Grounded Latents for Entity-Centric 4D Scene Generation

Jinhyung Park[1] Navyata Sanghvi[1] Erica Weng[1] Shawn Hunt[3] Shinya Tanaka[3]
Hironobu Fujiyoshi[2] Kris Kitani[1]
[1]Carnegie Mellon University [2]DENSO Corporation [3]DENSO International America, Inc.

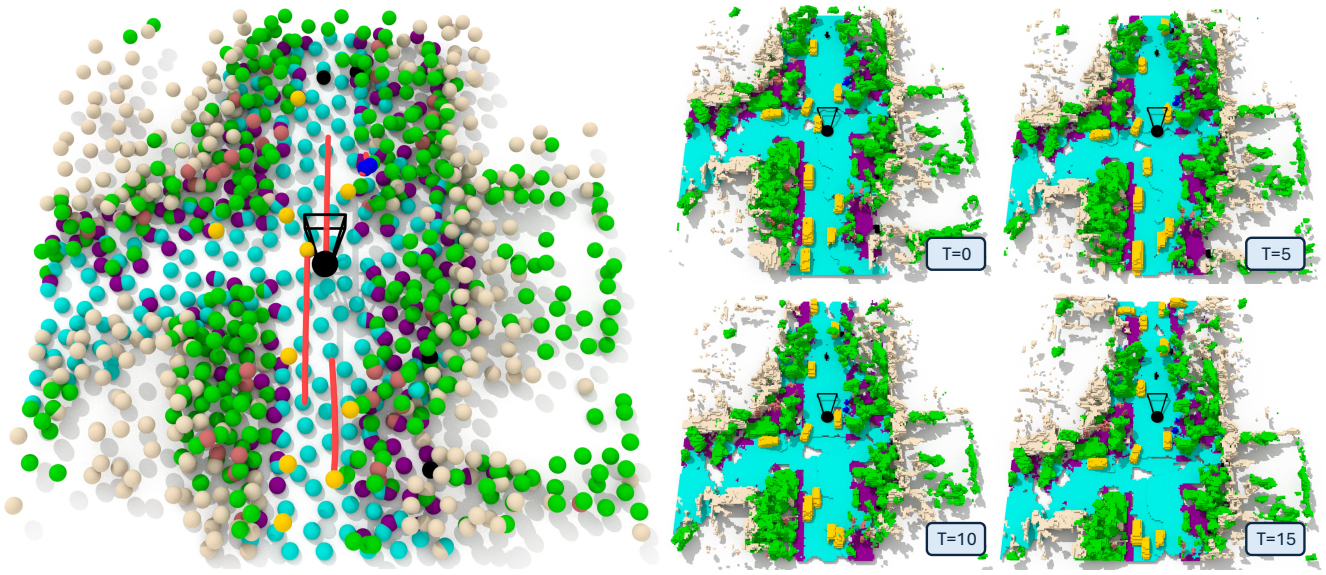

Figure 1. **Entity-centric 4D scene generation with grounded 3D latents.** We present **LatentWorld**, which introduces a grounded latent representation for controllable, interpretable 3D and 4D scene generation. On the left, we show a generated 3D latent layout in which each actor is a single editable latent with position and BEV orientation, together with generated future waypoints that define motion. On the right, we decode the same latents to semantic Gaussians and splat to voxels across future timesteps, producing coherent 4D scenes with precise foreground movement and stable background structure. *Zoom in for detail.*

Abstract

*Although recent work has explored generative modeling of 3D or 4D driving scenes, most approaches operate on dense voxel-based representations, which are computationally expensive and struggle to maintain temporal or structural consistency. These methods often produce blurred or merged entities (i.e., cars, trucks, pedestrians) and lack fine-grained control over individual scene elements. We propose **LatentWorld**, a framework for generative modeling in a compact, entity-centric latent space, where each grounded 3D latent represents a semantically meaningful local region of the scene. This formulation enables precise, consistent control of both foreground and background elements while preserving geometric detail. We further extend this representation to 4D by learning a motion diffusion model for both ego and dynamic actors, conditioned on the generated 3D scene, and by propagating the grounded latents through time. Our framework produces physically consistent and temporally coherent 4D scenes, supporting controllable and realistic generation.*

1. Introduction

Recent advances in generative 3D/4D scene modeling synthesize scenes by denoising in dense voxel/grid representations, producing realistic scene-level generations across driving scenarios [2, 18, 21, 33, 42]. While effective at scene-level generation, representing a scene as a dense, discretized volume provides no explicit modeling of foreground actors or background elements. For instance, a vehicle is a contiguous cluster of voxels without clear separation from its neighbors.

In such a dense voxel formulation, all content is generated jointly within a single discretized volume rather than

via explicit entities; foreground actors (vehicles, pedestrians) are not independently parameterized. As such, placement or manipulation of foreground actors rely on conditioning the generative model with coarse control signals, rather than direct and reliable, actor-specific adjustments. Extended to dynamic scenes, this lack of explicit entity modeling manifests as *merging*, *flickering*, *splitting* of entities in generated sequences.

In this work, we introduce **LatentWorld**, which factorizes generation around a compact, entity-centric latent representation. We express a scene as a sparse set of *grounded 3D latents*, each with (X, Y, Z) coordinates and a semantic class. Foreground actors (*e.g.*, vehicles, pedestrians) are each assigned a single latent for precise control, while multiple background elements (*e.g.*, road, buildings, vegetation) utilize a variable number of latents to capture fine-grained structure and larger spatial extent. Generation proceeds in three stages within a single pipeline: (i) a **layout diffusion** stage generates the latent set ((X, Y, Z) and classes), establishing an interpretable, controllable 3D layout; (ii) a **feature diffusion** stage predicts a per-latent feature that encodes local geometry; and (iii) a **motion diffusion** stage evolves the same latents through time. Each latent is decoded into a small set of semantic Gaussians [14, 15], which we splat to a voxel grid for occupancy training and evaluation. Because latents are *explicitly grounded* at (X, Y, Z) locations, modeling global motion is straightforward: generated ego movement is applied as a rigid transform to the entire latent set's (X, Y, Z) locations, and dynamic actors' latents follow trajectories produced by the motion diffusion model. This contrasts with dense voxel generators, which model ego motion indirectly through scene-wide changes and cannot reliably induce motion by directly transforming independently controllable actors.

