Supplementary Material for UrbanGIRAFFE: Representing Urban Scenes as Compositional Generative Neural Feature Fields

Yuanbo Yang, Yifei Yang, Hanlei Guo, Rong Xiong, Yue Wang, Yiyi Liao*

Zhejiang University

{yybbb, yfyang2018, guohl77701, rxiong, ywang24, yiyi.liao}@zju.edu.cn

Abstract

In this supplementary material, we first provide implementation details in Section 1. Next, we present more details of baselines in Section 3. We further describe the datasets in Section 4. Finally, we report additional experimental results in Section 5. The supplementary video shows synthesized animations including viewpoint control, stuff and object editing. We will make our code publicly available.

1. Implementation Details

1.1. Object Patch Rendering

We render object patches taking occlusions into consideration. First, this allows us to directly crop the object patches from the full composited image without additional computation to render the complete objects. Second, we can keep occluded object patches in the real images for training, as filtering out occluded objects is not trivial. More specifically, we first obtain the alpha value of a ray corresponding to the kth object via volume rendering:

$$\mathrm{A}_{obj}^{k} = \sum_{i=1}^{N} T_i \alpha_i \mathbb{1}[\mathbf{x}_i \in \{\mathbf{R}(\mathbf{s} \odot \mathbf{x}_{obj}^{k}) + \mathbf{t}\}] \tag{1}$$

This means $T_i\alpha_i$ is accumulated if the corresponding sampled point $\mathbf{x}_i$ belongs to the kth object. Here, we use the transmittance T_i of the composited scene obtained via Eq. 7 of the main paper, meaning that the alpha value is close to 0 if the object is occluded. Let $\mathbf{A}_{obj}^{k} \in \mathbb{R}^{H_f \times W_f}$ denote the alpha map consisting of object alpha values of all rays. Note that $\mathbf{A}_{obj}^{k}$ is obtained via volume rendering, and thus its resolution is lower than the final output image $\hat{\mathbf{I}} \in \mathbb{R}^{H \times W \times 3}$. Thus, we upsample $\mathbf{A}_{obj}^{k}$ via nearest neighbor sampling to obtain the object patches from $\hat{\mathbf{I}}$:

$$\hat{\mathbf{P}}_k = crop(\hat{\mathbf{I}} \odot up(\mathbf{A}_{obj}^{k})) \tag{2}$$

where $up(\cdot)$ denotes nearest neighbor upsampling and $crop(\cdot)$ denotes cropping the object based on its projected 3D bounding box. Fig. 1 displays patches and masks of rendered objects.

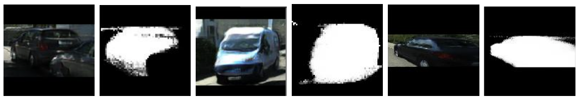


Figure 1: **Object patches and corresponding masks**

---

[*] Corresponding author.

	FID$_{win}$	FID$_{seq}$
GSN	127.1	118.8
GIRAFFE	110.9	121.3
Ours	**43.3**	**55.2**

Table 1: FID on test set.

1.2. Network Architecture

Generator Architecture: For the latent codes, we use a 256-dimension $\mathbf{z}_{wld}$ for the entire scene and a 256-dimension $\mathbf{z}_{obj}^{k}$ for each object. Our *stuff generator* consists of a semantic-conditioned feature grid generator and an MLP head. The feature grid generator G_{θ}^{vol} consists of 5 spatially-adaptive normalization blocks. Each block follows the structure of a ResNet [2] block with two convolutional layers, except that the batch normalization is replaced with spatial-adaptive normalization modulated by the semantic labels. As illustrated in Fig. 3 of the main paper, we inject the latent code $\mathbf{z}_{wld}$ at each block following [5]. The MLP head G_{θ}^{stf} is a 4-layer ReLU MLP with a hidden dimension of 256. The *object generator* G_{θ}^{obj} is an 8-layer ReLU MLP but with a lower dimension of 128. We use skip connections at the fourth layer for G_{θ}^{obj}. For the sky generator, we use a 5-layer ReLU MLP of hidden dimension 256 without a skip connection. As mentioned in the main paper, we apply positional encoding $\gamma(\cdot)$ to both $\mathbf{x}_{obj}^{k}$ and $\mathbf{x}_{wld}$:

$$\gamma(p) = (sin(2^{0}\pi p), cos(2^{0}\pi p), sin(2^{1}\pi p), cos(2^{1}\pi p), \cdots, sin(2^{L-1}\pi p), cos(2^{L-1}\pi p) \tag{3}$$

where $\gamma(p)$ is applied to each element of the coordinate. We use $L = 10$ for both $\mathbf{x}_{obj}^{k}$ as input to the object generator and $\mathbf{x}_{wld}$ as input to the stuff generator.

