# Supplementary Material for GIRAFFE: Representing Scenes as Compositional Generative Neural Feature 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, implementation details, and the training protocol in Sec. 1. Next, we describe data preprocessing and augmentation strategies in Sec. 2. Finally, we provide additional qualitative as well as quantitative experimental results in Sec. 3.*

## 1. Implementation

In this section, we first discuss the network architectures of our feature fields and discriminator (Sec. 1.1). Next, we provide details regarding the evaluation (Sec. 1.2) and training protocol (Sec. 1.3). Finally, we describe relevant volume rendering techniques (Sec. 1.4) and baseline implementations (Sec. 1.5).

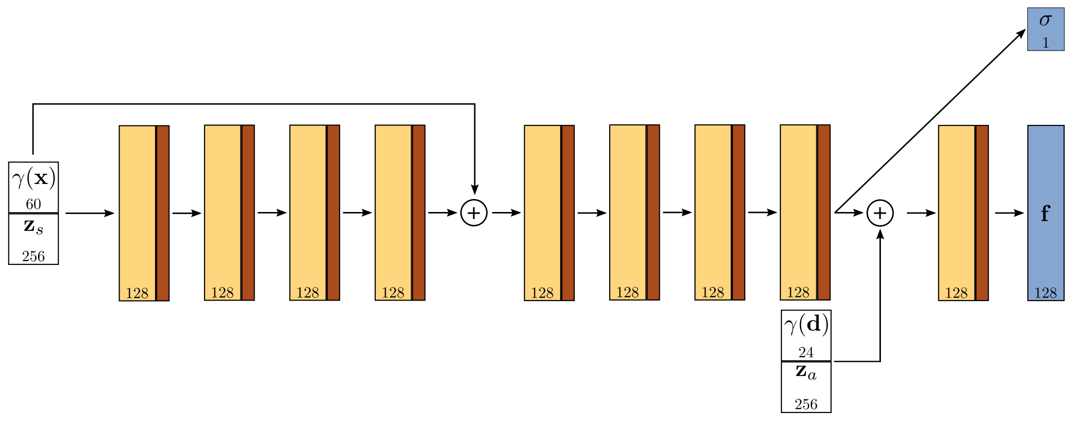
### 1.1. Network Architectures

**Feature Field Architecture:** We parameterize the object and background feature fields with 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 $\mathbf{z}_s, \mathbf{z}_a \sim \mathcal{N}(\mathbf{0}, I)$ to a one-dimensional density $\sigma$ and a $M_f$-dimensional feature vector $\mathbf{f}$ (see Fig. 1). As indicated in the main paper, we apply positional encoding to $\mathbf{x}$ and $\mathbf{d}$ before passing it to the network. Similar to previous works [16, 23], we use 10 and 4 frequency octaves for $\mathbf{x}$ and $\mathbf{d}$, respectively, such that we embed them in a $L_{\mathbf{x}} = 2 \cdot 3 \cdot 10 = 60$ and $L_{\mathbf{d}} = 2 \cdot 3 \cdot 4 = 24$ dimensional space. For the object feature field, we use 8 fully-connected layers with a hidden dimension of 128 and ReLU activation. We add a skip connection from the input to the fourth layer which was shown to improve results in the context of 3D reconstruction [19]. For the background feature field, we use 4 layers with a hidden dimension of 64 and ReLU activation due to its lower complexity. We half the hidden dimensions for experiments on scenes with three objects or more to facilitate training. In both networks, we project the features to the first output, the one-dimensional density $\sigma$, and use a single fully-connected layer with ReLU activation for the view-dependent feature prediction branch similar to previous works [16, 23]. The features are then projected to the second output, a $M_f$-dimensional feature vector $\mathbf{f}$. This way, we ensure that the density prediction only depends on the input point $\mathbf{x}$ and the latent shape code $\mathbf{z}_s$ as the geometry of objects is independent of the viewpoint. Further, we use only a single view-dependent layer to enable weight sharing between the two branches. We sample the latent codes $\mathbf{z}_s, \mathbf{z}_a$ from a 256-dimensional Gaussian for the object feature field, and from a 128-dimensional Gaussian for the background feature field. We reduce the dimensionality to 64 and 32, respectively, for experiments on scenes with three objects or more to facilitate training.

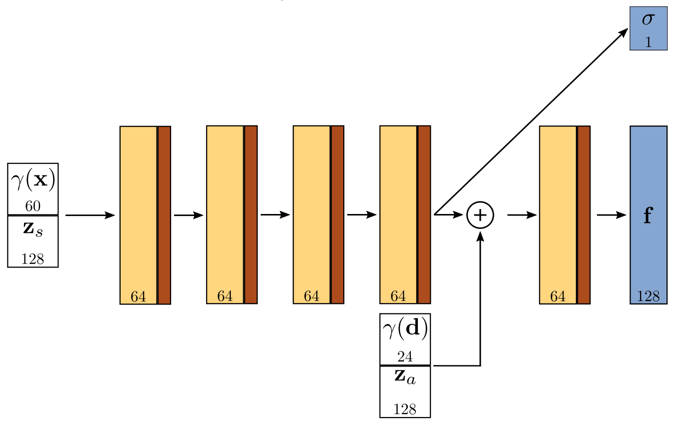
**Discriminator Architecture:** We parameterize our discriminator $D_\phi$ as a fully-convolutional neural network with leaky ReLU activations [22]. We use 5 and 7 layers for resolutions of $64^2$ and $256^2$ pixels, respectively (see Tab. 2).

### 1.2. Evaluation Metric

To quantify image quality, we report the Frechet Inception Distance (FID) score [6]. We use 20,000 real and fake samples to calculate the FID score. To be compatible with [23], we adhere to the authors' evaluation and use their evaluation script for reported experiments on datasets *Chairs*, *Cats*, *CelebA*, and *CelebA-HQ*.



(a) Object Feature Field.



(b) Background Feature Field.

Figure 1: **Feature Fields Architecture.** We parameterize the object (Fig. 1a) and background (Fig. 1b) feature fields with multi-layer perceptrons (MLPs) which take as input a 3D point $\mathbf{x}$ and viewing direction $\mathbf{d}$ together with latent shape and appearance codes $\mathbf{z}_s, \mathbf{z}_a$ and output a density $\sigma$ and feature $\mathbf{f}$. More specifically, we apply the positional encoding $\gamma$ to input point $\mathbf{x}$ and concatenate $\gamma(\mathbf{x})$ and the latent shape code $\mathbf{z}_s$. This is followed by blocks of fully-connected layers (yellow color) with ReLU activation (red color). We use 8 blocks with a hidden dimension of 128 and one skip connection to the fourth layer for the object feature field, and 4 blocks with a hidden dimension of 64 for the background feature field. We then project this to the first output, the one-dimensional density output $\sigma$ (blue). In a second branch, we apply the positional encoding $\gamma$ to the viewing direction $\mathbf{d}$, concatenate $\gamma(\mathbf{d})$ to the latent appearance code $\mathbf{z}_a$, and add it to the previous hidden features. We pass it through a single fully-connected layer with ReLU activation and project it to the second output, the $M_f$-dimensional feature output $\mathbf{f}$ (blue). We set $M_f$ to 128 and 256 for $64^2$ and $256^2$ pixels, respectively.

## 1.3. Training Protocol

We train with the RMSprop optimizer [24] and use a batch size of 32 and learning rates of $1 \times 10^{-4}$ and $5 \times 10^{-4}$ for the discriminator and generator, respectively. For experiments at $256^2$ pixels, we reduce the generator learning rate to $2.5 \times 10^{-4}$. We perform single-GPU training and train our models for one to four days. To determine when to stop training, we follow common practice and evaluate the FID metric every 10,000 iterations.

## 1.4. Volume Rendering

**3D Point Sampling:** We follow [16] and use stratified sampling to approximate the intractable volumetric projection integral. More specifically, we partition each ray into $N_s = 64$ evenly-spaced bins and sample uniformly one sample point within each bin. Let $\mathbf{r}(t) = \mathbf{r}_0 + t \cdot \mathbf{d}$ denote the ray starting from $\mathbf{r}_0$ with direction $\mathbf{d}$, and let $t_n, t_f$ be the near and far plane of the

| Layer Type | Kernel Size | Stride | Padding | Activation | Feature Dimension | Spatial Output Dimensions |
|---|---|---|---|---|---|---|
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 64 | $32 \times 32$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 128 | $16 \times 16$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 256 | $8 \times 8$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 512 | $4 \times 4$ |
| Conv | $4 \times 4$ | 1 | 0 | - | 1 | $1 \times 1$ |

(a) $64^2$ Pixel Resolution.

| Layer Type | Kernel Size | Stride | Padding | Activation | Feature Dimension | Spatial Output Dimensions |
|---|---|---|---|---|---|---|
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 16 | $128 \times 128$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 32 | $64 \times 64$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 64 | $32 \times 32$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 128 | $16 \times 16$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 256 | $8 \times 8$ |
| Conv | $4 \times 4$ | 2 | 1 | LReLU | 512 | $4 \times 4$ |
| Conv | $4 \times 4$ | 1 | 0 | - | 1 | $1 \times 1$ |

(b) $256^2$ Pixel Resolution.

Figure 2: **Discriminator Architecture.** Our discriminator consists of fully-convolutional layers with $4 \times 4$ kernels and leaky ReLU activation functions. We use 5 and 7 layers for $64^2$ and $256^2$ pixel resolutions, respectively.

camera. The $i$-the sample point is then defined as

$$\mathbf{x}_i = \mathbf{r}(t_i) \quad \text{where} \quad t_i \sim \mathcal{U}\left(t_n + \frac{i-1}{N_s}(t_f - t_n), t_n + \frac{i}{N_s}(t_f - t_n)\right) \tag{1}$$

**Camera:** In all experiments, we define the camera to be on a sphere with radius 2.732 and use $t_n = 0.5$ and $t_f = 6$ as near and far planes, respectively. Note that, as we learn from unposed and unstructured image collections, the camera radius is arbitrary. We adopt this setting from popular rendering scripts [11] as with this camera, the scene bounds are roughly described by the unit cube when using common fields of view.

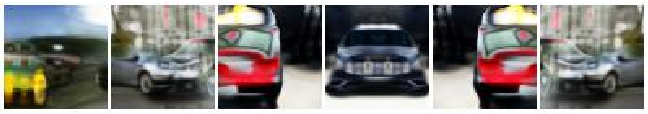
## 1.5. Baselines

**2D GAN:** For reference, We report results for a ResNet-based [4] 2D GAN [15] in our experiments. We adopt the training protocol and network architectures from [15]. We use the RMSprop optimizer with generator and discriminator learning rates of $1 \times 10^{-4}$. We train with the non-saturating GAN objective [2] and $R_1$ gradient penalty [15]. Similar to [23], we use minimum and maximum feature dimensions of 16 and 512, respectively, such that the generator has in total 1,689,267 parameters (our generator has 411,595 parameters).

