Supplementary Material for CAMPARI: Camera-Aware Decomposed Generative Neural Radiance Fields

Michael Niemeyer[1,2] Andreas Geiger[1,2]
[1]Max Planck Institute for Intelligent Systems, Tübingen [2]University of Tübingen
{firstname.lastname}@tue.mpg.de

Abstract

*In this **supplementary document**, we first discuss network architectures and baseline implementations in Section 1. Next, we discuss data generation and preprocessing steps in Section 2. Finally, we provide additional qualitative as well as quantitative experimental results in Section 3. In the **supplementary video**, we show animations in which we control the scene during image synthesis.*

1. Implementation

In this section, we discuss the network architectures of our 3D-aware image generator (Section 1.1), our camera generator (Section 1.2), and our discriminator (Section 1.3) as well as baseline implementations (Section 1.4).

1.1. Generator Architecture

We implement our fore- and background models as fully-connected multi-layer perceptrons (MLPs) which map an input point $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{S}^2$ together with latent shape and appearance codes $\mathbf{z}_s, \mathbf{z}_n \in \mathbb{R}^{L_z}$ to a density

Type	Layer	Activation	Input Dim	Output Dim.
Position Input	Linear	ReLU	$L_{\mathbf{x}} + L_z = 191$	128
Hidden Layer	Linear	ReLU	128	128
Hidden Layer	Linear	ReLU	128	128
Hidden Layer	Linear	ReLU	128	128
Hidden Layer	Linear	ReLU	$L_x + L_z + 128 = 319$	128
Hidden Layer	Linear	ReLU	128	128
Hidden Layer	Linear	ReLU	128	128
Hidden Layer	Linear	ReLU	128	128
Hidden Layer	Linear	ReLU	128	128
Density Head	Linear	ReLU	128	1
View Input	Linear	ReLU	$L_d + L_z = 155$	128
Color Head	Linear	Sigmoid	128	3

Table 1: **Image Generator Architecture.** We show the network architecture of our foreground and background model of our 3D-aware image generator, where $L_x = 2 \cdot 3 \cdot 10 + 3 = 63$ and $L_d = 2 \cdot 3 \cdot 4 + 3 = 27$ are the output dimensions of the positional encodings for the input point $\mathbf{x}$ and viewing direction $\mathbf{d}$, respectively, and $L_z = 128$ the dimension of the latent code. Note that we add the output of the last hidden layer onto the output of the view input layer before ReLU activation to enable weight sharing between the view-independent and the view-dependent branch.

Type	Layer	Activation	Input Dim	Output Dim.
Input Mapping	Linear	ReLU	5	64
Hidden Layer	Linear	ReLU	64	64
Hidden Layer	Linear	ReLU	64	64
Hidden Layer	Linear	ReLU	64	64
Hidden Layer	Linear	ReLU	64	64
Output Mapping	Linear	-	64	5

Table 2: **Camera Generator Architecture.** We show the network architecture of our camera generator for 180° rotation scenes where the 5-dimensional camera $\boldsymbol{\xi}$ consists of two focal lengths, a rotation angle, an elevation angle, and a camera radius (see Sec. 3.2 of the main paper). For 360° rotation scenes, we set the input and output dimension to 8 as we regress a 2×2 matrix instead of a scalar for the rotation, avoiding periodic boundary issues and ensuring well-behaved gradients (see Sec. 3.2 of the main paper).

$\sigma \in \mathbb{R}^3$ and a color value $\mathbf{c} \in \mathbb{R}^3$ (see Eq. 4 of the main paper). As indicated in the main paper, we apply a predefined positional encoding [8, 13] to the input and viewing direction before passing it to the MLP. More specifically, we apply

$$\gamma : \mathbb{R} \times \mathbb{N} \to \mathbb{R}^{L_\gamma} \qquad (t, L) \mapsto \big(t, \sin(2^0 t\pi), \cos(2^0 t\pi), \dots, \sin(2^L t\pi), \cos(2^L t\pi)\big) \tag{1}$$

elementwise to $\mathbf{x}$ and $\mathbf{d}$, where t is the scalar input and L the number of frequency octaves.

We use 8 hidden layers of dimension 128 and with ReLU activation for our fore- and background model (see Tab. 1) and add a skip connection to the fourth layer [10]. Similar to previous works [8, 12], we use 10 and 4 frequency octaves for the positional encoding applied to input point $\mathbf{x}$ and viewing direction $\mathbf{d}$, respectively. We project the hidden features to the first output, a one-dimensional density value, and use a single hidden layer with ReLU activation for the view-dependent branch (see Tab. 1) which we then project to the second output, a three-dimensional color value. To enable shape and appearance disentanglement, we concatenate the latent shape code to the input point and the latent appearance code to the viewing direction following [12]. We sample the latent codes from a 128-dimensional Gaussian distribution.