Our *2D neural renderer* π_{θ}^{neural} consists of two blocks of StyleGAN2-modulated convolutional blocks and one upsampling layer. In practice, we render the feature maps $\mathbf{I_F} \in \mathbb{R}^{H_f \times W_f \times M_f}$ at half resolution and upsample it to image $\hat{\mathbf{I}} \in \mathbb{R}^{H \times W \times 3}$ at the target resolution, i.e., $H_f = H/2, W_f = W/2$.

Discriminator Architecture: We adopt two independent discriminators to apply adversarial losses to the composited images and the object patches, respectively. Both discriminators follow the design choice of the StyleGAN2 discriminator, while the object discriminator takes a lower-resolution image as input and thus has fewer parameters. More specifically, all object patches are rescaled to 128×128 pixels, and the object discriminator $D_{\phi}^{\mathbf{P}}$ has 6 convolutional blocks with 5 downsampling layers. The discriminator of the full image $D_{\phi}^{\mathbf{I}}$ has 8 convolutional blocks with 7 downsampling layers.

1.3. Training and Inference

We train our model on four Nvidia GeForce RTX 3090 with a batch size of 16 for 100k iterations, taking 4 days in total. For inference, our method can render an image at the resolution of 188×704 at roughly 5 FPS.

2. Details about Evaluation

To better assess the model's generalization capabilities and its ability to produce convincing results across different layouts. We further split an entire sequence from the KITTI-360 dataset as validation set. The results, as presented in Table 4, demonstrate that even in the challenging scenario of an unseen sequence, the new evaluation results (referred to as FID$_{seq}$) only exhibit a slight degradation compared to the original FID reported in the main paper (referred to as FID$_{win}$)which remains reasonably good.Importantly, our method still significantly outperforms baselines methods

3. Baselines

GIRAFFE: We follow the original implementation[1] of GIRAFFE [4]. For the KITTI-360 dataset, we sample objects following the same object layout prior as used in our method. We also sample points within the objects using the same ray-box intersection strategy for a fair comparison. We train GIRAFFE on four Nvidia GeForce RTX 3090 with a batch size of 16 for 140k iterations.

GSN: We use the official implementation[2] of GSN [1]. GSN generates a scene based on a 2D grid of local latent codes. Following the original implementation, the spatial resolution of the 2D grid is set to 32×32. For the KITTI-360 dataset, we

---

[1] https://github.com/autonomousvision/giraffe

[2] https://github.com/apple/ml-gsn

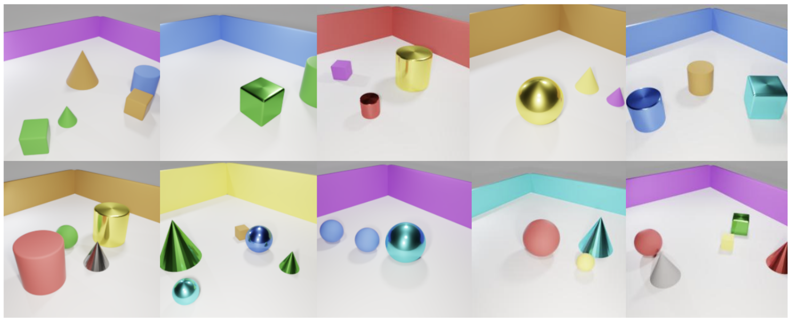


Figure 2: **Preview of CLEVR-W**. We add walls with randomly sampled colors and locations and render a set of images based on the rendering script of CLEVR [3].

set the maximum sampling distance as $80m$, with each pixel in the 2D grid corresponding to a region of $2.5m \times 2.5m$ in reality. Because of the limitation of computational resources, we sample 32 points per ray instead of 64. We train GSN for 400k iterations with a batch size of 4 on two 3090 GPUs and perform gradient accumulation every two iterations.

2D Baseline: We evaluate StyleGAN2 as a state-of-the-art 2D GAN following its PyTorch implementation[3]. We train the 2D baseline on four Nvidia GeForce RTX 3090 with a batch size of 32 for 120k iterations, taking 2 days in total.

4. Dataset

KITTI-360: We adopt the KITTI-360 dataset to evaluate our method on urban scenes. For the real object (i.e., cars) patches, we filter out heavily occluded and far away objects based on the following criteria: 1) The pixel number of the object is larger than a given threshold; 2) The projected 3D bounding box has at least four visible vertices. Note that this does not filter out all occluded objects. We use instance masks provided by KITTI-360 to crop the object patches for training. These masks may be noisy as they are obtained via label transfer algorithms instead of manually labeled. Therefore, using instance masks predicted by 2D segmentation methods might be possible. In order to obtain training images containing enough objects, we keep one image only when it contains one or more than one valid object patches for training. This leads to a total of 20k training images.

