GenAssets: Generating in-the-wild 3D Assets in Latent Space

Supplementary Material

Ze Yang[1,2] Jingkang Wang[1,2] Haowei Zhang[1]
Sivabalan Manivasagam[1,2] Yun Chen[1,2] Raquel Urtasun[1,2]
[1]Waabi [2]University of Toronto
{zyang, jwang, hzhang, siva, ychen, urtasun}@waabi.ai

In the supplementary material, we provide implementation details of our method (Appendix A) and baselines (Appendix B), experimental settings (Appendix C), additional results and analysis (Appendix D), and then discuss limitations and future directions of our method (Appendix E). Please refer to our project page https://waabi.ai/genassets for an overview of our methodology and video results of GenAssets for scalable content creation and sensor simulation.

A. GenAssets Details

Scene Representation Model: Our scene representation model is based on tri-plane feature maps and neural feature fields MLP network f_{feat}. For each actor, we use a tri-plane with spatial resolution of 128×128 and feature dimension of 12. For the background, we use a triplane with spatial resolution 256×256 and feature dimension of 12. The neural feature fields MLP network is composed of two sub-networks. The first one takes the interpolated feature $\{\texttt{interp}(\mathbf{x}^p, \mathbf{t}^p)\}_{p\in\{xy,xz,yz\}}$ as input and predicts the signed distance value s and an intermediate feature. The second network takes the concatenation of the intermediate feature and viewpoint encoding as input to predict the neural feature vector $\mathbf{f}$ with 32 channels. To model the unbounded scene, we adopted an inverted sphere parameterization similar to [1, 29]. To decouple shadows from actors, we identify the camera ray's intersection with the actor's bottom plane (derived from the actor's pose and dimension) and use a shadow head MLP to predict the actor's shadow RGB.

Rendering Details: To render the neural feature fields, we sample query points with a step size of 5 cm for regions inside the foreground actor bounding boxes, and 30 cm otherwise. To enable efficient volume rendering for unbounded background regions, we leverage geometry priors from LiDAR observations to localize near-surface regions, restricting radiance field evaluations to these areas. This approach significantly reduces the number of samples and radiance queries needed. Specifically, we construct an occupancy grid for the scene volume based on aggregated LiDAR point clouds similar to [29], with a voxel size of 0.5 m. Additionally, 8 extra points are sampled for the distant sky region during volume rendering.

Asset Decoder: Our asset decoder f_{dec} is designed to transform the asset latent representation into the tri-plane representation. The latent code has a spatial resolution of 8×8 and a feature dimension of 32. We regularize the latent space with a small Kullback-Leibler (KL) penalty using the reparameterization trick [14]. The asset decoder adopts a similar architecture to [10], where the latent code is concatenated with the class embedding and processed through a sequence of blocks: a residual block with 256 channels, a non-local block with 256 channels, and another residual block with 256 channels. This is followed by five residual blocks with channel channels $256, 128, 128, 64, 64$, with four $2\times$upsampling layers interleaved from the second to the last residual block, progressively increasing the spatial resolution to 128×128. Finally, a convolutional layer predicts the tri-plane representation. Group normalization with 32 groups is applied to normalize intermediate features.

Camera RGB CNN Network: The camera RGB network f_{rgb} consists of 6 residual blocks with 32 channels, and upsample the rendered image feature map from 480×270 to 1920×1080 resolution. A convolution layer is applied at the beginning to convert input feature to 32 channels, and another convolution layer is applied to predict the final output image. To get a larger receptive field, we set kernel size to 5 for all residual blocks. Two $2\times$ upsampling layers are inserted after the second residual block and the fourth residual block.

Denoising Network: The denoising network f_{diff} is implemented as a U-Net [20], following the design in DDPM [12]. The U-Net uses a base channel of 128 and includes two $2\times$ scaling module, each consisting of two residual blocks. The feature dimension increases progressively from the base channel size of 128 to 512. Timestep embeddings t and conditioning signals (*e.g.*, class labels, time-of-day) are incorporated into the intermediate layers. For the prediction format, we use the v-parameterization, as proposed in [21].

Training Details: In the first stage, we jointly train the asset code $\{\mathbf{c}_i\}_{i=1}^{N_c}$, class embedding $\{\mathbf{e}_i\}_{i=1}^{N_e}$, background neural fields $\mathbf{t}_{\mathcal{B}}$, asset decoder f_{dec}, neural feature fields MLP f_{feat}, and RGB CNN network f_{rgb} to minimize the reconstruction objective (Eqn. 5 in the main paper). The training set consists of 6 cameras from PandaSet across 96 training logs, each containing 80 images at 1920×1080 resolution, totaling approximately 46k images. Training is performed on 16 NVIDIA A10G GPUs, for 100 epochs, taking approximately 1.5 days to complete. The loss weights in the learning objective are set as follows: $\lambda_{\text{rgb}} = 1$, $\lambda_{\text{perp}} = 0.1$, $\lambda_{\text{adv}} = 0.001$, $\lambda_{\text{lid}} = 0.01$, and $\lambda_{\text{KL}} = 1e-5$. For the additional perceptual loss on object patches, we project the 3D instance boxes onto images to extract patches, which are resized to 256×256 resolution for perceptual loss computation. We adopt Adam optimizer [15] for training. Learning rate is 0.05 for asset latent code, 0.0001 for asset decoder and class embedding, 0.001 for neural feature fields network, and 0.001 for RGB CNN network. In the second stage, the denoising network is trained exclusively on 8 NVIDIA A10G GPUs for 100k iterations, requiring approximately 12 hours. The Adam optimizer is used with a learning rate of 0.0001 for the denoising network.

