Learning in-the-wild 3D Vehicle Generation for Realistic Sensor Simulation

Ze Yang, Jingkang Wang, Jiageng Mao
{zeyang, wangjk, jmao}@cs.toronto.edu

Abstract—Generating realistic 3D objects and rendering them at arbitrary viewpoints is important to scale the content creation and sensor simulation for robotics training the testing. To achieve this goal, we need to learn a 3D generative model that is able to synthesize 3D objects with diverse and accurate geometry, robust and photo-realistic appearance, and can be rendered efficiently. The generated 3D contents thus can be immediately deployed to facilitate downstream applications. Existing works on 3D objects generation usually cannot generate high-fidelity geometry, or cannot generate photo-realistic renderings compared to real images. This introduces a large domain gap when composing those assets with real images. In this project, we plan to utilize a 3D geometry-aware framework that learns 3D assets generation from real-world data. We model the shape and appearance of the 3D object using an implicit neural feature fields, and utilize differentiable rendering and neural rendering to synthesize the 2D image. The code is available at: https://drive.google.com/drive/folders/19uWlsDIyKh_pSXPwX9anr8osifBDrZaN?usp=share_link.

Index Terms—Neural Scene Representation, Volume Rendering, Genarative Model

1 Introduction

Inspired by the tremendous progress in 2D image generation [1], [2], [3], 3D content generation has attracted more and more attention in recent years. Existing works demonstrated the high-quality generation in different representations including point cloud [4], [5], [6], [7], voxel grid [8], [9], [10], [11], [12], mesh [13], [14], [15] or implicit geometry [16], [17], [18], [19]. However, these works usually focus on the synthetic datasets where the observations are dense and the objects are created with simplified materials and lighting conditions. Those assumptions will not hold in the real world. Specifically, the observations are often sparse and noisy (e.g., noisy segmentation masks, imperfect calibration and localization, etc). Therefore, the quality for generated meshes is not sufficient for the real applications (See Figure 8 in the state-of-the-art work [19]) such as realistic sensor simulation for self-driving.

In this project, we focus on the object-level in-the-wild 3D model generation. Built on top of the existing approaches (EG3D [16] and GET3D [19]), the ultimate goal is to generate a diverse set of 3D vehicles that contains realistic baked texture (more advanced material modeling is not considered) and can be rendered for actor insertion.

To achieve this goal, we first prepare multi-view images that contain vehicles in the real-world dataset Pandaset and Multi-view Marketplace (MVMC). We generate an object mask for each image using segmentation techniques and then filter out those low-quality images and occluded vehicles. In this way, we can obtain high-quality multi-view images that contain the same vehicles from real-world data.

As the next step, we explore two baseline approaches 1) volume rendering based approach (EG3D), 2) differentiable rendering based approach (GET3D). The overall pipeline can be summarized as follows: Given some latent codes, the network will produce some implicit feature grids or SDFs; We render the images at random viewpoints given the implicit representations either by volume rendering the feature grids or using differentiable mesh extraction and rendering; Finally, we use a GAN to judge whether the rendered images are real or fake. The full pipeline is differentiable and end-to-end trainable. We train these two baseline models on the real-world dataset. However, since real-world observations are quite sparse (several images with limited viewpoints) and localization/calibrations are noisy, the 3D generation results from the two baseline methods are not satisfactory.

To overcome the limitations and effectively handle the sparse viewpoints and noisy calibrations, we propose several techniques to improve the generation results: (1) Since the real-world observations are quite sparse (several images with limited viewpoints), it is usually challenging to generate without sufficient data priors. We add some template meshes (e.g., vehicle CAD models) as initialization and let the network predict the vertex offset, scale, etc. (2) We introduce a novel shape-aware energy function and a Laplacian loss to regularize the 3D vehicle shapes. With these improvements, our model could significantly outperform the baseline approaches and could generate high-quality 3D vehicle meshes from real-world data.

In summary, we would like to bridge the gap between synthetic and real 3D generation. The generated high-quality 3D vehicles can be potentially used to create an alternative asset bank compared to expensive/unrealistic 3D CAD models or inefficiently reconstructed assets that are widely used in the industry.

2 Related Work

We review recent advances in 3D generative models, neural rendering, and supervisory signals.

