Generative Point Light Fields

Dorsa Molaverdikhani

Abstract—Generating 3D urban scenes is a challenging task due to the complicated structure of city landscapes. In this work, we introduce Generative Point Light Fields, a generative model for synthesizing 3D urban scenes conditioned on point cloud data. We show the preliminary results of our proposed model on simple urban scenes. Our approach leverages point clouds as the 3D underlying representation, which are commonly available for urban scenes. We extend scene reconstruction techniques from point clouds to the generative domain, where we focus on synthesizing point cloud features. Our decoding pipeline for rendering RGB values for the rays is more efficient compared to neural volumetric rendering approaches, as it requires only a few points to render rays instead of numerous samples needed by neural volumetric rendering methods. We evaluate our proposed method by running overfitting experiments on Waymo Open Dataset.

Index Terms—3D-aware GAN, Sparse UNet, Point Cloud, Radiance Field, Light Field

1 Introduction

Recent scene generation methods have seen immense progress, with methods able to generate high-quality, multi-view consistent renderings. They play a crucial role in gaming, AR/VR, simulations, and autonomous driving [1]. For example, creating 3D urban scenes in simulations can serve as training and testing data for autonomous vehicles, and is particularly useful for unlikely or dangerous situations. Using 3D generative models can help speed up the time-consuming and labor-intensive process of making detailed and realistic city scenes [1].

There have been many attempts to generate photorealistic and consistent scenes of various types. Several methods employ image warping techniques to create a video of a nature scene given the first frame [2], [3], [4], but these methods are not 3D-aware as they are not using any underlying 3D representation for the scene, and as a result they may lack 3D and global consistency. Other works use 3D representation such as radiance fields [5], [6], [7] or mesh [8], [9], but they are all limited to single objects. Some works extend the problem of scene generation to indoor and nature scenes [10], [11], [12], [13]. However, indoor and nature scenes usually exhibit less complexity compared to urban environments. While other approaches tackle urban scene generation [1], [14], they often rely on priors that are challenging or costly to obtain.

In this work, we propose Generative Point Light Fields to address the challenging problem of urban scene generation. Our approach is 3D-aware and leverages point clouds as the underlying 3D representation of the scene which are widely accessible and mainly available for urban scenes. Our model generates point cloud features conditioned on a style code similar to StyleGAN2 [15]. It then renders an RGB frame using the generated point features similar to Neural Point Light Fields [16], as they demonstrate promising results on Waymo Open Dataset [17]. Due to the sparse nature of the point clouds, our feature generator uses Sparse UNet [18], [19] to synthesize point features. Similar to Neural Point Light Fields [16], our rendering pipeline consists of a multi-head attention module to render ray features from the features of the closest points followed by an MLP to derive RGB values given the ray features. Figure 1 illustrates the pipeline of our approach. We evaluate our proposed model by overfitting experiments on two sequences from Waymo Open Dataset [17]. We explore the performance of the model by experimenting with different point cloud sizes and various Sparse UNet [18], [19] model sizes for the generator.

Our contributions are as follows:

- We introduce a point cloud-conditioned generative model for photorealistic urban scene synthesis.
- We evaluate our method by overfitting to two sequences from Waymo Open Dataset [17] and explore the performance for various point cloud and generator model sizes.

1.1 Overview of Limitations

This work shows preliminary results towards our proposed method for generating urban scenes conditioned on point clouds. Our results are limited to the overfitting experiments conducted on a single scene. Our approach is further limited to rendering low-resolution frames.

2 Related Work

2.1 Neural Scene Representation and Rendering

Recently many efforts have been made to improve the efficiency of neural scene representation techniques and extend their capabilities to handle large and complex scenes. NeRF [20] encodes the radiance field of a scene using an MLP and employs volumetric ray marching to render the image. Due to the large number of samples required to render the rays, it takes hours to reconstruct a scene [20]. To address the time inefficiency of NeRF [20], EG3D [7] offers an explicit-implicit method by using an explicit triplane representation of features followed by a small MLP for implicit representation. Another line of work [21], [22], [23]

- *Dorsa Molaverdikhani is with the Department of Computer Science, University of Toronto.*

