent-space Scene Outpainting.** Given a known layout block P (green) with latent $\mathbf{z}^P$, the diffusion model predicts the latent $\mathbf{z}^Q$ for an adjacent block Q (outlined in red) that partially overlaps with P . During each sampling step, the overlapping area (yellow) is re-noised to match the current diffusion timestep, while denoising is applied to $\mathbf{z}_t$. The resulting latents are then combined through the mask $\mathbf{m}$, yielding a coherent representation $\mathbf{z}_{t-1}$ for the next iteration that preserves the known content and refines the unknown region (blue).

where $\mathbf{m}$ is a binary latent mask identifying the regions to be edited, $\odot$ denotes element-wise multiplication, and $\mathcal{N}(\mu_\theta(\mathbf{z}_t, t, y), \sigma_t^2 \mathbf{I})$ represents the learned reverse process of our pre-trained diffusion model. By properly setting the mask $\mathbf{m}$, we can edit an original scene in diverse ways.

Scene Outpainting. Scene outpainting extends a 3D layout beyond its original spatial extent by predicting plausible continuations of existing structures. We perform this extrapolation directly in the latent space of our diffusion model, using the same manipulation strategy employed for inpainting. As illustrated in Fig. 8, the procedure adopts a sliding-window scheme: given a known layout block P (green) with latent representation $\mathbf{z}^P$, we synthesize the latent $\mathbf{z}^Q$ of an adjacent block Q (outlined in red) that overlaps with P . The goal is to generate $\mathbf{z}^Q$ such that the overlapping part is faithfully preserved while the extended area remains semantically and structurally aligned with P . In all experiments, we use an overlap of 50%. This choice balances structural coherence with generative freedom. A large value strengthens conditioning on the known region, promoting continuity of geometry and semantics across block boundaries, but it also restricts the model’s ability to produce novel content. Conversely, overly reducing the overlap weakens the conditioning signal, often resulting in boundary artifacts and semantic misalignment. To initialize the diffusion process, we construct an input latent $\mathbf{z}_0$ by copying the overlapping region (yellow) from $\mathbf{z}^P$ and zeroing out the rest. A binary mask $\mathbf{m}$ marks the unknown area to be generated with ones and the known region with zeros. The process begins by sampling $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, following the standard reverse diffusion formulation. During each step t , the latent $\mathbf{z}_t$ is jointly denoised across both regions, yielding an

intermediate prediction $\mathbf{z}_{t-1}^{\text{unknown}}$ (see Eq. (21)). To maintain statistical compatibility between the two parts, the forward noising process is applied to the input latent $\mathbf{z}_0$, yielding $\mathbf{z}_{t-1}^{\text{known}}$ (see Eq. (20)). The two components are then merged through the mask $\mathbf{m}$ to form a unified latent representation $\mathbf{z}_{t-1}$ for the next denoising iteration (see Eq. (22)).

To expand a scene spatially, we apply this mechanism in a two-stage approach. In the first pass, we generate the four cardinal neighbors (top, bottom, left, and right), each conditioned on its corresponding overlapping region with the original layout. In the second pass, we synthesize the remaining four corner regions (top-left, top-right, bottom-left, and bottom-right), which depend on the previously generated neighbors. We note that within each pass, all blocks are generated in parallel. This two-stage process effectively doubles the spatial extent of the original layout. Iteratively applying this expansion enables the synthesis of large-scale 3D scenes, a task we refer to as large-scale scene extrapolation. For simplicity, and consistent with prior work [44], we restrict iterative extrapolation to a single spatial axis.

B.3.4. Implementation Details

All DiT variants are trained using the Diffusers library². Training is performed for 3 days on 4 A100 GPUs with a batch size of 256. Optimization uses AdamW with a constant learning rate of $1e^{-4}$ and a weight decay of $1e^{-6}$. An Exponential Moving Average (EMA) with a decay rate of 0.9999 is applied to stabilize training. We employ the DDPM scheduler with 1000 timesteps and a linear variance schedule ranging from $\beta_{\text{start}} = 0.0015$ to $\beta_{\text{end}} = 0.015$. Classifier-free

²<https://github.com/huggingface/diffusers>

guidance [27] is applied with a guidance scale of 4.0. During inference, we use 250 denoising steps, while inpainting and outpainting employ a roll-back of $J = 10$ and $R = 10$ resampling iterations.

C. Experimental Details

In this section, we describe the synthetic dataset and evaluation protocol used to compare our Cholesky-based encoding with a quaternion-based alternative. For this ablation study, we generate synthetic datasets of increasing size, where each sample contains a single zero-centered vehicle primitive with randomly sampled scales and yaw rotations. The training sets consist of 1.2K, 2.5K, 5K, and 10K samples, with each larger set including all samples from the smaller ones. A shared test set of 2K samples is used for evaluation. To reduce variance from random sampling, dataset generation is repeated with five different random seeds. For each seed and training size, both encoding variants are trained for 20 epochs, and the mean IoU3D is computed on the shared test set. Final results are reported as the average IoU3D over all seeds for each dataset size.

D. Baseline Methods

To the best of our knowledge, no prior work tackles 3D semantic urban layout generation using primitives. We therefore compare against three recent and well-established voxel-based methods, all trained from scratch on our voxelized dataset: SemCity [38], PDD [44], and XCube [64]. This comparison allows us to empirically demonstrate the benefits of primitives over voxels, the prevailing representation in this field, which is central to our contribution. SemCity adopts a continuous triplane diffusion model trained on the SemanticKITTI [4] and CarlaSC [87] datasets. It first trains a triplane autoencoder and then applies diffusion within the learned latent triplane space. The model follows a non-hierarchical design and supports downstream tasks such as semantic scene inpainting, outpainting, and completion refinement. PDD employs a discrete, hierarchical diffusion paradigm trained on the synthetic CarlaSC dataset. It generates scenes in a coarse-to-fine manner across three pyramid levels. The first level synthesizes scenes unconditionally from noise, while subsequent levels condition on the preceding outputs to progressively refine scene details. PDD supports large-scale scene outpainting and conditioned scene generation, the latter refining a ground-truth coarse layout into a more detailed semantic scene. XCube generates large-scale 3D scenes using a hierarchical sparse-voxel representation trained across the Kartan City [1] and Waymo [74] datasets. Each level in the hierarchy is trained independently and consists of a VAE-diffusion pair trained sequentially (VAE first, then diffusion). Both the VAE and diffusion components use 3D UNet backbones.

For urban scenes, XCube supports both unconditional and single-LiDAR-scan-conditioned generation, though the conditional implementation is not yet publicly available. Tab. 7 summarizes the model and training configurations for all baseline methods and PrITTI. Following the original implementations, PDD and XCube are trained using 4 and 8 A100 GPUs, respectively. In addition to training configurations, we also compare the peak GPU memory usage of all methods under the same inference setup. As shown in Tab. 8, our method is more memory-efficient than voxel-based baselines, consistent with the use of a primitive-based representation.

E. Additional Qualitative Results

In this section, we present additional qualitative results for all tasks described in the main paper.

3D Semantic Scene Generation. Fig. 14 presents 3D semantic scenes generated by our method. By conditioning the LDM on discrete scene labels controlling vegetation density, we can produce scenes with (a) high, (b) medium, and (c) low vegetation. Additional BEV semantic renderings of synthesized scenes, used for computing the generative metrics reported in the main paper, are shown in Fig. 15. Notably, conditioning on vegetation density serves only as a minimal illustration of controllability through our class-conditional DiT. This conditioning scheme is general and can be naturally extended to other primitive categories (e.g., vehicles) or their combinations. To demonstrate this, we additionally trained models conditioned on vehicle density and joint vegetation–vehicle densities. Qualitative results are shown in Fig. 16 and Fig. 17, respectively. For comparison, we provide 3D and BEV rendering results for all baselines, generated unconditionally: SemCity (Fig. 18, Fig. 19), PDD (Fig. 20, Fig. 21), and XCube (Fig. 22, Fig. 23). As seen in their BEV renderings and Fig. 9, these methods often produce scenes with limited semantic consistency and structural coherence, exhibiting blended geometries, irregular object shapes, and noisy semantic predictions. These artifacts reduce the perceived realism of the results, aligning with the lower quantitative generative performance in the main paper. **Scene Inpainting and Outpainting.** Fig. 24 and Fig. 27 present additional scene inpainting and outpainting results produced by our method, respectively. As illustrated in Fig. 24, by appropriately setting the binary latent mask $\mathbf{m}$ (see Eq. (22)), we can achieve flexible, localized scene edits in arbitrary directions (e.g. top, bottom, left, right). Among the baselines, only SemCity reports both inpainting and outpainting results. Fig. 26 compares our method with SemCity, demonstrating that PrITTI produces coherent and semantically consistent edits and extensions, whereas SemCity often yields fragmented or contextually inconsistent structures, particularly in the outpainting setting. XCube does not address these scene-level editing tasks, and PDD provides only qualitative results for large-scale scene extrapolation, which

Method	Hierarchical	Level	Voxel Size (m)	Voxel Resolution	VAE Train (days)	Diff. Train (days)
SemCity [38]	✗	Single	0.25	$256^2 \times 32$	3	3
PDD [44]	✓	Coarse	2.0	$32^2 \times 4$	—	1
		Medium	1.0	$64^2 \times 8$	—	2
		Fine	0.25	$256^2 \times 16$	—	14
XCube [64]	✓	Coarse	1.0	$64^2 \times 8$	1	2
		Fine	0.25	$256^2 \times 32$	6	7
PrITTI (Ours)	✗	Single	—	—	3	3

Table 7. **Comparison of Architectures and Training Configurations.** For each method, we report the model hierarchy, resolution settings, and the corresponding VAE and diffusion training times (where applicable).

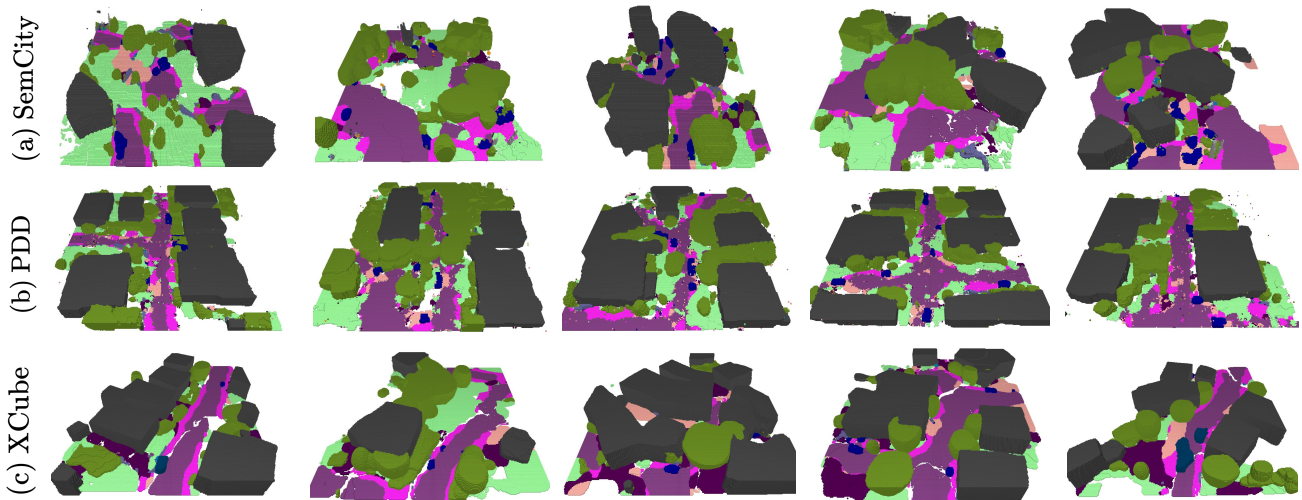


Figure 9. **Low-quality scene generation examples** from baseline methods exhibiting fragmented geometry and semantic inconsistencies.

Metric	SemCity	PDD	XCube	Ours (DiT-B)
Peak GPU Memory (GB)	8.30	1.33	4.84	0.89

Table 8. Peak GPU memory usage (GB) under identical inference settings (100 scenes, batch size 1, single GPU).

we compare against in a later section.