We train and evaluate our method at a half resolution of KITTI-360, i.e., 188×704 pixels. The same resolution is applied to the other baselines except for GSN, which requires large memory consumption and does not scale to the target resolution. Therefore, we train and evaluate GSN at the image resolution of 94 × 352 pixels.

In all of our experiments, we use voxel grids at the resolution of $64 \times 64 \times 64$ voxels. Note that the sampling interval of the semantic voxel is $1m$ horizontally and $0.25m$ vertically. Therefore, each semantic voxel grid covers an area of $4096m^2$ with a height of $16m$ in the real world.

CLEVR-W: We further create a dataset CLEVR-W to facilitate comparison with GIRAFFE [4] in more controlled environments. The dataset contains 10k images rendered at the resolution of 256×256. In contrast to the dataset used in GIRAFFE, we add walls as stuff regions and thus increasing the difficulty of modeling the background. The walls are sampled at different locations with random colors, see Fig. 2 for a preview.

4.1. Semantic Voxel Grids

In the case of KITTI-360, we take into account 16 categories, which include vegetation, terrain, ground, road, sidewalk, parking, rail track, building, gate, garage, bridge, tunnel, and wall. As for CLEVR-W, we focus exclusively on wall and ground within the semantic voxel grids. Additionally, we establish priorities for each semantic category. In situations where multiple stuff elements occupy a voxel, our selection process favors the element with the highest priority. This applies specifically to the categories within the semantic voxel grids.

---

[3] https://github.com/rosinality/stylegan2-pytorch

5. Additional Experimental Results

5.1. Additional Comparison to Baselines

KITTI-360: In Fig. 3, we show additional comparison results to the baselines on KITTI-360, where each column shows a single scene with the camera consecutively moving forward for up to 20 meters. Note that the background regions of GIRAFFE barely change despite the large camera movement, since GIRAFFE models the background as a far-away planar structure in the challenging urban scenario. GSN can model the camera movement faithfully but has lower image fidelity, especially when synthesizing an image with a large camera moving distance. In contrast, our method is able to synthesize high-fidelity images along the large camera moving distance.

CLEVR-W: In Fig. 4, we show additional comparisons to GIRAFFE on CLEVR-W. Our method enables control over objects, stuff, and the camera pose.

5.2. Additional Results on Controllable Image Synthesis

Fig. 6 shows additional scene editing results on KITTI-360, including stuff editing (including lower building in Fig. 6a, building to tree in Fig. 6b, road to grass in Fig. 6c) and object editing (Fig. 6d).

5.3. Limitations

Fig. 5 illustrates two limitations of our method. First, we observe in Fig. 5a that changing $\mathbf{z}_{wld}$ does not change the appearance of the stuff significantly. In contrast, we observe that editing the semantic layout can sometimes change the appearance of the scene. This is rational as the appearance is entangled with the semantic layout in the real world, e.g., to model shadows. Fig. 5b shows that our sky generator sometimes generates far-away buildings and thus yields artifacts. This is due to the fact that our semantic voxel grids can only model a region of 64×64 square meters.

5.4. Random Samples

We show randomly sampled, uncurated results on both KITTI-360 and CLEVR-W in Fig. 7.

References

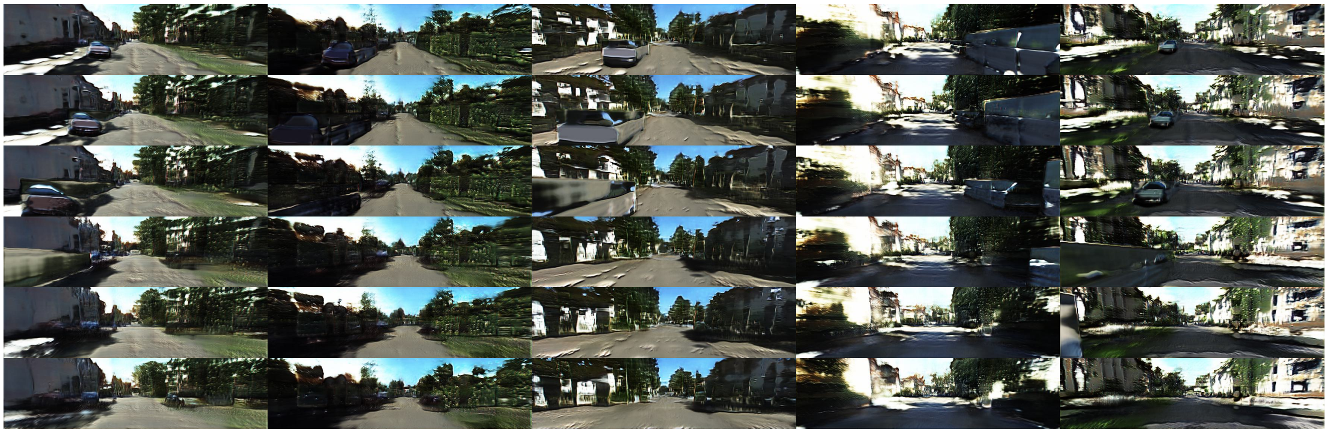
[1] Terrance DeVries, Miguel Ángel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