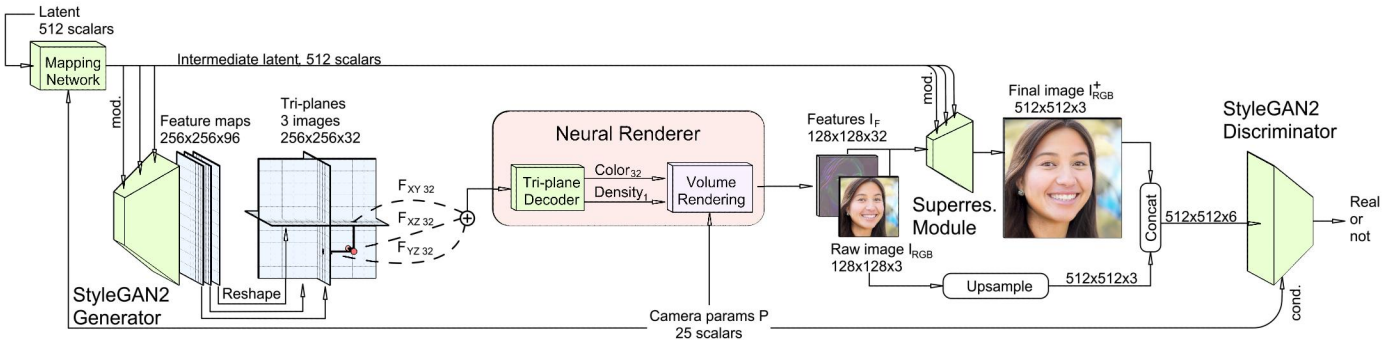


Fig. 1. Pipeline of the Efficient Geometry-aware 3D Generative Adversarial Networks (EG3D). Figure is from [16].

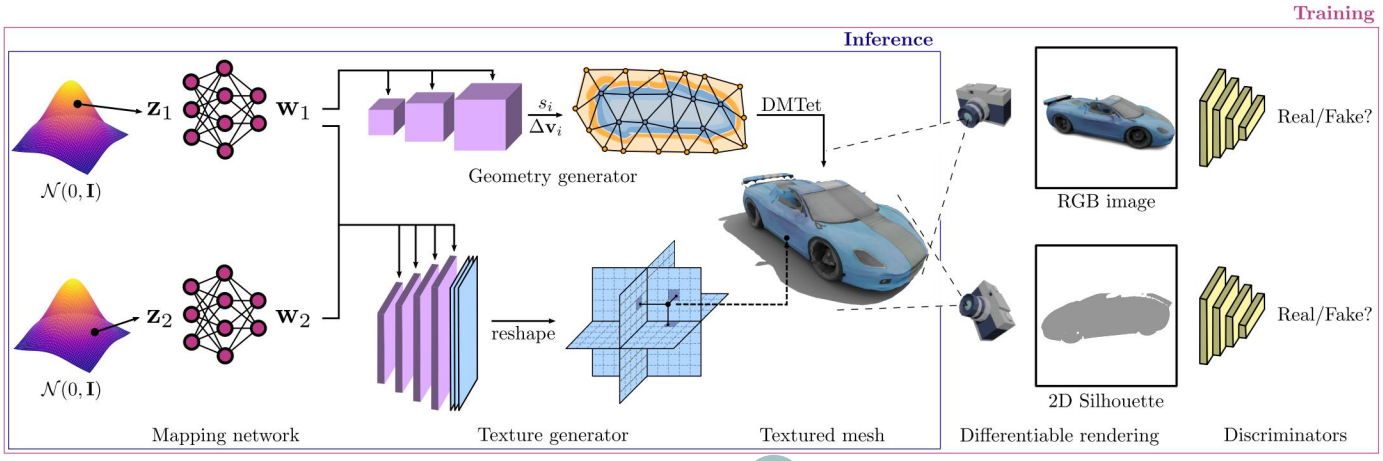


Fig. 2. Pipeline of GET3D. Figure is from [19].

2.1 3D generative models.

2D generative models have achieved remarkable progress in photo-realistic image synthesis, and many approaches attempt to extend the success of 2D generative models into the 3D space. Mesh-based GANs [20] rely on primitives used in computer graphics, but cannot generate high-fidelity images. Voxel-based GANs [21] extend the 2D convolutional generators into the 3D space. However, due to the high memory demands of voxels and the computational consumption of 3D convolutions, generating high-resolution volumes and images is quite difficult. Recent works [16], [19] incorporate implicit representations into the generative models and could produce high-quality images and 3D models efficiently. However, these approaches are trained on synthetic datasets and cannot be generalized to real-world scenarios.