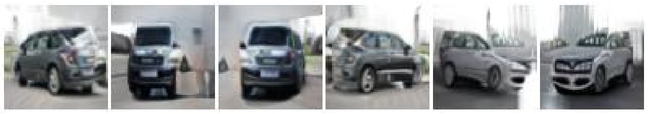
**HoloGAN:** We use the official HoloGAN [17] implementation[1] and follow their training protocol: We train with the Adam optimizer [12] with an initial learning rate of $1 \times 10^{-4}$ for experiments at $64^2$ pixels, and $5 \times 10^{-5}$ for higher resolutions. We train for 50 epochs, and linearly reduce the learning rate after the first 25 epochs. We further use the identity regularizer and style discriminator loss for an image resolution of $256^2$ pixels. If provided, we use the pose ranges from the authors, otherwise we use the same as for our method. For the non-object-centered datasets *CompCars* and *Churches*, we follow [18] and add random translation offsets during training. While the official implementation provides generator architectures for $64^2$ and $128^2$ pixel resolutions, we obtain the generator for $256^2$ pixels by adding one transposed convolutional layer with adaptive instance normalization [7] and leakly ReLU activation to the $128^2$ generator, and, similar to the previous layers, and apply the style discriminator loss to this intermediate output as well.

**PlatonicGAN:** We use the official PlatonicGAN [5] implementation provided by the authors.[2] We follow their training protocol and train with the Wasserstein loss and gradient penalty [3], and use a batch size of 16. We follow [23] and use

[1] https://github.com/thunguyenphuoc/HoloGAN
[2] https://github.com/henzler/platonicgan



(a) GRAF [23] on *CompCars* with Random Cropping.



(b) GRAF [23] on *CompCars* with Random Cropping.

Figure 3: **GRAF on CompCars.** In our experiments on *CompCars*, we apply random cropping to achieve more variety in the data wrt. the car positions within the scene. However, we find that GRAF [23] does not lead to consistent 3D representations when random cropping is applied (see Fig. 3a), such that we report results for GRAF without random cropping (see Fig. 3b). Note that also quantitatively results are improved from an FID score of 116 to 39 at $64^2$ pixel resolution.

| Name | Type | Number of Images | Object Rotation Range | Background Rotation Range | Camera Elevation Range | Horizontal Translation | Depth Translation | Object Scale | Field of View |
|---|---|---|---|---|---|---|---|---|---|
| Chairs [20] | Synth. | 152,680 | 360° | 0° | 90° | - | - | 1 | 49° |
| Cats [28] | Real | 9407 | 70° | 0° | 10° | - | - | 1 | 10° |
| CelebA [14] | Real | 202,599 | 70° | 0° | 10° | - | - | 1 | 10° |
| CompCars [26] | Real | 136,726 | 360° | 0° | 10° | $-0.12 - 0.12$ | $-0.22 - 0.22$ | $0.8 - 1$ | 10° |
| Churches [27] | Real | 126,227 | 360° | 90° | 0° | $-0.15 - 0.15$ | $-0.15 - 0.15$ | $0.8 - 1$ | 30° |
| CelebA-HQ [9] | Real | 30,000 | 90° | 0° | 10° | - | - | 1 | 10° |
| FFHQ [10] | Real | 70,000 | 70° | 0° | 10° | - | - | 1 | 10° |
| Clevr-2 [8] | Synth. | 54,336 | 0° | 0° | 0° | $-0.7 - 0.7$ | $-0.7 - 0.7$ | 1 | 49° |

Table 1: **Dataset Parameters.** We report relevant parameters for all datasets. We use the same dataset-specific parameters for experiments at $64^2$ and $256^2$ pixels.

reduced discriminator and generator learning rates of $5 \times 10^{-6}$ and $7.5 \times 10^{-4}$ to stabilize training. The reconstruction loss is weighted with factor 100 and the convolutional layers in the discriminator and generator have a minimum of 256 features. We use the adopted camera pose samping from [23].

**GRAF:** We use the authors' implementation of GRAF [23]. If reported, we use the authors' pose ranges, otherwise use the same as for our method, except for *Clevr-2* where we use a full rotation as we did not obtain competitive results with the correct rotation angle of 0°.

## 2. Data

In the following paragraphs, we discuss dataset-specific parameters, the data processing and augmentation strategies we use, and the data generation of the *Clevr* datasets.

**Dataset Parameters:** In Tab. 1, we report relevant parameters for all datasets. We use the same parameters for experiments at $64^2$ and $256^2$ pixels. As indicated in the main publication, we sample the object rotation, background rotation, camera elevation, horizontal and depth translation, and object size from uniform distributions over the indicated ranges. For the *Clevr* datasets, we sample object locations from the distribution we obtain during dataset generation (see below) to avoid collisions. We find that adding random background rotations for *Churches* slightly improves the quality of the learned disentanglement while the quantitative results are the same (both achieve an FID score of 17.27 at $64^2$ pixel resolution).

**Image Center Cropping:** To ensure a fair comparison, we follow [17, 23] and center crop the images of the *CelebA* and *CelebA-HQ* datasets to $108^2$ and $650^2$ pixels, respectively.

**Image Random Cropping:** For the *CompCars* dataset, we first rescale the image such that the smaller image dimension is 64 or 256 pixels, depending on the image resolution of the experiment. Next, we obtain our quadratic image by randomly selecting a $64^2$ or $256^2$ cropping window, respectively. This way, we achieve more variety wrt. the position of the cars within the scene. We find that PlatonicGAN [5] and GRAF [23] cannot handle this data augmentation but instead require the object to be centered in the image, such that we do not apply random cropping for these baselines to improve their performance (see Fig. 3).

**Data Augmentation:** For all experiments, we randomly flip images horizontally during training to achieve more variety in the data.

**Clevr Dataset Generation:** As discussed in the main paper, we use the script from [8] to render multi-object scenes of

| | WGAN-GP [3] | LR-GAN [25] | HoloGAN [17] | BlockGAN [18] | Ours |
|---|---|---|---|---|---|
| CompCars | $0.035 \pm 0.001$ | $0.014 \pm 0.001$ | $0.028 \pm 0.002$ | $0.016 \pm 0.001$ | $\mathbf{0.010 \pm 0.001}$ |

Table 2: **Quantitative Comparison to BlockGAN.** As the authors report results for BlockGAN [18] on *CompCars* at $64^2$ pixel resolution, we are able compare our method against their reported results. We follow their protocol and report the KID score (↓) for 10,000 real and fake samples.

random primitives. We adjust the camera position to have a rotation of 0° instead of 43°. We save renderings and positions of placed primitives to files. During training, we sample the translations of object feature fields from the saved positions.

## 3. Additional Experimental Results

In this section, we first provide additional baseline comparisons (Sec. 3.1) and ablation studies (Sec. 3.2). Next, we show more controllable image synthesis examples (Sec. 3.3), failure cases (Sec. 3.4), and random samples (Sec. 3.5).

### 3.1. Baseline Comparisons

**Comparison to Controllable Image Synthesis:** As discussed in the main publication, Controllable Image Synthesis (CIS) [13] is a 3D-aware method which allows for generating multi-object scenes with object-level control. However, as CIS requires additional supervision in the form of labeled (pure) background images, we restrict the comparison to the synthetic *Clevr* dataset and additionally render pure background images. Further, as we could not obtain competitive results for CIS on *Clevr-2345*, we compare CIS and our method on scenes with 0, 1, 2, or 3 primitives (*Clevr-0123*) at $64^2$ pixels which is the same setting as the authors consider.

Quantitatively, our method achieves a higher FID score than CIS (13 to 83). While both methods allow for controllable image synthesis, our method leads to more consistent results (see Fig. 4). Note that while CIS learns object-level locations, we sample them from the data distribution. In contrast to CIS which requires supervision in the form of labeled background images, object disentanglement emerges without any supervision for our method.

**Comparison to BlockGAN:** In Tab. 2, we show the KID results reported in [18] and ours.[3] We adhere to their evaluation protocol and sample 10,000 real and fake images to calculate the KID score.

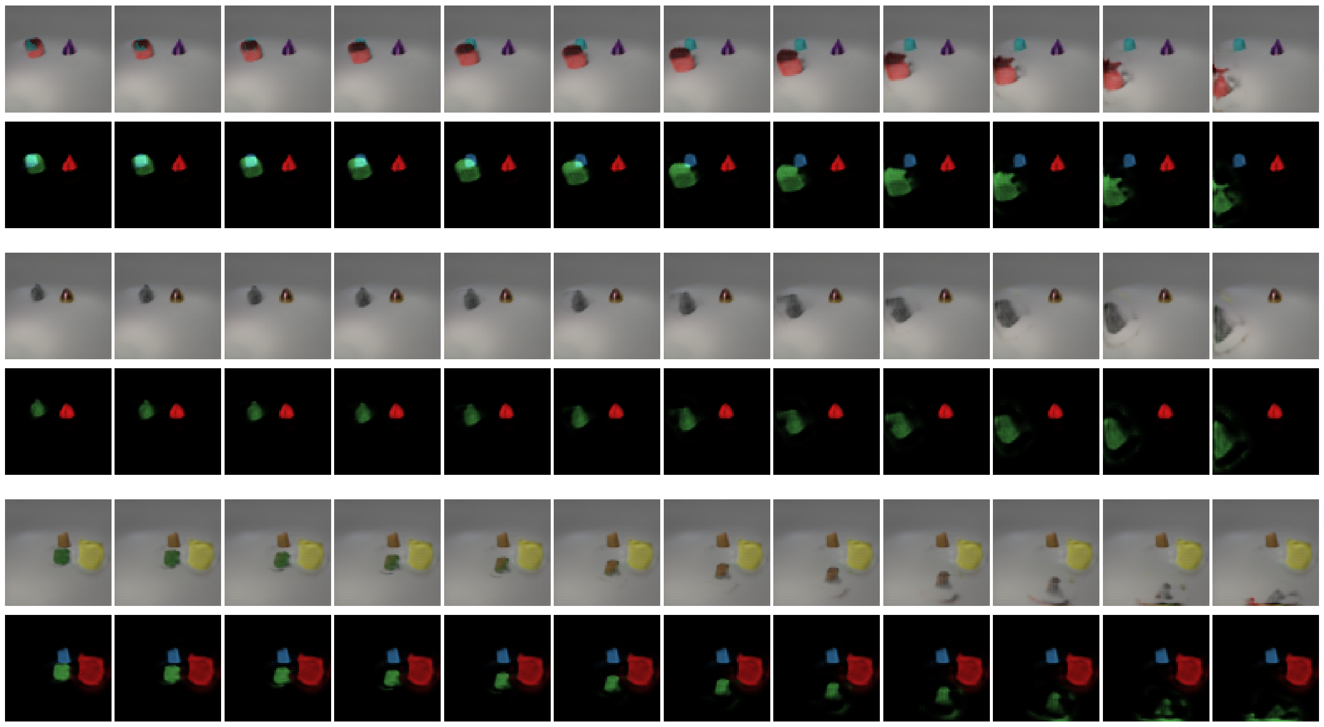
### 3.2. Ablation Studies

**Robustness Against Wrong Distributions:** To investigate the robustness of our method against wrong distributions over the object poses, we train our model on *CompCars* at $64^2$ pixel resolution with an object rotation of 15° instead of 360°, and on *CelebA* at $64^2$ pixel resolution with an object rotation of 360° instead of 15°. Surprisingly, the quantitative results do not change significantly: We obtain the same FID score for *CelebA* (6) and a slightly worse score for *CompCars* (19 compared to 16). Qualitatively, we observe that our model has two modes (front and back position) for the canonical pose for *CompCars*, and still learns a plausible 70° rotation. For *CelebA*, we observe that it learns two rotations with an identity flip after 180°. Note that for the latter, it is not possible to learn a plausible 360° rotation as the dataset only contains frontal images of human faces.