B. Baseline Implementation Details

B.1. Reconstruction Baselines

NeuRAD [22]: Following [29], NeuRAD leverages compositional neural radiance fields to handle dynamic scenes. It further proposes several techniques to handle more complex sensor phenonmena (*e.g.*, rolling-shutter, ray-dropping and beam divergence) and achieves superior performance in camera simulation. We adopt the public implementation[1] and train the models for 20,000 iterations. The models are trained on all input views (9 images for sparse view synthesis, and 80 images for novel camera synthesis and 360° view synthesis) for each validation snippet.

Street Gaussian [28]: Street Gaussian replaces NeRFs in [22, 29] with compositional 3DGS and achieves real-time camera simulation, but does not support LiDAR. We adopt the public implementation[2] and train the models for 30,000 iterations with the default hyperparameters (*e.g.*, density control, learning rate schedule). We use 800,000 downsampled aggregated LiDAR points and random 200,000 points for the initialization of 3D Gaussians. We train the models on all input views (9 images for sparse view synthesis, and 80 images for novel camera synthesis and 360° view synthesis) for each validation snippet.

PixelSplat [6]: PixelSplat is a generalizable scene reconstruction approach based on 3D Gaussian Splatting. It predicts 3D Gaussians with a 2-view epipolar transformer to extract features and then predicts the depth distribution and pixel-aligned Gaussians. The model is trained with 96 training PandaSet snippets with 9 source views and 71 target views. During sparse view synthesis evaluation, we select the two nearest source views to predict and render the representation. We adopt the public implementation[3] and use $2\times$ A6000 (48GB) to train the models. Due to GPU memory constraints, we downscale the image resolution to 360×640 for PandaSet (and rescale images to 1920×1080 for evaluation). We note that the original work uses an 80GB A100 for training and handles 256×256 resolution. We use `re10k` config and train each model for 100k iterations with a batch size of 1.

G3R [8]: G3R is a generalizable reconstruction approach that is designed for real-world large scenes and can efficiently predict high-quality 3D Gaussians taking many training images. It learns a reconstruction network that takes the gradient feedback signals from differentiable rendering to iteratively update a 3D scene representation. We adapt G3R to our evaluation setting by training models on sparse training views (sparse view synthesis) or all front-camera images (novel camera synthesis and 360° view synthesis). For all experiments, during training we follow [8] to randomly select 20 consecutive frames (10 source views, 10 target views) using the front camera. We train the models on $2\times$ A6000 (48GB) for 300 epochs. During the evaluation, we take all source images to reconstruct the 3D representations (9 images for sparse view synthesis, 80 images for novel camera synthesis and 360° view synthesis) for each validation snippet, and then render at the target views.

[1] https://github.com/georghess/neurad-studio
[2] https://github.com/ziyc/drivestudio
[3] https://github.com/dcharatan/pixelsplat

B.2. Generation Baselines

EG3D [5]: EG3D is a GAN-based model for high-quality 3D object synthesis with implicit representations. It combines StyleGAN-like latent space manipulation with a geometry-aware approach, leveraging a tri-plane representation to balance rendering quality and computational efficiency. The architecture integrates volumetric rendering and an efficient super resolution module, enabling fast and scalable training while maintaining photorealistic results and precise 3D structures. We adopt the official implementation[4] and train the models on our object-centric benchmark ($\approx$ 55k object images) using 2$\times$ A5000 (24GB) GPUs. We render the feature map at a resolution of 64×64 and use $4\times$ super resolution module to produce final images at 256×256 resolution. Each ray samples 64 coarse samples and 64 importance samples. The model is trained with a batch size of 8 images over a total of 1000k images. We apply R_1 regularization with $\gamma = 1$ to ensure stable training.

DiscoScene [27]: DiscoScene is a 3D-aware GAN-based generative model for controllable scene synthesis. It utilizes an abstract object-level representation as the scene layout prior and spatially disentangles the scene into object-centric generative radiance fields by learning solely from 2D images with the global-local discriminators. We use the official implementation[5] and follow the provided configuration[6] for Waymo. We train the model on all cameras from PandaSet, with a heuristic-based filtering process to remove noisy object samples. Specifically, we exclude objects smaller than 200 pixels in the original 1920×1080 resolution or those occluded by closer objects (*i.e.*, having $> 50\%$ IoU with a nearer object when projecting the 3D instance box onto the camera image). We train the model using 2$\times$ A5000 (24GB) GPUs for 300k iterations.