2.2 Neural scene representations and neural rendering.

Neural scene representations leverage differentiable 3D-aware representations that can be optimized with 2D multi-view images via neural rendering. Some approaches [24] resort to explicit representations such as voxels, but those methods are hard to scale up to high-resolution scenarios. Other approaches utilize implicit representations [25] or hybrid explicit–implicit representations [26] for more efficient neural rendering.

2.3 3D representations and supervisory signals.

Existing works on 3D generation from images can be divided based on the 3D representations they used and the supervisory signals. Occupancy networks [27] leverage implicit representations to learn 3D reconstruction from the functional space. PointFlow [28] learns to generate 3D point clouds from images with point-wise supervision. Texture3D [29] proposes to reconstruct 3D meshes and textures from images. Those methods generally rely on 3D supervisory signals. However, since 3D models are relatively expensive to obtain, these approaches are hard to generalize to real-world scenarios.

In addition to reconstruction leveraging 3D signals, there is also a category of works that generate novel views without 3D supervision. NeRF [30] is a pioneering work that proposes neural radiance field for novel view synthesis. Numerous papers [16], [17], [18], [31] have been trying to improve NeRF for 3D-aware novel view synthesis. However, these methods are restricted to view synthesis of objects or simple indoor scenes, and cannot effectively handle complex driving scenarios.

3 Method

In this paper, we have tried three approaches for realistic 3D vehicle generation. We first apply two popular 3D generation models, i.e., EG3D [16] and GET3D [19] on the real-world self-driving data. Then, we identify the limitations of the two models, and propose a novel approach for