1.2. Camera Generator

Implementation: We implement the camera generator as a fully-connected MLP with ReLU activation (see Tab. 2). We use 4 hidden layers of dimension 64. As discussed in Sec. 3.2 of the main paper, we add a skip connection from the input pose $\boldsymbol{\xi}^{\text{prior}} \sim p_\xi$ to the network output. We clamp the final output to be within a specified range. More specifically, we allow the focal lengths to be such that the resulting fields of view (FoVs) are in $[10^\circ, 60^\circ]$. We define the camera radius to be within $[0.5, 1]$, the camera elevation in $[-90^\circ, 90^\circ]$, and the camera rotation in $[-90^\circ, 90^\circ]$ for 180° scenes (CelebA and Cats), and in $[-180^\circ, 180^\circ]$ for full rotation scenes (Cars, Chairs1, and Chairs2). We set the radius of the foreground sphere r_{fg} to 0.5.

Prior Distributions: As indicated in the main publication, we define the prior distribution as Gaussian or uniform distributions. More specifically, we use the Gaussian distribution $\mathcal{N}(0, 13.5)$ for rotation and elevation on 180° rotation scenes (all values in degrees), and a uniform distribution over the entire rotation and elevation range on 360° rotation scenes. For the radius and focal lengths, we always use the Gaussian distributions $\mathcal{N}(0.75, 0.0375)$ and $\mathcal{N}(35, 3.75)$ as prior distributions, respectively.

1.3. Discriminator Architecture

We implement the discriminator D_ϕ using ResNet [4] blocks of CoordConv [6] layers (see Tab. 3) similar to [2]. More specifically, we map the RGB input image to a hidden dimension of 64 using a single 1×1 CoordConv layer and leaky ReLU activation. We then apply ResNet blocks consisting of two 3×3 CoordConv layers with leaky ReLU activation and an average pooling layer. We add a residual connection from input to output. Finally, we project the hidden features to the one-dimensional output using a 4×4 CoordConv layer.

As indicated in Sec 3.3 of the main publication, we train using progressive growing. For the earlier progressive growing stages, we add input mappings (see Tab. 3) from the lower-resolution input images to the respective ResNet blocks. When reaching a new progressive growing stage, we fade in newly added layers over 15,000 iterations following [2, 5].

Type	Layer	Kernel Size	Stride	Padding	Activation	Feature Out. Dim.	Spatial Out. Dim.
Input Mapping	CoordConv	1×1	1	0	LReLU	64	128×128
ResNet Block	CoordConv	3×3	1	1	LReLU	128	128×128
	CoordConv	3×3	1	1	LReLU	128	128×128
	AvgPool	-	-	-	-	128	64×64
ResNet Block	CoordConv	3×3	1	1	LReLU	256	64×64
	CoordConv	3×3	1	1	LReLU	256	64×64
	AvgPool	-	-	-	-	256	32×32
ResNet Block	CoordConv	3×3	1	1	LReLU	400	32×32
	CoordConv	3×3	1	1	LReLU	400	32×32
	AvgPool	-	-	-	-	400	16×16
ResNet Block	CoordConv	3×3	1	1	LReLU	400	16×16
	CoordConv	3×3	1	1	LReLU	400	16×16
	AvgPool	-	-	-	-	400	8×8
ResNet Block	CoordConv	3×3	1	1	LReLU	400	8×8
	CoordConv	3×3	1	1	LReLU	400	8×8
	AvgPool	-	-	-	-	400	4×4
Output Mapping	CoordConv	4×4	1	0	-	1	1×1

Table 3: **Discriminator Architecture.** We show the network architecture of our discriminator D_ϕ at 128^2 pixels. Note that in each ResNet block, we add a skip connection from the input to the output. For earlier stages of progressive growing, we have additional input mappings from the image of a smaller resolution to the respective ResNet block.

1.4. Baseline Implementations

We use the official implementations for the baselines HoloGAN [9] and GRAF [12].[1] In contrast to our method, the baselines require tuning of camera parameters, and we therefore report results for them in tuned and non-tuned settings (see Sec. 4 of the main paper). More specifically, we use the camera ranges reported by the authors for the tuned setting. For the non-tuned setting, we use the ranges of our prior distribution (see Section 1.2). As the baselines assume a uniform distribution, we use $\pm$ two standard deviations when converting from a Gaussian to a uniform range. Further, while our method also learns the camera's focal lengths, both baselines use fixed intrinsics. To account for this, we increase their camera radius range accordingly. More specifically, we sample the radius in $[7.5, 12.5]$ for GRAF and $[0.8, 1.5]$ for HoloGAN.

2. Data

We describe the generation of our Chairs datasets in Section 2.1 and report any data preprocessing steps in Section 2.2.

2.1. Dataset Generation

Next to the commonly-used CelebA [7], Cats [14], and Cars [12] datasets, we create two additional datasets to test our model on more complex camera rotation and elevation distributions (see Sec. 4 of the main paper). We create photorealistic renderings of the Photoshape chairs [11] using Blender [3]. More specifically, we predefine mixtures of Gaussians for the rotation and elevation distributions and render 4 images per chair, resulting in 21,808 images. In Fig. 1 and 2, we show random samples of the two datasets.