SSDNeRF [7]: SSDNeRF (Single-Stage Diffusion Neural Radiance Field) is a SoTA diffusion-based model for high-quality 3D object generation. It proposes a single-stage training paradigm with an end-to-end objective that jointly optimizes the tri-plane NeRF reconstruction and a diffusion model in the triplane space. This design enables simultaneous 3D reconstruction and generative prior learning. SSDNeRF achieves competitive results in both unconditional generation and sparse-view 3D reconstruction tasks. We adopt the official implementation[7] and use 2$\times$ A5000 (24GB) GPUs to train the models on our object-centric benchmark. We follow the config for unconditional generation detailed in the official repo[8]. We train the model with rendered images at a resolution of 256×256 and tri-plane resolution of 128×128. Since instances in our PandaSet object-centric benchmark contain varying numbers of images, we randomly sample three views per instance in each iteration. The model is trained with a batch size of 16 instances over 400k iterations.

C. Experiment Details

C.1. Pandaset Dataset

We evaluate on public real-world dataset PandaSet [25] which contains 103 urban driving scenes captured in San Francisco. Each scene spans 8 seconds (80 frames sampled at 10Hz). The data collection platform consists of a 360° mechanical spinning LiDAR as well as a forward-facing LiDAR, along with 6 HD (1920×1080) cameras. These cameras are facing front, front-left, left, back, front-right, and right. PandaSet also provides human annotated 3D bounding boxes in each frame. Following [8], we select 7 diverse scenes (`001, 030, 040, 080, 090, 110, 120`) for evaluation, and the remaining 96 scenes for training. As our focus is object reconstruction and generation, we leverage Segment Anything model (SAM) [16] to extract object segmentation masks for metric evaluation. We also use them to train object-centric baselines EG3D [5] and SSDNeRF [7]. Specifically, we project 3D bounding boxes onto the camera image to obtain 2D boxes. The 2D bounding boxes are used as prompts for the SAM, which generates per-pixel object masks. Next, we project the 3D LiDAR points within the 3D bounding box onto the camera image and calculate the ratio of points falling outside the object mask. To handle failures in SAM that are inconsistent with the 3D annotation, instances with more than 30% of LiDAR points falling outside the masks are removed. We also filter out instances that are too small (< 100 pixels) in the camera images. To further improve the data quality for training object-centric baselines, we exclude instances that are heavily occluded by other objects. Specifically, we project actor bounding boxes onto the camera image and determine the occluded areas caused by closer actor bounding boxes. Instances with more than 50% of their bounding box area occluded are removed. After processing, we have

[4] https://github.com/NVlabs/eg3d
[5] https://github.com/NVlabs/eg3d
[6] https://github.com/snap-research/discoscene/blob/master/scripts/discoscene/training/train_waymo256.sh
[7] https://github.com/Lakonik/SSDNeRF
[8] https://github.com/Lakonik/SSDNeRF/blob/main/configs/paper_cfgs/ssdnerf_cars_uncond.py

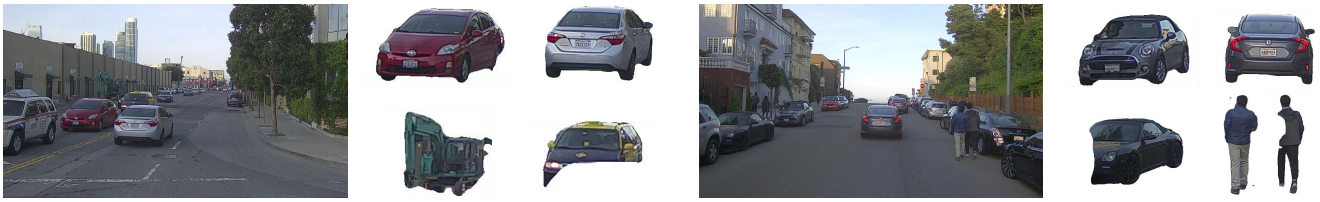

Figure A1. **Examples of object-centric benchmark.** We leverage Segment Anything [16] model and object 3D bounding boxes to create an object-centric benchmark for evaluating object reconstruction and generation, and training object-centric baselines [5, 7].

GenAssets Actor Class	PandaSet Actor Class
Car	Car
Truck	Pickup Truck Medium-sized Truck Semi-truck
Bus	Bus
Construction Vehicle	Other Vehicle - Construction Vehicle
Emergency Vehicle	Emergency Vehicle
Bicycle	Bicycle
Motorcycle	Motorcycle Motorized Scooter
Pedestrian	Pedestrian Pedestrian with Object

Table A1. Mappings between PandaSet actor class and our defined actor class for class conditional generation.

around 55k object images (with corresponding instance masks) to train the object-centric baselines and around 5k images for evaluation. Fig. A1 shows examples of object-centric images created from PandaSet.