Ground-conditioned 3D Primitives Generation. Fig. 25 illustrates a special case of inpainting enabled by our disentangled latent representation. By keeping the ground channels fixed and inpainting only the object channels, our model generates novel object primitives conditioned on a consistent ground layout. This is achieved using a pre-trained LDM without any additional training.

Large-scale Scene Extrapolation. Fig. 28 shows large-scale scene extrapolation results of our method, starting from an initially generated layout block (outlined in red) and progressively extending it upward. The extrapolated layouts exhibit strong global coherence, preserving smooth road connectivity and semantically consistent transitions between

synthesized regions. All examples use the medium vegetation density condition. Changing the label to low or high can enable extrapolation of the same initial block under different vegetation settings, highlighting our model’s controllability (see supplementary video for an example). For comparison, we also include results from PDD, the only baseline supporting large-scale scene extrapolation. SemCity provides code only for outpainting a generated block twice (see Fig. 26), and therefore cannot be evaluated here. As shown in Fig. 29, PDD often fails to maintain global structure: roads are disconnected between adjacent blocks, buildings are frequently misplaced, and abrupt semantic transitions occur across generated areas, resulting in overall fragmented and visually implausible large-scale layouts. In contrast, our method produces continuous, well-aligned extrapolations with consistent semantics and geometry across large regions.

Object-level Editing. Our framework maintains an instance-level vector representation for each primitive, parameterized by its 3D center location and 6D Cholesky parameters. This formulation allows intuitive manipulation of individual ob-

jects by directly updating their corresponding parameters (e.g. center for translation, Cholesky parameters for rotation and scaling). Fig. 30 illustrates additional editing results, where simple transformations are applied to randomly selected vehicle primitives from the original scene (column a). Columns (b-e) visualize examples of dropout, rotation, scaling, and translation, respectively. In contrast, object-level manipulation in voxel-based scene representations remains highly challenging. First, identifying which voxels correspond to a particular object typically requires explicit segmentation, which is often complicated by ambiguous object-background boundaries. Furthermore, even after segmentation, applying a transformation such as translation requires moving the corresponding voxel subset and then filling in the vacated space. Consequently, existing voxel-based approaches for 3D semantic urban scene generation provide little or no support for object-level editing. For instance, PDD and SemCity do not enable any form of direct instance-level manipulation, while XCube only supports manual voxel addition or removal for a single object at the coarsest level of its hierarchy through a dedicated user interface.

Photo-realistic Street View Synthesis. Fig. 31 presents additional examples demonstrating how 2D semantic maps rendered from our generated 3D scenes can be translated into photo-realistic street-view images. For this task, we employ the ControlNet [104] model from Urban Architect [45], which is fine-tuned on KITTI-360 3D semantic layouts. Although some visual artifacts exist, the rendered maps serve as effective conditioning signals and reliably guide the image generation process. Notably, despite the coarse geometry of our primitives, the synthesized images reflect object shapes beyond cuboids or ellipsoids. We emphasize that photo-realistic appearance synthesis is not the primary objective of our work but rather an additional downstream application.

F. Precision and Recall Metrics

To evaluate our generative models, among others, we use the improved precision and recall metrics introduced by Kynkäänniemi et al. [37]. Let Φ_r and Φ_g denote the feature sets extracted from real and generated samples, respectively. A binary function $f(\phi, \Phi)$ determines whether a feature vector ϕ lies within the estimated manifold of a reference set $\Phi \in \{\Phi_r, \Phi_g\}$:

$$f(\phi, \Phi) = \begin{cases} 1, & \text{if } \|\phi - \phi'\|_2 \leq \|\phi' - \text{NN}_k(\phi', \Phi)\|_2, \\ & \text{for at least one } \phi' \in \Phi, \\ 0, & \text{otherwise.} \end{cases} \quad (23)$$

Here, $\text{NN}_k(\phi', \Phi)$ denotes the k^{th} nearest neighbor of ϕ' within Φ . Using this function, precision and recall are computed as:

$$\text{precision}(\Phi_r, \Phi_g) = \frac{1}{|\Phi_g|} \sum_{\phi_g \in \Phi_g} f(\phi_g, \Phi_r) \quad (24)$$

$$\text{recall}(\Phi_r, \Phi_g) = \frac{1}{|\Phi_r|} \sum_{\phi_r \in \Phi_r} f(\phi_r, \Phi_g) \quad (25)$$

Intuitively, precision measures the proportion of generated samples that appear realistic (i.e. fall within the manifold of real data), whereas recall quantifies how well the generated samples cover the diversity of real samples. Following [37], we use an equal number of real and generated samples ($|\Phi_r| = |\Phi_g|$). However, we found out that these metrics are highly sensitive to several factors including: (i) implementation details, (ii) the diversity of real samples, (iii) the number of evaluation samples, and (iv) the neighborhood size k used for manifold estimation (see Eq. (23)). For (i), we follow [59], which forms the basis of our DiT models, and use the evaluation suite from [19] for consistency. To address (ii), we implement two separate strategies for sampling real poses. The first enforces a minimum spatial distance (in meters) between neighboring poses. The second, inspired by farthest point sampling (FPS), selects poses sequentially to maximize their distance from previously chosen ones. The number of evaluation samples (iii) depends on the applied sampling strategy. In the distance-based approach, the sample size is implicitly determined by the distance constraint, whereas in FPS we explicitly control it, evaluating with 1K, 5K, and 10K real samples. Finally, to study (iv), we vary the neighborhood size $k \in \{3, 5\}$, examining its effect on the generative performance. Comprehensive evaluations of precision and recall under these configurations are presented in Fig. 10, comparing our approach against SemCity, XCube, and PDD. For PDD and XCube, results are reported at their finest resolution, and for SemCity we use the 1M variant.

For both sampling strategies, we observe a consistent decrease in precision as the number of evaluation samples increases. In the distance-based strategy, this effect corresponds to smaller distance thresholds (e.g. 1 m yields 32,203 samples, whereas 10 m yields 3,855). This behavior follows Eq. (23): as more real samples are included, the hyperspheres defined by their k^{th} nearest neighbors become smaller and more densely packed. Consequently, even visually realistic generated samples are less likely to fall within any real sample’s hypersphere. In contrast, recall remains relatively stable across varying sample counts, likely because denser sampling refines the real manifold without altering its overall extent. Finally, both precision and recall increase with larger neighborhood sizes k . This trend follows directly from Eq. (23): increasing k expands each sample’s hypersphere, thereby loosening the inclusion criterion.

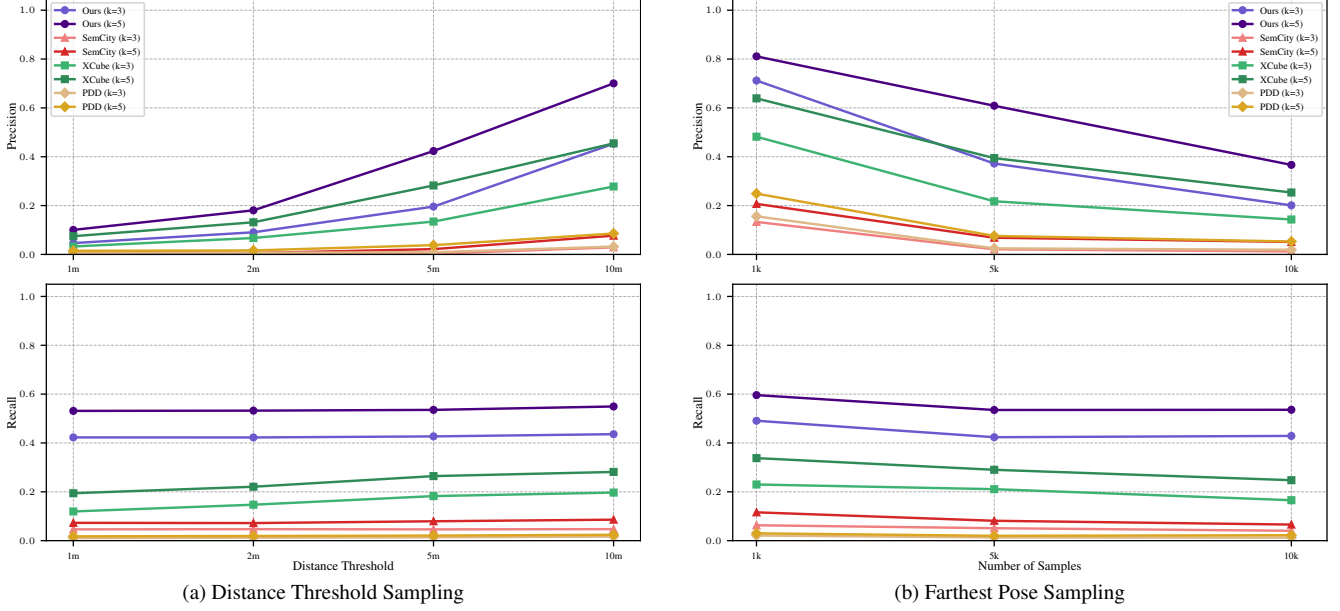


Figure 10. **Precision and recall comparison** between our method (using DiT-B), SemCity, XCube, and PDD under two evaluation protocols: (a) distance-threshold reference-pose sampling and (b) farthest-pose sampling, each tested across varying neighborhood sizes.

G. Overfitting Analysis

To assess potential memorization of dataset-specific patterns, we perform a nearest-neighbor (NN) analysis in feature space. We extract features using a pretrained VGG16 network and compute Euclidean NN distances between 50K generated BEV samples and a fixed subset of 50K training samples. To contextualize these values, we additionally compute real-to-real distances on the same subset by matching each training sample to its nearest neighbor excluding self-matches. We observe that generated samples are consistently farther from the training set (mean NN distance ≈ 52.2 , median ≈ 51.7 , minimum ≈ 27.2 , max ≈ 115.7) than real samples are from each other (mean NN distance ≈ 22.5 , median ≈ 22.0 , minimum ≈ 4.9 , max ≈ 58.7). This suggests that the model does not simply memorize or reproduce training examples. This is further supported by the qualitative examples in Fig. 11, where generated samples differ from their three nearest neighbors in the training set in layout and object arrangement.

H. Applicability beyond KITTI-360

We conduct our main experiments on KITTI-360, which, to the best of our knowledge, is the only real-world dataset providing primitive-level annotations for entire urban scenes (e.g. buildings, vegetation) rather than focusing on selected object classes (e.g. vehicles). This yields scenes with many primitives and thus a demanding setting for training and evaluation. Importantly, PrITTI does not require *dense* primitive annotations, but only parametric 3D cuboids or ellipsoids

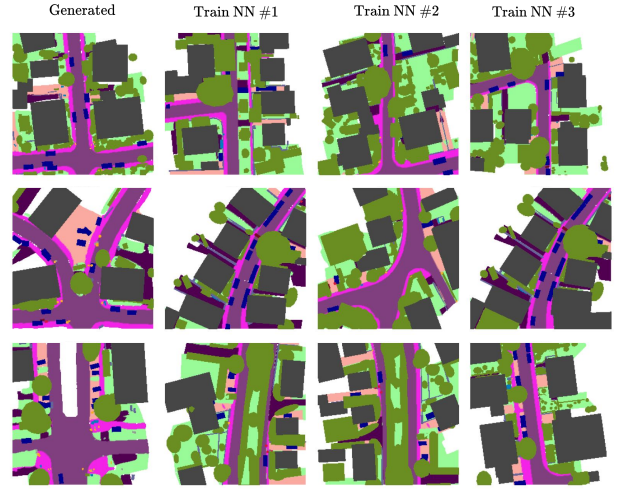


Figure 11. **Nearest-neighbor Analysis.** For each generated sample (left), we show its closest training samples (right). Generated scenes differ in layout and object placement, suggesting the model does not memorize training data.

for arbitrary semantic categories. When fewer categories are annotated, the model simply predicts fewer primitives overall, which makes the task easier. Many autonomous driving datasets [10, 11, 86] already provide such cuboid annotations. Training PrITTI on any of them involves grouping their labels into dataset-specific primitive categories and converting the cuboid parameters into our vectorized representation. For ground geometry, 3D HD maps (e.g. in [86]) enable BEV height fields rendering, while 2D maps (e.g.