For Chairs1, we use a mixture of $\mathcal{N}(-10, 5)$ and $\mathcal{N}(40, 20)$ for the rotation and $\mathcal{N}(55, 5)$ and $\mathcal{N}(-20, 10)$ for the elevation distribution (all values in degrees), where the mixture weights are 0.5. For Chairs2, we use a mixture of $\mathcal{N}(-140, 5)$, $\mathcal{N}(-80, 10)$, $\mathcal{N}(10, 5)$, $\mathcal{N}(50, 5)$, and $\mathcal{N}(100, 5)$ for the rotation distribution with equal mixture weights (0.2). For the elevation, we use a mixture of $\mathcal{N}(70, 5)$, $\mathcal{N}(15, 5)$, $\mathcal{N}(-30, 5)$, and $\mathcal{N}(-70, 5)$ with equal mixture weights (0.25). In Fig. 3, we visualize the camera rotation and elevation distributions for the two datasets.

[1] HoloGAN: https://github.com/thunguyenphuoc/HoloGAN, GRAF: https://github.com/autonomousvision/graf

Figure 1: **Random GT Samples from Chairs1 Dataset.**

2.2. Data Preprocessing

In contrast to previous works on 3D-aware image synthesis [9, 12], we use a center crop of the entire image for CelebA instead of a close-up region (see Sec. 4 of the main paper). Learning a 3D-consistent representation becomes more challenging as the data variety becomes larger. Further, the models ideally disentangle fore- from background. In Fig. 4, we show random samples of the processed CelebA dataset.

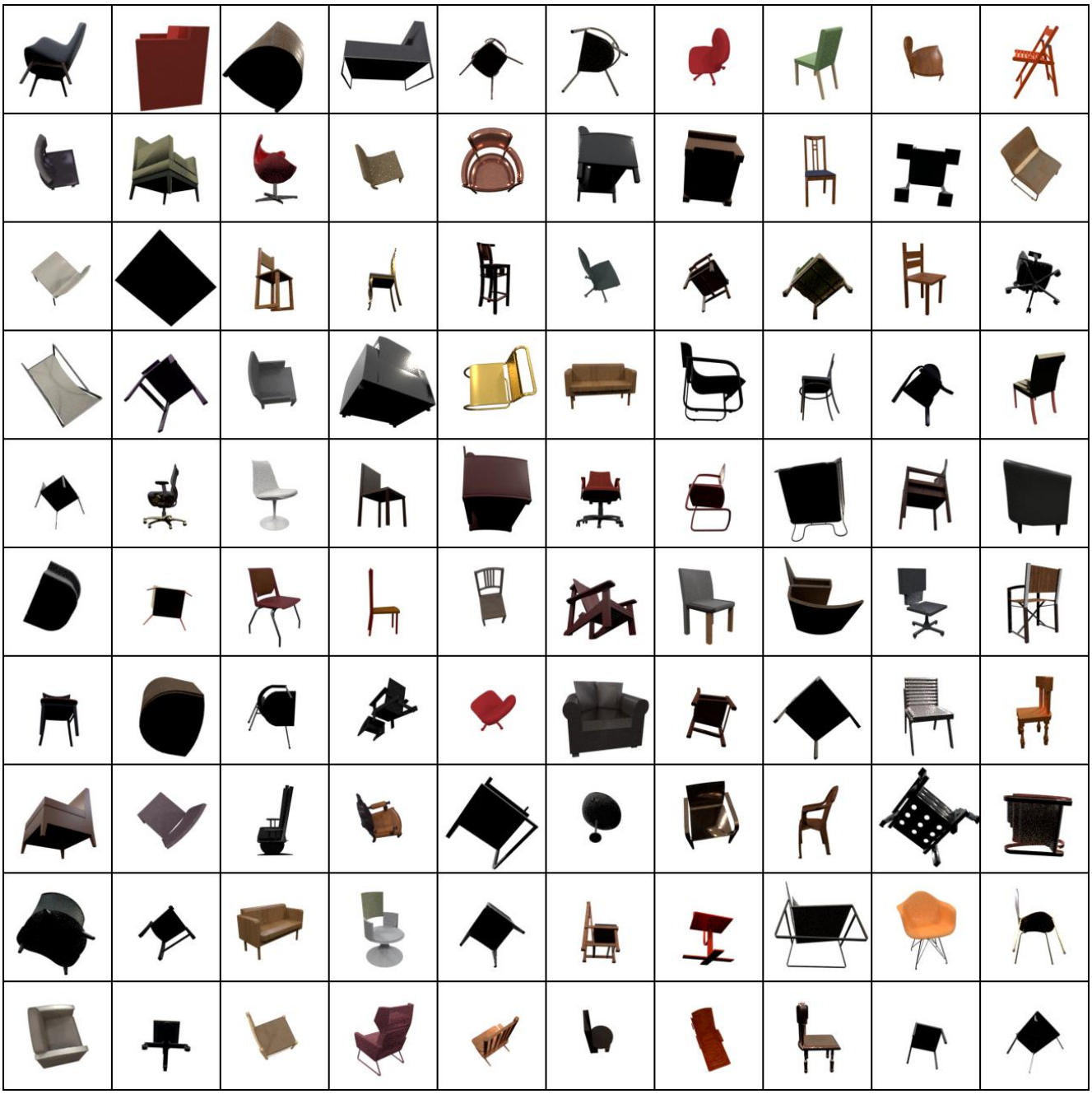
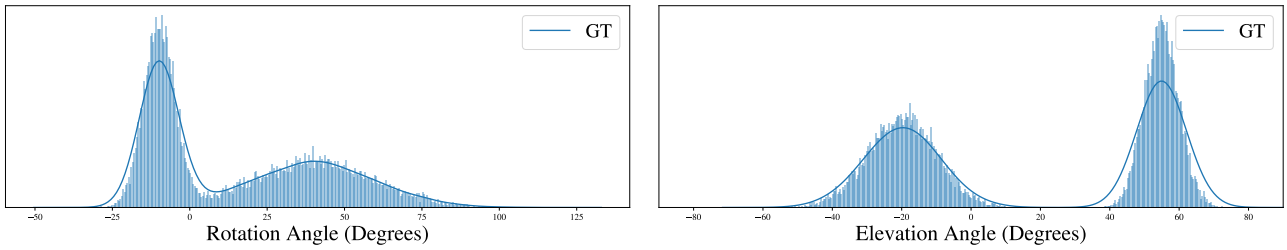