**Composition Operator:** We implement the composition operator (Eq. (8) in the main publication) as a sum for the density, and a density-weighted mean for the feature vector. This is a natural choice when combining non-solid objects [1]. To validate this choice, we evaluate our model on *CompCars* at $64^2$ pixels when using the max operation instead. In this case, we evaluate the density and feature for a given 3D point and viewing direction $(\mathbf{x}, \mathbf{d})$ by first identifying the feature field with the maximum density at 3D point $\mathbf{x}$, and then selecting the density and feature of this field. We observe that using our choice instead of the max operations leads to an FID score improvement from 18.47 to 16.16.

**Feature Rendering:** In Fig. 6, we show a qualitative comparison of our method with and without the neural renderer, where for the latter, we directly render RGB color instead of features. We find that removing the neural renderer leads to slower inference, higher memory consumption, and degraded results. Also quantitatively, we find that results are worse (61 to 16 in FID on *CompCars* at $64^2$ pixels).

[3]For KID evaluation, we use the code from https://github.com/abdulfatir/gan-metrics-pytorch with the official Tensorflow Inception network weights from https://github.com/mseitzer/pytorch-fid.


(a) Controllable Image Synthesis [13].

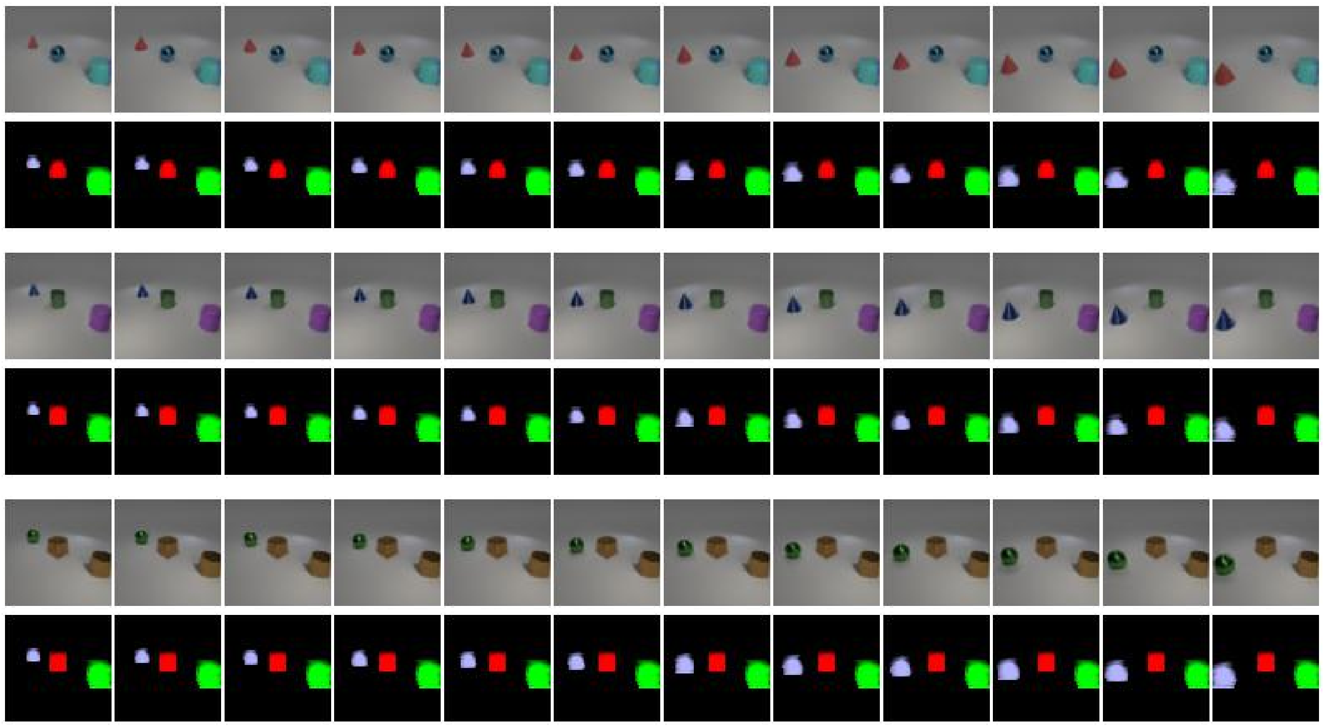
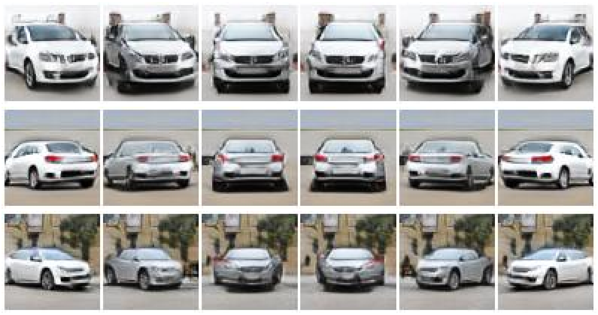
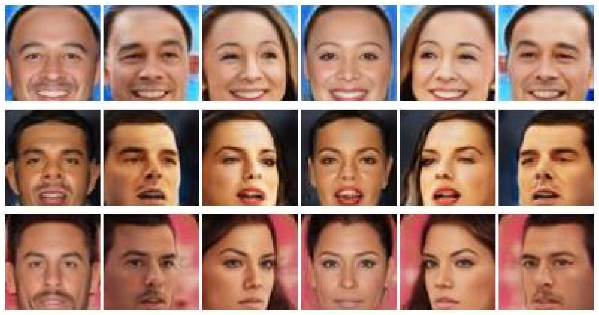

(b) Ours.

Figure 4: **Qualitative Comparison to CIS.** We show renderings (top rows) and color-coded alpha maps (bottom rows) for single-object depth translations for Controllable Image Synthesis (CIS) [13] and our method on *Clevr-0123* at $64^2$ pixels. While both methods allow for controllable image synthesis, we achieve more consistent and higher-quality results. Note that while CIS learns the object-level transformations, we sample them from the data distribution. Further, CIS requires additional supervision in the form of labelled background images, while ours learns to disentangle objects from the background unsupervised.

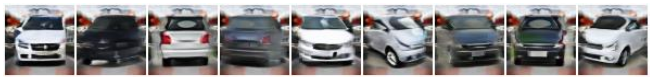


(a) 70° Rotation on *CompCars*.

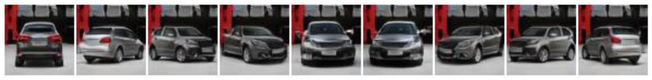


(b) 360° Rotation on *CelebA*.

Figure 5: **Robustness.** To investigate the robustness of our model against wrong distributions over object poses, we train our method on *CompCars* with only a 70° degree rotation instead of 360°, and on *CelebA* with a 360° degree rotation instead of 70°. For *CompCars*, our model has two modes (front and back) in the canonical pose, and still learns a plausible 70° rotation. For *CelebA*, it learns two half rotations where it flips the entity after 180° degrees. Note that for the latter, it is not possible to learn a plausible 360° rotation as the dataset only contains frontal images of human faces.



(a) Ours without Neural Renderer



(b) Ours

Figure 6: **Effect of Neural Renderer.** We compare our method without (Fig. 6a) and with the neural rendering pipeline (Fig. 6b) on *CompCars* at $64^2$. We find that incorporating a neural renderer not only leads to better quantitative results, but also qualitatively our method achieves more consistent representations.

## 3.3. Controllable Image Synthesis

**Many-Object Scenes:** As discussed in the main publication, our model is able to disentangle objects from the background without any explicit supervision. To test our model on scenes with even more objects, we render images with 10 random primitives in the scene (*Clevr-10*). In Fig. 11, we show renderings of only backgrounds and only foreground objects as well as color-coded object alpha maps and the synthesized images. We observe that even for 10-object scenes, our model is able to disentangle individual objects at both $64^2$ and $256^2$ pixel resolutions.

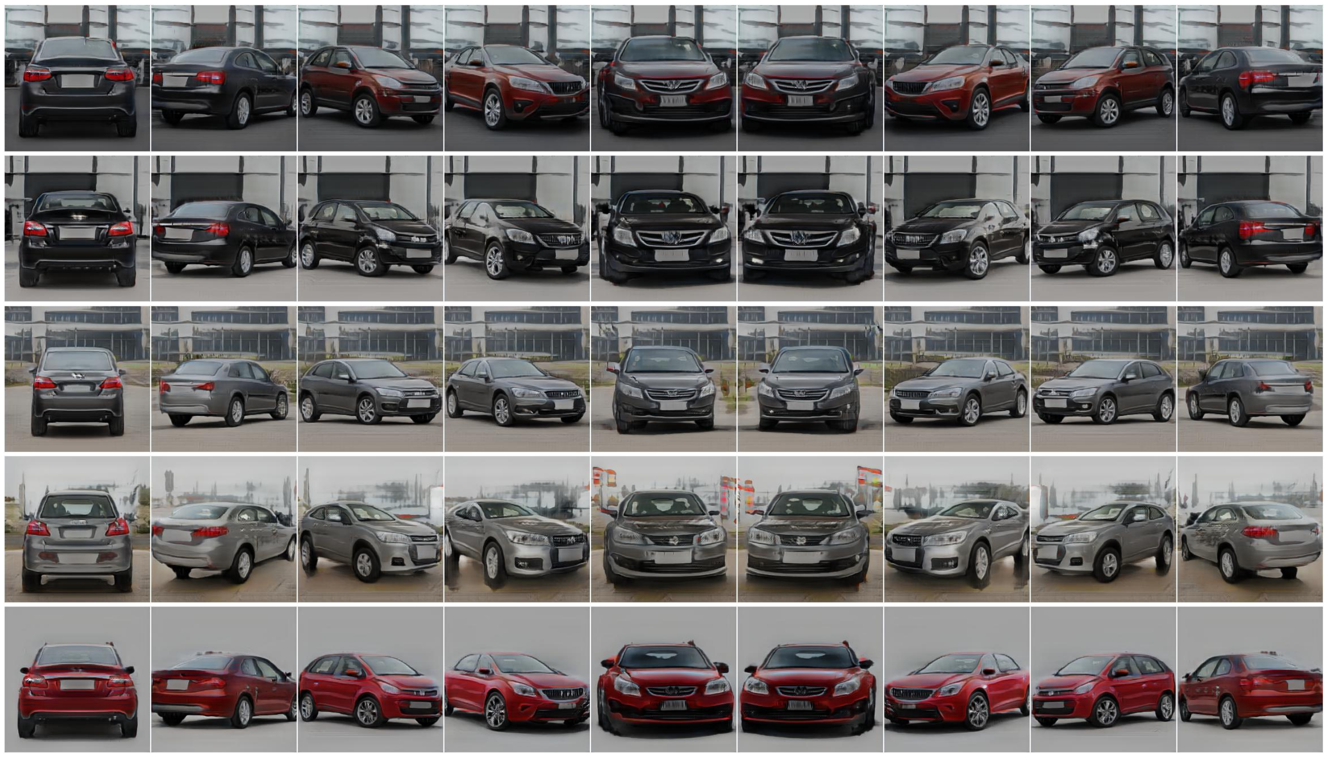
**Additional Results:** In Fig. 7, 8, 9, and 10, we show additional examples in which we control the scene during image synthesis.

