DisCoScene: Spatially Disentangled Generative Radiance Fields for Controllable 3D-aware Scene Synthesis Supplementary Materials

Yinghao Xu[1,2] Menglei Chai[2] Zifan Shi[3] Sida Peng[4]
Ivan Skorokhodov[5,2] Aliaksandr Siarohin[2] Ceyuan Yang[1] Yujun Shen[1]
Hsin-Ying Lee[2] Bolei Zhou[6] Sergey Tulyakov[2]

[1]CUHK [2]Snap Inc. [3]HKUST [4]ZJU [5]KAUST [6]UCLA

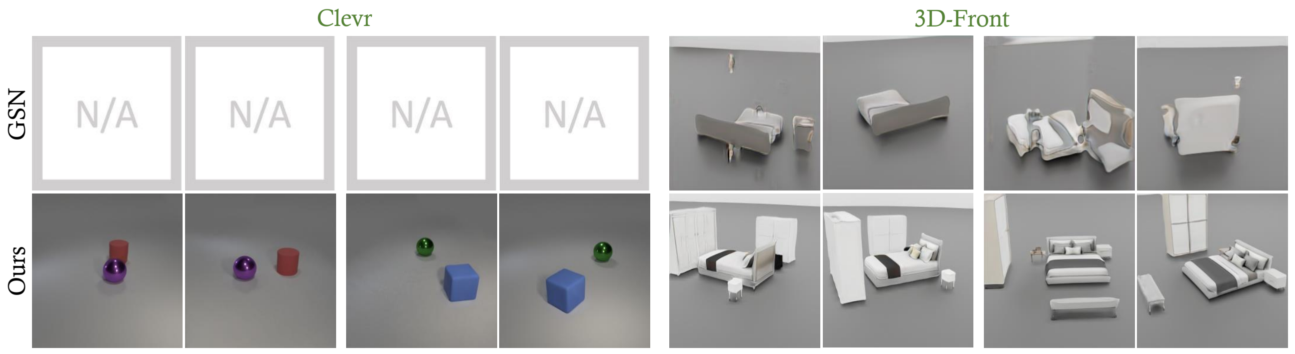

Figure 1. **Qualitative comparison between *DisCoScene* and GSN**. All images are in 256 × 256 resolution.

1. Overview

This supplementary material is organized as follows. We first introduce the discussions (Sec. 2) and additional comparison with pioneer works (Sec. 3). Then, we also present more implementation details of our *DisCoScene* (Sec. 4) and baseline approaches (Sec. 5), followed by details of data preparation (Sec. 6). We also evaluate the efficiency of our rendering pipeline against the naïve implementation (Sec. 7). Finally, we show additional results on various 3D manipulation applications to evaluate the flexibility and effectiveness of our approach (Sec. 8), as the attached demo video **id170_demo.mp4**.

2. Discussions with Related work.

Canonical Space and Spatial Condition. *Canonical Space* is a common representation for objects. Following many recent works [9, 10], we also adopt canonical space for object modeling. The difference is that we alleviate the view directions and rely on the upsampler to model view-dependent effects. *Ray-AABB Algorithm* [8] is originally proposed for efficient voxel rendering in graphics. We leverage it to improve the rendering efficiency, similar to recent works like NSVF [7] and [10] We use *Spatial Condition* to encode proper semantics for each object in 3D-Front only due to its complex layouts, while [10] uses it to model object-scene interaction and global illumination. Besides location [10], we also use the scale to provide spatial cues.

GIRRAFE. We train GIRAFFE on our own CLEVR dataset using official code rather than pre-trained models. We tried hard yet couldn't produce equally good results as shown in the paper in Fig. 2. The reason why GIRAFFE doesn't work well on Waymo is that the layouts sampled from human-designed priors do not fit real distributions. It also lacks object-level supervision, while ours benefits from the local-global discrimination. Besides, the renderer and upsampler are not as powerful as ours, harming the training convergence and performance.