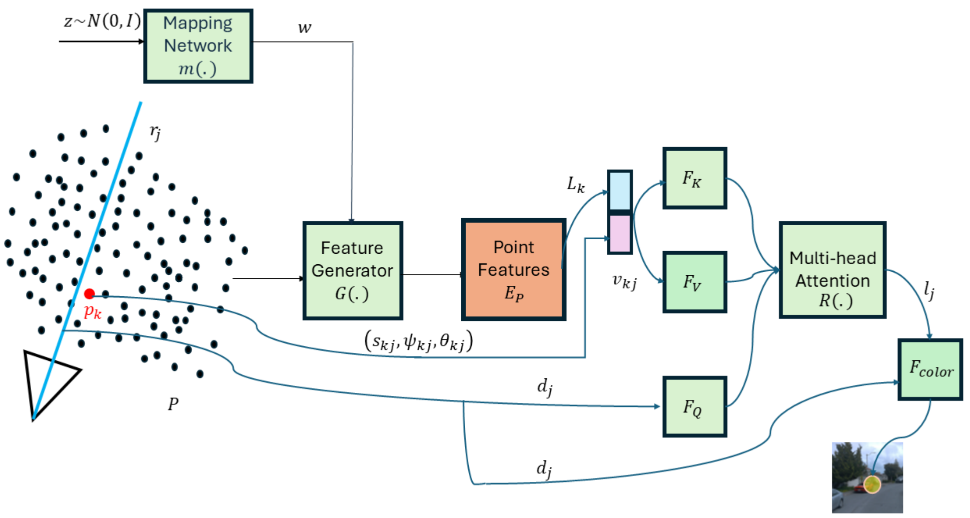

Fig. 1. The figure above illustrates the pipeline of our proposed method. Initially, we transform the noise vector z into a style vector w via a mapping network [15]. Subsequently, the generator synthesizes point cloud features conditioned on the style vector w. The features generated for the K closest points to ray r_j are utilized to derive the ray feature using the multi-head attention module [16]. Finally, the ray feature is decoded into RGB values using an MLP [16].

uses a voxel of features as explicit representation to reduce training time. InstantNGP [24] offers extremely fast training time by representing scenes using multi-resolution hash-encoding.

Closest to our work are the methods that use point cloud for the scene representation. Some works like Neural Point Light Fields [16] assume fixed point positions. Using a multi-head attention module along with an MLP, Neural Point Light Fields [16] renders RGB values of rays from the learnable features of N nearest points to the rays. Neural Point Light Fields [16] demonstrates high-quality reconstruction results on the urban scenes in Waymo Open Dataset [17]. Another line work like PAPR [25] and Gaussian Splatting [26] initialize the point positions and later optimize them. PAPR [25] starts with initializing the point positions on a sphere and refines them through training. It then renders the RGB values using attention and UNet modules [25]. However, it is limited to reconstructing single objects [25]. Gaussian Splatting [26] renders images in real-time at high resolution. Gaussian Splatting [26] optimizes the position, opacity, covariance, and spherical harmonic of the 3D Gaussians initialized using Structure-from-Motion [27] point cloud and render the Gaussians using α-blending [26]. In our work, we assume fixed point positions and use Neural Point Light Fields [16] point-based representation and rendering as they show high-quality results on Waymo Open Dataset [17]. We extend Neural Point Light Fields [16] to be generative by synthesizing features for the point cloud using a style-based feature generator and render them similar to Neural Point Light Fields [16].

2.2 Generative 3D-Aware Image Synthesis

Early 3D GANs [5], [6], [28], [29], [30] show promising results on learning 3D representations from unstructured 2D image collections, but they have limited visual quality. To generate high-quality features, later works [7], [31], [32], [33], [34], [35], [36], [37] use style-based generator [15], but most of these methods are limited to single objects. Other methods can generate simple compositional scenes by decomposing the background and foreground objects [38], [39]. Meanwhile, others enable the generation of indoor scenes with freely moving cameras [10], [11]. Other works introduce synthesis of unbounded nature scenes by employing Bird's Eye View (BEV) representation [13] or 2D layout grid [12] for scene representation. Nature scenes typically exhibit simpler structures compared to urban landscapes. Some methods tackle large urban scene generation by employing expensive CAD models [14], while others separately generate city background and building instances [40]. UrbanGIRAFFE [1] can synthesize high-quality urban scenes by introducing conditioning on semantic voxel grid as prior and utilizing separate generators for sky, stuff, and objects. In contrast to these methods, our method uses point clouds as the underlying 3D representation, which are easier to obtain for urban scenes and are available in most urban datasets.

3 Proposed Method

In this section, we introduce Generative Point Light Fields. Our approach has two main components. Given a point cloud of a scene $P = \{p_1, p_2, ..., p_N\}$, a sequence of M camera poses $C_1, ..., C_M$, camera intrinsics and noise $z \in \mathbb{R}^{512}$, our model outputs a generated RGB frame for each camera pose. Here, we overview our procedure for scene generation using point clouds and describe our approaches for point cloud feature generation and rendering. Figure 1 provides an overview of our framework.