## 3.4. Failure Cases

We sometimes observe disentanglement failures, e.g. for *Churches* where the background contains a church, or for *Cars*, where background elements are contained in the foreground object (see Fig. 12). We attribute these failures to mismatches between the assumed uniform distributions over object and camera poses (see above) and their real distributions. We identify learning the distributions instead from data as promising future work.

## 3.5. Random Samples

We show grids of random samples from our method on all datasets in Fig. 13, 14, 15, 16, 17 at $64^2$ pixel resolution and in Fig. 18, 19, 20, 21, 22, 23, 24, 25 at $256^2$ pixel resolution.


(a) Object Rotation for *CompCars* at $256^2$ pixels.

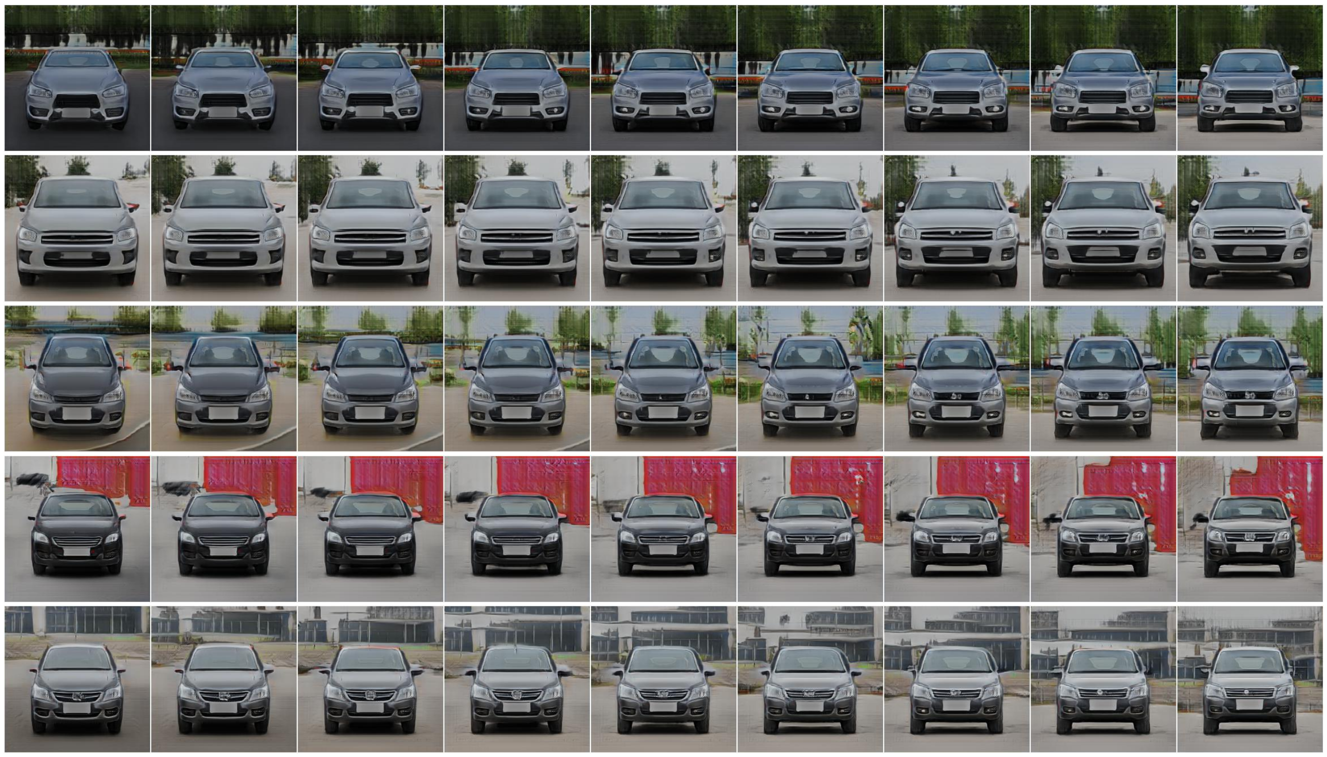
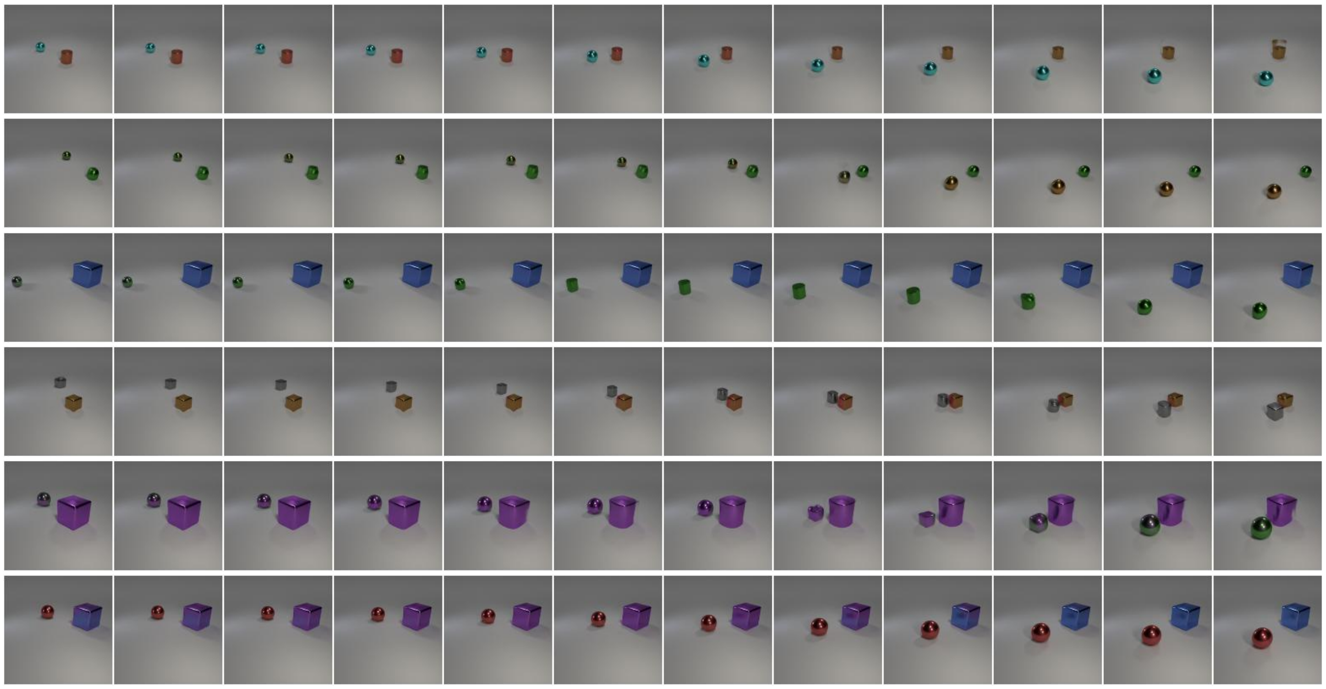
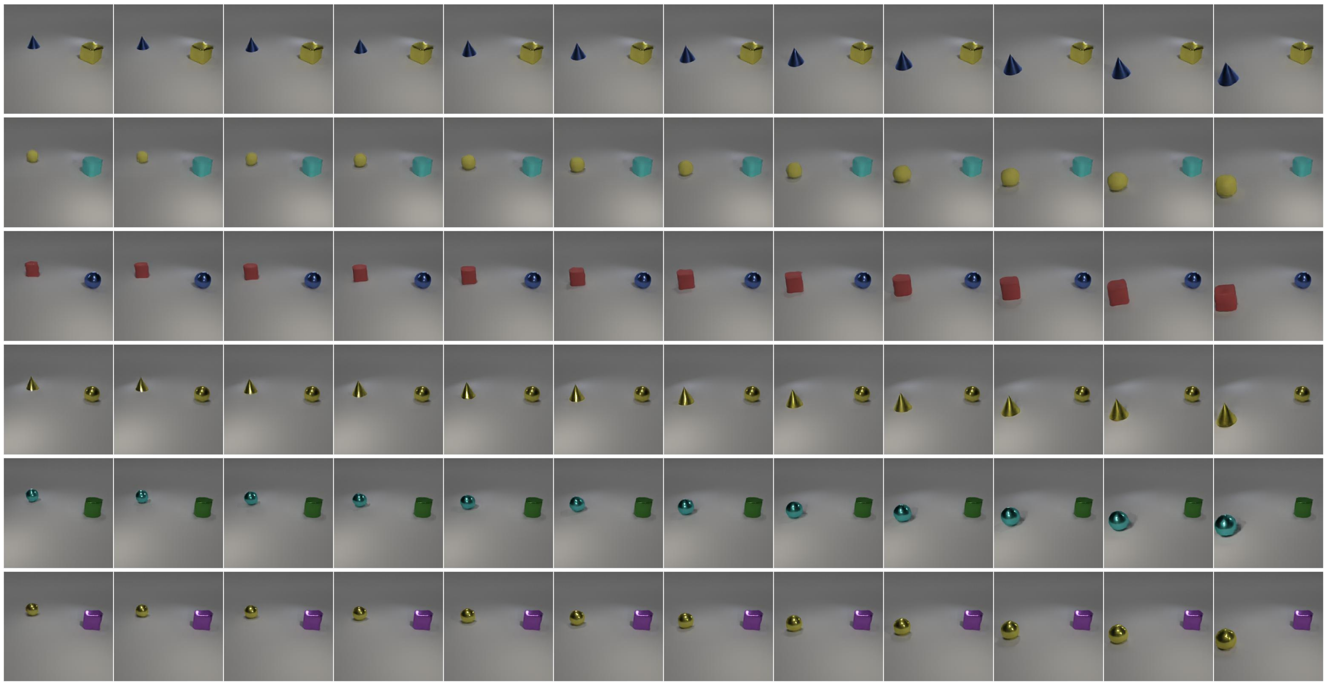

(b) Camera Elevation for *CompCars* at $256^2$ pixels.

Figure 7: **Controllable Image Synthesis.** We show object rotations and camera elevations for our method on *CompCars* at $256^2$ pixel resolution.



(a) Single-Object Translation for a 2D-based GAN [21] at $256^2$ pixels.



(b) Single-Object Translation for our Method at $256^2$ pixels.

Figure 8: **Controllable Image Synthesis.** We show single-object depth translations for a 2D-based GAN [21] and our method at $256^2$ pixels. Note how for the 2D-based method, translating one object might affect the other. In contrast, we incorporate 3D compositional scene structure into the generative model, leading to more consistent results.