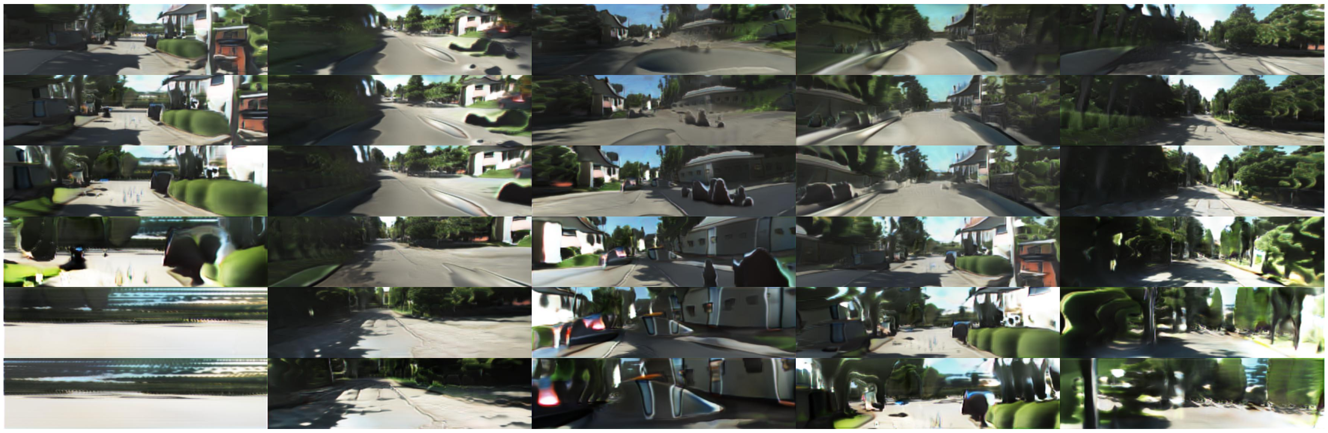
[3] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[4] Michael Niemeyer and Andreas Geiger. GIRAFFE: representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

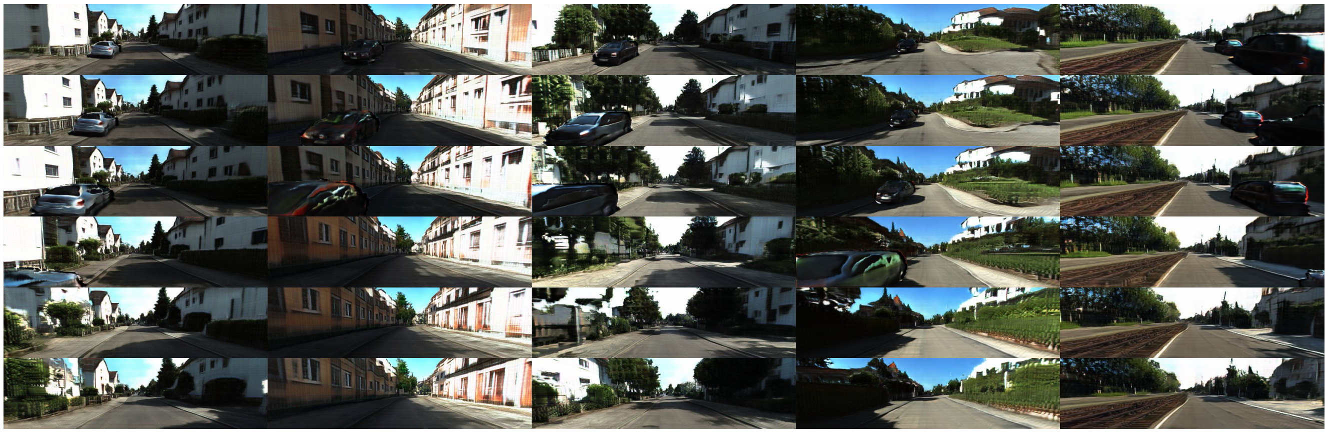
[5] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. 2