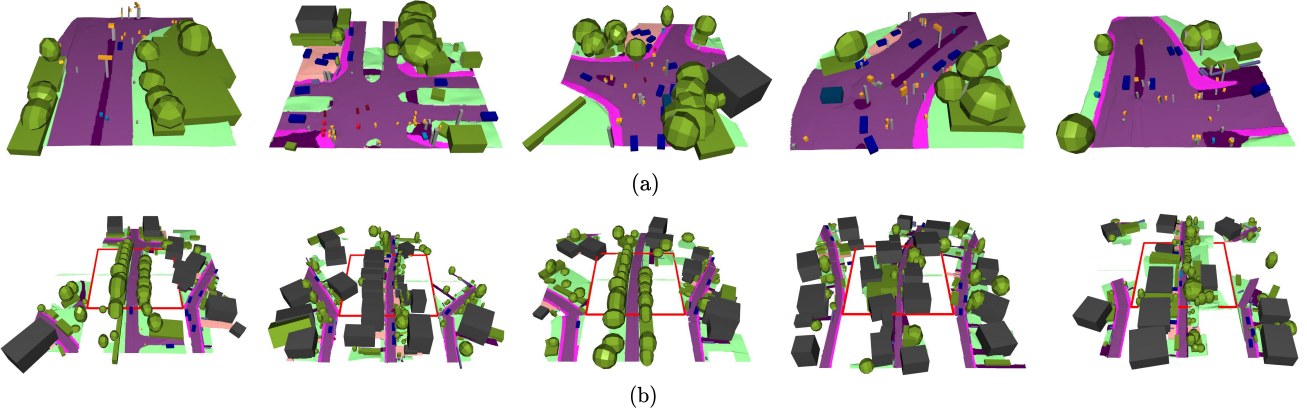


Figure 12. **PrITTI Failure Cases.** (a) Fine-grained attachments (e.g. pole-mounted lamps) are often not accurately reconstructed, yielding floating objects. (b) Large empty conditioning regions in outpainting can lead to weak or inconsistent completions.

Raster ($\times 10^{-2}$)		Primitive	
MSE $\downarrow$	IoU $\uparrow$	AP3D $\uparrow$	AP3D@50 $\uparrow$
0.015	99.98	56.495	39.490
(a) Reconstruction			
Precision $\uparrow$	Recall $\uparrow$	FID $\downarrow$	IS $\uparrow$
0.572	0.410	144.330	2.878
(b) Generation (DiT-B)			

Table 9. PrITTI evaluation results on AV2 [86], showing (a) reconstruction and (b) generation metrics.

in [10]) can be handled via a flat-ground assumption.

To demonstrate applicability beyond KITTI-360, we additionally train our method on Argoverse 2 (AV2) [86]. We group the original AV2 labels into seven primitive categories: vehicle big, vehicle small, two-wheelers, human, pole, traffic control, and object. We emphasize that PrITTI does not require a fixed category set across datasets, and for AV2 these groups happen to align well with a subset of those used in KITTI-360 (see Tab. 4). Notably, we find from dataset statistics that only 84 total queries are required, compared to 514 for [41], highlighting the reduced task complexity. We report qualitative generation results in Fig. 13 and quantitative results for both reconstruction and generation in Tab. 9.

I. Limitations and Failure Cases

Despite its promising results, our approach comes with certain limitations. First, as discussed in the main paper, the current framework is limited to static scenes and represents objects using coarse 3D primitives, which reduces geometric fidelity. Second, the model operates on a fixed set of semantic categories. Although this aligns with prior semantic

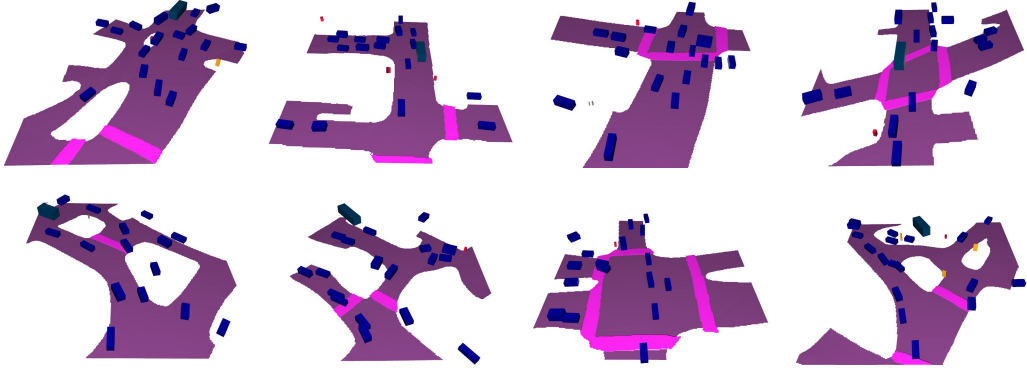
layout methods [38, 44, 64, 105], it limits the generation of unseen object types. Extending the framework toward open-vocabulary scene generation is an exciting future direction. In addition, ground elements are modeled as rasterized 2D representations that are subsequently extruded into 3D. Compared to voxel-based approaches that maintain a 3D grid over all scene elements, using a 2D representation yields better memory and computational scalability. However, the extrusion process is still inherently constrained by the resolution of the input raster maps, potentially resulting in less detailed ground surfaces. Furthermore, accurately reconstructing fine-grained attachments (e.g. lamps or traffic lights mounted on poles) and small, densely arranged structures (e.g. fences) remains quite challenging. Such geometric inaccuracies can be learned by the autoencoder and propagated to the diffusion model, occasionally producing scenes with floating or disconnected objects (see Fig. 12 (a)). Moreover, our framework does not explicitly enforce hard geometric constraints and instead relies on training data to capture such relationships. While the generated scenes are generally plausible, occasional geometric inconsistencies arise, especially in crowded scenes. In particular, primitives may intersect, which can sometimes be valid (e.g. vehicles with attached trailers) but not always appropriate (e.g. two vehicles placed unrealistically close to each other). Similarly, ground-primitive misalignment can produce semantically inconsistent configurations, such as buildings intersecting the drivable road surface (Fig. 15, right). Incorporating soft constraints or geometry-aware guidance during diffusion [46] could mitigate these issues and improve physical validity. Finally, in the outpainting setting, failures may occur when the conditioning block contains large empty regions along certain extrapolation directions. In such cases, the diffusion model is afforded greater generative freedom, which can lead to weakly connected or semantically inconsistent com-

pletions (see Fig. 12 (b)). Additionally, some visible seams can appear across block transitions in the ground surfaces.

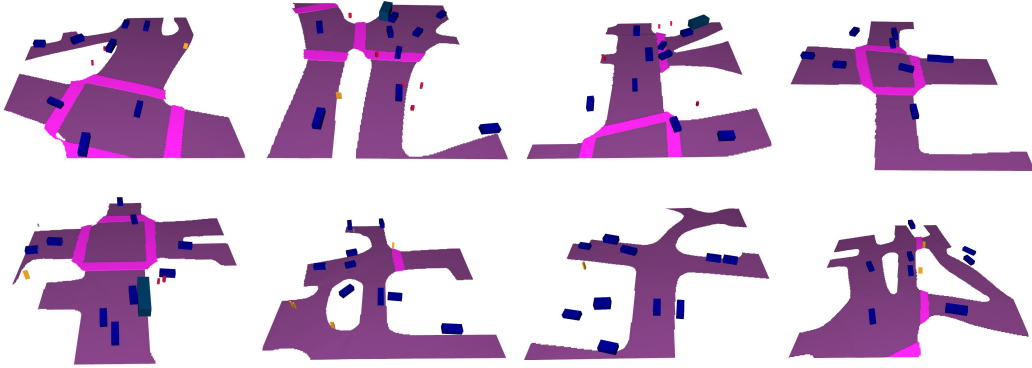
J. Potential Societal Impacts

Our work addresses 3D semantic scene generation for urban environments using primitive-based representations. Traditional approaches to constructing such 3D environments require expert knowledge and significant manual effort, making them costly and time-consuming. Moreover, reliance on pre-existing layout data, which remains scarce, particularly for outdoor scenes, limits diversity and leads to repetitive scene compositions. In contrast, our method enables the automatic generation of diverse and realistic 3D semantic scenes. This has the potential to benefit a range of applications in simulation, robotics, and autonomous driving. By providing interpretable and controllable 3D content, it can contribute to safer and more scalable environments for testing autonomous systems and improved scene understanding for embodied agents. Nevertheless, as with all generative models, there is potential for misuse. Synthetic scenes could, for instance, be employed to misrepresent real urban layouts, simulate favorable outcomes in planning proposals, or test vision-based monitoring systems (e.g. for traffic detection) under unrealistic conditions. Furthermore, our primitive-based representation simplifies real-world geometry, which, if over-relied upon, may lead to misleading conclusions in downstream tasks. These risks, however, do not diminish the framework’s practical value. As with existing modeling and simulation tools, effective use relies on a clear understanding of the method’s assumptions and limitations.

(a) High Vehicles



(b) Medium Vehicles



(c) Low Vehicles

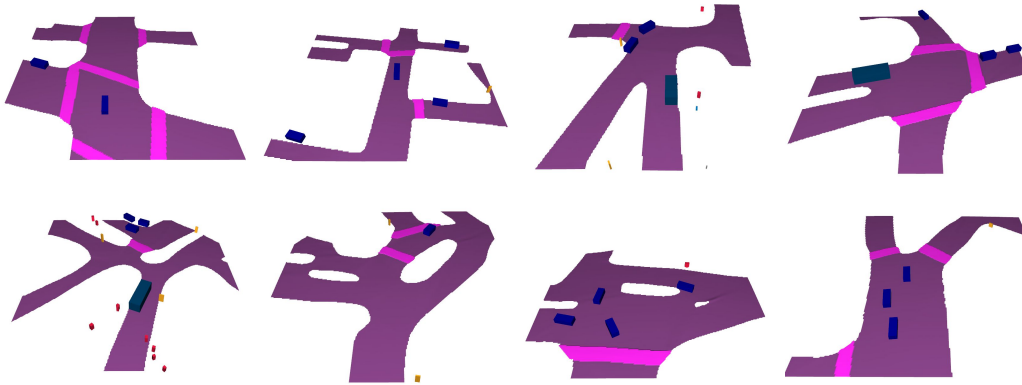
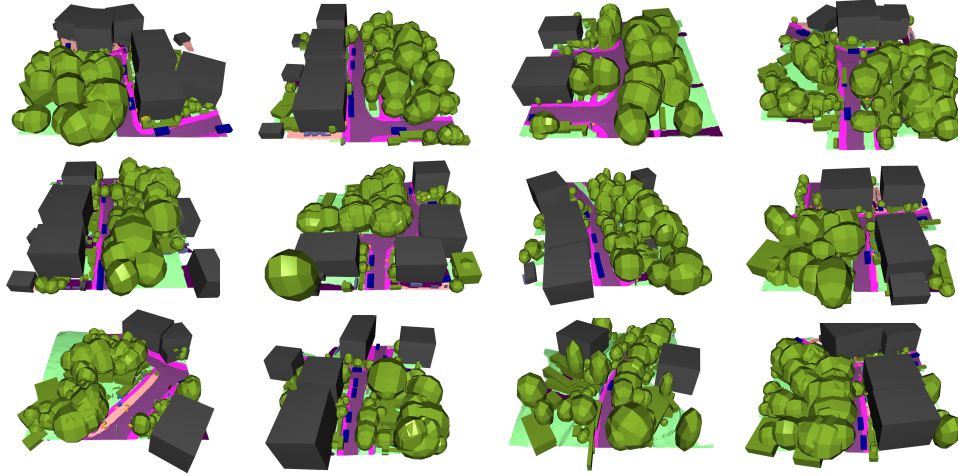
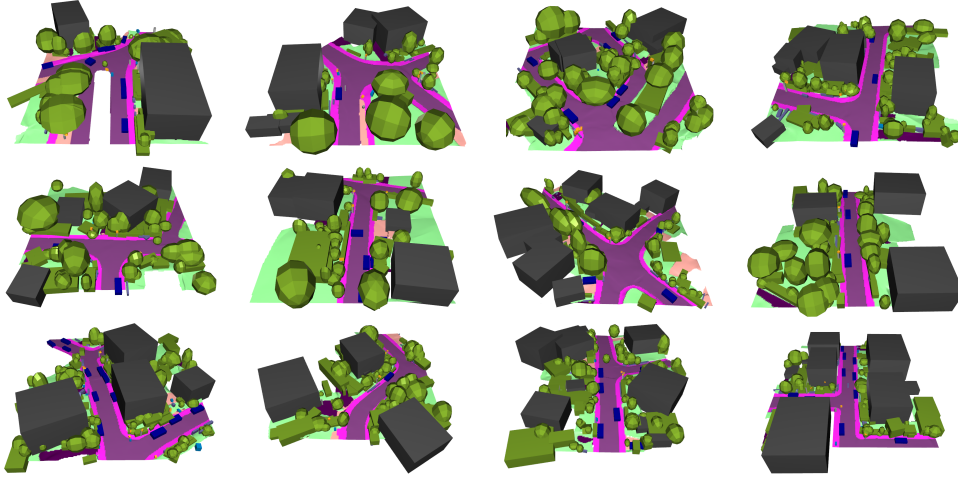


Figure 13. **PrITTI Scene Generation Results on Argoverse 2 (AV2) Dataset.** Generated 3D semantic scenes grouped under three vehicle-density conditions: (a) high, (b) medium, and (c) low. Vehicles appearing outside the drivable surface correspond to parked cars, as AV2 [86] HD maps do not provide explicit parking area annotations, unlike KITTI-360.