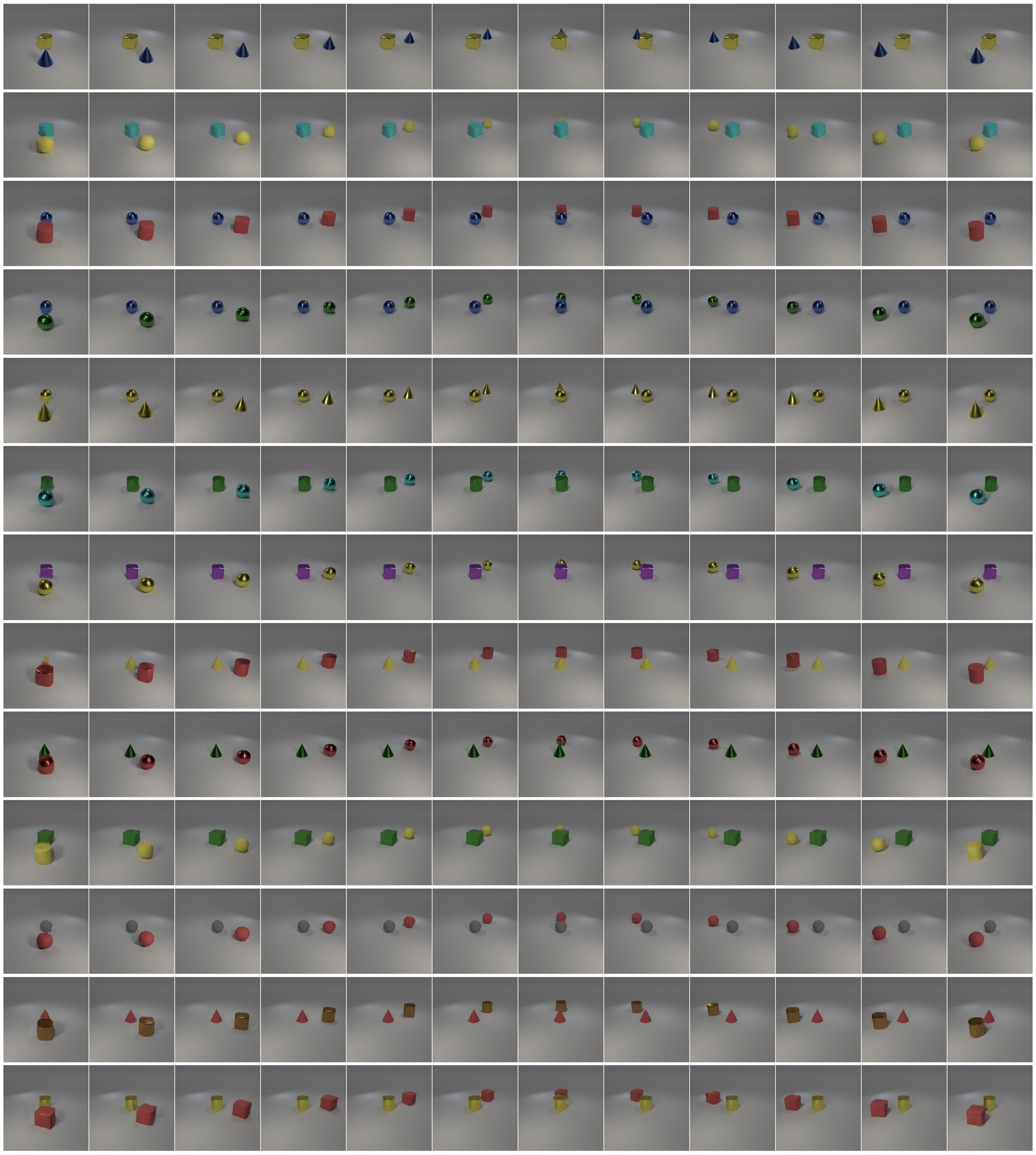
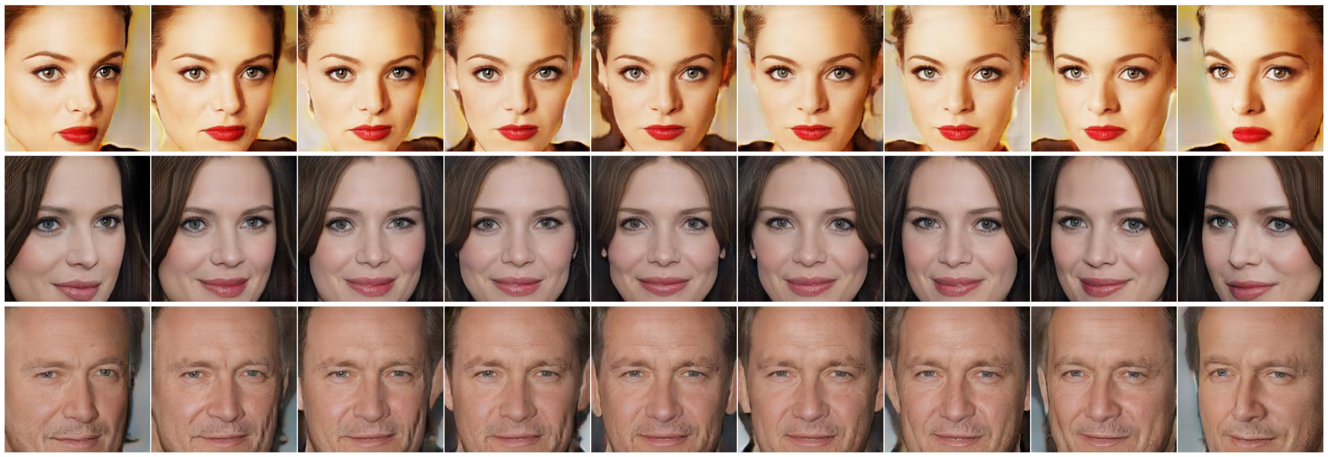
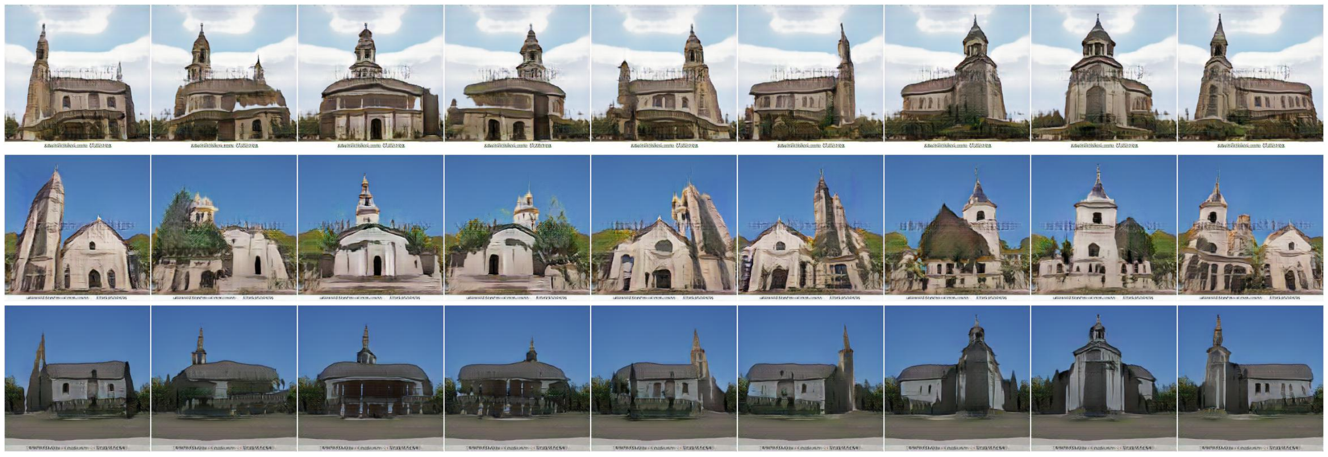


Figure 9: **Controllable Image Synthesis.** We show circular translations for our method on *Clevr-2* at $256^2$ pixel resolution.


(a) Object Rotation for *CelebA-HQ* at $256^2$ pixels.


(b) Object Rotation for *LSUN Churches* at $256^2$ pixels.

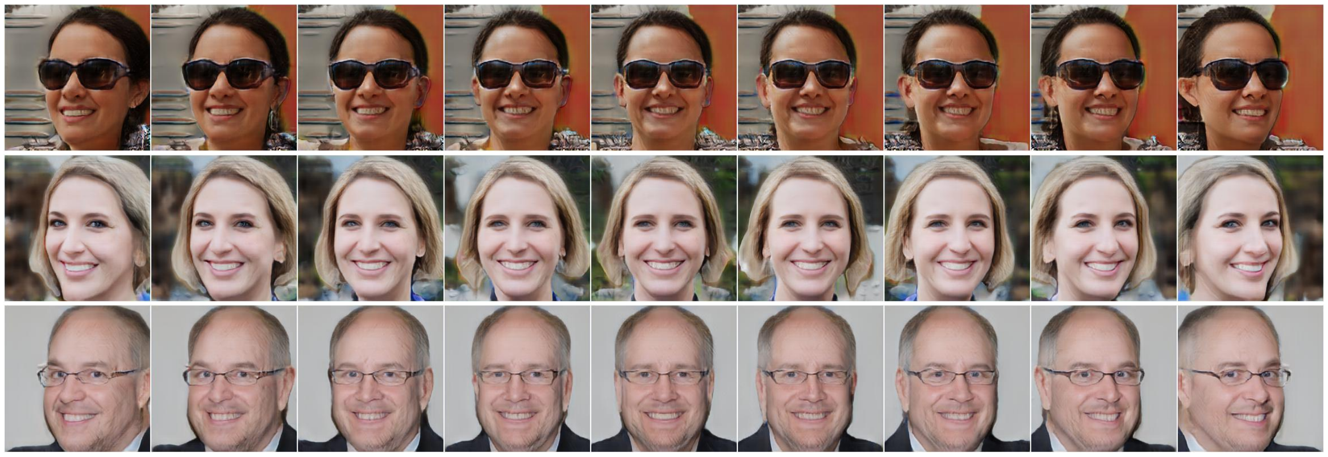
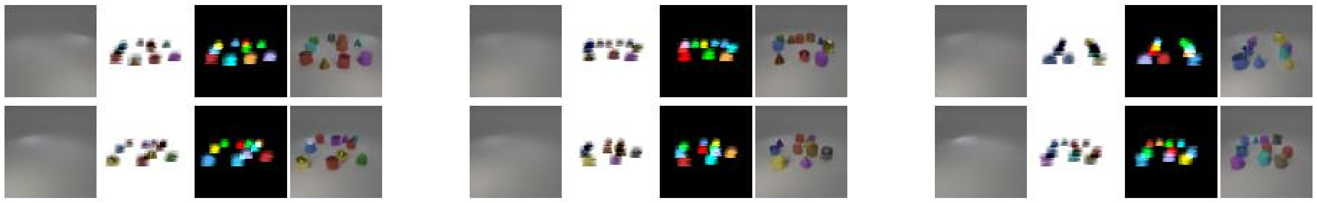

(c) Object Rotation for *FFHQ* at $256^2$ pixels.

Figure 10: **Controllable Image Synthesis.** We show object rotations for our method on *CelebA-HQ*, *LSUN Churches*, and *FFHQ* at $256^2$ pixel resolution.


(a) Ours on *Clevr-10* at $64^2$ pixels.