Figure 2: **Random GT Samples from Chairs2 Dataset.**

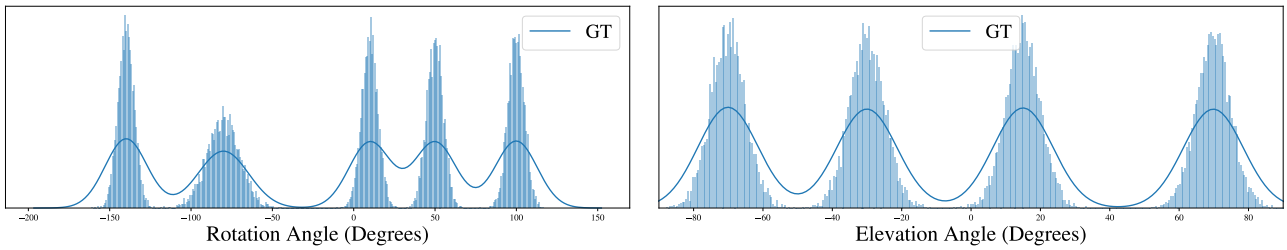
3. Additional Experimental Results

3.1. Baseline Comparison

In Tab. 4 and Fig. 5, we report additional quantitative and qualitative comparisons to baselines. Although our method does not require any tuning, we achieve similar or better KID scores [1] compared to baselines if they are tuned. Further, the baselines' results degrade if the data's camera distribution is not matched. In contrast, we learn a camera generator jointly with the image generator leading to more 3D consistent results (see Fig. 5) although no tuning is required.



(a) Camera Rotation and Elevation Distribution for Chairs1



(b) Camera Rotation and Elevation Distribution for Chairs2

Figure 3: **Camera Distributions of Chairs Datasets.** We visualize the camera rotation and elevation distributions for the created datasets Chairs1 and Chairs2.

	Tuned?	Cats	CelebA	Carla	Chairs1	Chairs2
HGAN [9]	yes	2.32	5.42	11.55	-	-
	no	3.41	7.89	12.78	9.02	8.65
GRAF [12]	yes	1.46	2.48	1.48	-	-
	no	5.83	6.12	4.95	3.40	3.48
Ours	no	1.74	2.19	2.42	1.57	1.75

Table 4: **Quantitative Comparison.** We report KID$\times$100 ($\downarrow$) for our method and baselines at 128^2 pixels.

3.2. Controllable Image Synthesis

We show additional examples for controllable image synthesis on Cats in Fig. 6, on CelebA in Fig. 7 and on Cars in Fig. 8. Further, we show random samples from our learned camera distribution for Chairs1 in Fig. 9 and for Chairs2 in Fig. 10. Next, we show visualizations of the learned depth for CelebA in Fig. 11 and for Chairs2 in Fig. 12. Additional examples for fore- and background disentanglement on CelebA are shown in Fig. 13. In Fig. 14 we show outputs of our model when fixing the prior instrinsics distribution and in Fig. **??** we show changes in focal length at test time. Finally, in Fig. 16 we show qualitative results for π-GAN [2] with tuned and non-tuned cameras as well as in combination with our camera generator.

3.3. Random Samples

We show random samples for our method on CelebA in Fig. 17, on Cats in Fig. 18, on Cars in Fig. 19, on Chairs1 in Fig. 20, and on Chairs2 in Fig. 21.