This formulation preserves foreground integrity and enables precise control: translating or rotating a single actor is handled by a direct edit to its latent, and background detail is expressed via the density of background latents rather than higher grid resolution. With persistent latents, trajectories remain coherent and editable, and failure modes common to voxel generation such as merging, flickering, and splitting of entities are substantially reduced. Furthermore, because complexity scales with the number of latents rather than grid size, computation naturally concentrates on occupied regions, compared to grid-based methods which misuse compute on free space. Compared to dense grid methods, our generations exhibit more reliable foreground handling and improved temporal stability, which leads to state-of-the-art results on the CarlaSC [8, 45] and Waymo [38, 39] datasets.

Our main contributions are as follows:

- We introduce **LatentWorld**, a novel entity-centric generative framework that represents a scene with sparse grounded 3D latents.
- We factorize 4D scene generation into layout diffusion (XYZ positions and semantic classes), per-latent geometry diffusion (local features), and motion diffusion over a persistent latent set.
- We enable precise, interpretable control of foreground actors and fine-grained background structure, substantially reducing merging, flickering, and splitting artifacts and yielding temporally coherent trajectories.
- Our framework achieves state-of-the-art generation quality on CarlaSC [8, 45] and Waymo [38, 39].

2. Related Work

2.1. 3D Object Generation

3D generation has been extensively explored for single objects [4]. Early approaches used GANs [9] and VAEs [17], while recent work leverages diffusion [12] and flow matching [20]. These pipelines span diverse 3D representations, including sparse point clouds [24, 28, 53, 54], flexible query sets [55], explicit voxel grids [25, 26, 47, 48], and factorized triplanes [37, 43]. Although these methods produce high-fidelity 3D assets, they are largely object-centric rather than scene-centric [33, 49], limiting their applicability to autonomous driving. We instead target realistic, dynamic scenes and represent them with sparse, grounded 3D latents that capture multi-entity structure and relationships.

2.2. 3D Scene Generation

Beyond single objects, autonomous driving works have recently adopted semantic occupancy [39, 40] as the central 3D state, as it captures scene geometry and semantics for high-fidelity perception [39, 40] and integrates naturally with planning [40, 56]. While LiDAR sequence synthesis is an active line of work [13, 27, 30, 32, 50, 57], we focus on semantic occupancy because it models unseen extents and fine semantics. Recent static 3D scene generators operate directly on semantic occupancy grids, often training a VAE or VQ-VAE [17, 41] and denoising in the learned latent space [25, 34]. SemCity [18] uses triplane diffusion for efficient 3D modeling, XCube [33] and PDD [21] adopt coarse-to-fine generation, and InfiniCube [23] and Control-3D-Scene [35] introduce diverse controls. These approaches excel at layout and static geometry but rely on grid-aligned intermediates without explicit entity factorization. We instead develop a grounded 3D latent representation designed for downstream 4D motion.

2.3. 4D Scene Generation

Extending from 3D to 4D, the goal is to synthesize scene evolution over time. One line of work addresses 4D occupancy *forecasting* [56], typically conditioned on current observations and planned trajectories [10, 36, 42, 44, 56].

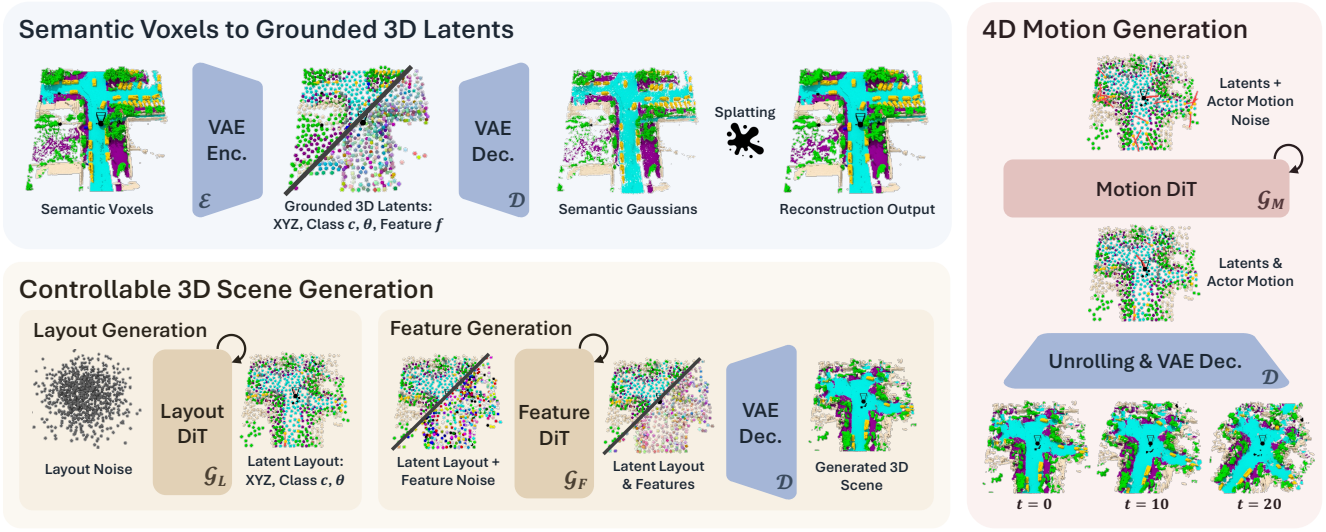

Figure 2. Method Overview. **Semantic Voxels to Grounded 3D Latents:** A VAE encodes semantic voxels into a sparse, editable latent point set $\mathcal{Z} = \{(\mathbf{x}, c, \theta, \mathbf{f})\}$, and the decoder maps latents to semantic Gaussians for voxel splatting. **Controllable 3D Scene Generation:** A layout diffusion transformer G_L generates the latent layout (positions, classes, orientations), then a feature diffusion transformer G_F predicts per-latent features to capture fine geometry; decoding yields a realistic 3D scene. **4D Motion Generation:** A motion diffusion transformer G_M produces future ego and actor trajectories; we move the corresponding latents and unroll the decoder across timesteps to obtain coherent 4D semantic occupancy.

Another line focuses on 4D occupancy *generation* [2, 42] for open-ended scene simulation. In this latter setting, OccSora [42] and UniScene [19] compress scenes into dense grid-aligned latents for denoising; DrivingSphere [51] generates static scenes similarly, then populates foreground actors and dynamics using a traffic simulator. DynamicCity [2] decomposes scenes into HexPlane features [3], achieving more efficient, higher-fidelity synthesis than dense voxels [42]. In contrast to voxel-centric pipelines, our flexible, grounded 3D latents support direct, interpretable edits and couple naturally with our motion diffusion model, enabling ego- and actor-level control without external heuristic-based controllers.