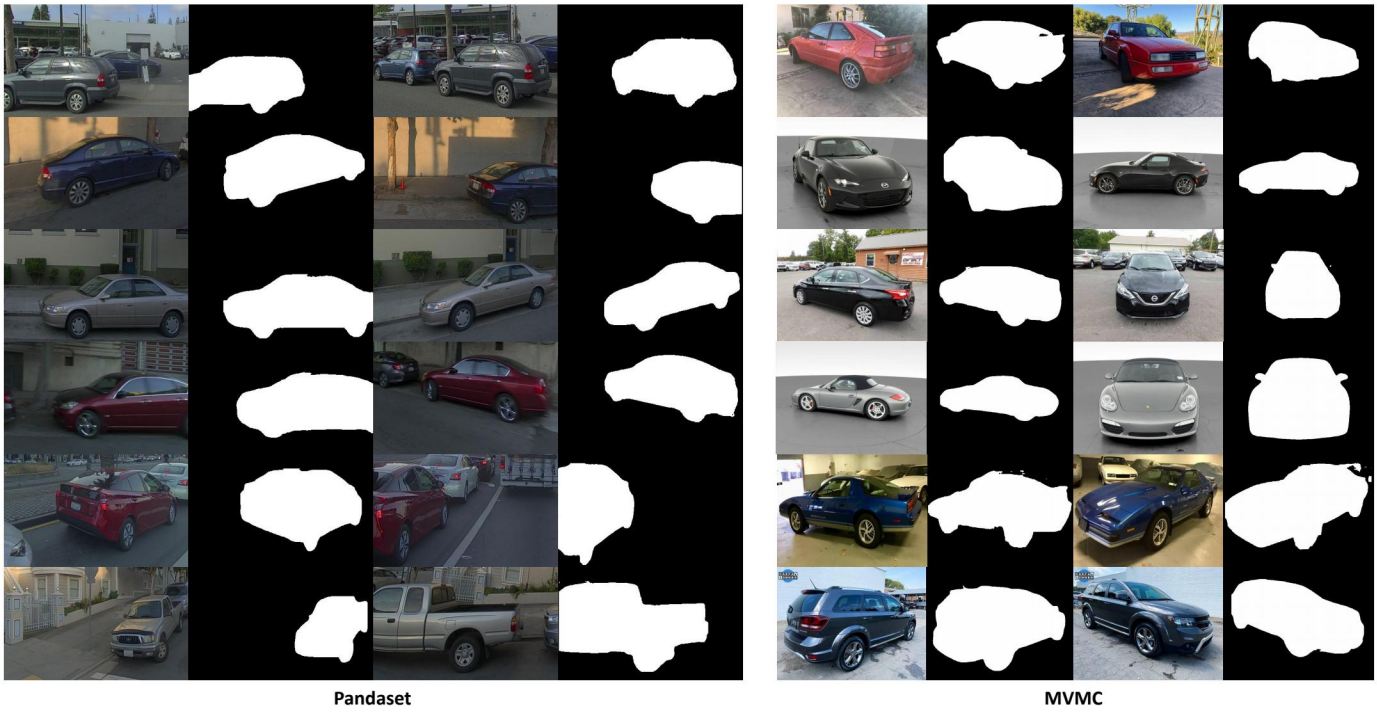


Fig. 3. Visualization of the post-processed datasets from Pandaset [22] and MVMC [23].

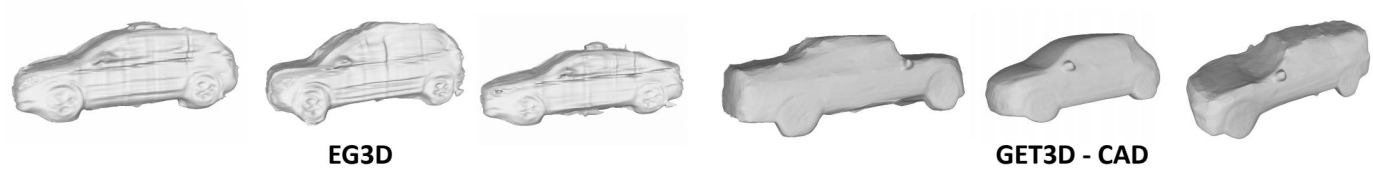


Fig. 4. Generated geometries from EG3D and GET3D - CAD on real-world datasets.

more realistic in-the-wild vehicle generation. We will first introduce the driving data preparation, and then briefly introduce the architecture of EG3D and GET3D models. Finally, we discuss our modifications to improve the 3D vehicle generation process.

3.1 Data preparation

We conduct 3D vehicle generation using the data from PandaSet [22]. PandaSet is a real-world autonomous driving dataset that contains 103 driving scenes and 8240 data frames. The data collection vehicle is equipped with 6 cameras (front, front-left, left, front-right, right and back cameras) and 2 LiDAR sensors (a 360° mechanical spinning LiDAR and a solid forward-facing LiDAR), providing 6 images from different viewpoints at the same location.

For realistic 3D object generation, we must have images that contain this object from multiple viewpoints. To achieve this, for each vehicle in PandaSet, we first obtain the corresponding multi-view images from sequential frames and different cameras. And for each image, we can also get its respective pose information from the dataset. Next, we employ PointRend [32] to generate object masks for each image. We further filter out those low-quality images where the objects inside are occluded. Specifically, we use aggregated LiDAR points to help us identify the occlusion. We also filter out those objects that only exists in several images (≤ 10 images).