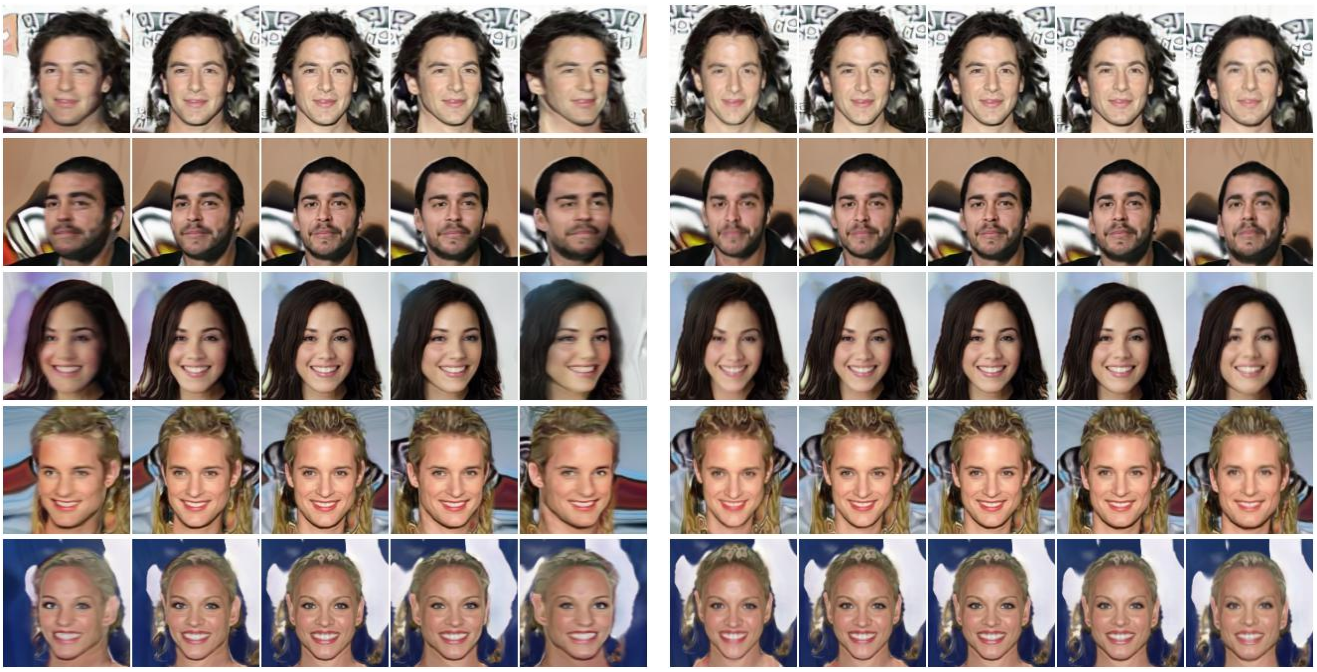
Figure 4: **Random GT Samples for the CelebA Dataset.** In contrast to previous works on 3D-aware image synthesis [9,12], we use a center crop of the entire image for CelebA instead of a close up region (see Sec. 4 of the main paper) resulting in more challenging and more varied data.

(a) GRAF [12] (w./ Camera Tuning) (b) GRAF [12] (wo./ Camera Tuning) (c) Ours (No Tuning Required)

Figure 5: **Qualitative Comparison.** We show camera rotation examples for GRAF [12] with and without camera parameter tuning and our method for CelebA at 128×128 pixels. State-of-the-art 3D-aware synthesis models typically require hand-tuning of camera parameters (Fig. 5a), and results degrade if the camera distribution does not match the data distribution (Fig. 5b). In contrast, we learn a camera generator jointly with the image generator leading to more consistent 3D representations (Fig. 5c) without the need for any tuning.

(a) Rotation (b) Elevation

Figure 6: **Controllable Image Synthesis on Cats.** We show examples in which we control the rotation or elevation angle of the camera for Cats at 128×128 pixels.