3. Additional Comparison

Qualitative Comparison with GSN [2]. Here, we also include GSN to compare with our method in Fig. 1. GSN highly depends on the training camera trajectories. Thus in our setting, where the camera positions are randomly sampled, it suffers from poor image quality and multi-view consistency. *

*We fail to train GSN on CLEVR and WAYMO with the official implementation, hence we do not report the quantitative results.

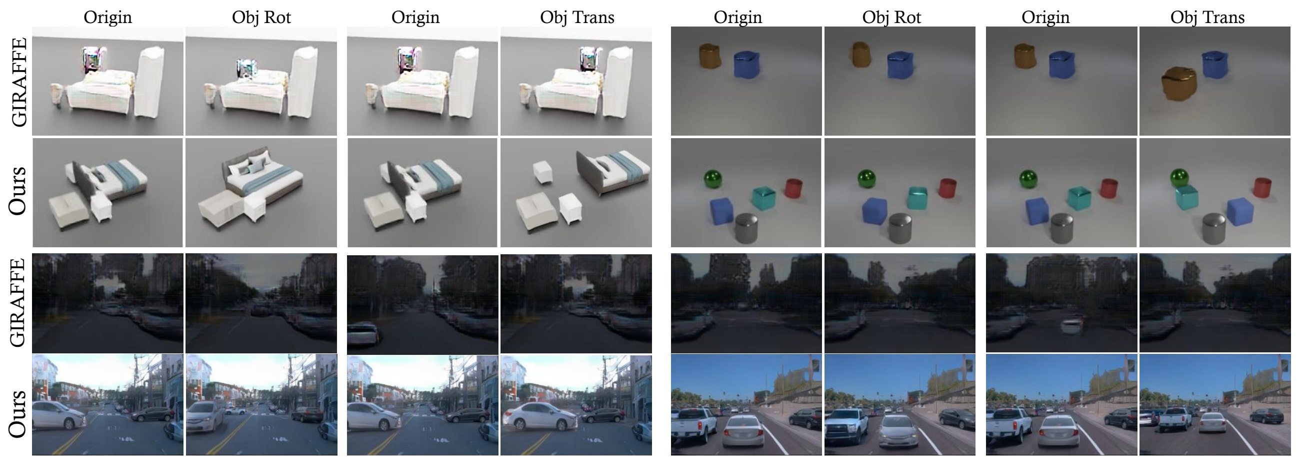

Figure 2. **Comparison with GIRAFFE** on all datasets. All images are in 256×256 resolution

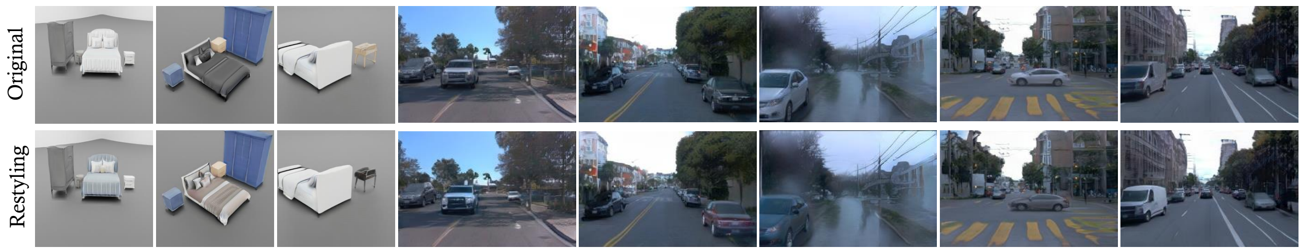

Figure 3. **Well-Curated recoloring results** on 3D-FRONT and WAYMO

Qualitative Comparison with GIRAFFE [9]. We also supplement the comparison results with GIRRAFE on CLEVR and 3D-FRONT in Fig. 2. Even in these diagnostic datasets, GIRRAFFE also suffers from inconsistency when editing the object pose and location. However, *DisCoScene* always synthesizes very realistic objects whatever the rotation and translation are carried out on the objects.