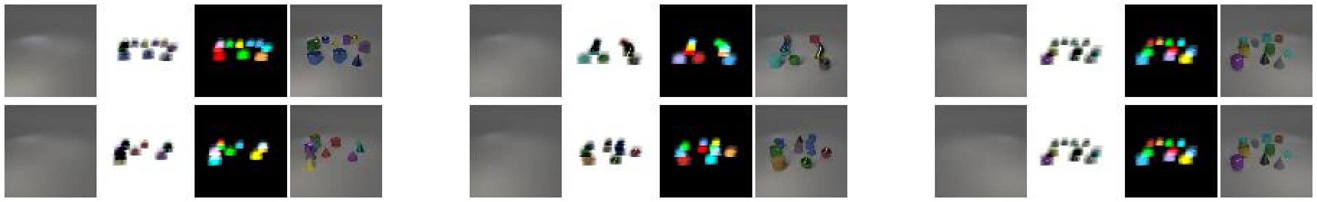
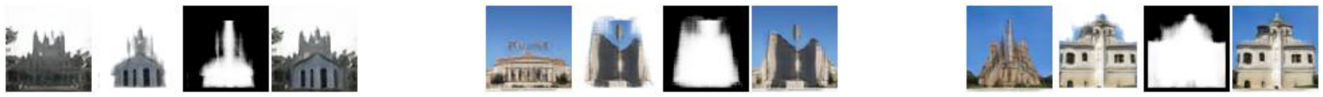

(b) Ours on *Clevr-10* at $256^2$ pixels.

Figure 11: **Unsupervised Disentanglement.** From left to right, we show only background, only objects, color-coded object alpha maps, and synthesized images for our method on *Clevr-10* at $64^2$ and $256^2$ pixel resolutions, respectively. Note that our model is able to disentangle individual objects for images of 10-object scenes without any explicit supervision.


(a) Disentanglement Failure on *Churches*.

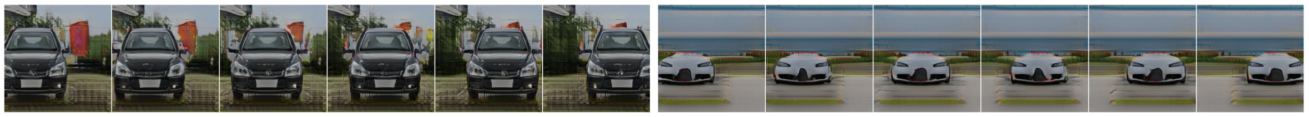

(b) Disentanglement Failure on *CompCars*.

Figure 12: **Disentanglement Failures.** For *Churches*, the background sometimes contains a church, and for *CompCars*, the object sometimes contains background parts or vice versa. We attribute these to mismatches between the assumed uniform distributions over object and camera poses and their real distributions, and identify learning them instead as interesting future work.

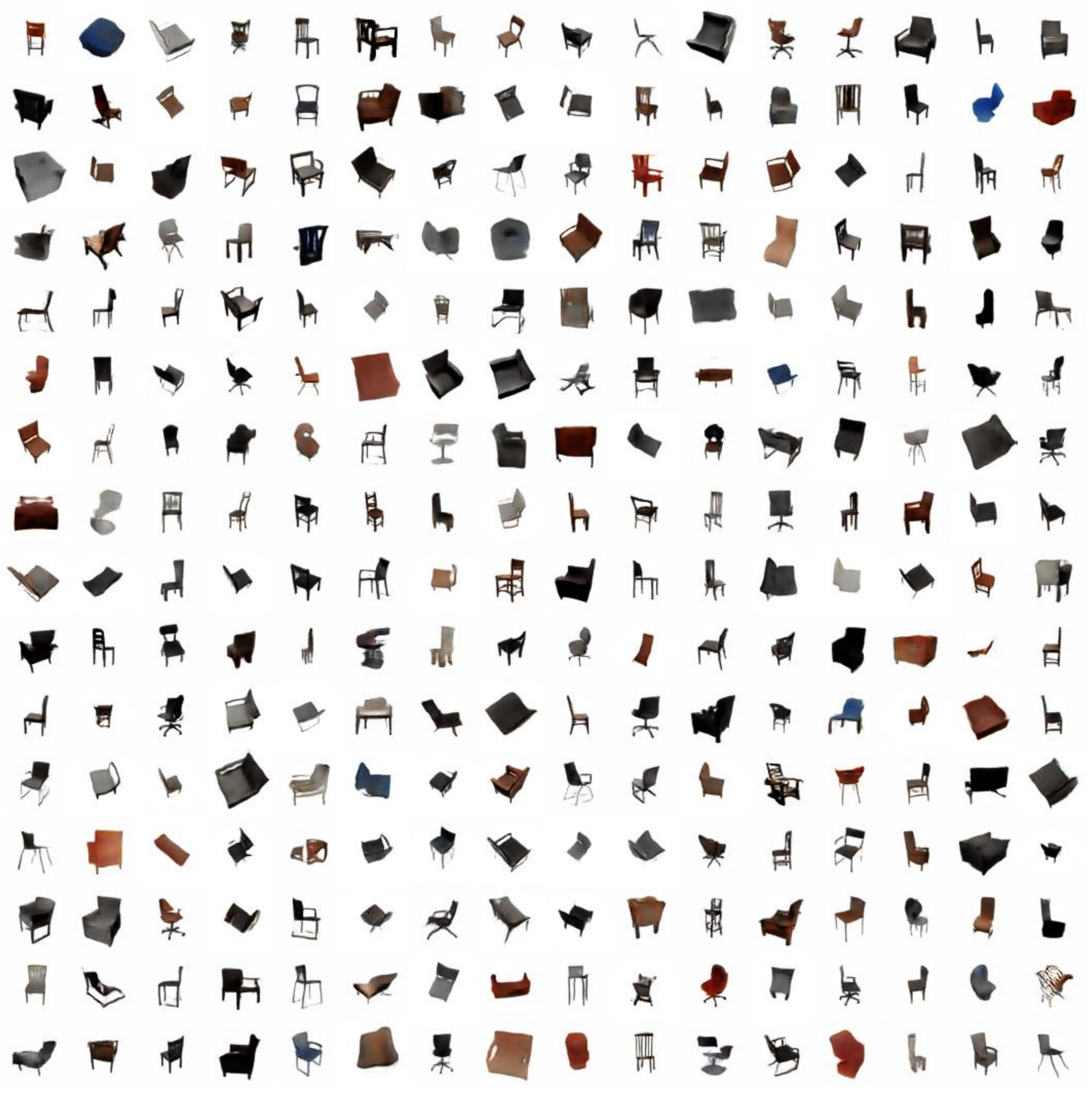


Figure 13: **Random Samples.** We show random samples for our method on *Chairs* at $64^2$ image resolution.

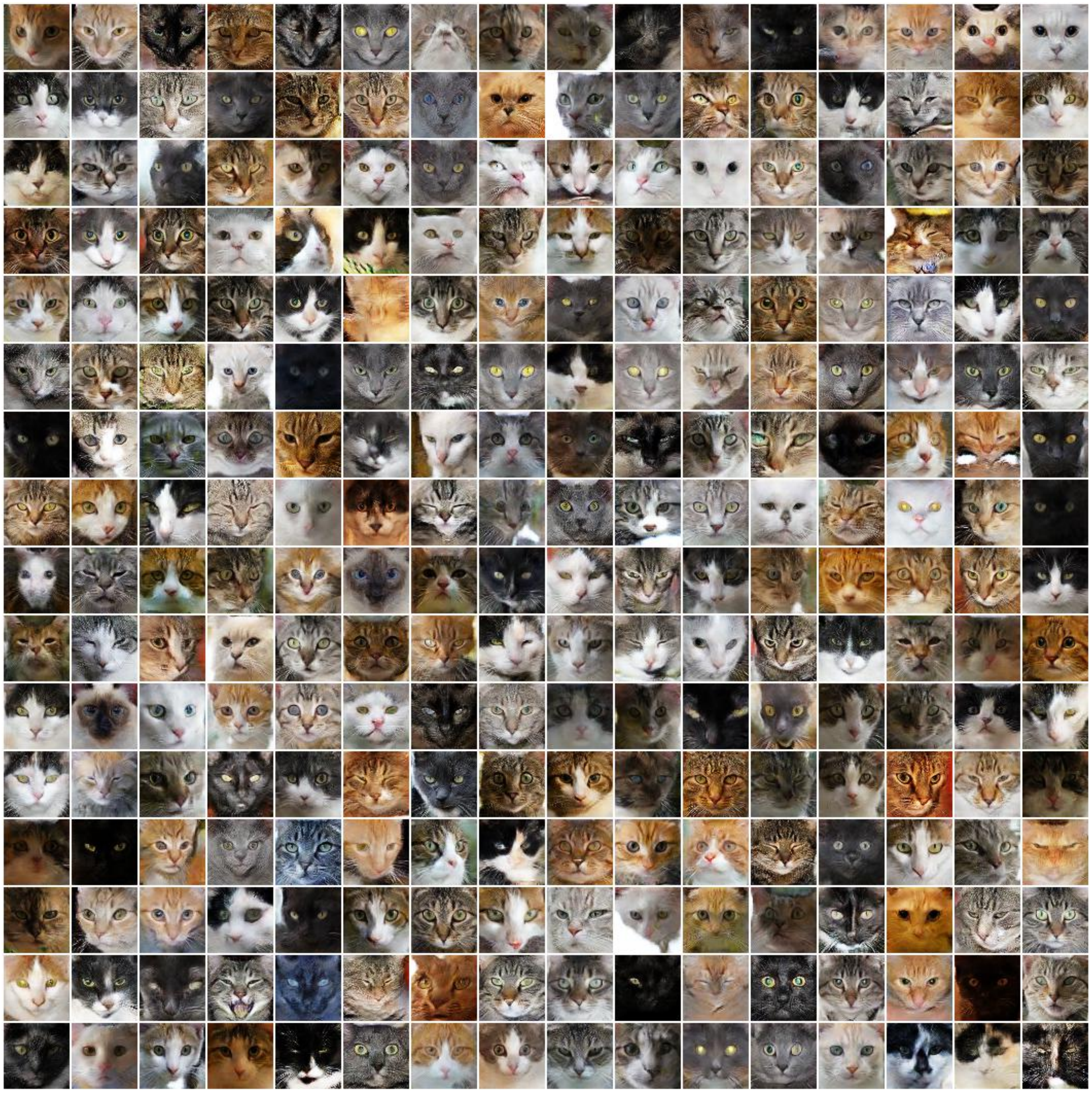


Figure 14: **Random Samples.** We show random samples for our method on *Cats* at $64^2$ image resolution.