(a) Rotation (b) Elevation

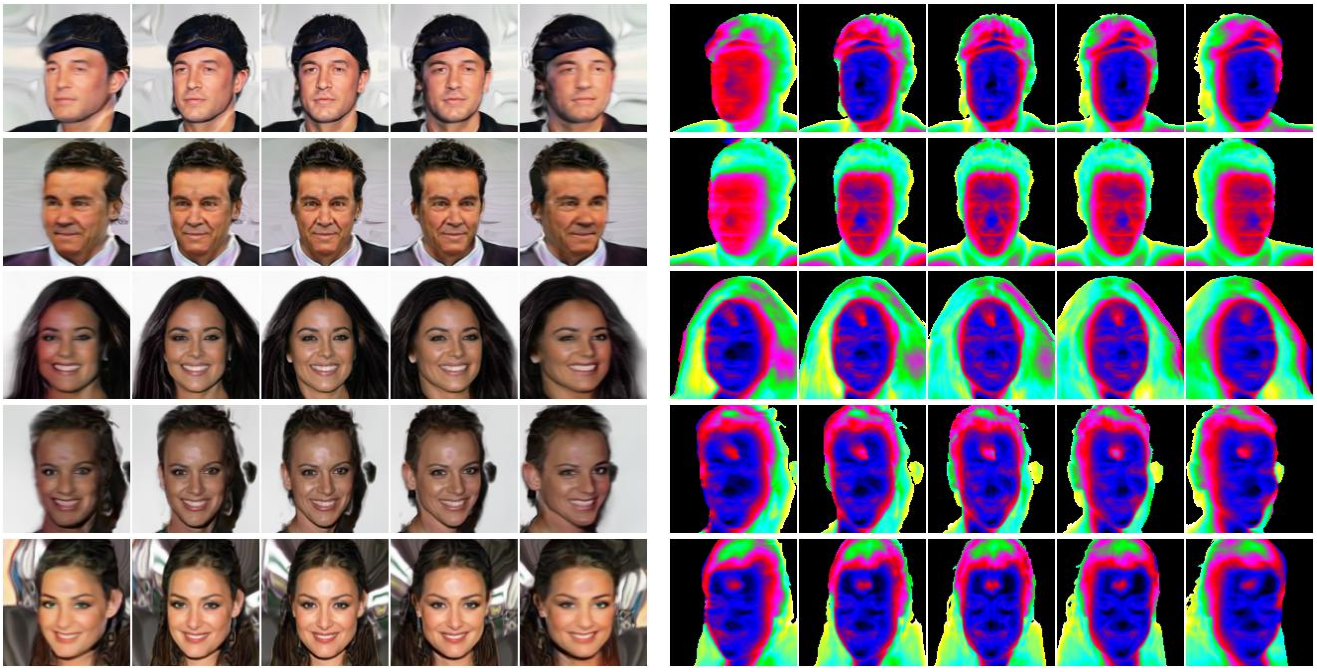
Figure 7: **Controllable Image Synthesis on CelebA.** We show examples in which we control the rotation or elevation angle of the camera for CelebA at 128×128 pixels.

(a) Rotation (b) Elevation

Figure 8: **Controllable Image Synthesis on Cars.** We show examples in which we control the rotation or elevation angle of the camera for Cars at 128×128 pixels.

Figure 9: **3D- and Camera-Aware Image Synthesis on Chairs1.** We show examples in which we sample camera poses from our learned distribution for Chairs1 at 128×128 pixels.

Figure 10: **3D- and Camera-Aware Image Synthesis on Chairs2.** We show examples in which we sample camera poses from our learned distribution for Chairs2 at 128×128 pixels.



(a) RGB Color (b) Depth

Figure 11: **Foreground Depth Visualization on CelebA.** As we disentanglement fore- from background, we can render individual components. We visualize the foreground depth for CelebA at 128×128 pixels. Note that as we train from raw image collections, fore- and background disentanglement as well as 3D consistency are learned completely unsupervised.