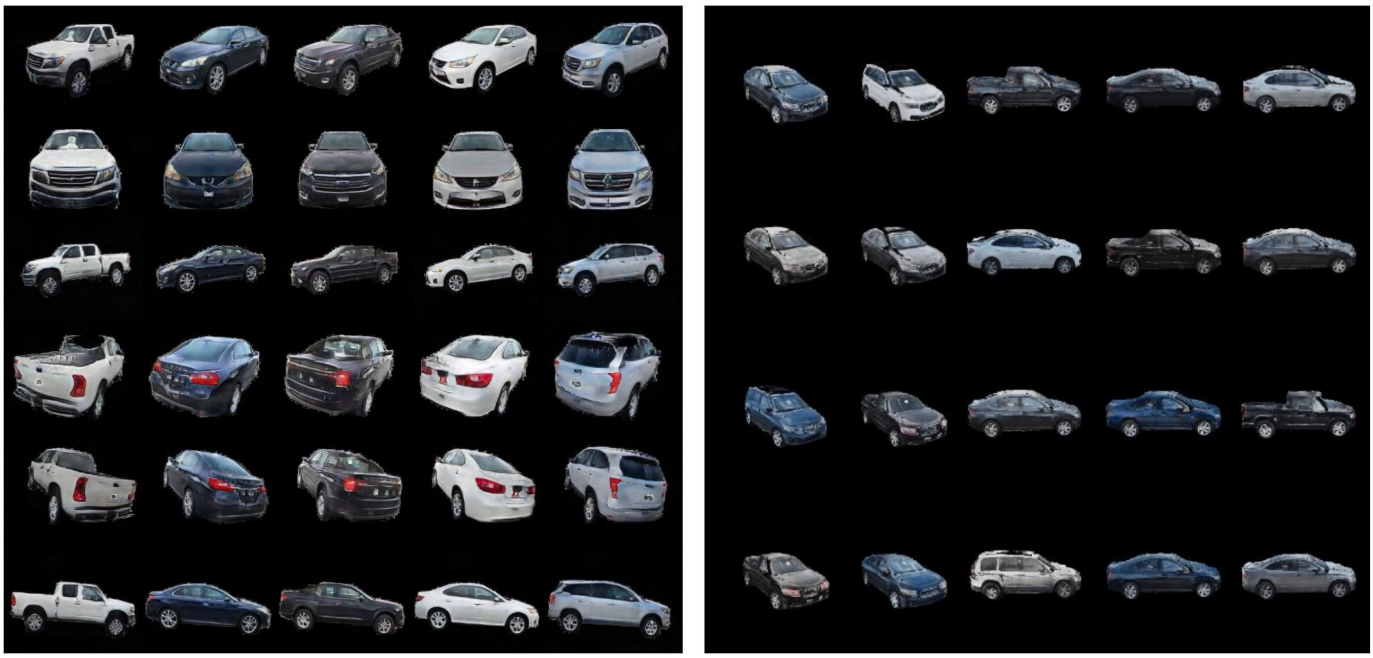
Eventually, for each select vehicle, we can get a group of high-quality images from different viewpoints and their respective pose information. We will use those images to reconstruct high-fidelity 3D vehicle models.

3.2 EG3D

The first method we have tried is the Efficient Geometry-aware 3D Generative Adversarial Networks (EG3D). The pipeline of EG3D is shown in Figure 1. This framework is mainly composed of 5 parts: a StyleGAN2-based feature generator, a novel tri-plane based feature decoder, a neural volume render, a super-resolution module, and a StyleGAN2 discriminator.

3.2.1 *StyleGAN2 feature generator.*

In the feature generator, latent code and camera parameters are fed into a mapping network to modulate the convolutional kernels of the 2D generative network. The 2D generative network produces a $256 \times 256 \times 96$ feature image, which is further sliced into three 32-channel feature planes for the following tri-plane feature decoder.



EG3D on MVMC

GET3D - CAD on MVMC

Fig. 5. Generated appearance on MVMC dataset.

3.2.2 Tri-plane feature decoder.

For each 3D querying location $Q \in R^3$, we will project this query into three orthogonal feature planes and obtain the three respective features from these planes. Then, the three feature vectors are aggregated by summation, and passed into a feature decoder that contains several MLP layers to generate color and density for this 3D querying position.

3.2.3 Neural Volume render.

The neural volume render is following NeRF [30]. We render a 128^2 feature image and for each ray, we sample 96 3D querying positions via importance sampling. The feature image is then converted into a low-resolution RGB image.

3.2.4 Super-resolution module.

To obtain a high-resolution reconstructed image and maintain a manageable computational cost, we upsample the low-resolution feature image and refine the features using convolutional kernels modulated by the mapping network in the StyleGAN2 feature generator.

3.2.5 StyleGAN2 discriminator.

To give a strong and pose-aware supervisory signal to the reconstructed images, we apply GAN-based discriminative losses on both the low-resolution and the high-resolution images. A conditional label that contains camera intrinsics and extrinsics is also fed into the discriminator for pose-aware discrimination.

3.3 GET3D

The second method we have tried is the GET3D [19] model. GET3D shares similar ideas with EG3D in terms of tri-plane and GAN-based designs. The overall framework of GET3D is shown in Figure 8. It mainly contains three parts: geometry generator, texture generator, and differentiable rendering.

3.3.1 Geometry generator.

The geometry generator maps a sample from the Gaussian distribution to a mesh. It borrows the design of DMTet [15], generates a signed distance field (SDF) from the latent code, and extracts a 3D surface mesh from the SDF.