3. Method

3.1. Preliminary: Gaussians Occupancy Modeling

We draw from GaussianFormer [14, 15] and model a scene as semantic 3D Gaussians $\mathcal{G} = \{\mathbf{G}_i\}_{i=1}^{P}$, where each $\mathbf{G}_i$ has center $\mathbf{x}_i$, rotation $\mathbf{r}_i$, scale $\mathbf{s}_i$, opacity a_i, and class probabilities $\mathbf{c}_i \in \mathbb{R}^C$. The covariance is $\mathbf{\Sigma}_i = \mathbf{R}(\mathbf{r}_i)\,\mathrm{diag}(\mathbf{s}_i)^2\,\mathbf{R}(\mathbf{r}_i)^\top$. For a voxel center $\mathbf{x}$, occupancy is the probability that at least one Gaussian explains $\mathbf{x}$:

$$\alpha(\mathbf{x}) = 1 - \prod_{i=1}^{P} \big(1 - \exp(-\tfrac{1}{2}(\mathbf{x}-\mathbf{x}_i)^\top \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{x}_i))\big). \quad (1)$$

The voxel semantics is an opacity- and density-weighted mixture of the semantics of nearby Gaussians,

$$\mathbf{e}(\mathbf{x}) = \frac{\sum_i p(\mathbf{x} \mid \mathbf{G}_i)\, a_i\, \mathbf{c}_i}{\sum_j p(\mathbf{x} \mid \mathbf{G}_j)\, a_j}. \quad (2)$$

Figure 3. Rotating foreground Gaussians by their latent's yaw enables interpretable, reliable heading control.

where $p(\mathbf{x} \mid \mathbf{G}_i) \propto \exp\big(-\tfrac{1}{2}(\mathbf{x}-\mathbf{x}_i)^\top \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{x}_i)\big)$. We express the voxel's distribution empty space and semantic classes as $[1-\alpha(\mathbf{x}); \alpha(\mathbf{x})\,\mathbf{e}(\mathbf{x})] \in \mathcal{R}^{(1+C)}$.

3.2. Controllable Grounded Scene Latents

Representation. To support editable 4D scenes with inherent, interpretable control over both foreground actors and background structure, we replace dense grid–aligned latents with a sparse, grounded latent point set

$$\mathcal{Z} = \{z_n\}_{n=1}^{N}, \qquad z_n = \big(\mathbf{x}_n,\ c_n,\ \theta_n,\ \mathbf{f}_n\big),$$

where $\mathbf{x}_n \in \mathbb{R}^3$ is the position, $c_n \in \{1, \ldots, C\}$ is the semantic class, $\theta_n \in \mathbb{R}$ is the BEV yaw (used for foreground actors), and $\mathbf{f}_n \in \mathbb{R}^D$ encodes local geometry. We assign exactly one latent to each foreground actor to maintain identity and enable direct control: multiple latents per object tend to split an actor over time, and one latent explaining multiple objects entangles motion and causes merging. Background regions are covered by multiple latents to cap-

ture detail and allow localized edits. To derive this representation, we use a VAE with an encoder $\mathcal{E}$ that maps semantic voxels to $\mathcal{Z}$, and a decoder $\mathcal{D}$ that maps $\mathcal{Z}$ to semantic Gaussians as shown in Figure 2 (top).

Encoding to the latent point set. Given a semantic occupancy grid $\mathbf{V} \in \{0, \dots, C\}^{X \times Y \times Z}$, we compute per-voxel features $\mathbf{F}$ using a sparse voxel transformer with shifted 3D windows [22, 48, 52]. To obtain latent positions, we take the 3D center of each foreground instance and apply furthest-point sampling (FPS) over voxels from background regions. Each sampled point indexes its voxel feature in $\mathbf{F}$ to obtain $\mathbf{f}_n$. Latents take the class c_n from the corresponding voxels, and foreground latents also take the instance yaw θ_n.

Decoding to semantic occupancy. Given $\mathcal{Z}$, the decoder predicts a set of semantic Gaussians per latent, then splats them to the voxel grid. We encode position $\mathbf{x}_n$ and yaw θ_n with positional encodings, map class c_n and feature $\mathbf{f}_n$ with linear layers, and sum these embeddings as input to a transformer. The transformer then predicts for each latent z_n a set $\{(\Delta\mathbf{m}_{n,k}, \mathbf{r}_{n,k}, \mathbf{s}_{n,k}, a_{n,k})\}_{k=1}^{K_n}$ of Gaussians (offsets, orientations, scales, opacities). Each Gaussian is assigned the class of its latent parent c_n. We place Gaussian centers using the predicted offsets relative to the latent. For background latents we apply the offsets directly: $\mathbf{m}_{n,k} = \mathbf{x}_n + \Delta\mathbf{m}_{n,k}$. For foreground latents, we apply the latent's yaw θ_n to both offsets and orientations: $\mathbf{m}_{n,k} = \mathbf{x}_n + \mathbf{R}(\theta_n)\,\Delta\mathbf{m}_{n,k}$ and $\mathbf{r}_{n,k} = \mathbf{R}(\theta_n) \circ \tilde{\mathbf{r}}_{n,k}$. This ensures the user-specified yaw is directly reflected in the generated geometry, and it natively enables orientation control for downstream motion. In Figure 3, we demonstrate this rotation control on a vehicle latent. Finally, we convert the predicted Gaussians to a reconstructed semantic occupancy grid $\hat{\mathbf{V}}$ via Gaussian-to-occupancy splatting (Sec. 3.1).

Training. We train the VAE with cross-entropy ($\mathcal{L}_{\text{CE}}$), Lovász ($\mathcal{L}_{\text{Lovasz}}$), and β-weighted KL regularization:

$$\mathcal{L}(\hat{\mathbf{V}}, \mathbf{V}) = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{Lovasz}} + \beta\,\mathcal{L}_{\text{KL}}(\mathbf{f}). \tag{3}$$

This objective yields a compact, controllable latent point set where translating a latent moves the corresponding structure, and adjusting a foreground latent's yaw rotates its predicted geometry.

3.3. Generating a Controllable 3D Scene

Given our grounded latent representation, we separate 3D scene generation into two stages as shown in Figure 2 (bottom). In the first stage, we generate the overall layout of the latent set $\mathcal{Z}$: the positions $\mathbf{x}$, semantic classes c, and orientations θ. In the second stage, we condition on this layout to generate per-latent deep features $\mathbf{f}$ which encode fine-grained structure.