3.1 Point Feature Generator

Given a set of 3D points $P = \{p_1, p_2, ..., p_N\}$ and a noise vector z, the point cloud feature generator G outputs feature vectors for the points in P, which are later used to render a frame given camera pose C_i as discussed in section 3.2. Due to the sparse nature of point clouds, we employ Sparse UNet [18], [19] backbone as the feature extractor for the points. Drawing inspiration from StyleGAN2 [15], we introduce the mapping network $m : \mathbb{R}^{512} \rightarrow \mathbb{R}^{512}$ that maps noise vector $z \sim \mathcal{N}(0, I)$ to style vector $w \in \mathbb{R}^{512}$:

$$w = m(z) \tag{1}$$

We condition the Sparse UNet [18], [19] on w similar to StyleGAN2 [15]. Inspired by StyleGAN2 [15], we use the style vector w to condition the convolution layers of Sparse UNet [18], [19] by replacing the affine transformation of the batch norms in Sparse UNet [18], [19] with channel-wise multiplication with style vector. As a result, our style-based [15] feature encoder G can be expressed as follows:

$$E_P = G(P, w) \tag{2}$$

where E_P denotes the generated features corresponding to the points in the point cloud P.

3.2 Point Cloud Rendering

We follow Neural Point Light Fields [16] rendering approach and make a few changes to adjust their method for our model. Let C_i denote the ith camera pose in a given sequence of camera poses $C_1, ..., C_M$. Similar to Neural Point Light Fields [16], we follow the pinhole camera model to get the rays r_{ji} for the jth pixel and the ith camera. We parameterize each ray r_{ji} by its origin o_{ji} and its direction d_{ji} [16]. Then, we render each ray from the features of the closest points to that ray using a multi-head attention module and an MLP [16].

3.2.1 Point Selection

Similar to Neural Point Light Fields [16], we are interested in finding the set of K closest points to each ray and using their generated features from G to render the ray features [16]. Unlike Neural Point Light Fields [16], which chooses the closest points to each ray among the points visible in camera frustum, we use the entire point cloud P to find the K closest points to each ray. As proposed in Neural Point Light Fields [16], for the ith camera in a sequence, we define the distance s_{kj} between a point $p_k \in P$ and the ray r_{ji} with origin o_{ji} and direction d_{ji} as follows:

$$\cos(\phi_{kj}) = d_{ji} \cdot \left(\frac{p_k - o_{ji}}{\|p_k - o_{ji}\|}\right) \tag{3}$$

$$s_{kj} = \left(\sqrt{1 - \cos^2(\phi_{kj})}\right) \cdot (p_k - o_{ji}) \tag{4}$$

To find the set of closest points $P_{ji} \subset P$ to ray r_{ji}, we pick the K points with the smallest distance s, as proposed by Neural Point Light Fields [16].

3.2.2 Ray Feature Encoding

There are various ways to combine the features of the K closest points to a ray to obtain the ray feature such as pooling or linear combination [16]. However, when two rays share the same set of closest points, these methods yield identical features for those rays [16]. To overcome these ambiguities for ray features introduced by these methods, we follow the ray feature attention module from Neural Point Light Fields [16]. For the ray r_{ji} and the point $p_k \in P_{ji}$, we first define three parameters θ_{kj}, ψ_{kj} as derived in equations (5) and (8) below and s_{jk} as obtained earlier in equation (4) [16].
As proposed in Neural Point Light Fields [16], θ_{kj}, which is the angle between the point $p_k \in P_{ji}$ and ray direction d_{ji} in world coordinates, is formulated as follows:

$$\theta_{kj} = \arccos\left(d_{ji} \cdot \frac{p_k}{\|p_k\|}\right) \tag{5}$$

Below we define c_j which is a projection of global $Y-$axis to the plane orthogonal to the ray direction d_{ji} [16]:

$$c_j = \frac{Y - (Y \cdot d_{ji})d_{ji}}{\|Y - (Y \cdot d_{ji})d_{ji}\|} \tag{6}$$

We formulate projection of the point $p_k \in P_{ji}$ to the plane defined by c_j and the vector perpendicular to c_j and ray direction d_{ji} [16]:

$$\begin{bmatrix} a_{kj} \\ b_{kj} \end{bmatrix} = \begin{bmatrix} c_j^T \\ (d_{ji} \times c_j)^T \end{bmatrix} p_k \tag{7}$$