(a) High Vegetation



(b) Medium Vegetation



(c) Low Vegetation

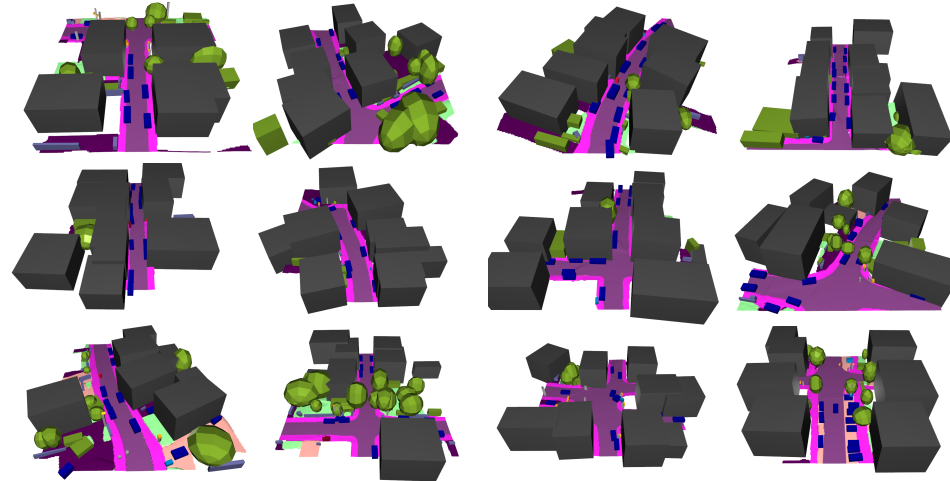


Figure 14. **PrITTI Scene Generation Results.** Generated 3D semantic scenes grouped under three vegetation-density conditions: (a) high, (b) medium, and (c) low. Across all settings, PrITTI produces diverse and realistic urban layouts with coherent ground surfaces and well-shaped primitives placed in semantically appropriate locations.

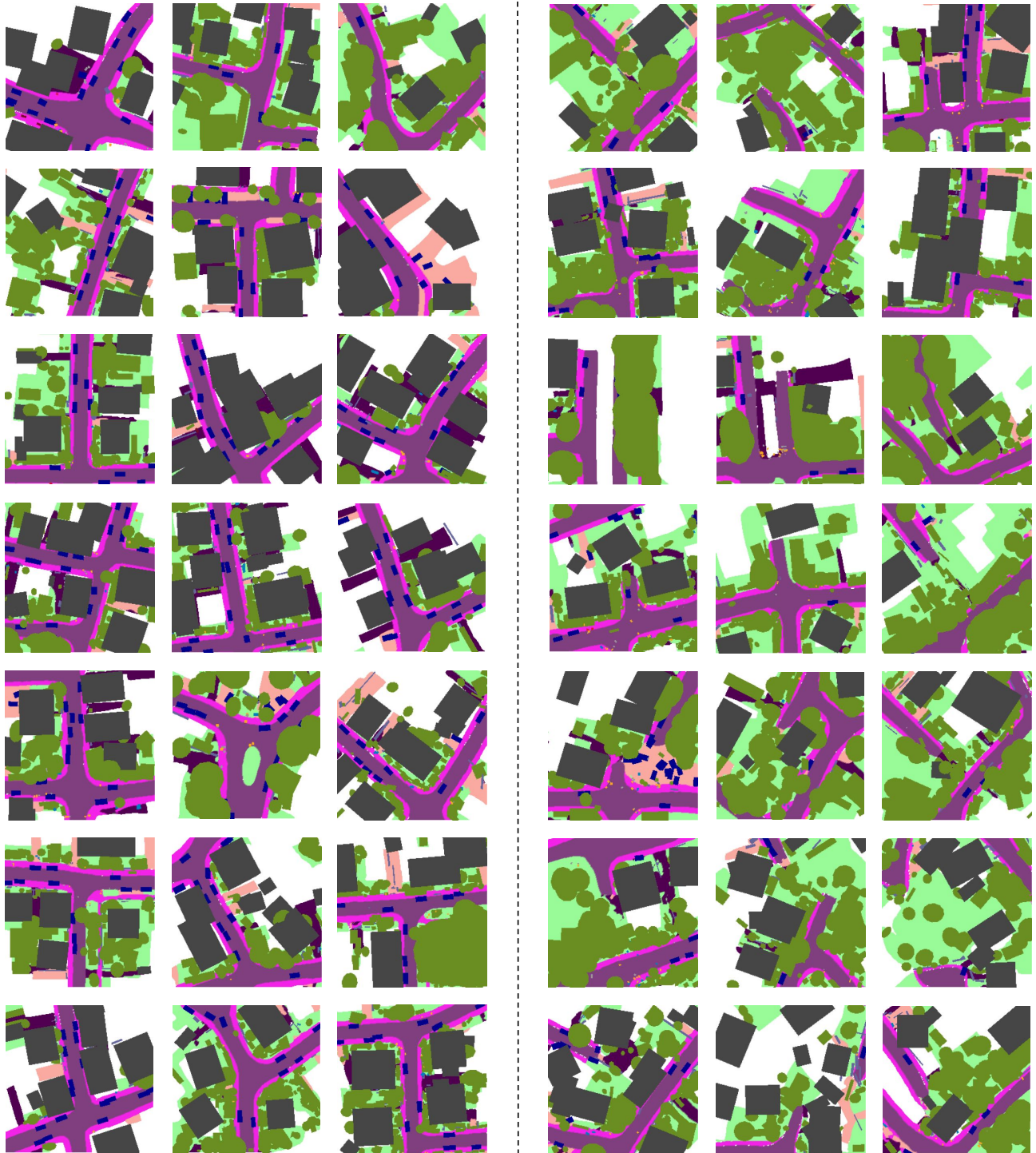


Figure 15. **BEV Renderings of PrITTI-generated Scenes.** Examples on the left illustrate well-structured and realistic scene generations, while those on the right show poorly synthesized samples with geometric or semantic inconsistencies.

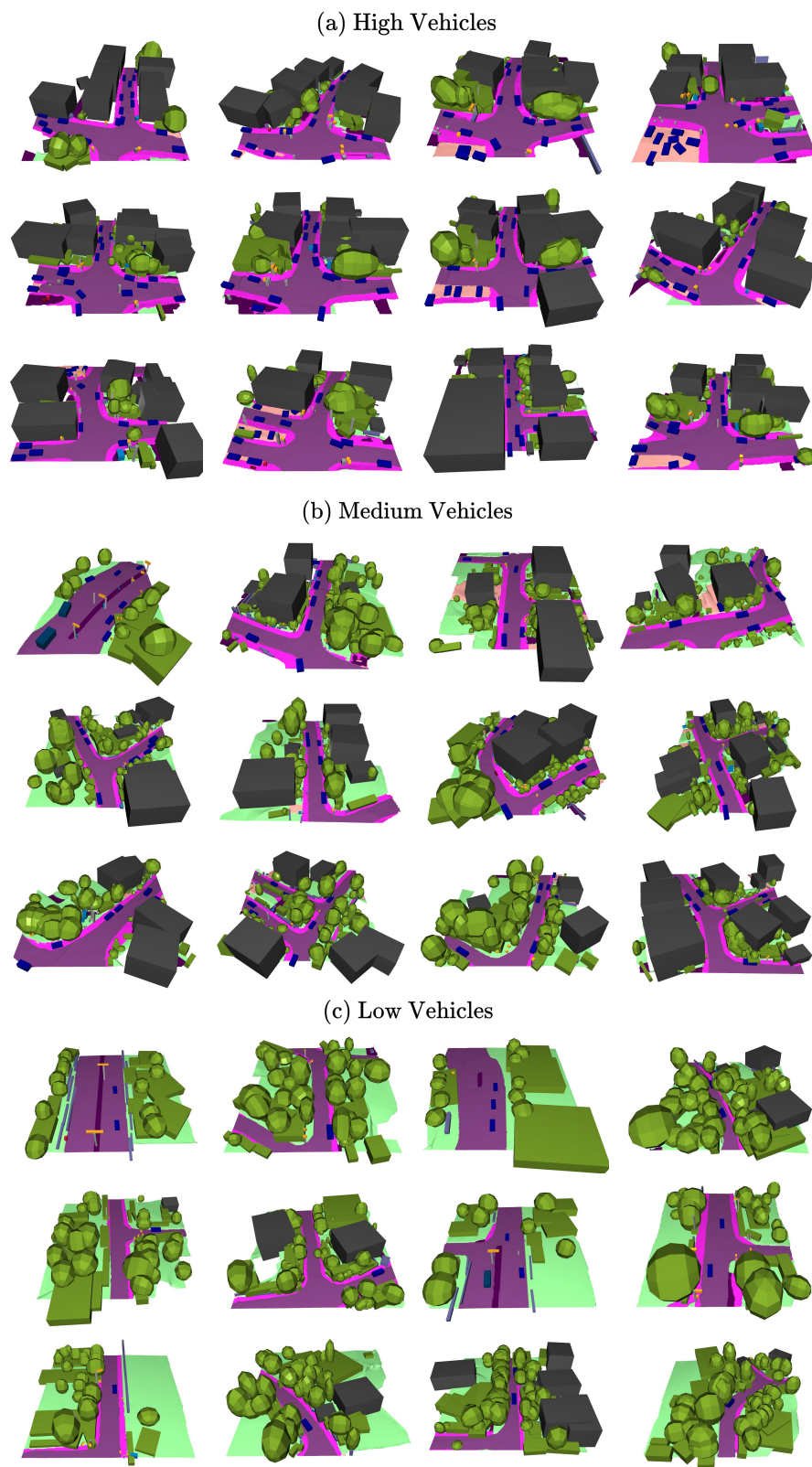
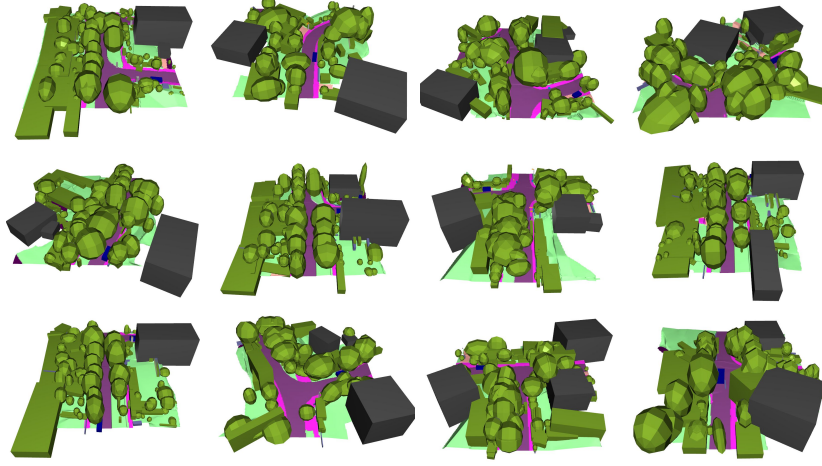
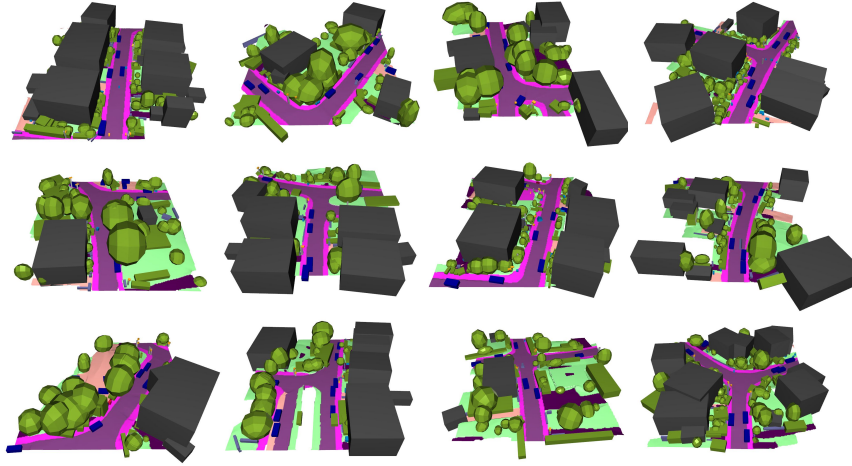


Figure 16. **PrITTI Vehicle Density-Conditioned Scene Generation.** Generated 3D semantic scenes grouped under three vehicle-density conditions: (a) high, (b) medium, and (c) low, demonstrating controllability over primitive categories beyond vegetation.