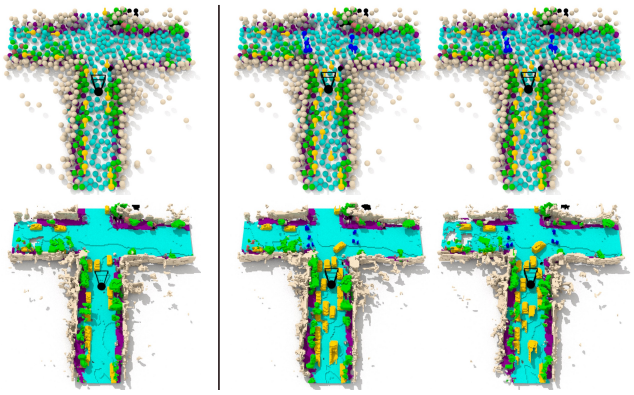

Figure 4. *Left:* Generated layout and semantic grid. *Right:* We manually arrange the layout to simulate a complex traffic scene and sample two sets of latent features to decode. LatentWorld's grounded latent representation enables interpretable, explicit control of scene elements, with high-fidelity geometric variations captured by our feature generation model.

We use two stages for several reasons. *First*, prior work shows that producing scene structure before local detail improves fidelity [48]. *Second*, we want the layout to be interpretable and editable: a user can inspect latent placements, move or rotate elements, and then produce high-fidelity geometry that follows these edits. *Third*, keeping coarse layout and fine geometry separate allows the same layout to be paired with multiple fine-grained generations, giving users coarse control together with fine-grained diversity.

Layout Generation To generate this layout, we use a layout diffusion transformer $\mathcal{G}_L$ operating on the latent point set. For each latent n, we encode its fields for diffusion as

$$\bar{\mathbf{z}}_{n,0} = \big[X_n,\, Y_n,\, Z_n,\, \sin\theta_n,\, \cos\theta_n,\, \text{bits}(c_n)\big]. \tag{4}$$

We normalize (X, Y, Z) to $[-1, 1]$ using dataset bounds and encode yaw as $(\sin\theta, \cos\theta)$. For classes, we follow Bit Diffusion [1] and convert $c_n \in \{1, \dots, C\}$ to $\lceil \log_2 C \rceil$ bits so classes are modeled within the same continuous diffusion as the real-valued channels. We do this because unlike PDD [21] which applies only discrete diffusion on a fixed voxel grid and DynamicCity [2] which uses only continuous diffusion on a latent feature space without explicit classes, our layout includes both continuous $(X, Y, Z, \sin\theta, \cos\theta)$ and discrete (class) components. Converting classes to bits enables a single continuous diffusion schedule. We train $\mathcal{G}_L$ with the ϵ-prediction objective following [12, 29]:

$$\mathcal{L}_{\text{layout}} = \mathbb{E}_{t,\epsilon}\left\|\epsilon - \mathcal{G}_L(\bar{\mathbf{z}}_{n,t}, t)\right\|_2^2 \tag{5}$$

with $\bar{\mathbf{z}}_{n,t} = \sqrt{\bar{\alpha}_t}\,\bar{\mathbf{z}}_{n,0} + \sqrt{1-\bar{\alpha}_t}\,\epsilon$, and sample layouts with the reverse process. This yields editable, semantically grounded layouts that provide a reliable scaffold for the subsequent per-latent geometry and feature generation. Figure

4 illustrates this process: we first generate a layout, edit specific entities (e.g., translate or rotate actors) as desired, then generate per-latent features and decode to semantic occupancy. The resulting semantics adhere to the edited layout, demonstrating precise, interpretable control.

Feature Generation. Given the coarse layout $\{(\mathbf{x}_n, c_n, \theta_n)\}_{n=1}^{N}$ from $\mathcal{G}_L$, we generate a deep feature $\mathbf{f}_n$ for each latent to capture local geometry. We employ a diffusion transformer $\mathcal{G}_F$ that conditions on the layout: at each timestep it receives the current noisy feature $\bar{\mathbf{f}}_{n,t}$ together with embeddings of the latent's position $\mathbf{x}_n$, orientation θ_n, and class c_n from the previous layout stage, and uses this context to denoise $\bar{\mathbf{f}}_{n,t}$ into $\mathbf{f}_n$ while leaving the layout unchanged. Combining the layout $(\mathbf{x}_n, c_n, \theta_n)$ with the generated features $\mathbf{f}_n$ yields the grounded latent set passed to $\mathcal{D}$ for Gaussian decoding and occupancy splatting. This simple conditioning cleanly separates editable layout from fine detail, allowing multiple feature realizations per layout as shown in Figure 4.

3.4. Generating Dynamic Scenes

Motion Generation. In driving scenarios, scene dynamics primarily come from two sources: the ego vehicle's motion and the motion of dynamic actors. We generate trajectories for both with a diffusion transformer $\mathcal{G}_M$. In grid-based methods, ego motion is implicit in synthesized frames and actor motion is baked into voxels; steering requires conditioning on control trajectories that the generator may not reliably respect because there are no explicit, controllable entities. In our representation, motion is explicit edits to latents. Applying an ego transform moves the entire scene consistently, and updating a single foreground latent places that actor exactly at the desired waypoint and heading. Users may edit latents directly, or $\mathcal{G}_M$ can learn to generate realistic trajectories from data.

We now formalize our model $\mathcal{G}_M$ as shown in Figure 2 (right). We identify dynamic foreground classes (e.g., vehicles, pedestrians, cyclists) and include the ego vehicle as an agent. The model generates $T{=}20$ future steps at 10Hz. For each agent a, it outputs future waypoints and headings in the current ego frame,

$$\{(\mathbf{p}_{a,t},\, \phi_{a,t})\}_{t=1}^{T}, \qquad \mathbf{p}_{a,t} \in \mathbb{R}^3,\ \phi_{a,t} \in \mathbb{R}, \tag{6}$$

where $\mathbf{p}_{a,t}$ is an (X, Y, Z) waypoint and $\phi_{a,t}$ is a BEV heading angle. We form a sequence of agent–timestep tokens where each token's input feature is the sum of 1) an embedding of the noised $(\mathbf{p}_{a,t}, \phi_{a,t})$ to be denoised, 2) a time-step embedding for t, and 3) the agent identity from its latent $(\mathbf{x}_a, \theta_a, c_a, \mathbf{f}_a)$. The denoiser cross-attends to all current-time latents (foreground and background) for scene context. For autoregressive generation, each agent is AdaLN-conditioned [31] on an encoding of its past 10 steps (10Hz), dropped 10% of the time for unconditional sampling. $\mathcal{G}_M$ uses the same ϵ-prediction objective and schedule as in other models.