Finally, ψ_{kj} can be defined as follows [16]:

$$\psi_{kj} = \arctan\left(\frac{a_{kj}}{b_{kj}}\right) \tag{8}$$

We use a multi-head attention module to derive the ray features similar to Neural Point Light Fields [16]. We first apply positional encoding $\gamma(x) = [\sin(2^0\pi x), \cos(2^0\pi x), ..., \sin(2^4\pi x), \cos(2^4\pi x)]$ to θ_{kj}, ψ_{kj} and s_{kj} derived earlier [16]. Similar to Neural Point Light Fields [16], we define a unique vector v_{kj} for each point $p_k \in P_{ji}$ and ray r_{ji} that encapsulates information on both feature of point p_k and its position relative to the ray r_{ji}. The vector v_{kj} is constructed by concatenating the feature vector for point p_k, L_k, obtained from the generator, and positional encodings of θ_{kj}, ψ_{kj} and s_{kj} i.e. $v_{kj} = L_k \oplus \gamma(\theta_{kj}) \oplus \gamma(\psi_{kj}) \oplus \gamma(s_{kj})$ [16]. As proposed in Neural Point Light Fields [16], the keys and values for each closest point p_k to ray r_{ji} and the query for ray r_{ji} needed for the attention module are obtained as follows:

$$V_{kj} = F_V(v_{kj}), K_{kj} = F_K(v_{kj}), Q_j = F_Q(\gamma(d_{ji})) \tag{9}$$

where F_V, F_K and F_Q are MLPs with two-layers and $\gamma(d_{ji})$ is the positional encoding of the ray direction d_{ji} in world coordinate system [16].
Given the keys and values for the K closest points to r_{ji} and the query for ray r_{ji}, we use the multi-head attention module R to derive the ray feature l_{ji} for the ray r_{ji} [16]:

$$l_{ji} = R(V_{kj}, K_{kj}, Q_j) \tag{10}$$

3.2.3 Ray Feature Decoder

To obtain an RGB value for the pixel corresponding to ray r_{ji}, we concatenate the feature vector l_{ji} outputted by the multi-head attention module for ray r_{ji} and the direction of the ray, d_{ji}, and map it to the RGB value c_{ji} using an 8-layer MLP F_{color} [16]. As proposed by [16], the RGB value for ray r_{ji} is obtained as follows:

$$c_{ji} = F_{color}(d_{ji} \oplus l_{ji}) \tag{11}$$

4 Experimental Results

4.1 Dataset and Prepossessing

We conduct our experiments on Waymo Open Dataset [17], which provides sequences of urban scenes. Waymo Open Dataset [17] contains lidar data for four short-range lidars and one mid-range lidar, alongside camera data from the front and four side cameras [17]. For our experiments, we focus solely on the front camera and mid-range lidar data, along with their corresponding poses and intrinsics, discarding the 4 short-range lidars and the data from the side cameras. To accommodate memory constraints, we resize the frames of the sequences to 64×64.

To obtain a coherent representation of a scene for a sequence of M frames with point clouds $\{P_1, P_2, ..., P_M\}$, we use ICP algorithm [41] to merge $\{P_1, P_2, ..., P_M\}$ into a single point cloud P for the entire scene [16]. Unlike Neural Point Light Fields [16], which merges points for subsequences within a sequence, we align all point clouds for a sequence. Due to the large size of the resulting merged point cloud, we downsample it using voxel downsampling provided in Open3D library [42], which averages the points within each voxel. Both merging and subsampling processes are conducted as part of the data preparation before training and inference.

4.2 Implementation and Training Details

The multi-head attention used to extract the ray features adheres to the architecture proposed in Neural Point Light Fields [16]. We use $K = 8$ closest points to each ray to derive the ray features. Our approach integrates the same mapping network, discriminator, and objective function as those presented in StyleGAN2 [15]. We train our model using Adam [43] optimizer with learning rate 0.0005 and $\beta_1 = 0.0$ and $\beta_2 = 0.99$ for both generator and discriminator. Our experiments are conducted on a single NVIDIA RTX A6000, and each experiment is trained for approximately one day.

4.3 Ablation Experiments

We conducted overfitting experiments on one sequence utilizing two distinct sequences from Waymo Open Dataset [17], and we refer to these sequences as Sequence 1 and Sequence 2. We select the sequences with no moving objects, and both sequences have 198 frames. For each sequence, we run overfitting experiments using point clouds with varying numbers of points and employing generator models of different sizes.