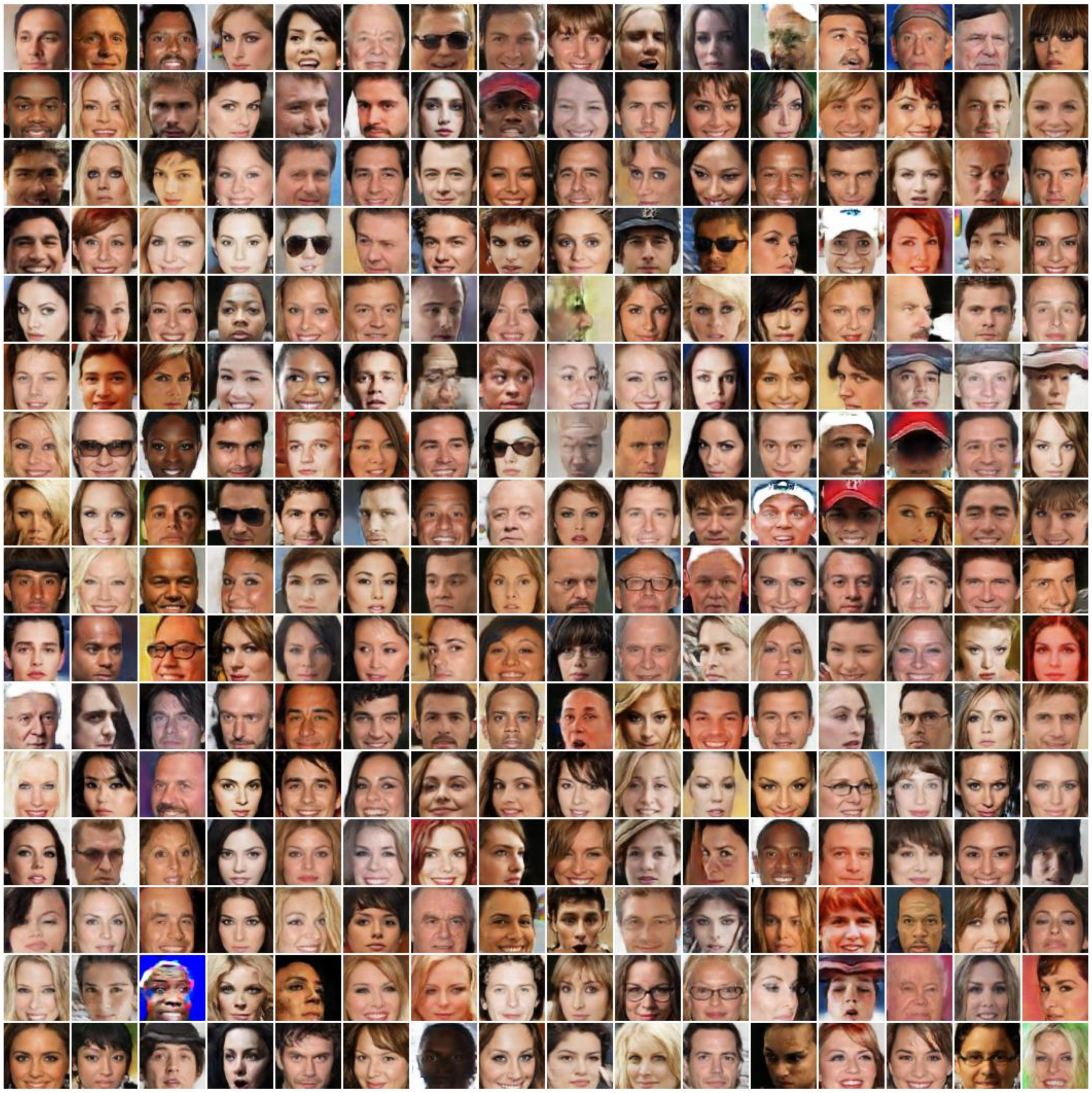


Figure 15: **Random Samples.** We show random samples for our method on *CelebA* at $64^2$ image resolution.

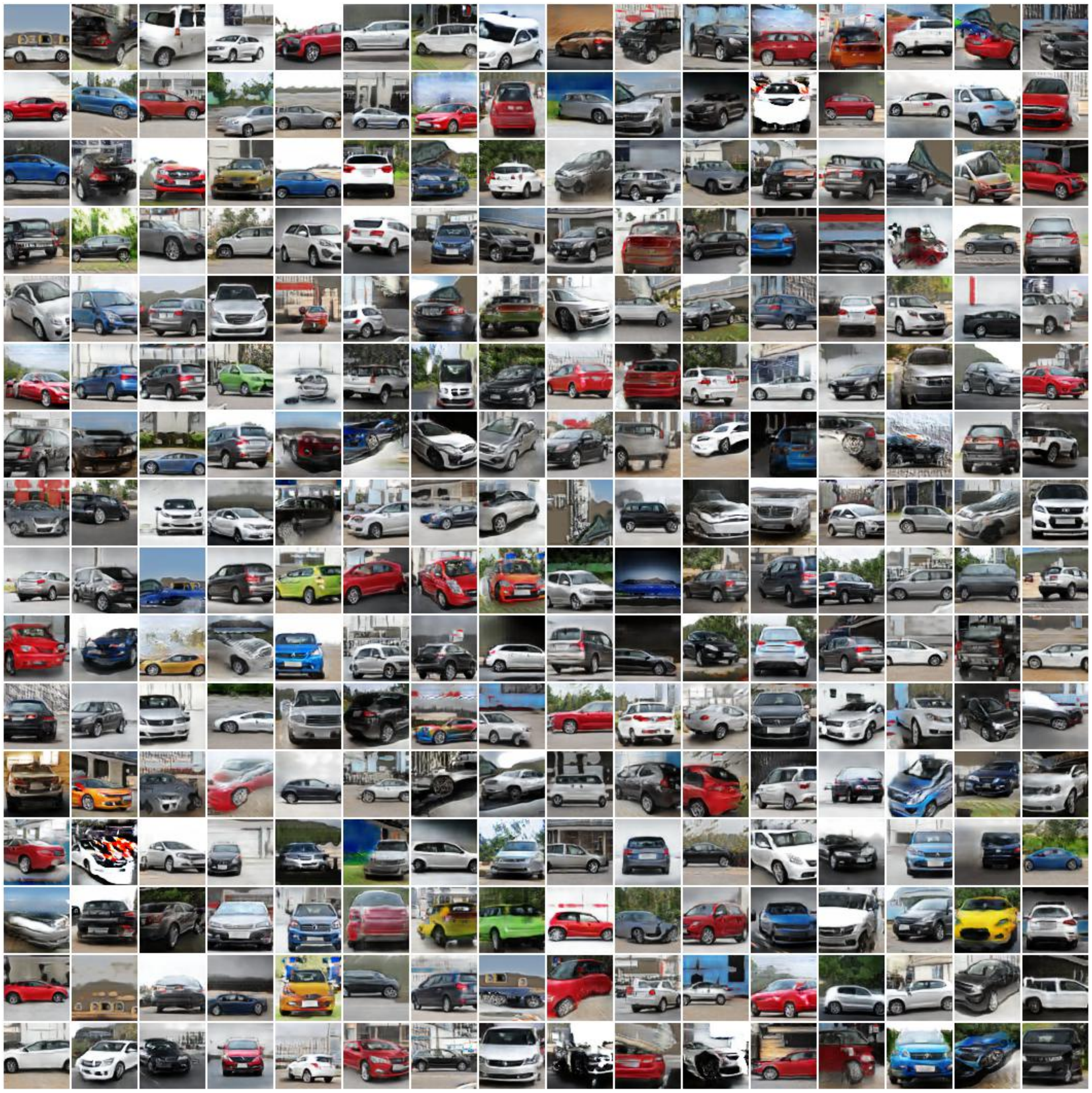


Figure 16: **Random Samples.** We show random samples for our method on *CompCars* at $64^2$ image resolution.

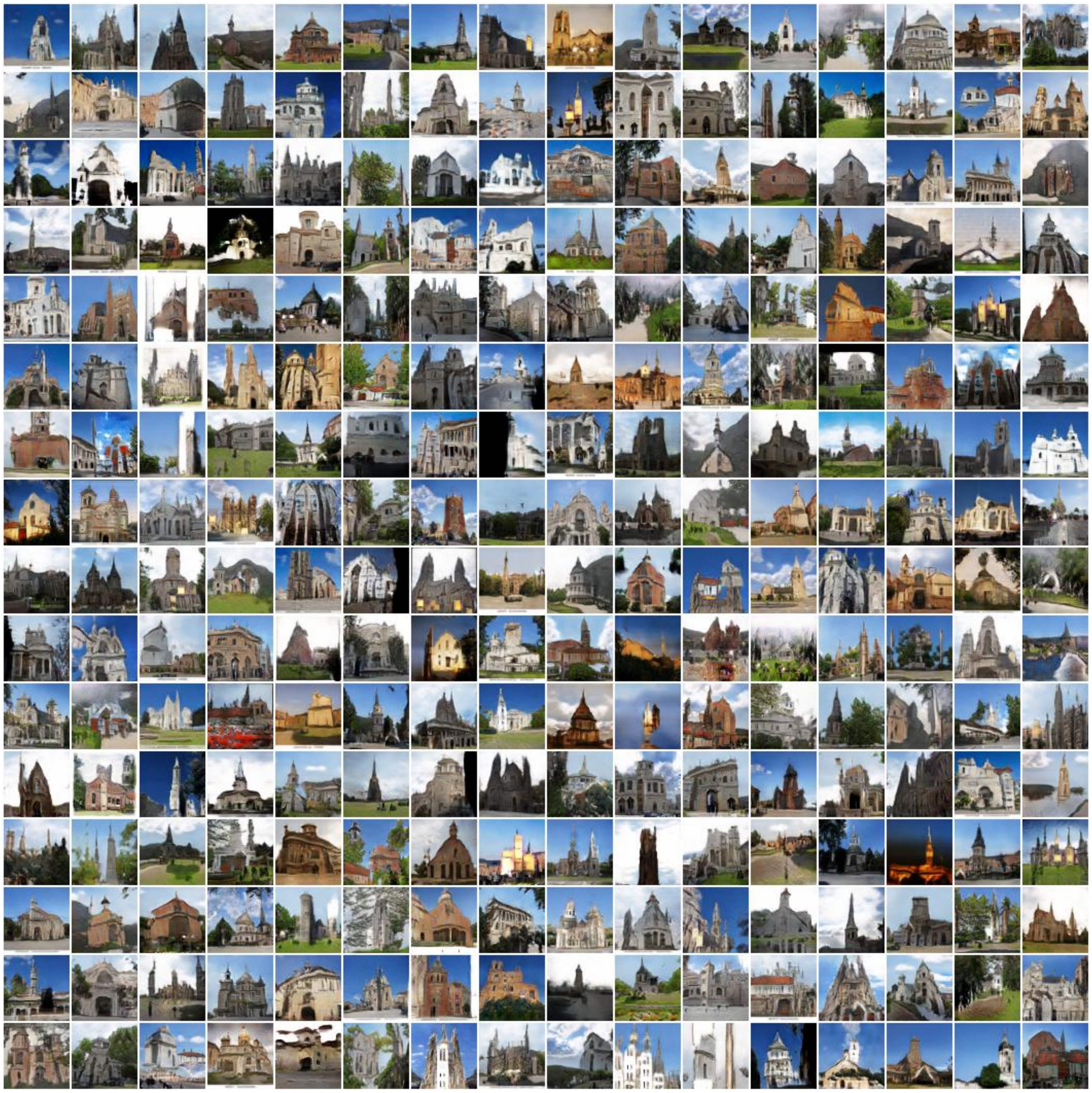


Figure 17: **Random Samples.** We show random samples for our method on *Churches* at $64^2$ image resolution.