(a) GIRAFFE

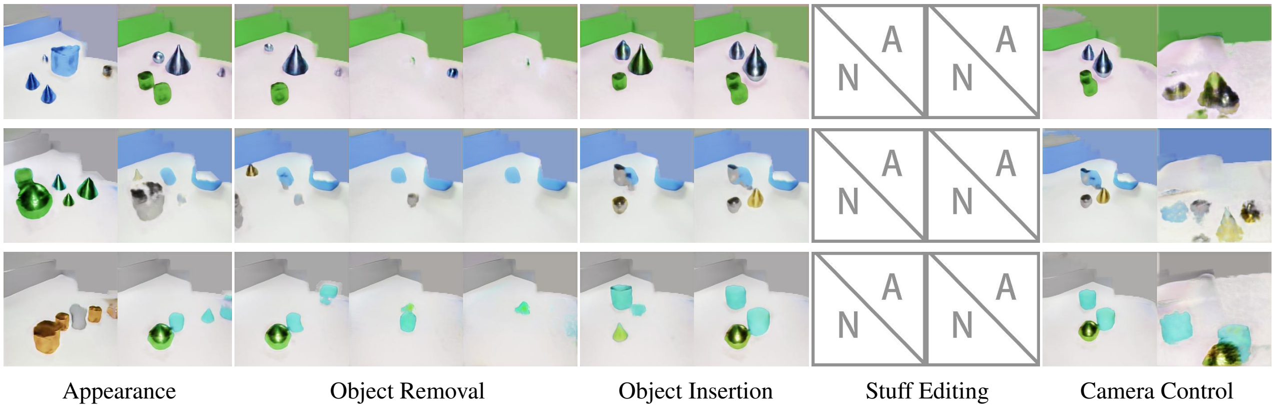


(b) GSN

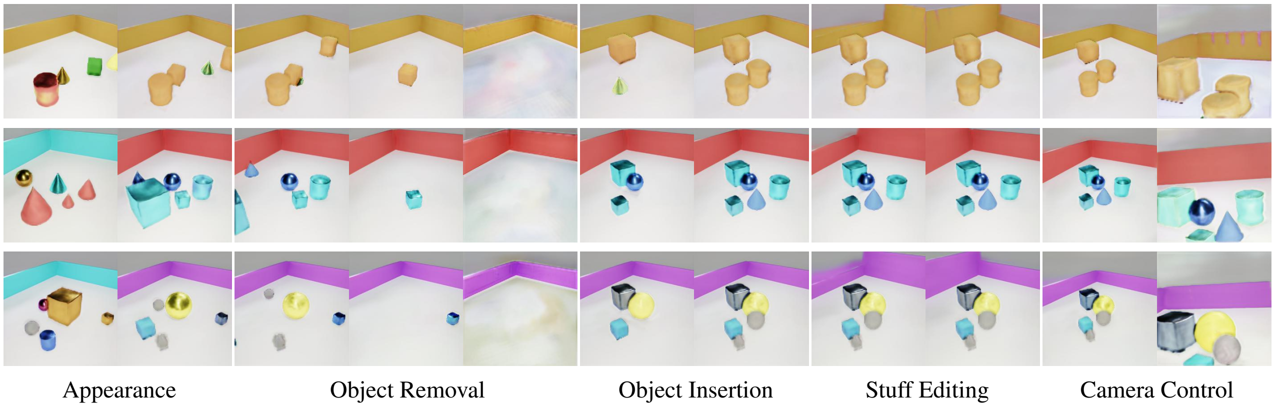


(c) UrbanGIRAFFE (Ours)

Figure 3: **Additional Qualitative Comparison on KITTI-360**. The camera moves up to 20 meters in each scene.

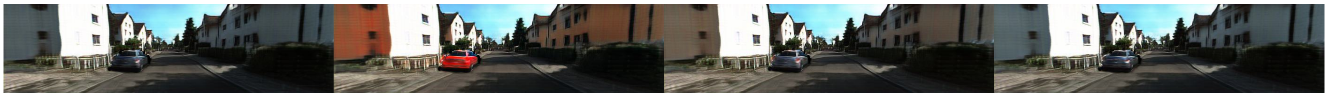


(a) GIRAFFE

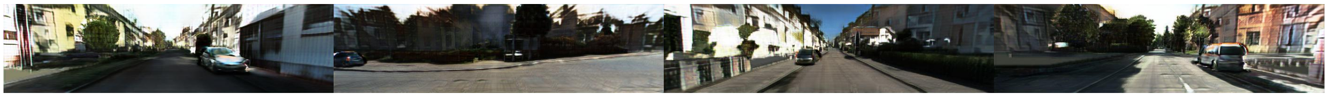


(b) UrbanGIRAFFE (Ours)

Figure 4: **Additional Qualitative Comparison on CLEVR-W**. Our method enables stuff editing and shows more convincing results in camera viewpoint control.