C.2. Evaluation Metric Details

Asset Reconstruction Metrics: For *Sparse view synthesis* and *Novel camera synthesis* settings, we report peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [23], and perceptual similarity (LPIPS) [31] metrics to evaluate the photorealism of novel views. We mask out the background using our inferred object segmentation masks. Our approach, along with generalizable reconstruction baselines PixelSplat [6] and G3R [7], are trained on the 96 training logs to learn generalizable priors. For *Sparse view synthesis*, we evaluate using the front camera from 7 evaluation logs. All methods take every 10th frame as source frames and evaluate on the remaining frames. For *Novel camera synthesis*, we use all frames from the front camera as source frames and evaluate on all frames from the front-left camera across the 7 evaluation logs. For 360° *View synthesis*, we rotate actors by $0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°$ to simulate various behaviors to evaluate the asset completeness. All frames from the front camera are used as source frames, and evaluations are conducted on the 7 evaluation logs. Since ground-truth images for rotated actors are unavailable, we measure Fréchet Inception Distance (FID) [11] between the rotated images and source images.

Asset Generation Metrics: To evaluate the asset generation quality, we report both the Fréchet Inception Distance (FID) [11] and Kernel Inception Distance (KID) [3] metrics. The FID metric is defined as:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}\right) \tag{1}$$

where μ_r and μ_g are the means of the feature representations for the real and generated images, respectively. And Σ_r and Σ_g are the corresponding covariance matrices. Tr denotes the trace of a matrix. The KID metric measures the squared Maximum

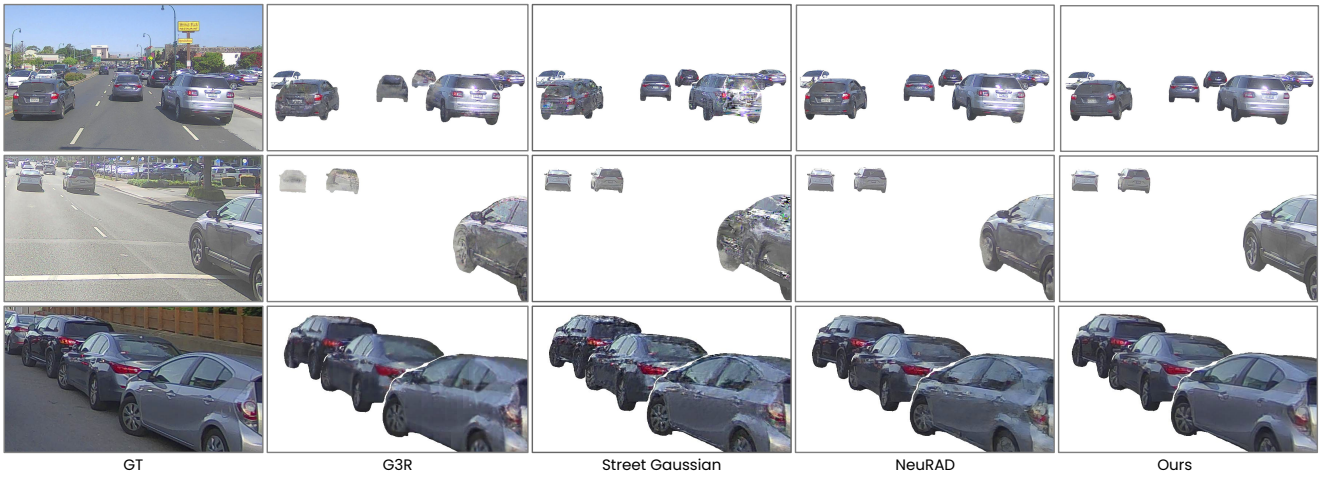

Figure A2. **Qualitative results on sparse view synthesis.** All methods use every 10th frame as source frames and evaluate on the remaining frames. GenAssets generalizes well under this challenging setting, whereas SoTA reconstruction methods show reduced robustness and introduce noticeable visual artifacts.