(a) High Vegetation, Low Vehicles



(b) Medium Vegetation, Medium Vehicles



(c) Low Vegetation, High Vehicles

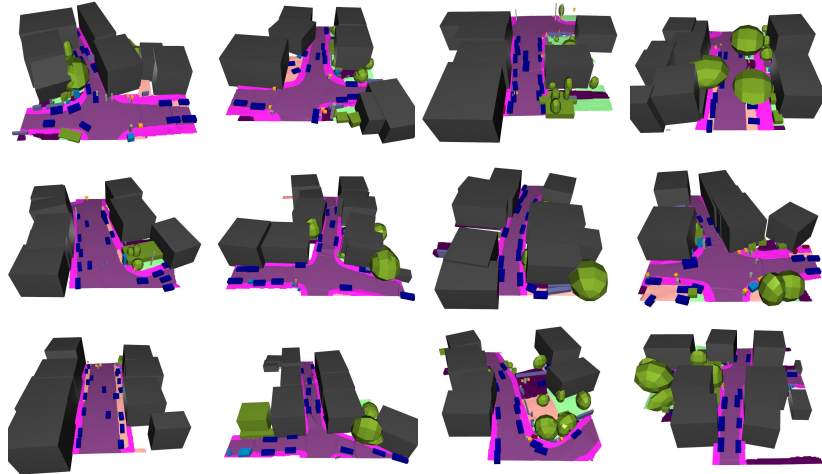


Figure 17. **PrITTI Joint Vegetation–Vehicle Density-Conditioned Scene Generation.** Generated scenes conditioned on combinations of vegetation and vehicle densities. From the nine possible settings, we show three examples: (a) high vegetation, low vehicles, (b) medium vegetation, medium vehicles, and (c) low vegetation, high vehicles, illustrating controllability over combinations of primitive categories.

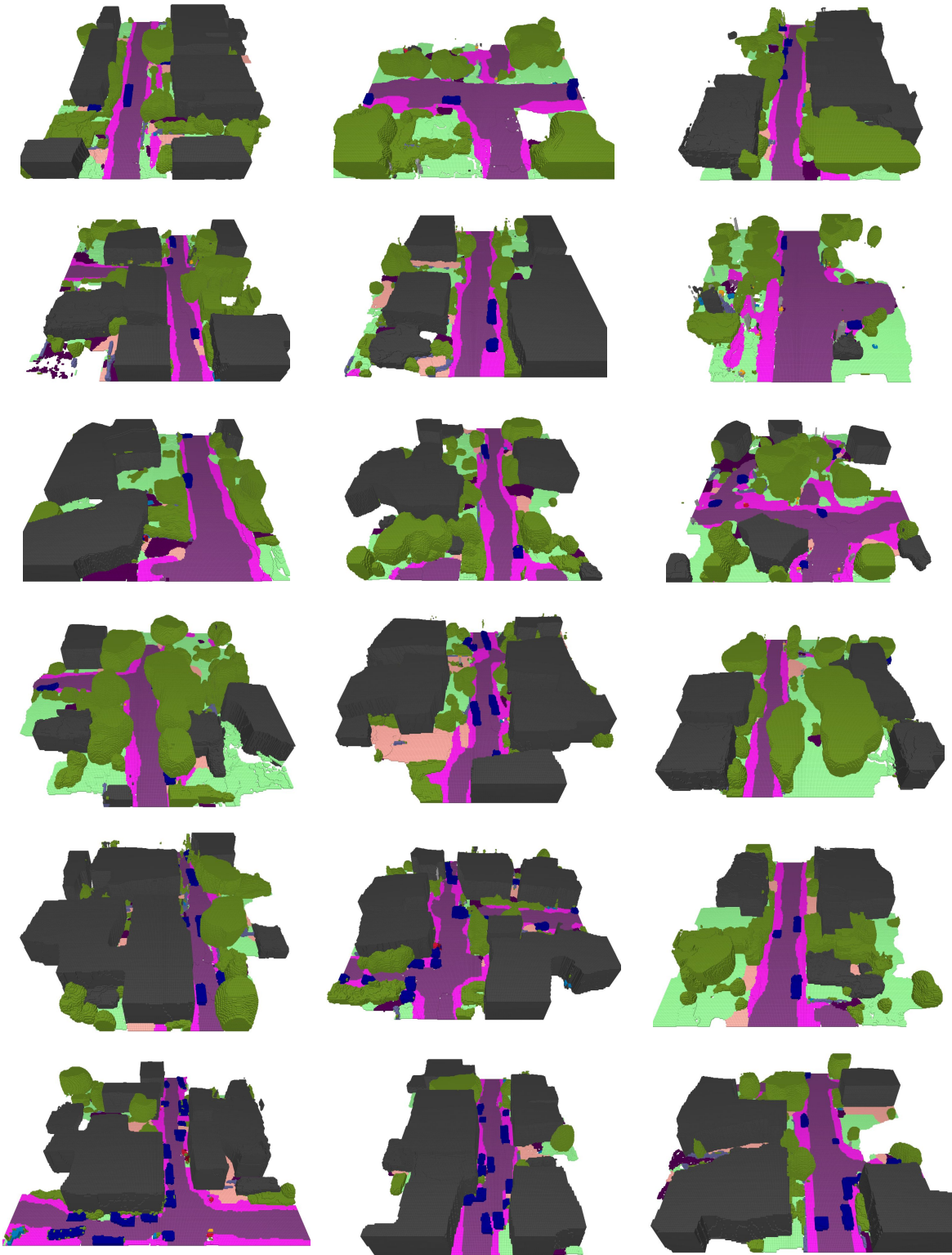


Figure 18. **SemCity Scene Generation Results.** Unconditional 3D semantic scene samples generated by SemCity [38], occasionally exhibiting irregular object shapes, blurred boundaries, and inconsistent semantic structures.

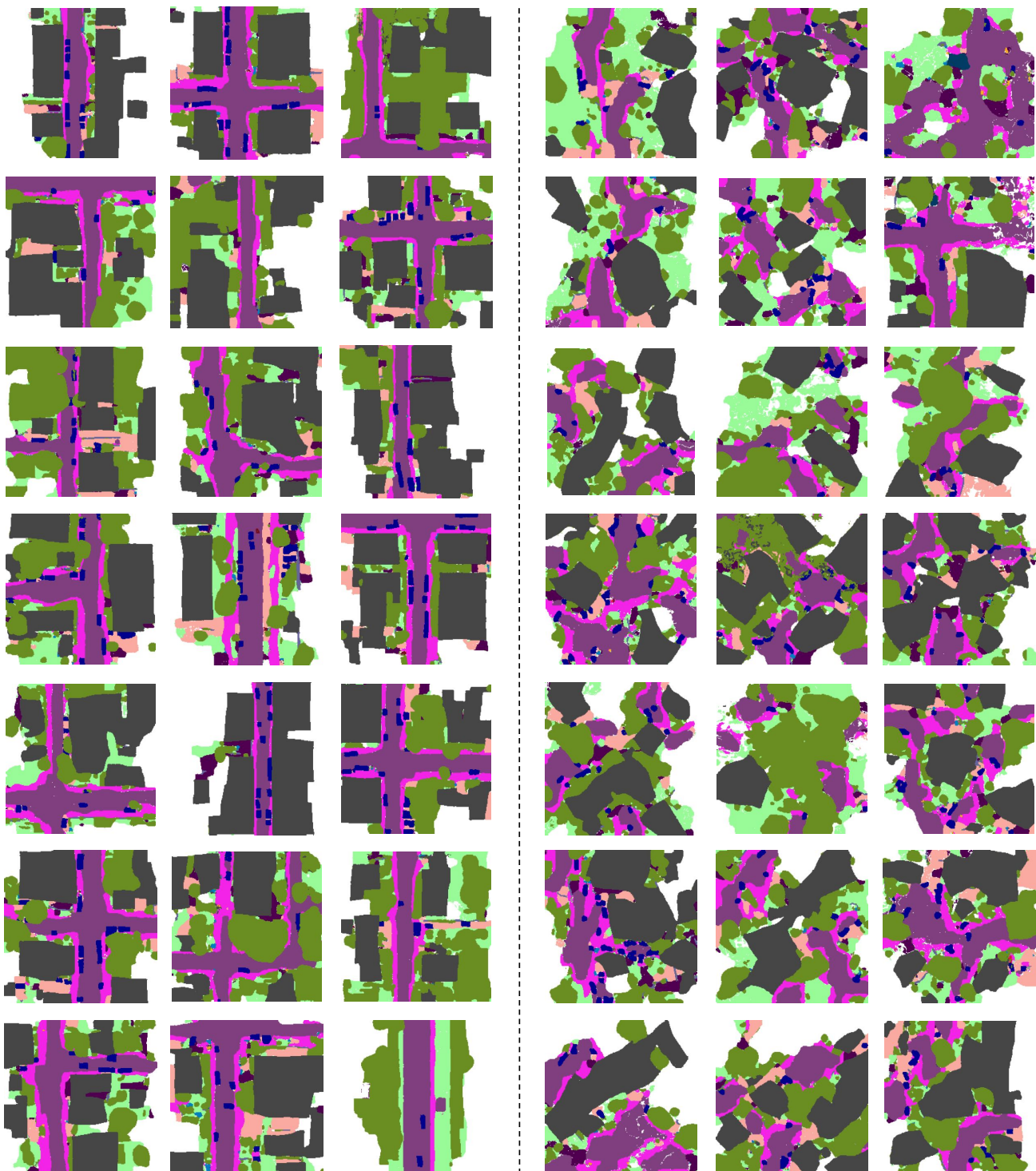


Figure 19. **BEV Renderings of SemCity-generated Scenes.** The examples on the left show relatively well-structured and semantically coherent layouts, while those on the right exhibit fragmented, irregular geometry and inconsistent semantics.