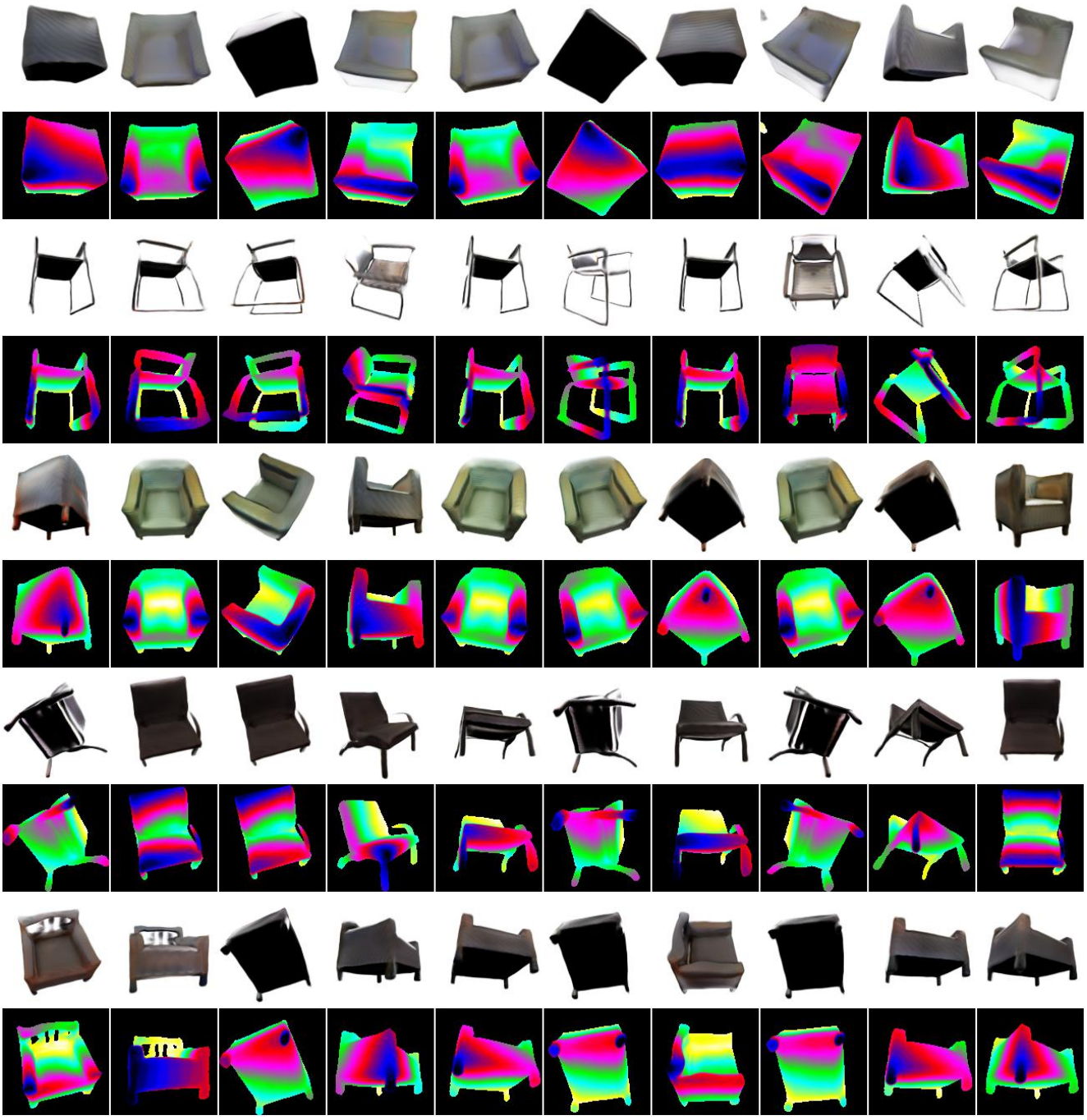


Figure 12: **Depth Visualization on Chairs2.** We show RGB color (top) and depth (bottom) for samples from our learned camera distributions for Chairs2 at 128×128 pixels. Note that as we train from raw image collections, no depth information has been used during training.

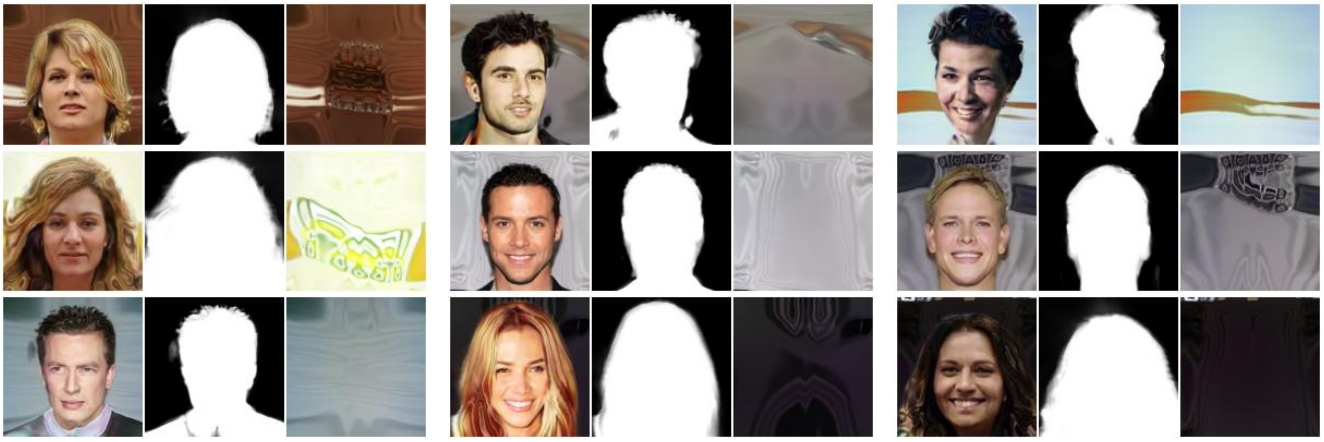


Figure 13: **Fore- and Background Disentanglement on CelebA.** We show the generated image, foreground alpha mask, and generated background to visualize fore- and background disentanglement for CelebA at 128×128 pixels.

Figure 14: **Importance of Learning Instrinsics.** We show rotations for our model without learning instrinsics on CelebA and Carla (i.e. fixing the instrinsics distribution to the prior). Similar to the quantitative results, they are qualitatively degraded and less consistent.

(a) Change in x direction (b) Change in y direction (c) Change in both direction

Figure 15: **Change of Focal Length.** We change the focal length in x, y, and both directions.