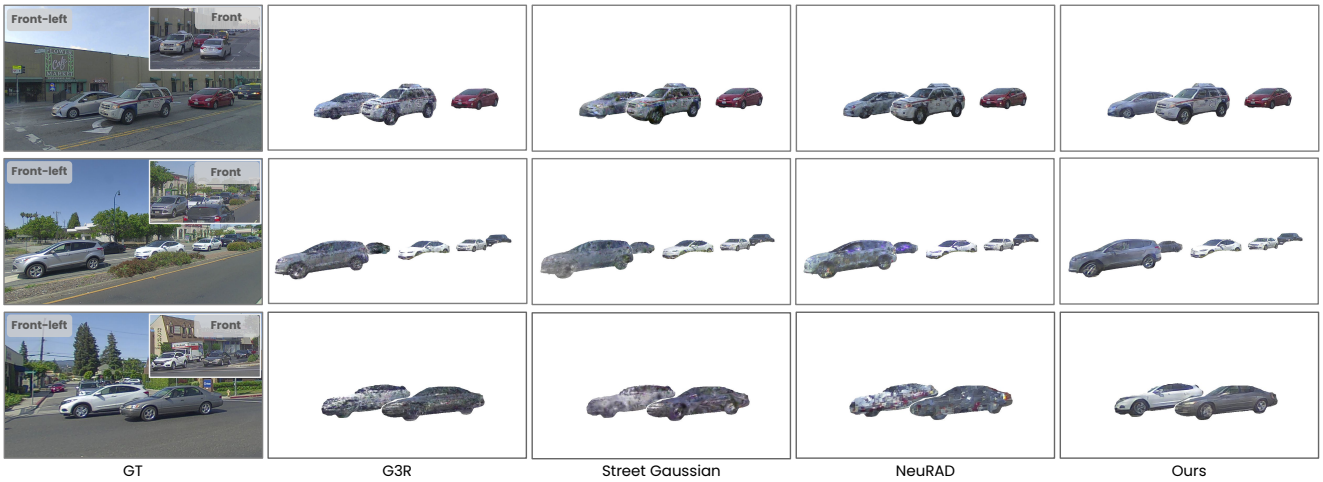

Figure A3. **Qualitative results on novel camera synthesis.** We train on front-camera frames and evaluate on front-left-camera frames. GenAssets achieves better extrapolation, while baseline methods exhibit significant visual artifacts.

Mean Discrepancy (MMD) between two distributions using a polynomial kernel. For both FID and KID, we generate 10k object images and compute the metrics w.r.t. all object images in the evaluation set. We render the images at a resolution of 256×256, and resize the real images to the same resolution.

C.3. GenAssets Experiments Details

Asset Reconstruction Details: We train GenAssets on 96 training logs and fine-tune on the 7 evaluation logs for evaluating asset reconstruction metrics. During fine-tuning, we optimize the asset latent code and background tri-planes to minimize the reconstruction objective (Eqn. 5 in the main paper). In addition, we incorporate a diffusion prior term (Eqn. 7 in the main paper), resulting in the final training objective: $\mathcal{L} = \mathcal{L}_{\text{rend}} + \lambda_{\text{diff}}\mathcal{L}_{\text{diff}}$. By guiding the gradient to minimize the diffusion loss, we achieve a more complete reconstruction of the actor latent code. This approach, known as score distillation sampling (SDS) [18], is widely used in radiance fields generation.

Asset Generation Details: We train GenAssets on 96 training logs, which include diverse assets classes and different times of day. For class-conditional generation, we map the PandaSet classes into 8 fine-grained GenAssets classes according to

Figure A4. **Qualitative results on** 360° **view synthesis.** Leveraging multi-scene training and a structured latent space, GenAssets enables higher-fidelity asset completion across 360° orientations.

Tab. A1. We use an `Embedding` layer to encode the one-hot representation of class information, which is then combined with the timestep embedding before being fed into the U-Net denoising network. For time-of-day conditional generation, actors from night logs (`057, 058, 059, 062, 063, 064, 065, 066, 067, 068, 069, 070, 071, 072, 073, 074, 077, 078, 079, 149`) are classified as night actors, while the others are categorized as day actors. The time-of-day information is then used to condition the diffusion model in the same way as the class-conditional model. For single-image-to-3D generation, we employ a rendering-guided denoising process (Eqn. 10 in main paper). We jointly generate multiple actors within the same image and applying the rendering loss to the composited actor representations, which helps to mitigate the shape ambiguity caused by occlusions.

Data Augmentation Details: We show that our generated assets boost downstream performance when training the BEVFormer detector. Specifically, we augment the training dataset by swapping out existing actors with generated ones for the same scene layouts. Fig. A5 shows examples of the augmented dataset. We adapt the official BEVFormer repository[9] to support PandaSet, focusing on single-frame vehicle detection using only the front camera. Any actor outside the camera's field of view is ignored. Our models are trained in vehicle coordinates following the FLU convention (x: forward, y: left, z: up), with the region of interest defined as . $x \in [0, 80m]$,$y \in [-40m, 40m]$, and $z \in [-2m, 6m]$. We use the BEVFormer-tiny architecture with a batch size of 4 per GPU. We use the AdamW optimizer and employ the default cosine learning rate schedule for 25 epochs. For augmented training, we generate a simulated (sim) dataset by replacing existing actors with generated ones while keeping the same scene layouts, and then mix this with the original data (real). We only consider the