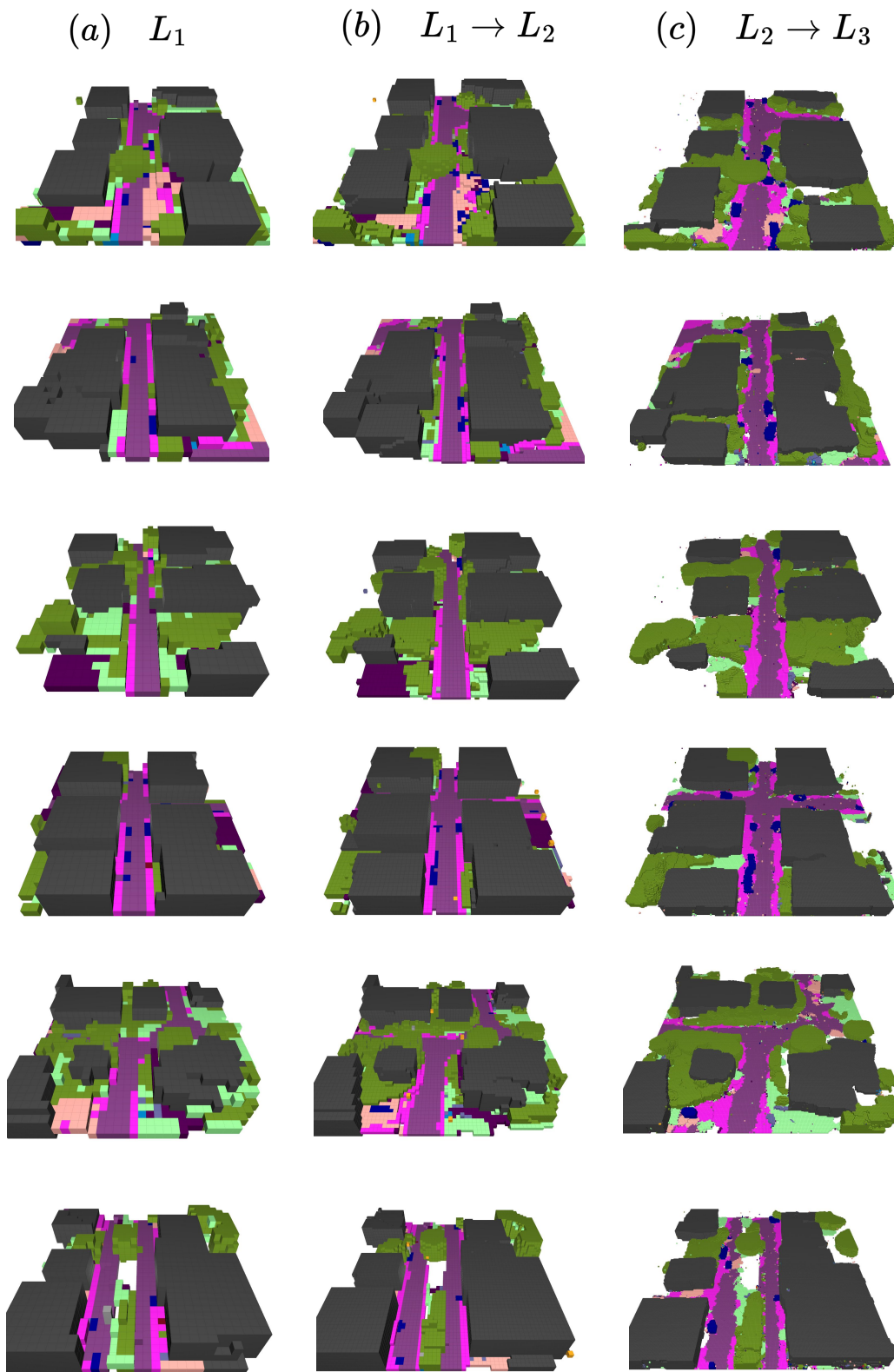


Figure 20. **PDD Hierarchical Scene Generation Results.** Unconditional 3D semantic scenes generated by PDD [44] at successive pyramid levels. While scene details increase across levels, the final stage also introduces noticeable noise and semantic inconsistencies.

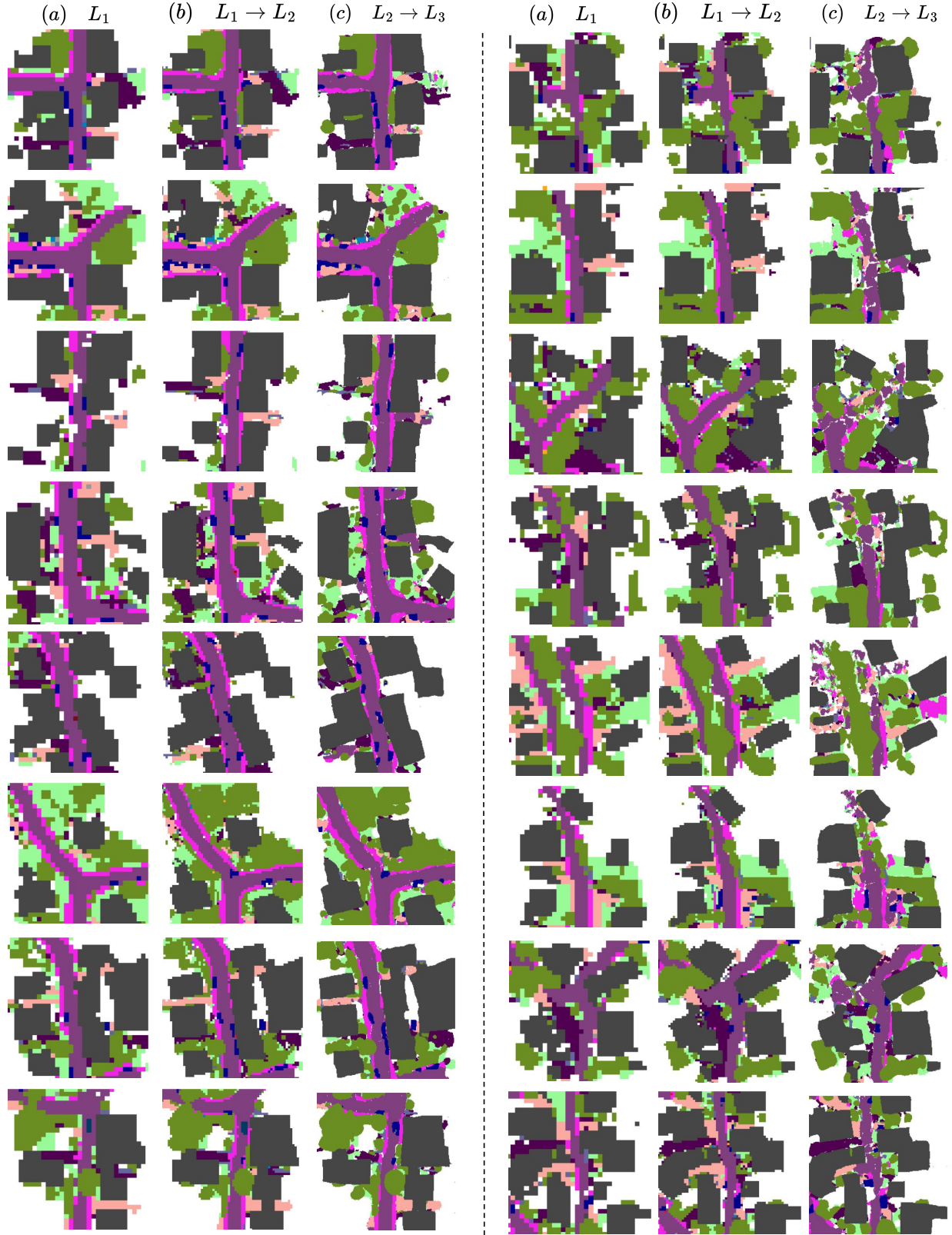


Figure 21. **Hierarchical BEV Renderings of PDD-generated Scenes.** Each column triplet shows results across successive hierarchy levels. Examples on the left exhibit relatively coherent scene structure, whereas those on the right clearly show that the last refinement stage introduces noticeable noise and semantic inconsistencies, thereby reducing the perceived realism of the final outputs.

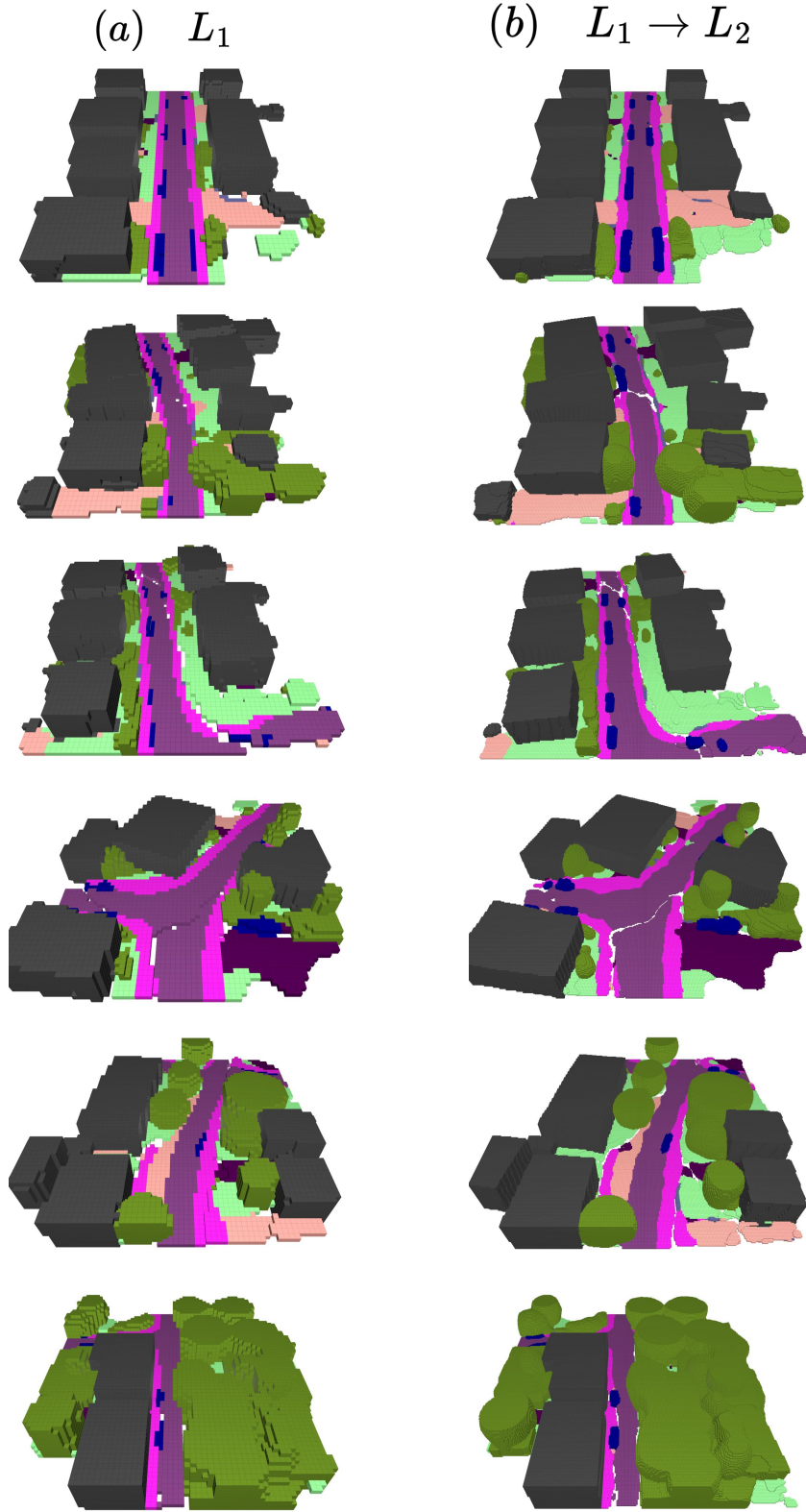


Figure 22. **XCube Hierarchical Scene Generation Results.** Unconditional 3D semantic scenes generated by XCube [64] across successive hierarchy levels. Scene details improve across levels, and the overall structure appears realistic, but irregular object shapes (e.g. vehicles) and occasional road-geometry discontinuities remain visible.