Driving Latents with Motion. To generate 4D sequences, we update actors first, then advance the full scene by the ego transform. At each future step t, dynamic actors move to their predicted waypoints and headings (in the current ego frame), background latents remain in place, and the ego transform is applied uniformly to all latents:

$$(\mathbf{x}_n^t,\, \theta_n^t) = \begin{cases} \big(\mathbf{R}_{\text{ego}}^{(t)}\mathbf{p}_{n,t} + \mathbf{t}_{\text{ego}}^{(t)},\ \phi_{n,t} + \Delta\theta_{\text{ego}}^{(t)}\big) \\ \qquad \text{if latent } n \text{ is dynamic,} \\ \big(\mathbf{R}_{\text{ego}}^{(t)}\mathbf{x}_n^{t-1} + \mathbf{t}_{\text{ego}}^{(t)},\ \theta_n^{t-1} + \Delta\theta_{\text{ego}}^{(t)}\big) \\ \qquad \text{otherwise.} \end{cases} \tag{7}$$

where $\mathbf{R}_{\text{ego}}^{(t)} = \mathrm{Rot}\big(\Delta\theta_{\text{ego}}^{(t)}\big)$. With latents $\{(\mathbf{x}_n^t, c_n, \theta_n^t, \mathbf{f}_n)\}$, we decode via $\mathcal{D}$ to semantic Gaussians and splat to occupancy at time t. Because our actor motion is not baked into voxels, such control is straightforward; Figure 1 demonstrates this operation.

Outpainting for Unbounded Generation. As the ego vehicle advances, the scene must extend beyond the initially generated window. Grid-based methods can hold a BEV crop fixed and denoise only the unknown area. With points, however, naively denoising new latents lets them appear anywhere, risking interference with content already generated and yielding sparse, underpopulated new regions. To address this challenge, we freeze existing latents and denoise only *new* latents to populate the *new forward half* of the BEV window while using the rear half as context. During denoising we (i) clip new latents to the forward half and (ii) apply a quadratic mean shift away from the frontier, inspired by guided diffusion [7]. Let $\tilde{x}$ denote a new latent's x coordinate (the ego vehicle faces $+x$). During diffusion, we adjust the model mean for the forward coordinate by

$$\mu' \;=\; \mu \;+\; \eta\,\lambda\left(1 - \mathrm{clip}(\tilde{x}, 0, 1)^2\right), \tag{8}$$

where μ is the denoiser's predicted mean at the current denoising step, η follows the diffusion variance schedule, and λ is a push weight. This in-loop mean-shift guidance concentrates *new* latents in the forward half while leaving the context half unchanged, enabling automatic, stable outpainting as the ego vehicle moves forward. Additional analysis is in the ablations and the supplement.

4. Experiments

4.1. Experimental Details

Datasets. We evaluate on two datasets. CarlaSC [8, 45] is a synthetic dataset that provides 10 classes with sparse semantic occupancy grids of size $128 \times 128 \times 8$, covering a

Table 1. **CarlaSC scene generation.** MMD↓ between generated and real features using geometry, semantics, and joint (geometry+semantics) metrics; lower is better. LatentWorld consistently achieves the best performance, with large gains on foreground classes.

Method	Avg	All	Building	Barrier	Other	Pedestrian	Pole	Road	Ground	Sidewalk	Vegetation	Vehicle
Geometry												
SemCity [18]	10.47	9.07	9.42	**27.31**	8.18	9.17	**4.56**	**6.06**	23.39	4.90	17.38	8.30
PDD [21]	12.36	8.83	10.90	50.94	7.14	10.91	5.88	18.98	15.15	6.15	**7.35**	25.45
DynamicCity [2]	20.45	19.66	7.01	28.96	13.54	11.38	17.54	30.27	40.52	15.94	22.14	25.09
LatentWorld (Ours)	**6.44**	**3.89**	**4.57**	30.13	**5.22**	**6.42**	5.38	6.80	**11.85**	**2.99**	11.30	**5.34**
Semantics												
SemCity [18]	15.90	10.81	31.00	51.95	**13.71**	31.36	12.94	10.58	8.82	11.94	19.07	18.49
PDD [21]	13.17	8.60	29.19	**42.95**	14.56	24.35	9.64	**6.18**	7.50	11.04	10.59	21.41
DynamicCity [2]	12.74	6.96	26.42	44.91	19.79	26.84	10.97	8.78	8.73	9.37	11.39	17.98
LatentWorld (Ours)	**11.30**	**6.81**	**24.62**	43.38	16.40	**18.63**	**9.58**	7.44	**5.95**	**8.17**	**9.12**	**14.54**
Geometry & Semantics												
SemCity [18]	10.04	3.90	13.04	51.92	13.96	24.60	8.20	4.47	12.93	7.73	15.22	9.84
PDD [21]	13.27	5.89	14.48	81.84	12.77	25.26	6.68	6.68	12.38	10.77	**12.68**	22.98
DynamicCity [2]	9.98	4.13	10.61	**44.44**	12.65	14.61	9.07	9.61	17.83	4.93	19.13	15.46
LatentWorld (Ours)	**6.69**	**1.70**	**8.09**	48.15	**9.52**	**8.87**	**5.41**	**3.56**	**8.77**	**4.03**	14.11	**6.35**

Table 2. **Waymo 4D scene generation.** MMD↓ under geometry, semantics, and joint metrics. LatentWorld outperforms DynamicCity overall, with clear improvements on foreground categories.

Method	Avg	All	Gen. Obj.	Vehicle	Pedestrian	Sign	Cyclist	Traff. Light	Pole	Cons. Cone	Bicycle	Motorcycle	Bldg.	Veg.	Tree Trunk	Road	Walk.
Geometry																	
DynamicCity [2]	3.93	1.20	3.56	3.46	4.77	2.54	16.23	2.96	2.71	34.83	11.44	8.49	**0.48**	0.91	4.15	2.51	0.91
LatentWorld (Ours)	**1.62**	**0.19**	**1.98**	**2.31**	**0.92**	**0.26**	**14.05**	**1.59**	**0.51**	**11.64**	**3.55**	**6.66**	0.63	**0.38**	**0.55**	**0.51**	**0.24**
Semantics																	
DynamicCity [2]	3.32	0.80	6.55	2.77	6.19	3.46	**9.34**	4.33	4.48	9.26	24.51	10.25	1.27	1.34	1.47	1.21	1.25
LatentWorld (Ours)	**1.50**	**0.29**	**1.88**	**1.38**	**1.91**	**0.46**	10.91	**1.07**	**0.66**	**3.64**	**11.28**	**5.14**	**0.50**	**0.38**	**0.48**	**0.53**	**0.48**
Geometry & Semantics																	
DynamicCity [2]	2.61	1.63	2.09	1.82	2.16	1.50	7.35	1.64	1.62	9.36	9.11	7.77	0.76	1.38	3.32	2.66	1.31
LatentWorld (Ours)	**0.96**	**0.16**	**0.81**	**0.71**	**0.60**	**0.27**	**7.27**	**1.00**	**0.37**	**2.79**	**6.17**	**4.48**	**0.46**	**0.29**	**0.51**	**0.25**	**0.22**

$51.2\times51.2\times3$ m region centered on the ego vehicle. Occ3D-Waymo [38, 39] is a large-scale, real-world dataset generated by voxelizing annotated LiDAR point clouds. It provides 15 semantic classes with grids of $200\times200\times16$ resolution, spanning $80\times80\times6.4$ m and is a challenging testbed for 3D/4D generation. Due to computational constraints, we temporally subsample Waymo to 2Hz to train baselines and our framework.