4.3.1 Ablation on Point Cloud Size

To analyze the performance of the model as the number of points representing the scene is reduced, we conduct two overfitting experiments for each sequence, using two different point cloud sizes. For each sequence, we align all the point clouds to obtain a single point cloud, as mentioned in section 4.1. We use voxel sizes of 1.5 and 3.5 for the first sequence, and 2 and 4 for the second sequence for voxel downsampling. Consequently, we obtain two point clouds for each sequence, with the latter being more heavily downsampled than the first.

Figures 2 and 3 show the results of the overfitting experiments for varying number of points in the first and second sequences, respectively. Both sets of results exhibit aliasing artifacts and lack high-frequency details, suggesting a deficiency in the model's capacity to generate and render such details. The output frames of the second sequence with $N = 1382$, demonstrate better quality with fewer aliasing artifacts compared to the same sequence with a larger point cloud. The generated frames for Sequence 1, regardless of point cloud size, show comparable levels of blurriness and aliasing artifacts. However, Sequence 1 with $N = 1035$ occasionally fails to capture certain visible objects or details present in the ground truth, possibly due to limited point coverage. Table 1 illustrates the quantitative results of point cloud size ablation experiments on each sequence, obtained by computing PSNR and SSIM metrics for the frames generated with ground truth poses. Using a smaller point cloud leads to a minor reduction in both PSNR and SSIM metrics for both sequences. Consequently, the ablation analysis indicates a minor impact of point cloud size on the results.

4.3.2 Ablation on Model Size

We run overfitting experiments using different generator model sizes for each sequence. For one experiment per sequence, we use a Sparse UNet [18], [19], consisting of 2, 3, 4, 6, 2, 2, 2, and 2 layers with 32, 64, 128, 256, 256, 128, 96, and 96 channels respectively. For the second set of experiments, we utilize a smaller generator model with fewer layers, where the Sparse UNet [18], [19] has 2, 3, 4, 4, 2, and 2 layers with 32, 64, 128, 256, 256, and 96 channels respectively.

Figures 4 and 5 illustrate the outcomes of the overfitting experiments with various generator model sizes for the first and second sequences, respectively. The results across different generator model sizes show aliasing artifacts and blurriness. However, in some cases, there are more noticeable mismatches between some objects visible in the ground truth frame and the output result for the smaller generator size. The quality of the outputs of both model sizes for both sequences is nearly identical. Conversely, some frames in the second sequence exhibit more artifacts in the full model size compared to the smaller one. Table 1 presents the quantitative results of model ablation experiments on each sequence, obtained by computing PSNR and SSIM metrics for the frames generated with ground truth poses. As shown in Table 1, reduction in the model size leads to a slight decrease in both PSNR and SSIM metrics for both sequences.

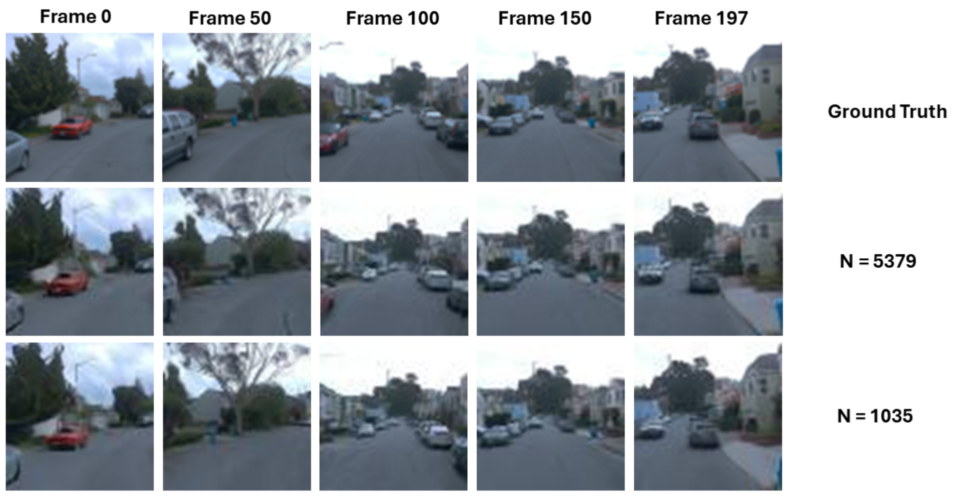

Fig. 2. Qualitative results for overfitting experiments on Sequence 1 using point clouds with 5379 and 1035 points. The images illustrate selected frames from the ground truth alongside frames generated by the models using identical camera poses as the ground truth.