3.3.2 Texture generator.

The texture generator aims to produce a texture map consistent with the output mesh. Similar to EG3D, we represent the texture field as a tri-plane representation. Features of each 3D querying point are extracted from three orthogonal feature planes, and the features are then mapped to textures.

3.3.3 Differentiable rendering.

With the textured meshes produced by the geometry and texture generator, a differentiable rasterizer is employed to render the 3D textured mesh into a 2D silhouette and an RGB image. Finally, GAN-based discrimination loss is applied to the 2D silhouette and image as supervision.

3.4 Real world data adaptation

3.4.1 Limitations of existing approaches.

Both EG3D and GET3D rely on hundreds of noise-free multi-view images and dense coverage on the camera viewpoints to reconstruct a single 3D object. However, in real-world settings, vehicles are normally captured by very sparse viewpoints with few images. And the reconstruction process also suffers from localization and calibration

EG3D on Pandaset

GET3D - CAD on Pandaset

Fig. 6. Generated appearance on Pandaset dataset.

errors. Hence both two methods cannot effectively handle the in-the-wild 3D vehicle generation. Formally, the vertex prediction V can be written as

$$V = V_{\text{CAD}} + V_{\text{offset}}, \tag{1}$$

where V_{CAD} denotes the coordinate for CAD template and V_{offset} represents the predicted vertex deformation.

3.4.2 Better handle real-world data

Considering the above-mentioned difficulties, we propose two techniques to resolve the in-the-wild 3D vehicle reconstruction problem.

3.4.3 Introducing CAD priors.

Both EG3D and GET3D directly generate 3D vehicle geometry with only image-level supervisions. However, due to the sparse viewpoints of images in the wild, the reconstructed vehicles can be easily collapsed or have incomplete 3D shapes. To overcome the limitations, we incorporate shape priors from CAD models into the reconstruction process. Specifically, we initialize the 3D vehicle meshes from a CAD library. Then instead of directly generating 3D shapes, we propose to learn the vertices' movements based on the template meshes for vehicle reconstruction. Compared to EG3D and GET3D, this method can stabilize the reconstruction process and produce more complete 3D shapes.

3.4.4 Shape-prior loss function.

Following [33], we adopt a shape prior function $\mathcal{L}_{\text{shape}}$ to regularize the 3D vehicle shapes. This loss function encourages the 3D shapes to be smooth. The formula is represented as:

$$\begin{aligned}
\mathcal{L}_{\text{shape}} &= \mathcal{L}_{\text{lap}} + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}} + \lambda_{\text{edge}}\mathcal{L}_{\text{edge}} + \lambda_{\text{sym}}\mathcal{L}_{\text{sym}} \\
\mathcal{L}_{\text{lap}} &= \sum_i ||\delta_i||_2^2, \\
\mathcal{L}_{\text{normal}} &= \frac{1}{N_F}\sum_{f\in F}\sum_{f'\in N(f)} ||n(f)\mathcal{L}_{\text{edge}} \cdot n(f')||_2^2, \\
\mathcal{L}_{\text{edge}} &= \frac{1}{N_E}\sum_{v\in V}\sum_{v'\in N(v)} ||v - v'||_2^2, \\
\mathcal{L}_{\text{sym}} &= \text{Chamfer}(V, V_{\text{flip}}),
\end{aligned} \tag{2}$$

where $f \in F$ and $v \in V$ are the face and vertex respectively, $N(f)$ and $N(v)$ are the neighboring faces and vertices, and N_F and N_E are the number of neighboring faces and edges. The laplacian coordinate is defined as the difference between the coordinates of each vertex and the center of mass of its immediate neighbors:

$$\delta_i = v_i - \frac{1}{N_i}\sum_{v'\in N(v_i)} v', \tag{3}$$

where N_i denotes the number of immediate neighbors (denoting by $N(v_i)$) for vertex v_i. Finally, V_{flip} denotes the flipped vertices according to the symmetry axis and Chamfer represents symmetric Chamfer distance.

4 Experiments

4.1 Datasets

We conduct experiments on the real world PandaSet [22] and MVMC dataset [23]. Please see Figure 3 for some examples of our post-processed samples.