[9] https://github.com/fundamentalvision/BEVFormer

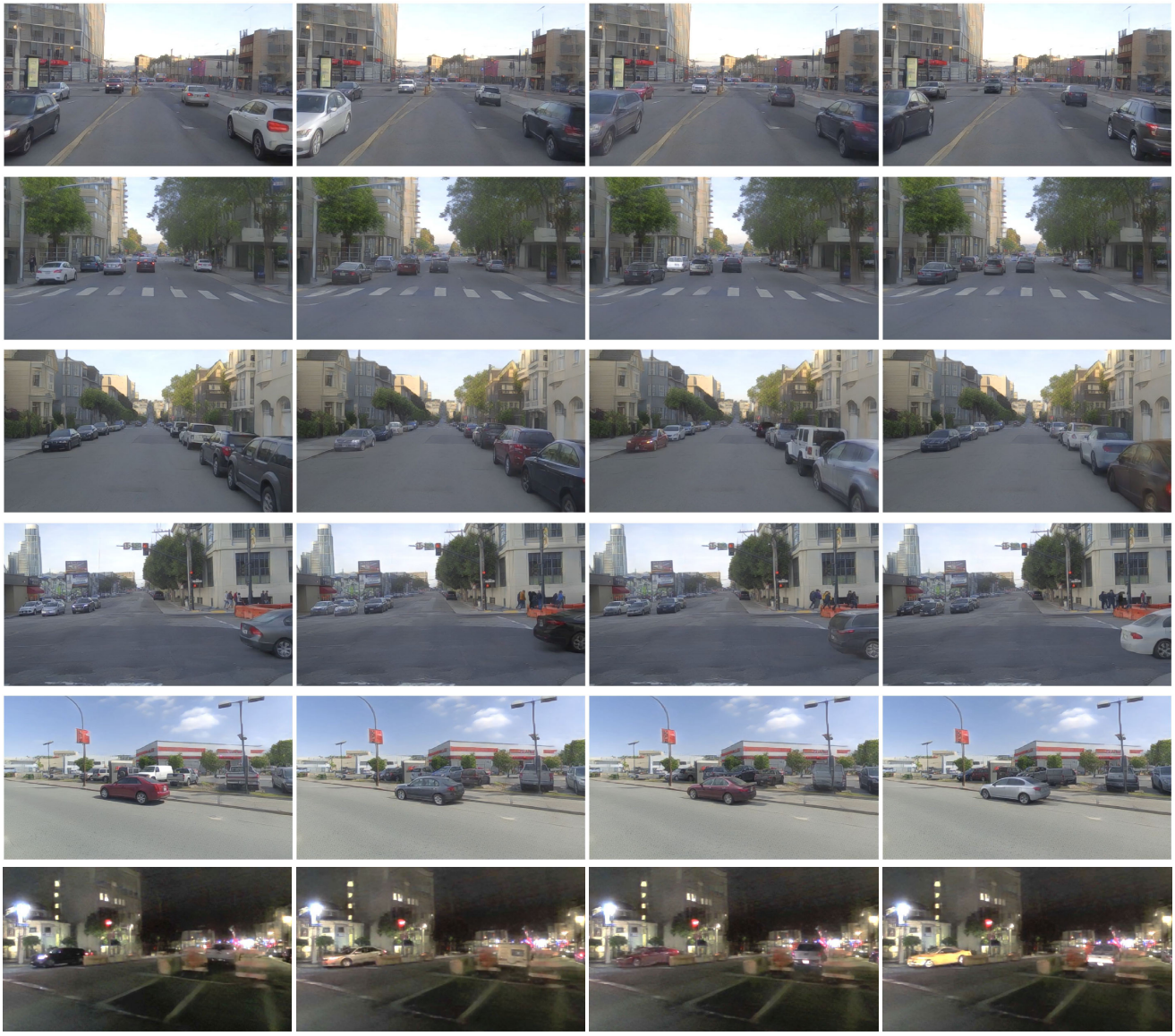

Figure A5. **Examples of sensor simulation generated by GenAssets.** For each scene layout (row), existing actors are replaced with generated ones, showcasing different variations per layout.

vehicle class and report distance-based AP [4] at $1m$, $2m$, and $4m$, and average the three metrics to obtain the final mAP.

D. Additional Experiments and Analysis

D.1. Additional Qualitative Results on Reconstruction

Sparse View Synthesis: Additional qualitative comparisons of sparse view synthesis are presented in Fig. A2. GenAssets demonstrates strong generalization in this challenging setting, while SoTA reconstruction methods exhibit reduced robustness and introduce noticeable visual artifacts.

Novel Camera Synthesis: We compare GenAssets against SoTA methods for novel camera synthesis in Fig. A3. The inset shows source images from the front camera. Rendering from a new camera (front-left) involves significant viewpoint changes, with previously unobserved regions becoming visible. GenAssets handles these challenges effectively, whereas