(a) π-GAN (not tuned) (b) π-GAN + Ours (not tuned) (c) π-GAN (tuned)

Figure 16: **π-GAN as Image Generator.** We show random samples for π-GAN [2] with and without tuned camera poses, and with our camera generator. Similar to the quantitative results (Tab. 3 of main publication), π-GAN's performance drops for non-tuned cameras, but using our camera generator leads to similar results to π-GAN (tuned) while no tuning is required.

Figure 17: **Random Samples for CelebA.**

Figure 18: **Random Samples for Cats.**

Figure 19: **Random Samples for Cars.**

Figure 20: **Random Samples for Chairs1.**

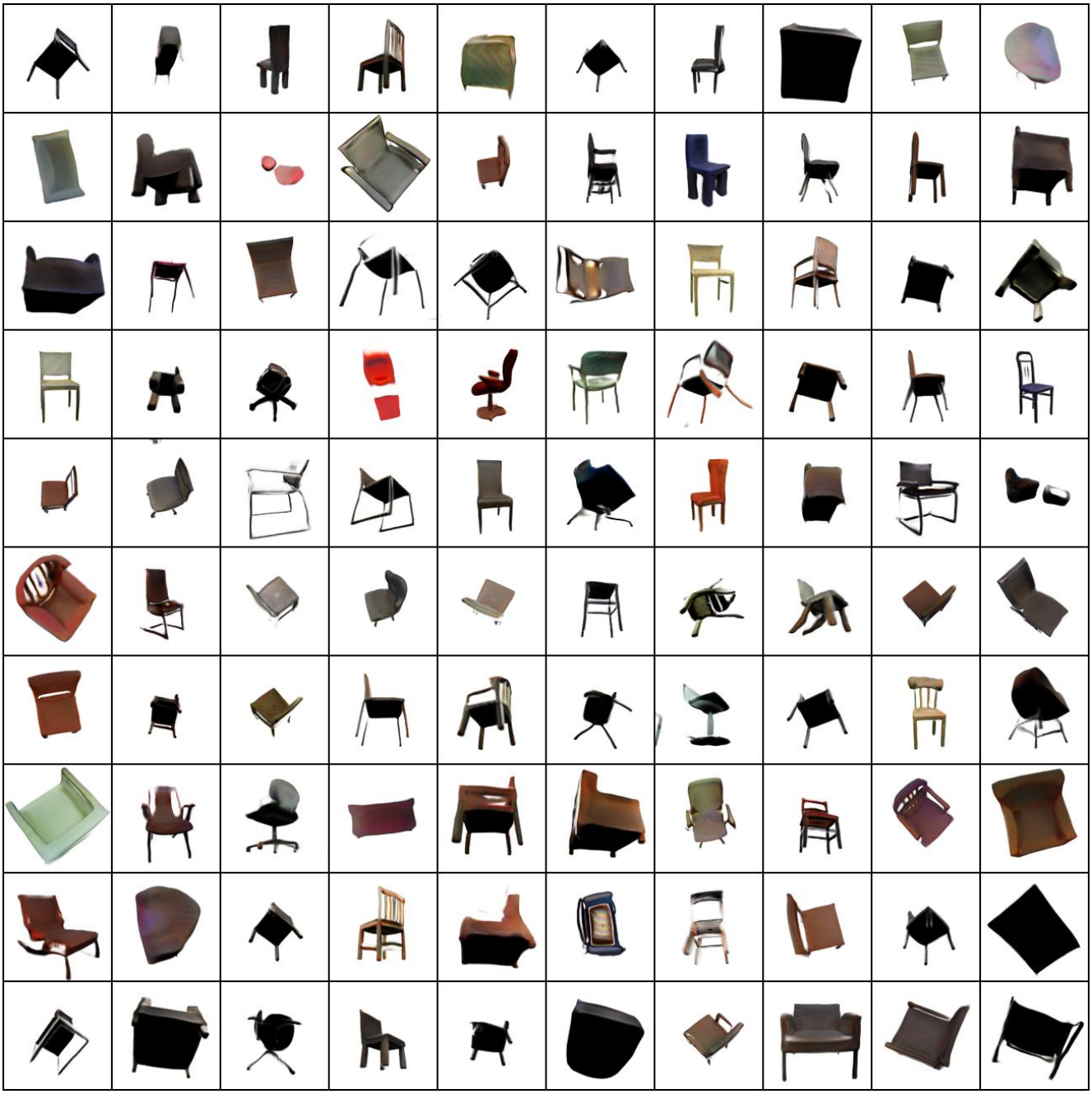


Figure 21: **Random Samples for Chairs2.**

References

[1] Mikolaj Binkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD gans. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 5

[2] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *arXiv.org*, 2012.00926, 2020. 2, 6, 13

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 3

[4] 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

[5] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 2

[6] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2

[7] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015. 3

[8] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2

[9] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 3, 4, 6, 7

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[11] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *Communications of the ACM*, 2018. 3

[12] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3, 4, 6, 7, 8

[13] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[14] Li Zhang, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multi-view stereo. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2003. 3