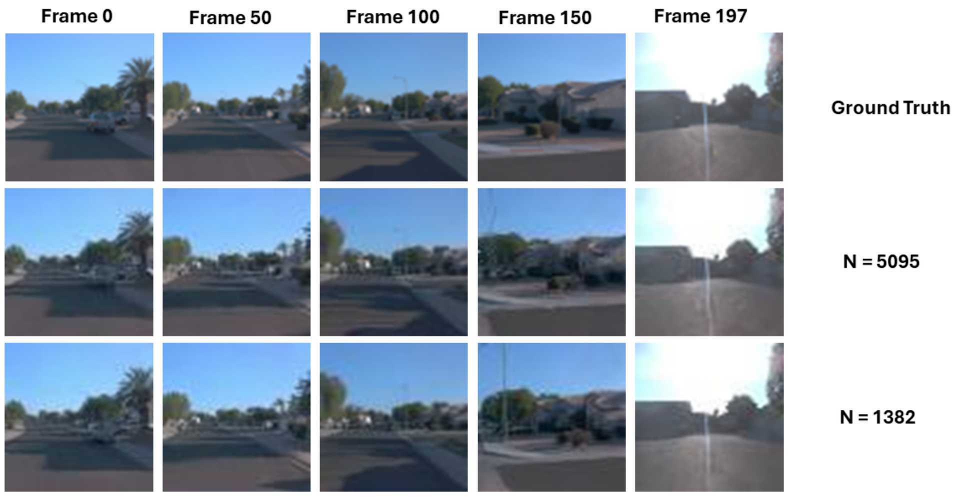

Fig. 3. Qualitative results for overfitting experiments on Sequence 2 using point clouds with 5095 and 1382 points. The images illustrate selected frames from the ground truth alongside frames generated by the models using identical camera poses as the ground truth.

TABLE 1
Metrics for ablation experiments computed across all frames generated from the ground truth poses.

	Sequence 1		Sequence 2		Sequence 1		Sequence 2	
	Full Model	Small Model	Full Model	Small Model	N = 5379	N = 1035	N = 5095	N = 1382
PSNR(db) ↑	20.16	19.54	26.55	24.49	20.16	18.84	26.55	27.12
SSIM ↑	0.651	0.616	0.804	0.734	0.651	0.593	0.804	0.826

5 Conclusion

We present a framework for generating urban scenes by leveraging point clouds as the underlying 3D representation. Incorporating point clouds into our framework helps us efficiently render rays using the generated features, in contrast to neural volumetric rendering approaches such as NeRF [20], which require a large number of samples for ray rendering. Our ablation results regarding point cloud size indicate that utilizing smaller point clouds has

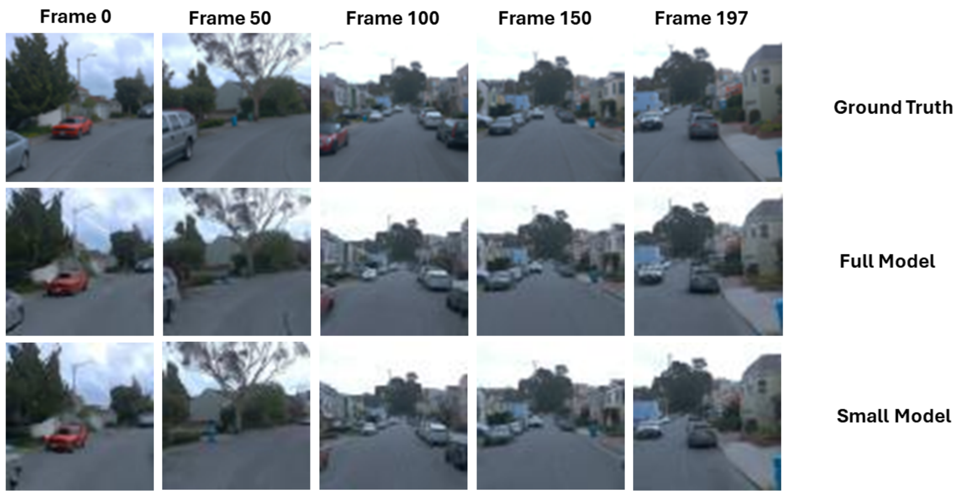

Fig. 4. Qualitative results for overfitting experiments on Sequence 1, comparing full and small generator models as described in Section 4.3.2. The images depict selected frames from the ground truth alongside frames generated by both small and full generator models using the same camera poses as the ground truth.