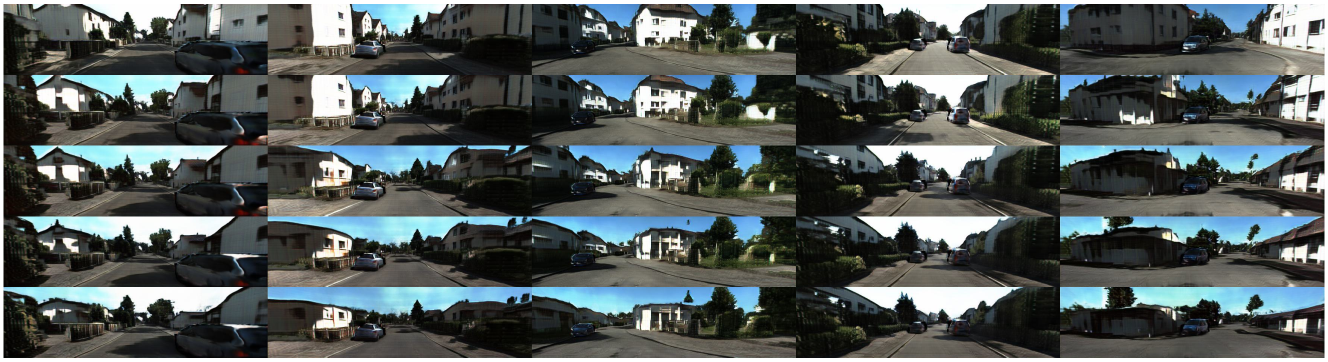


(a) Sampling $\mathbf{z}_{wld}$ can only slightly adjust the stuff appearance

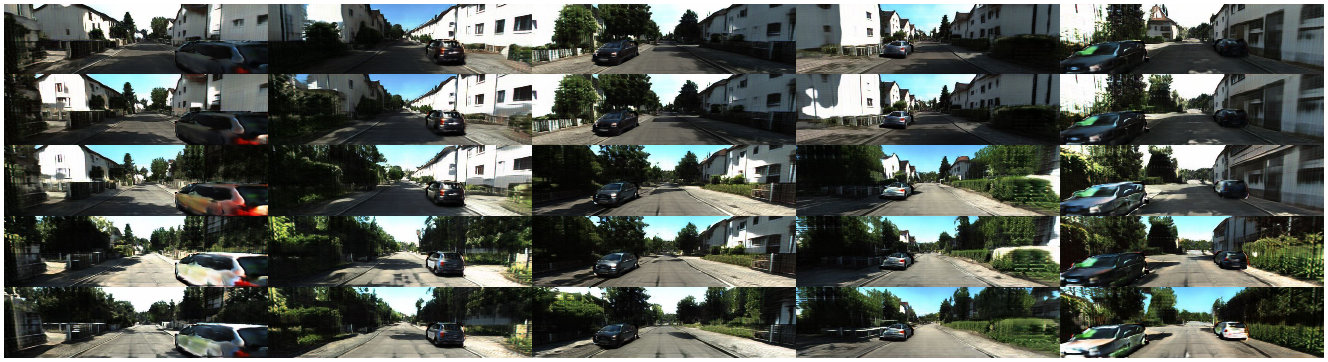


(b) Sky occasionally models far buildings

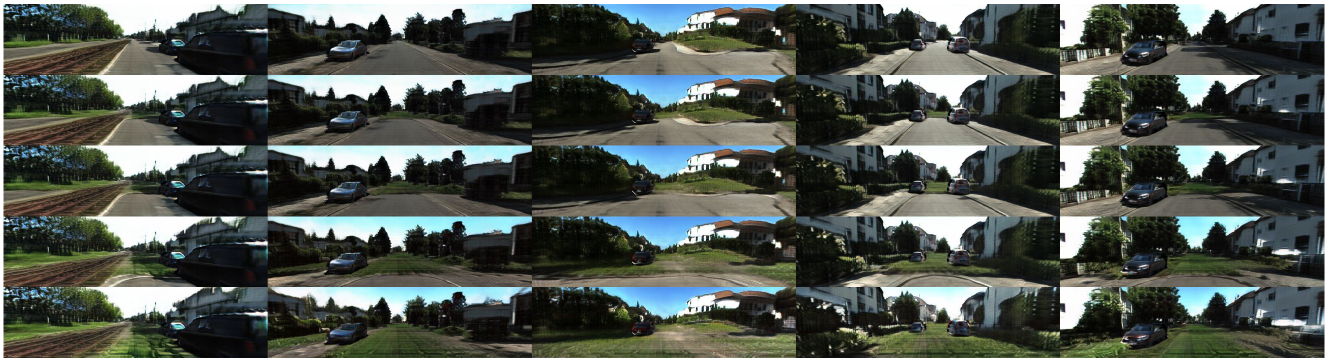
Figure 5: **Limitations** of UrbanGIRAFFE.