Evaluation Settings. Following prior work [2, 21], we compare feature distributions of generated vs. real scenes using a pretrained 3D autoencoder [6]. We make two improvements for finer-grained evaluation. First, we compute metrics per semantic class. The encoder maps an $X\times Y\times Z$ scene to a BEV feature map $(X/d)\times(Y/d)\times C$. For each BEV cell, we identify the classes present in its covered 3D region, gather real and generated features per class, and then average across classes. More details are in the supplement. This reduces background bias and preserves instance-level detail compared to globally pooling the bottleneck. Second, we use multiple encoders to capture different aspects: a geometry-only autoencoder to assess occupancy/shape fidelity, a semantics-only autoencoder to assess semantic plausibility regardless of exact geometry, and a joint geometry+semantics autoencoder provides an overall holistic metric. Distributional differences are measured with Maximum Mean Discrepancy (MMD), following prior work [5, 16] that show FID [11] is unreliable due to non-Gaussian feature embeddings. We report both averaged and per-class MMD.

Implementation Details. We closely follow the settings in prior work [2], training our VAE for 20 epochs and the DiT for 1200 epochs. We use 768 latents for CarlaSC [46] and 1024 latents for Occ3D-Waymo [39]. Our VAE has a hidden size of 384, with 6 blocks in the encoder and decoder. Our layout, feature, and motion generators also have a hidden size of 384 with 12 DiT blocks each. As CarlaSC does not provide instance-level tracks, we only evaluate static scene generation on CarlaSC, and we evaluate our 4D generation pipeline on Waymo. Additional details are in the supplementary.

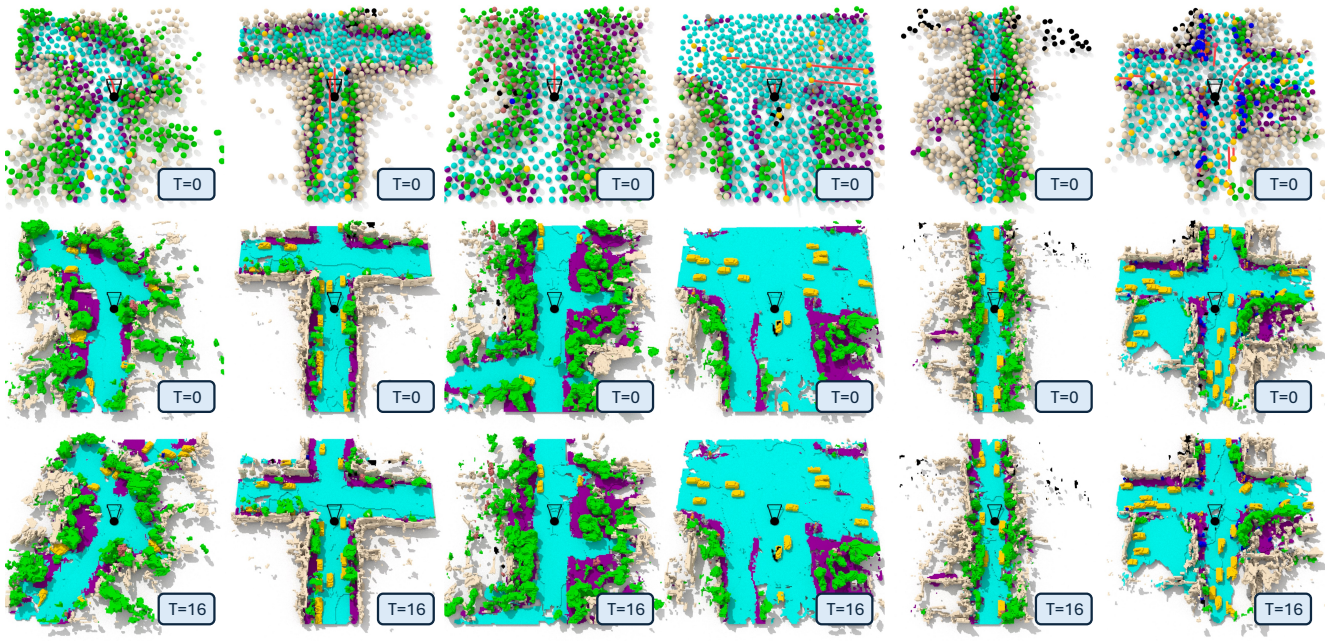

Figure 5. **Factorized generation with grounded latents.** Row 1: generated latent layouts with generated actor waypoints, capturing coarse structure and multi-actor placement. Row 2: feature generation and decoding to semantic Gaussians, splatted to voxels, producing diverse, realistic 3D scenes faithful to the layout. Row 3: applying the generated motion to the same latents produces coherent 4D sequences with precise actor movement and stable background. *Zoom in for details.*

4.2. Quantitative Results

We evaluate **3D scene generation** on CarlaSC (Table 1). Our sparse, grounded latent point model achieves consistent gains, with large improvements on fine-grained foreground classes such as *Pedestrian* and *Vehicle*. Compared to grid-based methods, each actor with a single grounded latent makes multi-actor layout explicit, yielding sharper shapes and more reliable placement in crowded regions.

For **4D scene generation** on Waymo (Table 2), LatentWorld substantially outperforms DynamicCity, especially on foreground categories such as *Vehicle*, Pedestrian, and Motorcycle that require precise localization of small or fast-moving actors. While DynamicCity performs reasonably on background classes like Building and Vegetation, its dense voxel representation lacks explicit actor factorization, causing spatial drift and blurred actor geometry over time. In contrast, our grounded latents maintain a one-to-one actor representation and apply ego motion as an explicit rigid transform, producing coherent trajectories and stable inter-actor separation alongside strong background fidelity.