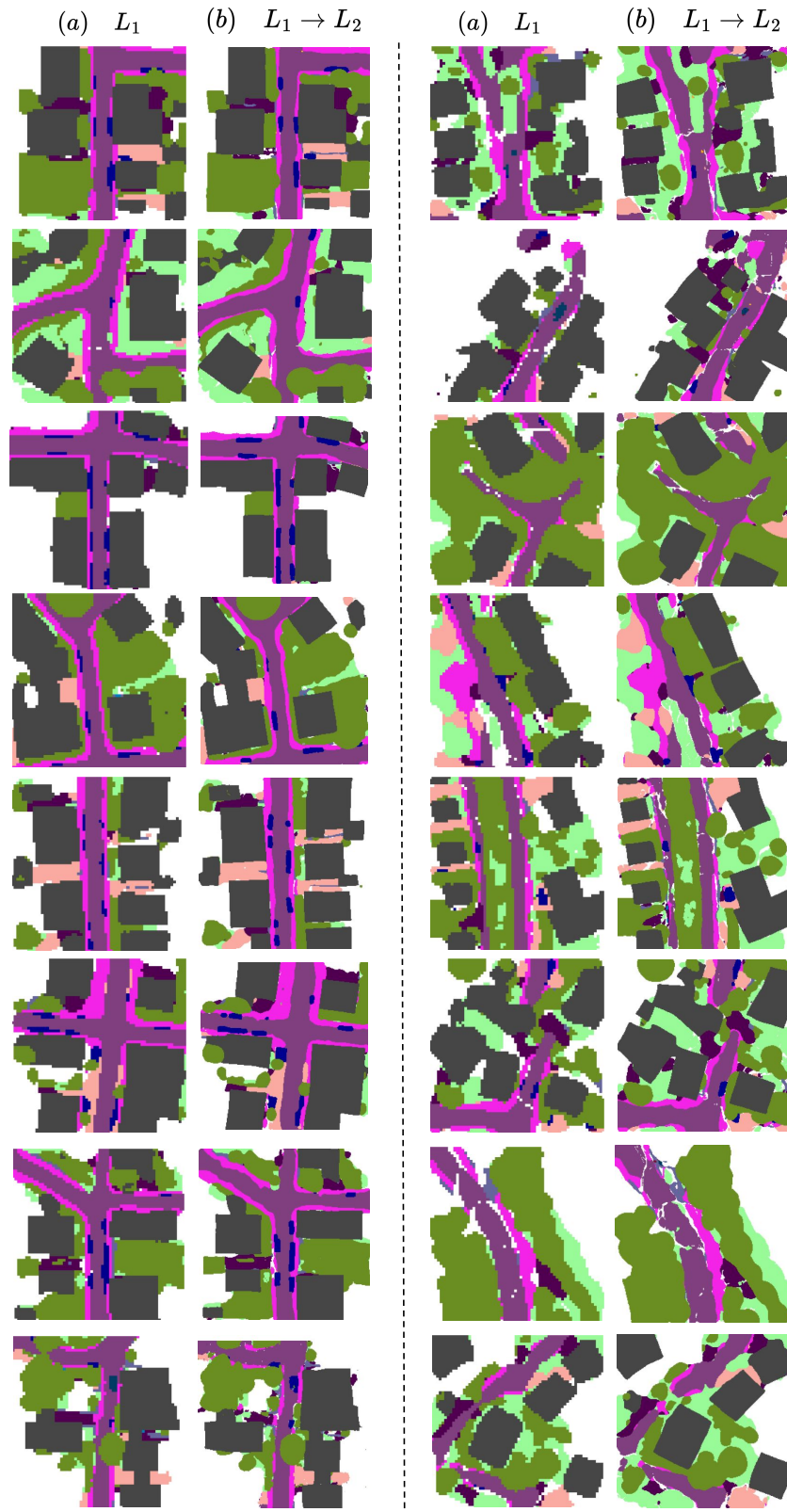


Figure 23. **Hierarchical BEV Renderings of XCube-generated Scenes.** Each column pair shows results across successive hierarchy levels. Examples on the left demonstrate high-quality generations, while those on the right exhibit geometric distortions and semantic inconsistencies.

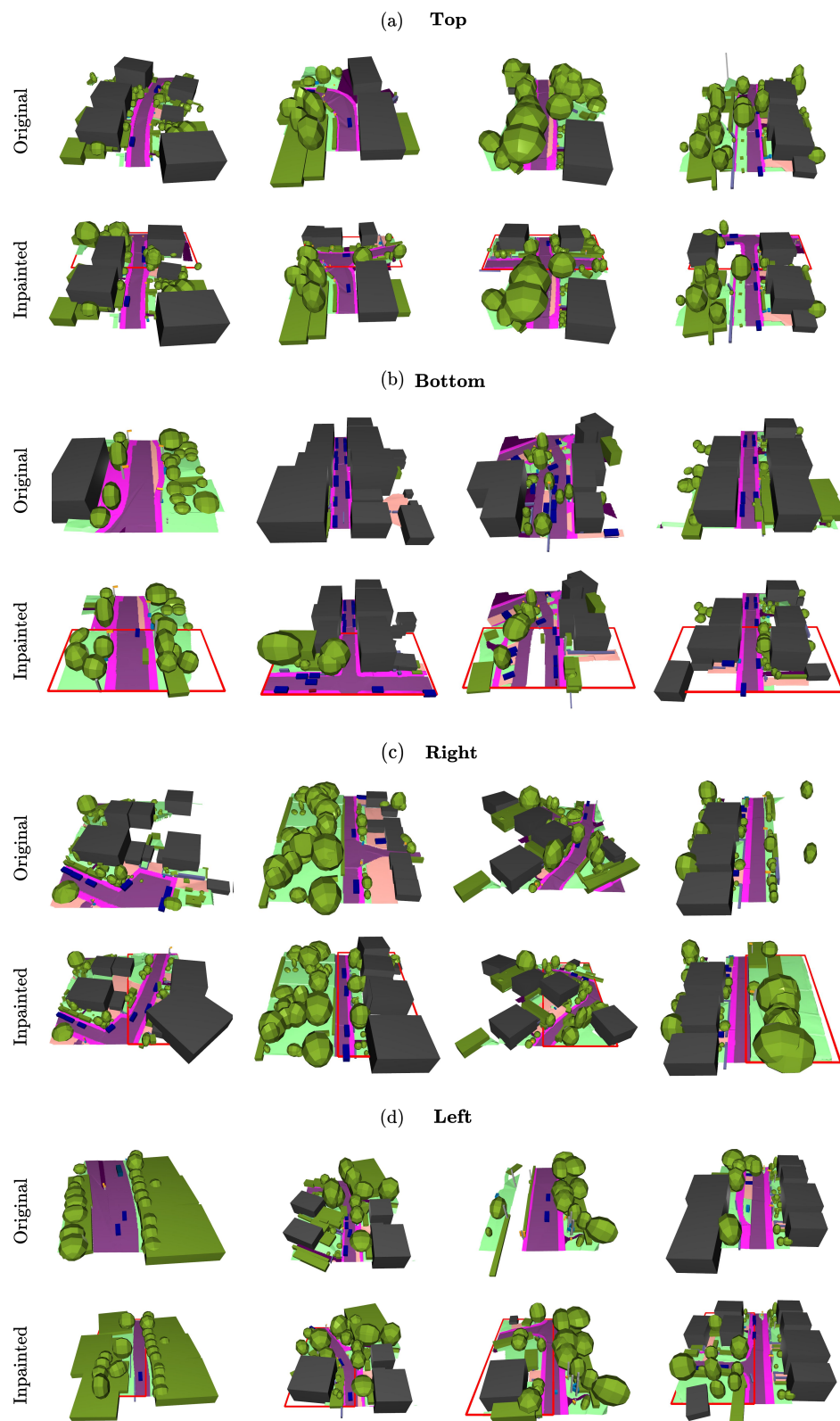


Figure 24. **PrITTI Inpainting Results.** Inpainting applied to (a) top, (b) bottom, (c) right, and (d) left regions of an input scene. PrITTI produces localized edits that blend seamlessly with the surrounding geometry and semantics, yielding consistent and plausible completions.



Figure 25. **PRITTI Ground-to-Object Inpainting Results.** Leveraging our disentangled joint latent structure, we can inpaint object primitives while keeping the ground layout fixed. The results show that the ground geometry is faithfully preserved from the original scene, while newly generated objects are placed in semantically appropriate locations.

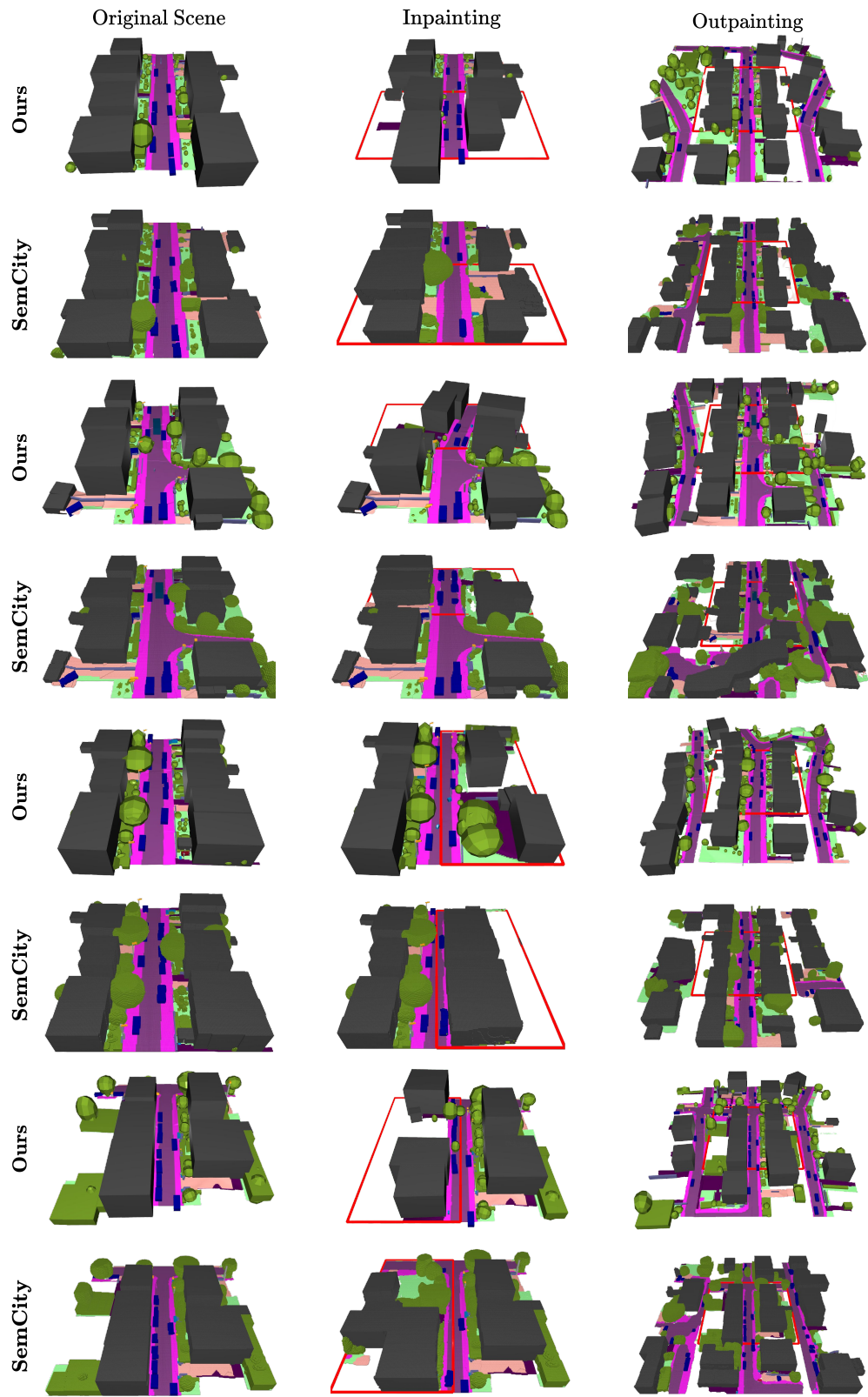


Figure 26. **Comparison of scene inpainting and outpainting results** on identical test scenes using our method and SemCity [38]. PrITI (Ours) yields coherent and semantically consistent edits and extensions, while SemCity often produces fragmented or contextually inconsistent structures, particularly during outpainting.



Figure 27. **PrITTI Outpainting Results.** Outpainting examples produced by our method showing that it can expand a scene beyond its spatial extent while preserving ground layout continuity and generating new objects that integrate naturally with the surrounding context.