Restyling Results. Although we don't model the object's shape and color explicitly with specific latent codes, we can still achieve disentangled control over color and shape using a hierarchical latent space. However, this approach can sometimes result in minor artifacts. To demonstrate the control over recoloring objects, we provide additional curated samples in Fig. 3.

4. Implementation Details of *DisCoScene*

Background with NeRF++ [14]. The outdoor datasets, *i.e.* WAYMO, have unbounded backgrounds. It is insufficient to model the whole scene in the image within a fixed bounding box. Therefore, we inherit the inverse parametrization of NeRF++ to model the background in WAYMO:

$$\boldsymbol{x} = (x/r, y/r, z/r), \tag{1}$$

where $r = ||\boldsymbol{x}||^2$. The background points are uniformly sampled in an inverse depth range of $[1/R, 0)$ where $R = 2.0$ denotes the starting depth of the background.

Constant Latents for Upsampler. We adopt similar architecture and parameters of the synthesis network from StyleGAN2 [6] as the upsampler for the rendered 2D feature map. Note that since our model handles multiple radiance fields, different spatial locations of the convolution feature maps should be modulated by different $\boldsymbol{w}$ codes belonging to specific objects, making it costly to upsample the feature map. Thus we disable the spatial-aware modulation by setting $\boldsymbol{w}$ as a constant tensor with value 1, which significantly reduces the computation overhead.

5. Implementation Details of Baselines

Because of the wildly divergent data distribution, the training parameters vary greatly on different datasets. Tab. 1 and Tab. 2 list the detailed training configurations of different datasets for each baseline. *FOV*, $Range_{depth}$, and *#Steps* denote the field of view, the depth range, and the number of sampling steps along a camera ray, respectively. $Range_h$ and $Range_v$ denote the horizontal and vertical angle ranges of the camera pose ξ. *Sample_Dist* denotes the sampling scheme of the camera pose. We only use Gaussian or uniform sampling in our experiments. λ is the loss weight of the gradient penalty.

VolumeGAN [13]. We use the official implementation of VolumeGAN.[†] We train VolumeGAN with $25K$ images. The coordinates range of feature volume is adjustable for different datasets. We adopt the training configuration in Tab. 1 to train VolumeGAN models.

[†]https://github.com/genforce/volumegan

Table 1. **Training configurations** regarding different datasets for VolumeGAN and EpiGRAF.

Datasets	FOV	Radius	Range$_{\text{depth}}$	#Steps	Range$_h$	Range$_v$	Sample_Dist	λ
CLEVR	12.0	1.0	$[0.8, 1.2]$	24	$[\pi/2 - 0.5, \pi/2 + 0.5]$	$[\pi/4 - 0.15, \pi/4 + 0.15]$	Uniform	1
3D-FRONT	12.8	1.0	$[0.7, 1.3]$	24	$[0, 2\pi]$	$[3\pi/8 - 0.2, 3\pi/8 + 0.2]$	Uniform	1
WAYMO	12.0	1.0	$[0.7, 1.3]$	24	$[\pi/2 - 0.5, \pi/2 + 0.5]$	$[\pi/2 - 0.15, \pi/2 + 0.15]$	Uniform	1

Table 2. **Training configurations** regarding different datasets for EG3D.

Datasets	FOV	Radius	Range$_{\text{depth}}$	#Steps	Range$_h$	Range$_v$	Sample_Dist	λ
CLEVR	18.0	1.7	$[0.1, 2.6]$	96	$[\pi/2 - 0.5, \pi/2 + 0.5]$	$[\pi/4 - 0.15, \pi/4 + 0.15]$	Uniform	2
3D-FRONT	18.8	2.7	$[2.2, 3.3]$	96	$[0, 2\pi]$	$[3\pi/8 - 0.2, 3\pi/8 + 0.2]$	Uniform	2
WAYMO	18.0	1.7	$[0.1, 2.6]$	96	$[\pi/2 - 0.5, \pi/2 + 0.5]$	$[\pi/2 - 0.15, \pi/2 + 0.15]$	Uniform	5