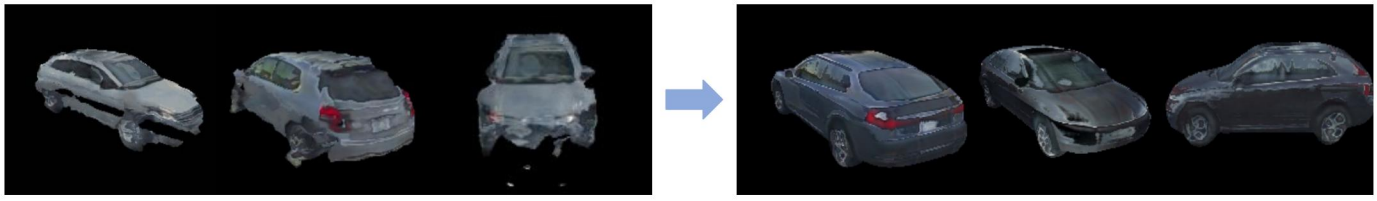


Fig. 7. Comparison of generated appearance for GET3D and GET3D-CAD.

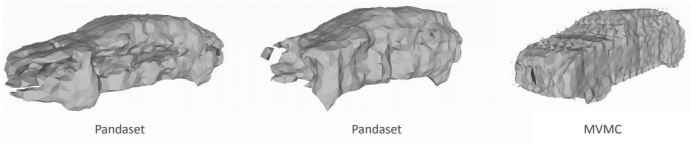


Fig. 8. Generated geometry for GET3D baseline.

4.1.1 *PandaSet.*

PandaSet is a dataset captured by the self-driving vehicle platform equipped with 6 cameras (front, front-left, left, front-right, right and back cameras) and two LiDARs (a 360° mechanical spinning LiDAR and a solid forward-facing LiDAR). The cameras and LiDARs are calibrated. PandaSet annotates instance-level 3D bounding boxes for common traffic participants in urban scenes, which can be used to extract camera images and LiDAR sweeps for diverse set of vehicles, motorcycles, *etc.* We are primarily interested in learning vehicle generation model, since vehicles are the most common actor in self-driving scenes. We use off-the-shelf segmentation algorithm [32] to estimate the instance mask for each vehicle.

4.1.2 *Multi-view Marketplace (MVMC).*

MVMC dataset is collected from online marketplace where sellers listed their cars for sale. Each listing contains multi-view images of the same vehicle. The are approximately 500 vehicles and each has ≈ 10 views. The camera calibrations are estimated from [23] and the segmentation masks are estimated from [32]. MVMC dataset contains diverse vehicle assets under different illumination, weather condition, and camera exposure settings.

4.2 Implementation details

4.2.1 *EG3D.*

We follow the official implementation[1]. We resize the input image to 256×256. We found the discriminator is getting too strong on real-world data and the generator cannot learn from the discriminator and training will fail. And we empirically found that using non-saturating GAN loss [1] with a large R1 regularization ($\gamma = 40$) can alleviate this phenomenon. We use the efficient AABB ray-box intersection algorithm to determine the entry point and exit point for volume rendering given the instance boxes. And we use 64 stratified samples and 64 important samples for volume rendering.

1. https://github.com/NVlabs/eg3d

4.2.2 *GET3D-CAD.*

We build on top of the official repo[2] and follow the same image resolution 256×256 as EG3D. We also set the R1 regularization coefficient $\gamma = 40$ for all experiments. Empirically, we find that the vanilla GAN training is not stable without adding CAD priors. The generator will crash occasionally even the model already obtains reasonable results. After adding CAD priors, we observe the training procedure becomes more stable and faster in convergence.

4.3 Results

We show the generated meshes for GET3D and EG3D in Fig. 4. We observe GET3D-CAD learns more smooth and well-regularized surface, while EG3D learns more fine-grained details but is more noisy. Both GET3D-CAD and EG3D can obtain a varierty of vehicles including pickup truck, sedan and SUV. Compared to vanilla GET3D (Fig. 8), GET3D-CAD learns more complete shapes and less noisy geometry thanks to CAD shape priors. In terms of photorealism metrics, we find the best FID (1k generated images) for GET3D-CAD is 93.21 which is better than FID for GET3D is 101.92.

In Fig. 5 and Fig. 6, we show that both EG3D and GET3D are able to produce diverse rendering results with reasonable texture quality. However, in some generated examples on Pandaset, the generated vehicle texture is not complete (e.g., missing vehicle top) as the vehicle top is rarely observed by data collection vehicle.