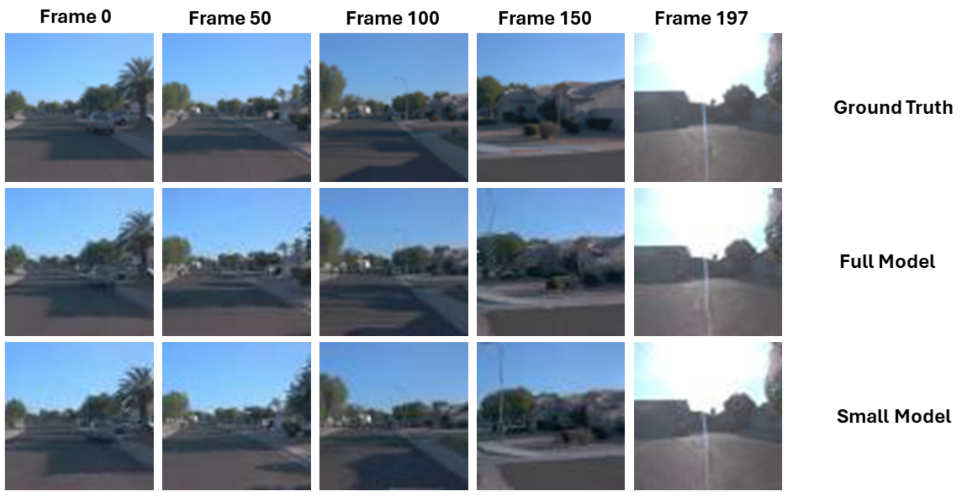

Fig. 5. Qualitative results for overfitting experiments on Sequence 2, comparing full and small generator models as described in Section 4.3.2. The images depict selected frames from the ground truth alongside frames generated by both small and full generator models using the same camera poses as the ground truth.

a minor impact on the quality of the generated outputs. This suggests that we can further improve efficiency by using small point cloud size. However, the experimental results are preliminary, and the approach is not yet adopted to handle multiple sequences. Future work may explore extending the method to be trained on a large portion of urban datasets. Furthermore, the current approach renders the images at 64×64 resolution since rendering at high resolution is costly for training and requires lots of memory. To address this limitation, the current approach can be extended using patch training methods like EpiGRAF [34] to train with image patches and subsequent high-resolution rendering during inference. Another strategy to generate high-resolution outputs involves integrating an upsampler module, which takes low-resolution rendered images as input and outputs high-resolution images, and it can be trained end-to-end with the rest of the model. The suggested approaches for training and inference for high-resolution data may be explored in future work.

Acknowledgments

The author would like to express her sincere gratitude to Professor David Lindell for his invaluable support and guidance throughout this project. I would also like to thank Sherwin Bahmani, Roland Gao, and JJ Park for their assistance in this work.

References

[1] Y. Yang, Y. Yang, H. Guo, R. Xiong, Y. Wang, and Y. Liao, "UrbanGIRAFFE: Representing urban scenes as compositional generative neural feature fields," in *Proc. ICCV*, 2023.

[2] A. Liu, R. Tucker, V. Jampani, A. Makadia, N. Snavely, and A. Kanazawa, "Infinite Nature: Perpetual view generation of natural scenes from a single image," in *Proc. ICCV*, 2021.

[3] Z. Li, Q. Wang, N. Snavely, and A. Kanazawa, "InfiniteNature-Zero: Learning perpetual view generation of natural scenes from single images," in *Proc. ECCV*, 2022.

[4] S. Cai, E. R. Chan, S. Peng, M. Shahbazi, A. Obukhov, L. Van Gool, and G. Wetzstein, "DiffDreamer: Towards consistent unsupervised single-view scene extrapolation with conditional diffusion models," in *Proc. ICCV*, 2023.

[5] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "GRAF: Generative radiance fields for 3D-aware image synthesis," in *Proc. NeurIPS*, 2020.

[6] E. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis," in *Proc. CVPR*, 2021.

[7] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein, "Efficient geometry-aware 3D generative adversarial networks," in *Proc. CVPR*, 2022.

[8] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler, "GET3D: A generative model of high quality 3D textured shapes learned from images," in *Proc. NeurIPS*, 2022.

[9] D. Pavllo, J. Kohler, T. Hofmann, and A. Lucchi, "Learning generative models of textured 3D meshes from real-world images," in *Proc. ICCV*, 2021.

[10] T. DeVries, M. A. Bautista, N. Srivastava, G. W. Taylor, and J. M. Susskind, "Unconstrained scene generation with locally conditioned radiance fields," in *Proc. ICCV*, 2021.