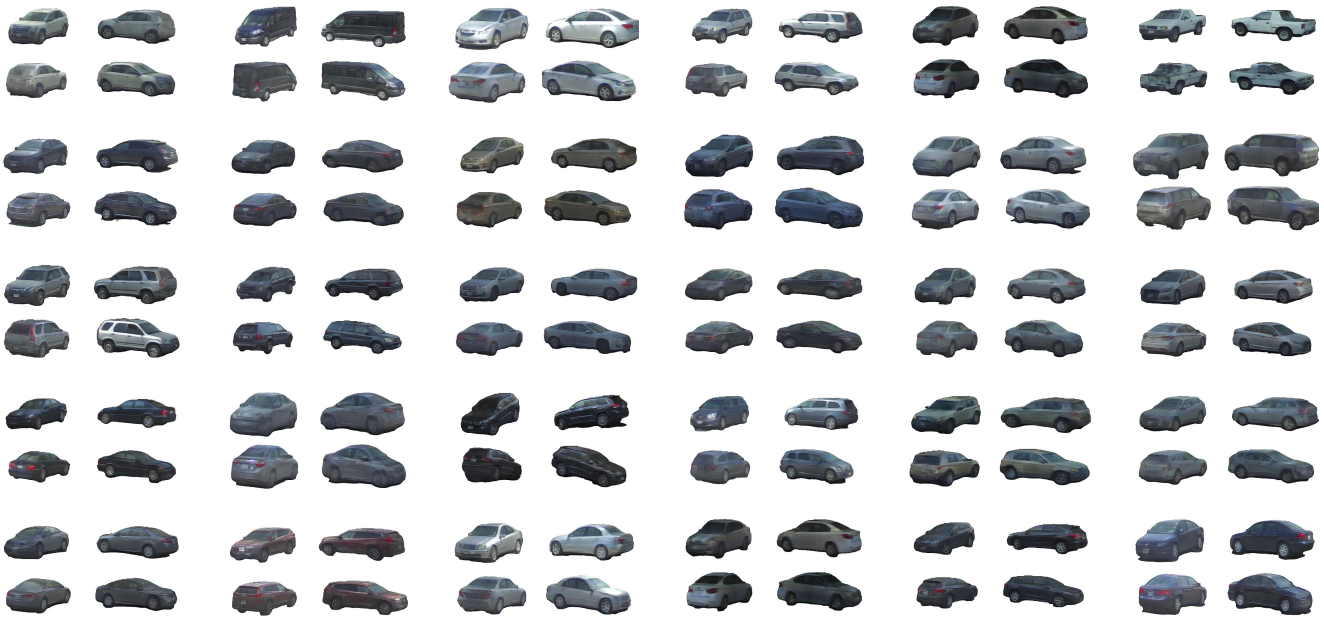

Figure A6. **Additional qualitative results on unconditional generation.**

Figure A7. **Additional qualitative results on conditional generation.**

baselines struggle with robustness and produce visible artifacts.

360° **View synthesis:** Additional 360° view synthesis results are shown in Fig. A4. GenAssets captures complete asset shapes and appearances, enabling high-fidelity rendering across 360° viewpoints, allowing for scalable sensor simulation.

D.2. Additional Qualitative Results on Generation

Unconditional Generation: We provides additional unconditional generation results in Fig. A6. GenAssets generates diverse, complete and higher-quality 3D assets and enables scalable content creation for sensor simulation.

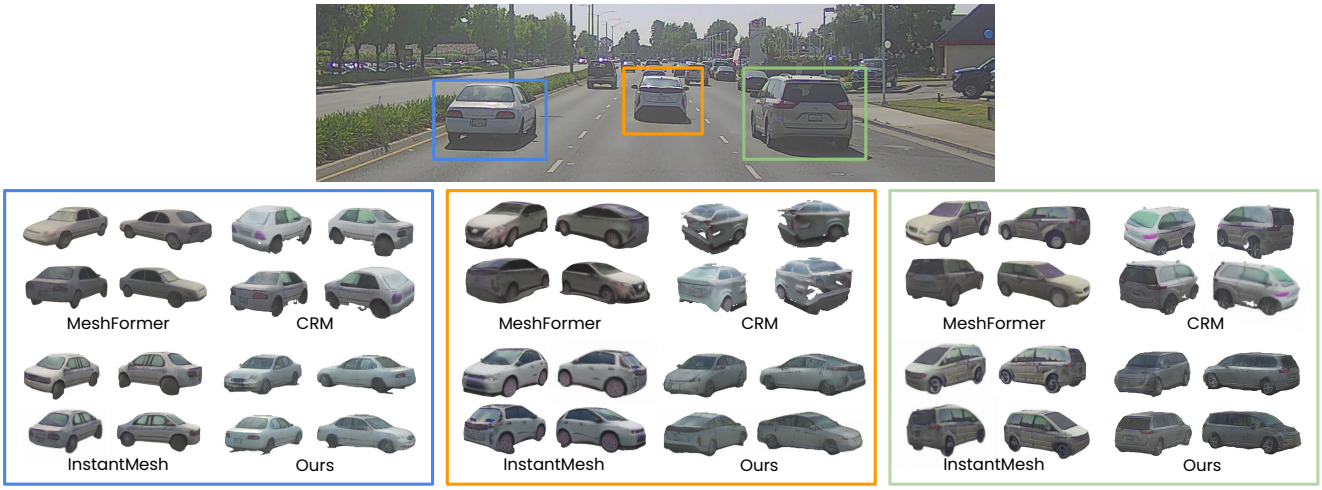

Figure A8. **Additional qualitative results on single image to 3D.**

Methods	Sparse View Synthesis			Novel Camera Synthesis			360° View Synthesis
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	FID↓
Full model	21.34	0.825	0.113	**18.36**	**0.805**	**0.147**	**100.28**
w/o KL regularizer	21.48	0.825	**0.112**	18.06	0.800	0.150	110.87
w/o perceptual supervision	**21.52**	**0.829**	0.155	18.15	0.803	0.177	110.19
Tri-plane opt. w/o asset decoder	19.31	0.784	0.186	17.58	0.778	0.195	169.44

Table A2. **Ablation study on asset reconstruction.**

Methods	FID↓	KID↓
Full model	**59.50**	**28.32**
w/o KL regularizer	82.67	39.06

Table A3. **Ablation study on unconditional generation.**

Conditional Generation: Fig. A6 provides additional asset generation results conditioned on class and time-of-day. GenAssets allows us to control the generation process for diverse asset creation.