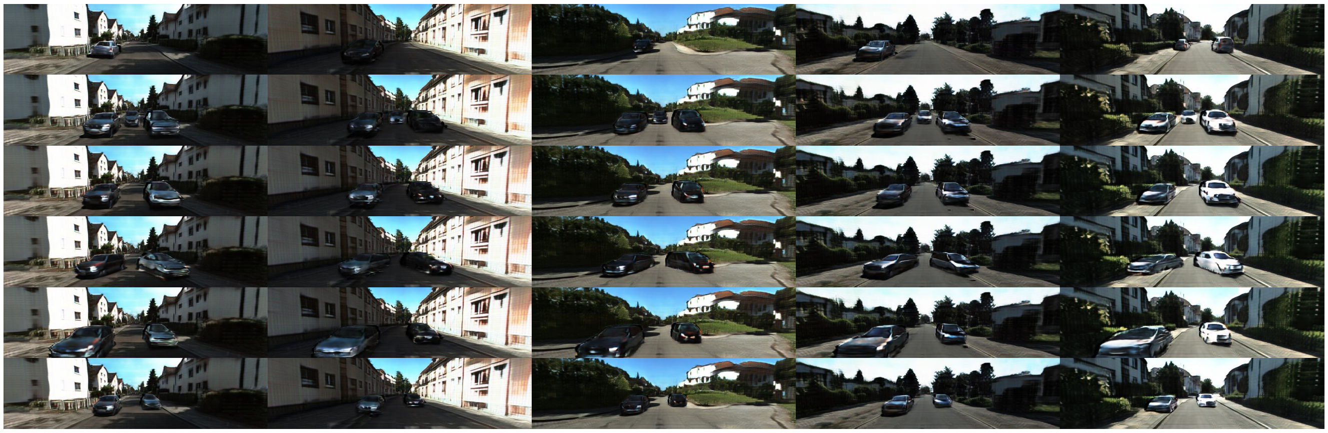
(a) Lower building



(b) Building to tree

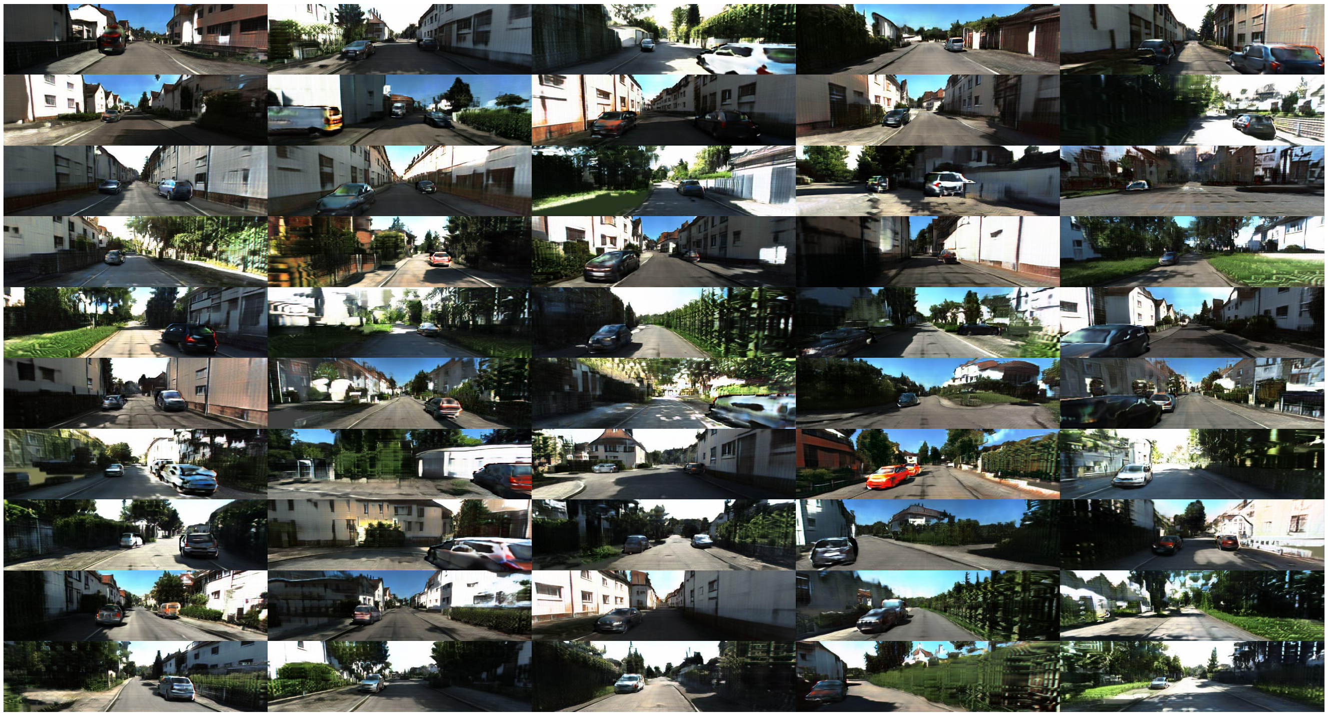


(c) Road to grass

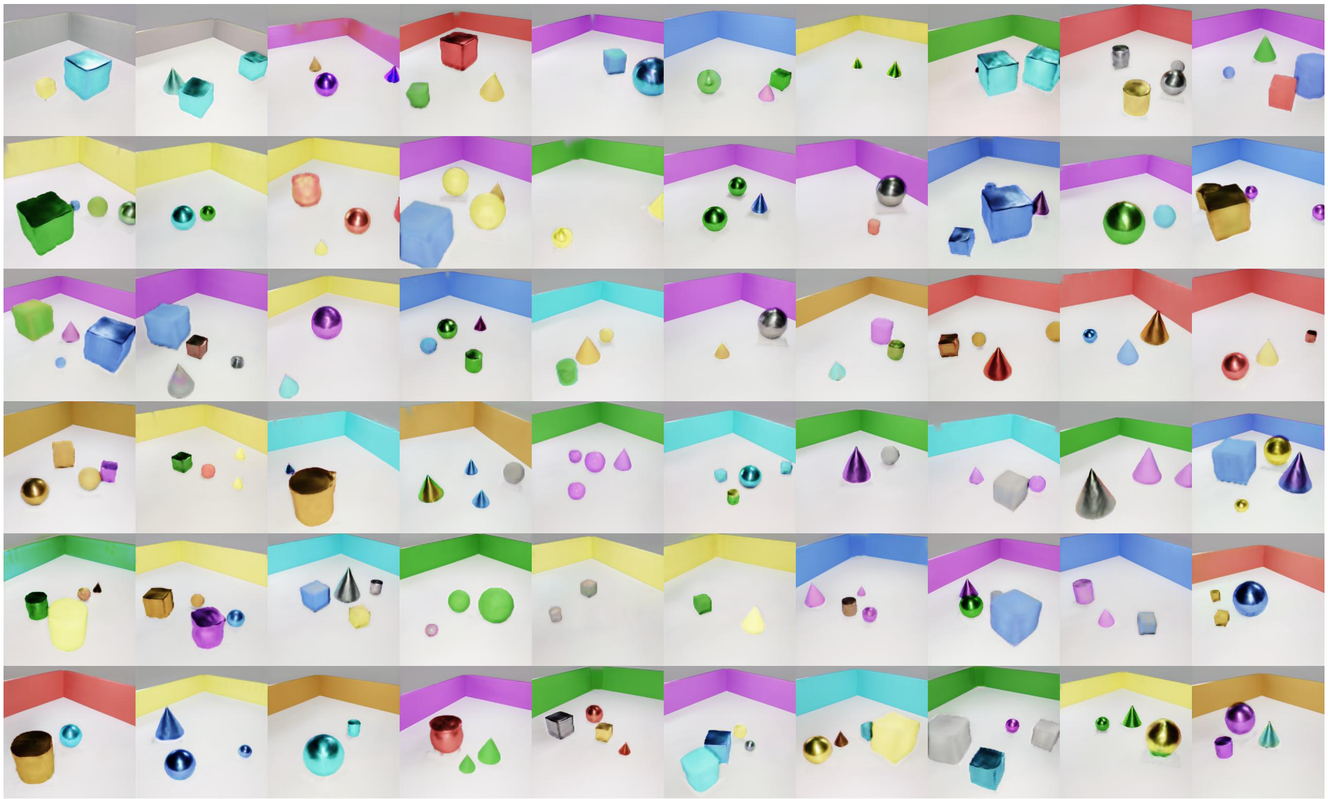


(d) Object editing

Figure 6: **Additional Controllable Image Synthesis Results** on KITTI-360 dataset.



(a) KITTI-360



(b) CLEVR-W

Figure 7: **Uncurated Samples**. We show randomly sampled images of UrbanGIRAFFE.