[11] M. A. Bautista, P. Guo, S. Abnar, W. Talbott, A. Toshev, Z. Chen, L. Dinh, S. Zhai, H. Goh, D. Ulbricht, A. Dehghan, and J. Susskind, "GAUDI: A neural architect for immersive 3D scene generation," in *Proc. NeurIPS*, 2022.

[12] L. Chai, R. Tucker, Z. Li, P. Isola, and N. Snavely, "Persistent Nature: A generative model of unbounded 3d worlds," in *Proc. CVPR*, 2023.

[13] Z. Chen, G. Wang, and Z. Liu, "SceneDreamer: Unbounded 3D scene generation from 2D image collections," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 45, no. 12, pp. 15 562–15 576, Dec 2023.

[14] C. H. Lin, H.-Y. Lee, W. Menapace, M. Chai, A. Siarohin, M.-H. Yang, and S. Tulyakov, "InfiniCity: Infinite-scale city synthesis," in *Proc. ICCV*, 2023.

[15] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proc. CVPR*, 2020.

[16] J. Ost, I. Laradji, A. Newell, Y. Bahat, and F. Heide, "Neural point light fields," in *Proc. CVPR*, 2022.

[17] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. CVPR*, 2020.

[18] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. CVPR*, 2019.

[19] X. Wu, Z. Tian, X. Wen, B. Peng, X. Liu, K. Yu, and H. Zhao, "Towards large-scale 3D representation learning with multi-dataset point prompt training," in *Proc. CVPR*, 2024.

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. ECCV*, 2020.

[21] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensoRF: Tensorial radiance fields," in *Proc. ECCV*, 2022.

[22] C. Sun, M. Sun, and H. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *Proc. CVPR*, 2022.

[23] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proc. CVPR*, 2022.

[24] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022.

[25] Y. Zhang, S. Peng, S. A. Moazenipourasil, and K. Li, "PAPR: Proximity attention point rendering," in *Proc. NeurIPS*, 2023.

[26] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023.

[27] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *Proc. SIGGRAPH*, 2006.

[28] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "HoloGAN: Unsupervised learning of 3D representations from natural images," in *Proc. ICCV*, 2019.

[29] T. H. Nguyen-Phuoc, C. Richardt, L. Mai, Y. Yang, and N. Mitra, "BlockGAN: Learning 3D object-aware scene representations from unlabelled images," in *Proc. NeurIPS*, 2020.

[30] M. Niemeyer and A. Geiger, "CAMPARI: Camera-aware decomposed generative neural radiance fields," in *Proc. THREEDV*, 2021.

[31] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, and I. Kemelmacher-Shlizerman, "StyleSDF: High-resolution 3D-consistent image and geometry generation," in *Proc. CVPR*, 2022.

[32] J. Gu, L. Liu, P. Wang, and C. Theobalt, "StyleNeRF: A style-based 3D-aware generator for high-resolution image synthesis," in *Proc. ICLR*, 2022.

[33] Y. Deng, J. Yang, J. Xiang, and X. Tong, "GRAM: Generative radiance manifolds for 3D-aware image generation," in *Proc. CVPR*, 2022.

[34] I. Skorokhodov, S. Tulyakov, Y. Wang, and P. Wonka, "EpiGRAF: Rethinking training of 3d GANs," in *Proc. NeurIPS*, 2022.

[35] I. Skorokhodov, A. Siarohin, Y. Xu, J. Ren, H.-Y. Lee, P. Wonka, and S. Tulyakov, "3D generation on ImageNet," in *Proc. ICLR*, 2023.

[36] K. Deng, G. Yang, D. Ramanan, and J.-Y. Zhu, "3D-aware conditional image synthesis," in *Proc. CVPR*, 2023.

[37] Y. Xu, S. Peng, C. Yang, Y. Shen, and B. Zhou, "3D-aware image synthesis via learning structural and textural representations," in *Proc. CVPR*, 2022.

[38] M. Niemeyer and A. Geiger, "GIRAFFE: Representing scenes as compositional generative neural feature fields," in *Proc. CVPR*, 2021.

[39] Y. Xue, Y. Li, K. K. Singh, and Y. J. Lee, "GIRAFFE HD: A high-resolution 3D-aware generative model," in *Proc. CVPR*, 2022.

[40] H. Xie, Z. Chen, F. Hong, and Z. Liu, "CityDreamer: Compositional generative model of unbounded 3D cities," in *Proc. CVPR*, 2024.

[41] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001.

[42] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[43] D. Kingma, "Adam: a method for stochastic optimization," in *Proc. ICLR*, 2014.