Single Image to 3D: We provides additional single image to 3D comparisons in Fig. A8. Compared to SoTA 3D large models MeshFormer [17], CRM [24] and InstantMesh [26], our approach generates higher quality 360° completion and is more multi-view consistent. Due to the reliance on object-centric synthetic dataset training, existing 3D large models usually produce cartoonish generation results especially on unobserved views.

D.3. Ablation Study

In this section, we study the effectiveness of several key components of GenAssets on PandaSet. Tab. A2 reports the reconstruction metrics. The KL term plays a important role in regularizing the latent space and learning complete asset representation, which can impacts the novel camera synthesis and 360° view synthesis results where the viewpoint changes are significant. The perceptual supervision enhances the overall image quality (LPIPS and FID) by preserving more details. We also investigate a setting where per-actor triplane representation are learned directly instead of using latent codes with a shared asset decoder in latent space. We use the same resolution of 128×128 for the per-actor triplane representation and learn them on the 7 evaluation logs in our experiment. This approach struggles to capture complete assets, resulting in notably worse performance particularly for 360° view synthesis. Finally, we study the impact of KL regularizer in diffusion

learning in Tab. A3. The KL regularizer is essential for learning complete asset representations for diffusion learning in our in-the-wild setting.

E. Discussions

E.1. Artifacts and potential enhancements

Our reconstructed or generated assets are not perfect. Our method can still lack sharp details for unobserved views and have inconsistent color tone. Increasing model capacity and dataset size may ameliorate these artifacts. The triplane representation can have boundary-like smearing artifacts. Our work may benefit from enhanced 3D representations and rendering techniques [2, 13]. We also do not model various sensor effects (*e.g.*, sensor calibration [30]) or label noise [29] in the real-world data, which might result in degraded performance.

E.2. Limitations and Future Work

GenAssets assumes all objects are rigid. Despite this, we still find that the method reconstructs reasonable shape and appearance for pedestrians, as depicted in Fig. A7 and Fig. 1 of the main paper. Additionally, GenAssets are not fully lighting-aware. Future works include 4D modelling and animation [9], and intrinsic decomposition [19], as well as more powerful conditioning techniques to further control content creation.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 1

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023. 10

[3] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv*, 2018. 4

[4] 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 multimodal dataset for autonomous driving. In *CVPR*, 2020. 7

[5] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 3, 4

[6] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024. 2, 4

[7] Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. In *ICCV*, 2023. 3, 4

[8] Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3r: Gradient guided generalizable reconstruction. In *ECCV*, 2025. 2, 3

[9] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, et al. Omnire: Omni urban scene reconstruction. *arXiv*, 2024. 10

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 1

[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. *NeurIPS*, 2017. 4

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 2

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 2023. 10

[14] Diederik P Kingma. Auto-encoding variational bayes. *arXiv*, 2013. 1

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 2

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 3, 4

[17] Minghua Liu, Chong Zeng, Xinyue Wei, Ruoxi Shi, Linghao Chen, Chao Xu, Mengqi Zhang, Zhaoning Wang, Xiaoshuai Zhang, Isabella Liu, et al. Meshformer: High-quality mesh generation with 3d-guided reconstruction model. *arXiv*, 2024. 9

[18] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 5

[19] Ava Pun, Gary Sun, Jingkang Wang, Yun Chen, Ze Yang, Sivabalan Manivasagam, Wei-Chiu Ma, and Raquel Urtasun. Neural lighting simulation for urban scenes. In *NeurIPS*, 2023. 10

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[21] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv*, 2022. 2

[22] Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. NeuRAD: Neural rendering for autonomous driving. In *CVPR*, 2024. 2

[23] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 4

[24] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. Crm: Single image to 3d textured mesh with convolutional reconstruction model. In *ECCV*, 2025. 9

[25] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *ITSC*, 2021. 3

[26] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv*, 2024. 9

[27] Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skorokhodov, 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 *CVPR*, 2023. 3

[28] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. In *ECCV*, 2024. 2

[29] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *CVPR*, 2023. 1, 2, 10

[30] Ze Yang, George Chen, Haowei Zhang, Kevin Ta, Ioan Andrei Bârsan, Daniel Murphy, Sivabalan Manivasagam, and Raquel Urtasun. Unical: Unified neural sensor calibration. In *ECCV*, 2025. 10

[31] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4