4.3. Qualitative Analysis

Generation Component Visualizations. In Figure 5, we present our factorized 4D scene generation. The first row shows synthesized grounded latent layouts, visualized with generated actor future waypoints; the layout cleanly captures coarse scene structure and multi-actor placement. We then run feature generation and decode to semantic Gaussians before splatting to voxels, yielding diverse, realistic 3D scenes that remain faithful to the underlying layout. Finally, because latents are explicitly grounded and controllable, we apply the generated motion to the latent set and decode across time, producing coherent, diverse 4D sequences with precise actor movement and stable background structure.

Dynamic Scene Generation Comparison. Figure 6 compares generated 4D scenes on Waymo from DynamicCity and our method. DynamicCity often fails to maintain vehicle coherence: actor voxels flicker in and out across frames in the first sequence and split or merge in the second. Furthermore, because grid-based methods represent ego motion only implicitly via global scene changes, they struggle to decouple actor motion from scene motion during turns, leading to additional merging and frame-to-frame inconsistencies in background geometry.

In contrast, our grounded latent formulation, with one latent per actor, explicit ego transforms applied to all latents, and a motion diffusion model over persistent actors, preserves foreground identity even in fast sequences. Notably, it also yields precise pedestrian trajectories in crowded scenes and maintains consistent actor and scene geometry through generated ego-vehicle turns. Qualitatively, this produces cleaner, more stable 4D generations also aligns with our stronger metrics on foreground classes in Table 2.

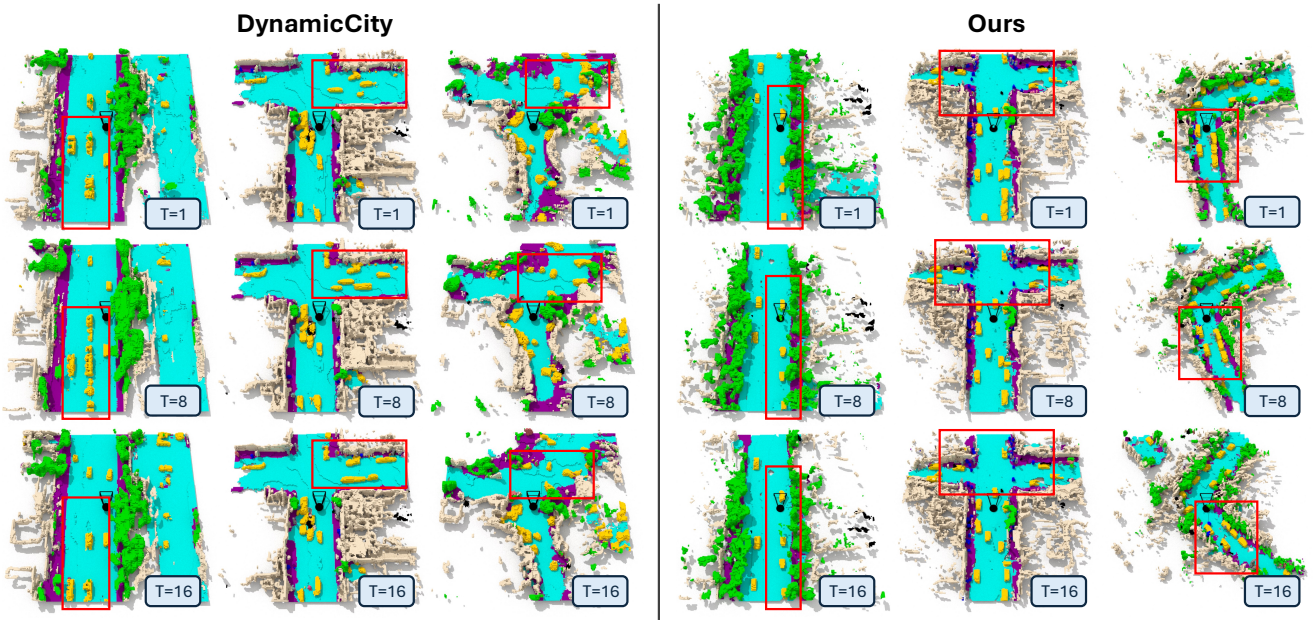

Figure 6. **Waymo qualitative comparison.** DynamicCity's generations exhibit foreground flicker, cause frequent vehicle splitting, and ego turns yield background inconsistencies across frames. LatentWorld's grounded latents, with one latent per actor and explicit ego transforms, preserve identity and inter-actor separation, giving stable trajectories and consistent background. *Zoom in for details.*

Table 3. Ablation on the number of latents in CarlaSC [46]. mIoU↑: reconstruction; MMD↓: generation.

# Latents	mIoU↑	Geo ↓	Sem ↓	Geo+Sem ↓
256	85.45	13.32	**10.95**	7.33
512	92.90	9.90	11.11	6.97
768	93.63	6.44	11.30	**6.69**
1024	**94.71**	**6.38**	12.36	7.30

Table 4. Ablation on outpainting push weight λ. MMD↓: generation.

λ	Geo ↓	Sem ↓	Geo+Sem ↓
0.0	3.98	3.56	2.31
0.5	1.81	**1.81**	1.07
1.0	**1.57**	1.86	**1.04**
1.5	1.63	1.85	1.09

4.4. Ablations

Number of latent points. In Table 3, we ablate the number of latent points on CarlaSC. Reconstruction mIoU improves with more latents (due to a wider bottleneck), and geometric fidelity in generation also rises as added latents capture finer detail. However, semantic-only quality worsens at very high counts. Denser background coverage places multiple latents within small neighborhoods, making class assignment more difficult. Nearby latents receive mixed semantic supervision and their predicted classes become less stable even as geometry becomes more accurate. Balancing geometric improvements against this semantic instability, we choose 768 latents based on the joint geometry+semantics metric.

Outpainting. In Table 4, we ablate the guidance weight λ used in the outpainting mean shift (Eq. 8). With $\lambda = 0$ (no guidance), newly denoised latents may emerge anywhere, often drifting back into already generated regions. This perturbs existing content and leaves the new forward half underpopulated. Introducing guidance *steers* latents toward the outpaint region, improving coverage and consistency. Performance peaks at $\lambda = 1.0$, which provides sufficient attraction without over-concentrating points. Larger values (e.g., $\lambda = 1.5$) push new latents *too far*, leaving the boundary region sparse.