EpiGRAF [11] We use the official implementation of EpiGRAF.[‡] We inherit the patch-wise training scheme to train EpiGRAF with the same data and camera parameters at the target resolution shown in Tab. 1.
EG3D [1]. We use the official implementation of EG3D. [§] Different from VolumeGAN and EpiGRAF, EG3D renders the whole radiance field within a bounding box, so we inherit the larger camera radius than the ones of EpiGRAF and VolumeGAN for training. Since the original EG3D requires pose annotations for training, we add a pose sampler in it to enable the training on all three datasets as the global annotations are not always available. We adjust the loss weight of gradient penalty on different datasets to achieve the best performance. Hyperparameters used for training are available in Tab. 2
GSN [2]. We use the official GSN implementation.[¶]. GSN highly dependents on input camera sequences and we find it very difficult to converge at a narrow camera distribution, *i.e.* WAYMO and CLEVR. On 3D-FRONT, we set the length of camera sequence to 1, and it can converge to some extent. We don't leverage depth supervision for a fair comparison with our method.
GIRAFFE [9]. We use the official implementation of GIRAFFE.[‖] The number of boxes for training follows the configuration of ours on each dataset. The bounding box generator of GIRAFFE is tuned specifically for each dataset for a fair comparison.

6. Data Preparation

Clevr [5]. We use the official script [5] to render scenes with Cube, Cylinder, and Sphere primitives. The camera position is jittered in a small scale. And the dataset is rendered in a 256×256 resolution with $80K$ samples.
3D-Front [3, 4]. We use BlenderProc to render 20 images per room in 3D-FRONT. We move the center of each room to the coordinate origin and then sample the camera positions on the upper sphere between $2r$ to $3r$ where r is the diagonal length of room.
Waymo [12]. We only keep the front view of WAYMO for the model training. However, there exist lots of occluded and noisy cars in WAYMO, we design several heuristic rules to filter it. Specifically, we require the camera depth of car is less than $40m$ and the area of cars is larger than 40000 pixels in original image size (1920×1280). We then adopt the black padding to make images square and then resize it in to 256×256 resolution.

[‡]https://github.com/universome/epigraf
[§]https://github.com/NVlabs/eg3d
[¶]https://github.com/apple/ml-gsn
[‖]https://github.com/autonomousvision/giraffe

Table 3. **Ablation analysis** of efficient rendering (ER).

	TR.↓		INF.↓	
	w/ ER	*w/o* ER	*w/* ER	*w/o* ER
CLEVR	**18.1**	29.2	**95**	180
3D-FRONT	**22.3**	38.1	**110**	330
WAYMO	**19.2**	30.9	**100**	195

7. Efficiency of Rendering Pipeline

Naïve point sampling solutions where the density and color of spatial points are inferred with multiple object radiance fields can lead to prohibitive computational overhead. Therefore we propose an efficient rendering pipeline by only focusing on the valid points within the bounding boxes. Tab. 3 presents the training cost in *V100 days* and testing cost in *ms/image* (on a single V100 over $1K$ samples) with our efficient rendering and Naïve rendering. Our rendering pipeline can handle multiple objects efficiently, with nearly 1.5 and 2 times faster training and inference speed, respectively.

8. Additional Results

We include a demo video, which shows more results of various 3D manipulation applications. From the video, we can see that our method both achieves good generation quality and enables precise object control. We also include comparisons with the state-of-the-art methods, *i.e.*, EG3D [1] and GIRRAFE [9], in the video.

References

[1] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 3, 4

[2] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Int. Conf. Comput. Vis.*, 2021. 1, 3

[3] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3

[4] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *Int. J. Comput. Vis.*, 2021. 3

[5] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 3

[6] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[7] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 2020. 1

[8] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 2018. 1

[9] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 1, 2, 3, 4

[10] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2856–2865, 2021. 1

[11] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. In *Adv. Neural Inform. Process. Syst.*, 2022. 3

[12] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 3

[13] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2

[14] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2