5 Conclusion

In this project, we focus on the large-scale 3D vehicle generation from real-world data. Firstly, we process two datasets PandaSet [22] to get more than 6k images (300+ actors) without occlusion (reasoned by shooting rays with aggregated LiDAR points). We experiment two state-of-the-art 3D GAN frameworks - EG3D [16] and GET3D [19]. We observe that 3D generation in more challenging in real world due to different noise sources (imperfect masks, camera calibration errors, etc) and sparse viewpoints. Moreover, GAN training using real-world data is much more unstable and sensitive to the parameter choices (e.g., R1 regularization coefficient). To mitigate this issue, we propose to leverage the shape priors from CAD models. Specifically, instead of extracting meshes using DMTet, we extract features from triplane feature grids and predict the per-vertex offset. We also adopt some regularization terms to guarantee the surface

2. https://github.com/nv-tlabs/GET3D

smoothness. With proper training schedule and CAD priors, EG3D and GET3D can generate a variety of real vehicles in the wild although the generation quality is worse than using synthetic datasets. EG3D geometries contain more fine-grained details and noise compared to GET3D.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[2] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[3] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8780–8794, 2021.

[4] L. Caccia, H. Van Hoof, A. Courville, and J. Pineau, "Deep generative modeling of lidar data," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5034–5040.

[5] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4541–4550.

[6] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.

[7] L. Zhou, Y. Du, and J. Wu, "3d shape generation and completion through point-voxel diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5826–5835.

[8] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," *Advances in neural information processing systems*, vol. 29, 2016.

[9] S. Lunz, Y. Li, A. Fitzgibbon, and N. Kushman, "Inverse graphics gan: Learning to generate 3d shapes from unstructured 2d data," *arXiv preprint arXiv:2002.12674*, 2020.

[10] E. J. Smith and D. Meger, "Improved adversarial systems for 3d object generation and reconstruction," in *Conference on Robot Learning*. PMLR, 2017, pp. 87–96.

[11] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *European Conference on Computer Vision*. Springer, 2016, pp. 484–499.

[12] P. Henzler, N. J. Mitra, and T. Ritschel, "Escaping plato's cave: 3d shape from adversarial rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9984–9993.

[13] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 216–224.

[14] G. Gkioxari, J. Malik, and J. Johnson, "Mesh r-cnn," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9785–9795.

[15] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler, "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6087–6101, 2021.

[16] 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 *et al.*, "Efficient geometry-aware 3d generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 123–16 133.

[17] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 154–20 166, 2020.

[18] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.

[19] 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," *arXiv preprint arXiv:2209.11163*, 2022.

[20] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger, "Towards unsupervised learning of generative models for 3d controllable image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5871–5880.

[21] M. Gadelha, S. Maji, and R. Wang, "3d shape induction from 2d views of multiple objects," in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 402–411.

[22] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang *et al.*, "Pandaset: Advanced sensor suite dataset for autonomous driving," in *ITSC*, 2021.

[23] J. Zhang, G. Yang, S. Tulsiani, and D. Ramanan, "Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 835–29 847, 2021.

[24] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *arXiv preprint arXiv:1906.07751*, 2019.

[25] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction," in *European Conference on Computer Vision*. Springer, 2020, pp. 608–625.

[26] T. DeVries, M. A. Bautista, N. Srivastava, G. W. Taylor, and J. M. Susskind, "Unconstrained scene generation with locally conditioned radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 304–14 313.

[27] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4460–4470.

[28] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4541–4550.

[29] D. Pavllo, J. Kohler, T. Hofmann, and A. Lucchi, "Learning generative models of textured 3d meshes from real-world images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 879–13 889.

[30] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[31] J. Gu, L. Liu, P. Wang, and C. Theobalt, "Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis," *arXiv preprint arXiv:2110.08985*, 2021.

[32] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2020.

[33] J. Wang, S. Manivasagam, Y. Chen, Z. Yang, I. A. Bârsan, A. J. Yang, W.-C. Ma, and R. Urtasun, "Cadsim: Robust and scalable in-the-wild 3d reconstruction for controllable sensor simulation," in *6th Annual Conference on Robot Learning*.