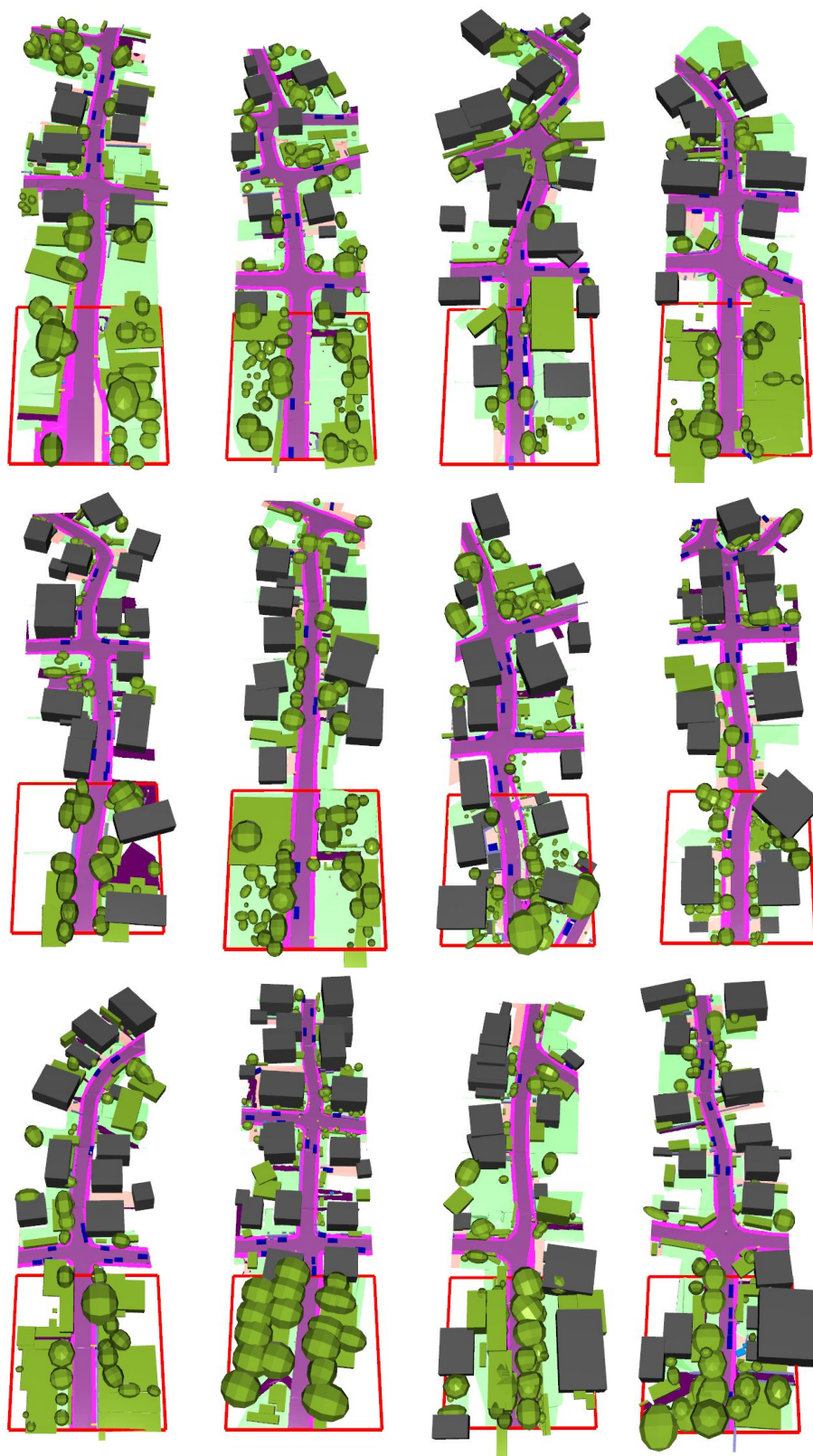


Figure 28. **PrITTI Large-scale Scene Extrapolation Results.** Starting from an initially generated block (outlined in red), our method produces extended layouts that remain globally coherent and semantically consistent, with smooth structural transitions across blocks.

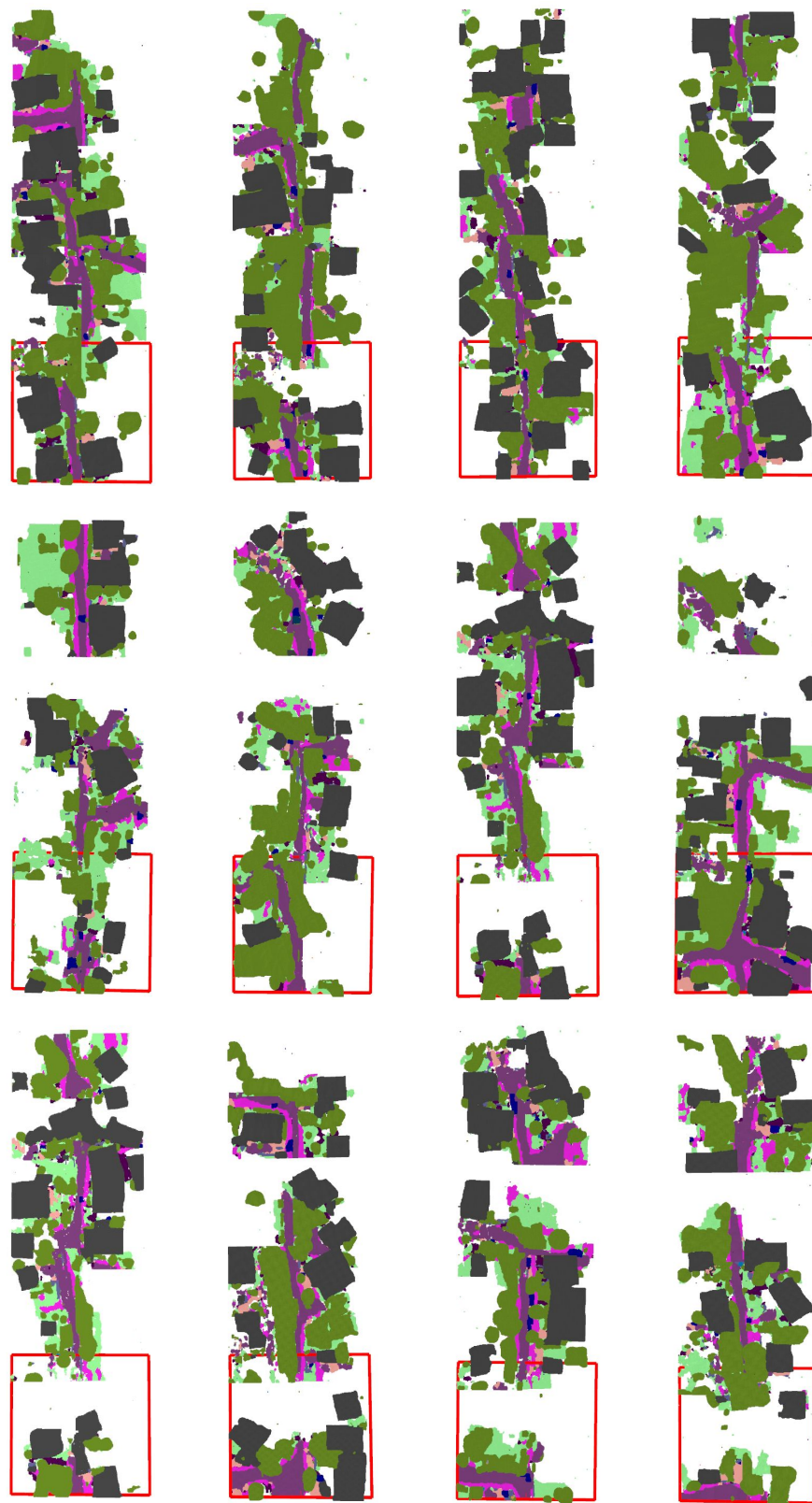


Figure 29. **PDD Large-scale Scene Extrapolation Results.** Starting from an initially generated block (outlined in red), PDD progressively extends the scene upward. The extrapolated regions exhibit broken road connectivity, misplaced structures, and inconsistent semantic transitions, leading to fragmented large-scale layouts.



Figure 30. **PrITTI Object Editing Results.** Enabled by our instance-level primitive representation, PrITTI naturally supports a wide range of individual-object manipulations. Columns (b-e) show dropout, rotation, scaling, and translation applied to the same randomly selected vehicle primitives from (a), highlighted in white.

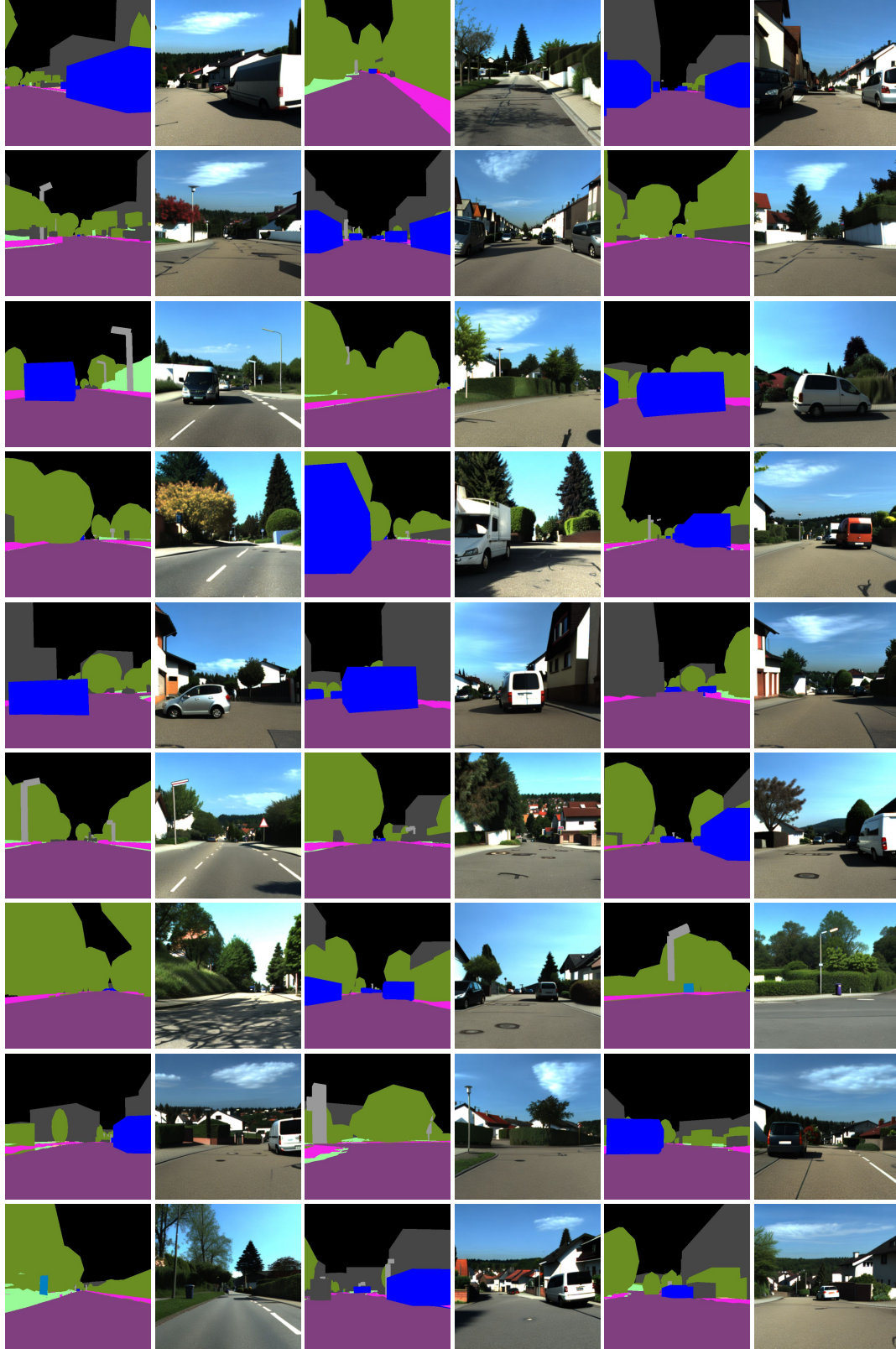


Figure 31. **PrITTI Photo-realistic Street View Synthesis Results.** Each pair shows a 2D semantic rendering (left) and the corresponding photo-realistic image (right), generated with ControlNet [104]. Despite the coarse geometry of our primitives, the synthesized scenes exhibit diverse object appearances that go beyond cuboids or ellipsoids while remaining consistent with the input semantics.