5. Conclusion

We presented **LatentWorld**, an entity-centric generative framework for 4D scene synthesis that replaces dense voxel generation with a sparse set of grounded 3D latents. By factorizing generation into layout diffusion, per-latent geometry diffusion, and motion diffusion over a persistent latent set, our approach produces interpretable, controllable 3D scenes that naturally lift to realistic 4D motion. Decoding each latent into semantic Gaussians and splatting to voxels enables standard training and evaluation while concentrating computation on occupied regions. This design preserves the stability of foreground actors, supports direct entity edits, reduces merging, flickering, and splitting artifacts, and enables 4D occupancy forecasting and motion prediction.

References

[1] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021. 4

[2] Hengwei Bian, Lingdong Kong, Haozhe Xie, Liang Pan, Yu Qiao, and Ziwei Liu. Dynamiccity: Large-scale 4d occupancy generation from dynamic scenes. In *Proc. Int. Conf. Learn. Represent.*, 2025. 1, 3, 4, 6

[3] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 3

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[5] Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6070–6079, 2020. 6

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 6

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 5

[8] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on Robot Learning*, 2017. 2, 5

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. Adv. Neural Inf. Process. Syst.*, 2014. 2

[10] Songen Gu, Wei Yin, Bu Jin, Xiaoyang Guo, Junming Wang, Haodong Li, Qian Zhang, and Xiaoxiao Long. Dome: Taming diffusion model into high-fidelity controllable occupancy world model. *arXiv preprint arXiv:2410.10429*, 2024. 2

[11] 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. *Advances in neural information processing systems*, 30, 2017. 6

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2, 4

[13] Qianjiang Hu, Zhimin Zhang, and Wei Hu. Rangeldm: Fast realistic lidar point cloud generation. *arXiv preprint arXiv:2403.10094*, 2024. 2

[14] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Gaussianformer: Scene as gaussians for vision-based 3d semantic occupancy prediction. *arXiv preprint arXiv:2405.17429*, 2024. 2, 3

[15] Yuanhui Huang, Amonnut Thammatadatrakoon, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, and Jiwen Lu. Probabilistic gaussian superposition for efficient 3d occupancy prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 2, 3

[16] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9307–9315, 2024. 6

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[18] Jumin Lee, Sebin Lee, Changho Jo, Woobin Im, Juhyeong Seon, and Sung-Eui Yoon. Semcity: Semantic scene generation with triplane diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024. 1, 2, 6

[19] Bohan Li, Jiazhe Guo, Hongsi Liu, Yingshuang Zou, Yikang Ding, Xiwu Chen, Hu Zhu, Feiyang Tan, Chi Zhang, Tiancai Wang, et al. Uniscene: Unified occupancy-centric driving scene generation. *arXiv preprint arXiv:2412.05435*, 2024. 3

[20] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023. 2

[21] Yuheng Liu, Xinke Li, Xueting Li, Lu Qi, Chongshou Li, and Ming-Hsuan Yang. Pyramid diffusion for fine 3d large scene generation. In *European Conference on Computer Vision (ECCV)*, 2024. 1, 2, 4, 6

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 4

[23] Yifan Lu, Xuanchi Ren, Jiawei Yang, Tianchang Shen, Zhangjie Wu, Jun Gao, Yue Wang, Siheng Chen, Mike Chen, Sanja Fidler, et al. Infinicube: Unbounded and controllable dynamic 3d driving scene generation with world-guided video models. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2025. 2

[24] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021. 2

[25] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 306–315, 2022. 2

[26] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023. 2

[27] Kazuto Nakashima and Ryo Kurazume. LiDAR data synthesis with denoising diffusion probabilistic models. In

IEEE International Conference on Robotics and Automation, pages 14724–14731, 2024. 2

[28] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2

[29] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021. 4

[30] Lucas Nunes, Rodrigo Marcuzzi, Benedikt Mersch, Jens Behley, and Cyrill Stachniss. Scaling diffusion models to real-world 3D LiDAR scene completion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14770–14780, 2024. 2

[31] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 4195–4205, 2023. 5

[32] Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with LiDAR diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14738–14748, 2024. 2

[33] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2

[35] Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*, 2025. 2

[36] Yining Shi, Kun Jiang, Qiang Meng, Ke Wang, Jiabao Wang, Wenchao Sun, Tuopu Wen, Mengmeng Yang, and Diange Yang. Come: Adding scene-centric forecasting control to occupancy world model. *arXiv preprint arXiv:2506.13260*, 2025. 2

[37] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. *arXiv preprint arXiv:2211.16677*, 2022. 2

[38] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2446–2454, 2020. 2, 6

[39] Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving. *Advances in Neural Information Processing Systems*, 36:64318–64330, 2023. 2, 6

[40] Wenwen Tong, Chonghao Sima, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, et al. Scene as occupancy. In *ICCV*, pages 8406–8415, 2023. 2

[41] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Proc. Adv. Neural Inf. Process. Syst.*, 2017. 2

[42] Lening Wang, Wenzhao Zheng, Yilong Ren, Han Jiang, Zhiyong Cui, Haiyang Yu, and Jiwen Lu. Occsora: 4d occupancy generation models as world simulators for autonomous driving. *arXiv preprint arXiv:2405.20337*, 2024. 1, 2, 3

[43] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4563–4573, 2023. 2

[44] Julong Wei, Shanshuai Yuan, Pengfei Li, Qingda Hu, Zhongxue Gan, and Wenchao Ding. Occllama: An occupancy-language-action generative world model for autonomous driving. *arXiv preprint arXiv:2409.03272*, 2024. 2

[45] 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*, 7:8439–8446, 2022. 2, 5

[46] 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*, 7(3):8439–8446, 2022. 6, 8

[47] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016. 2

[48] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 2, 4

[49] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. Generative gaussian splatting for unbounded 3D city generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 2

[50] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Learning compact representations for lidar completion and generation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1074–1083, 2023. 2

[51] Tianyi Yan, Dongming Wu, Wencheng Han, Junpeng Jiang, Xia Zhou, Kun Zhan, Cheng-zhong Xu, and Jianbing Shen. Drivingsphere: Building a high-fidelity 4d world for closed-loop simulation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2025. 3

[52] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding, 2023. 4

[53] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. *arXiv preprint arXiv:2210.06978*, 2022. 2

[54] Biao Zhang, Matthias Nießner, and Peter Wonka. 3DILG: Irregular latent grids for 3d generative modeling. In *Advances in Neural Information Processing Systems*, 2022. 2

[55] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023. 2

[56] Wenzhao Zheng, Weiliang Chen, Yuanhui Huang, Borui Zhang, Yueqi Duan, and Jiwen Lu. Occworld: Learning a 3d occupancy world model for autonomous driving. In *ECCV*, 2024. 2

[57] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *Proc. Eur. Conf. Comput. Vis.*, pages 17–35. Springer, 2022. 2