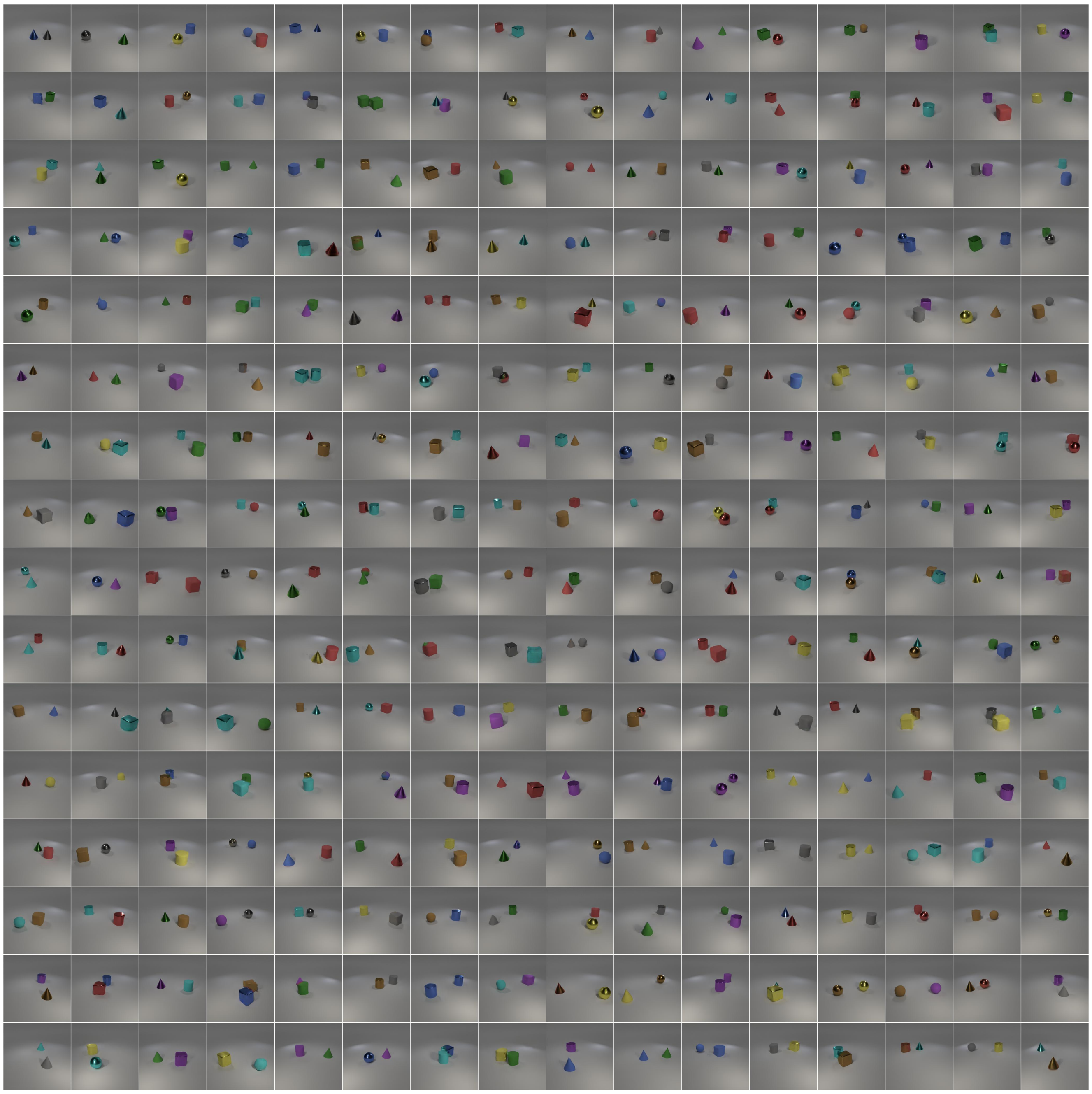


Figure 18: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *Clevr-2*.

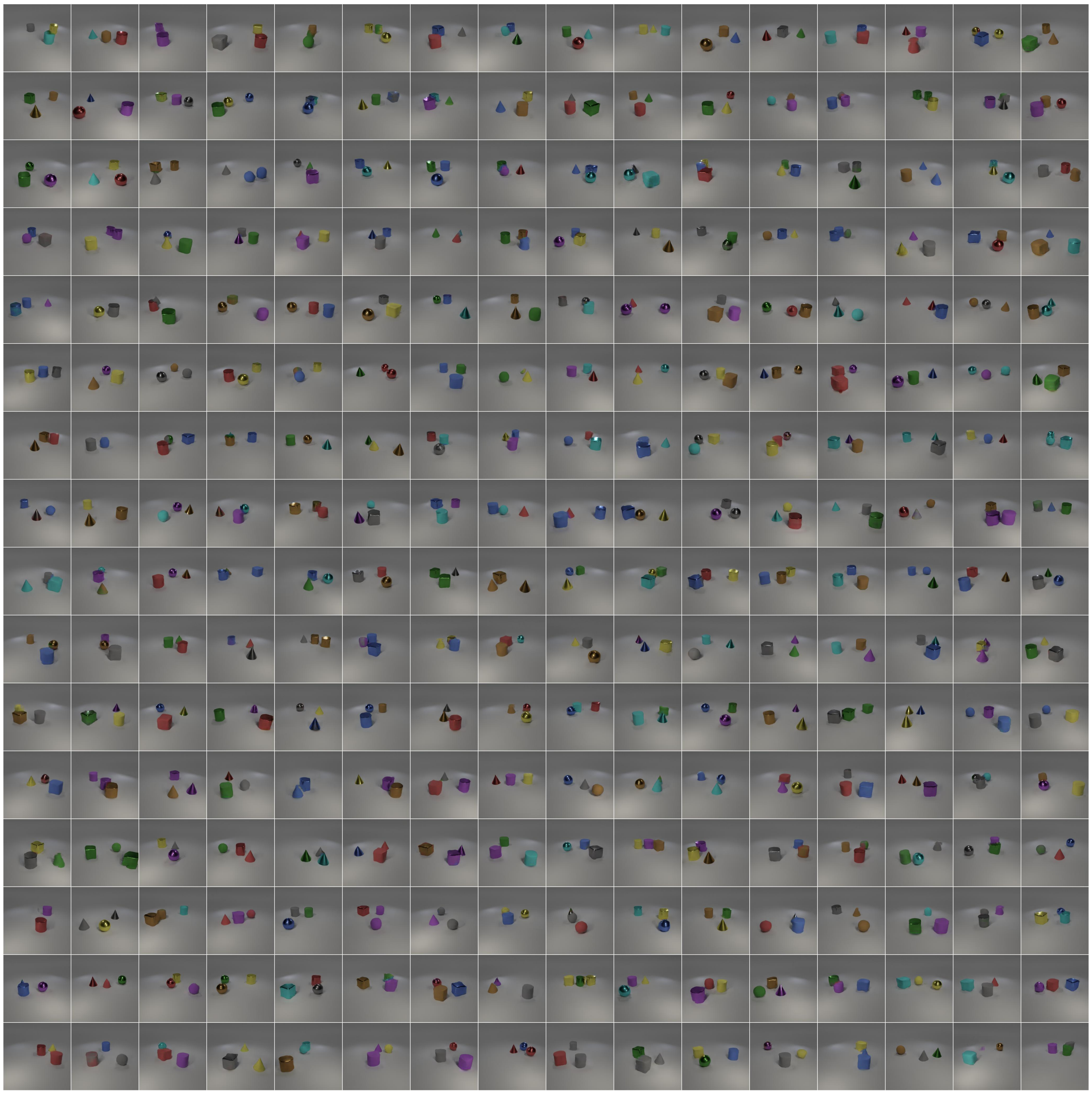


Figure 19: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *Clevr-3*.

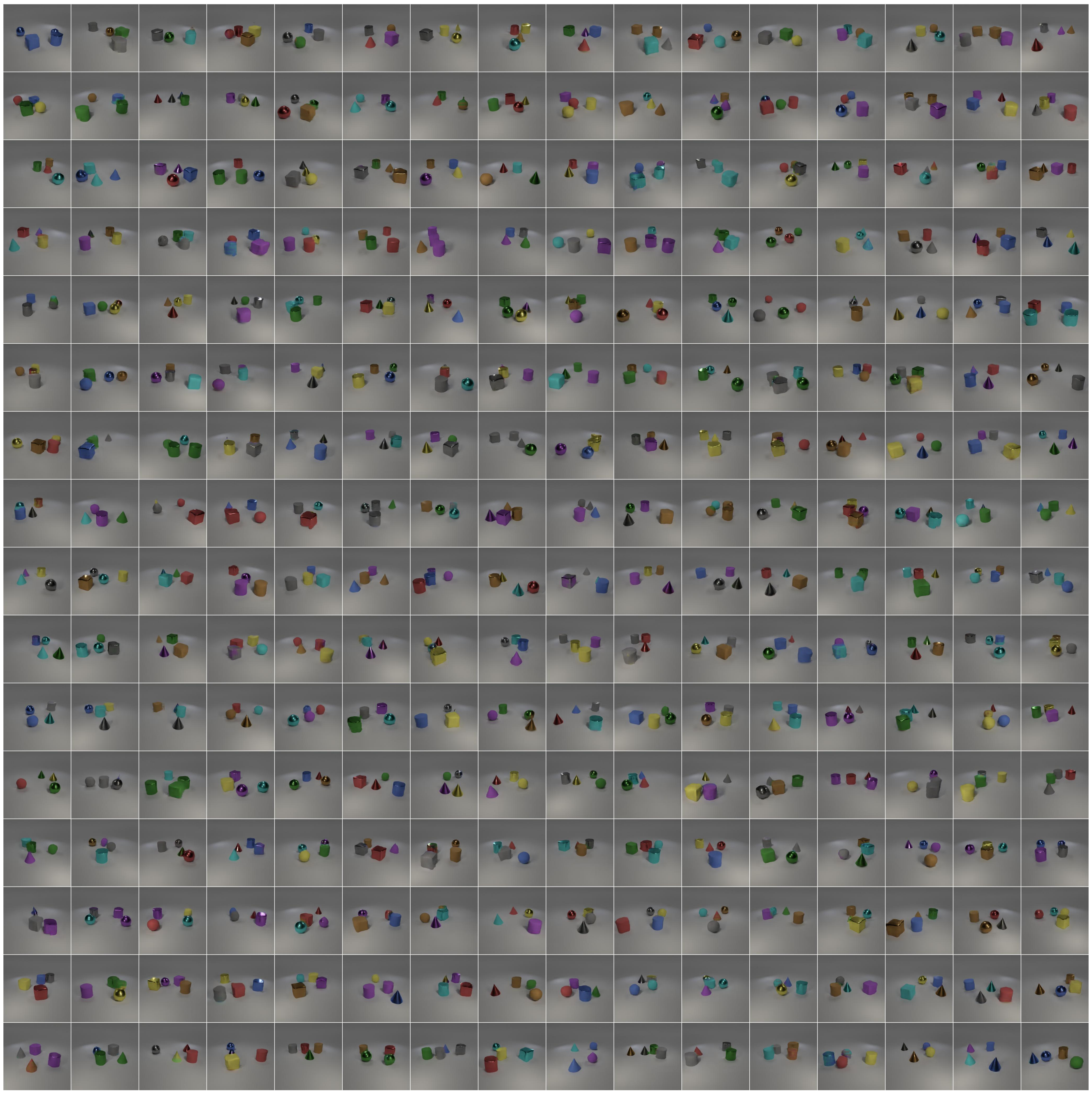


Figure 20: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *Clevr-4*.

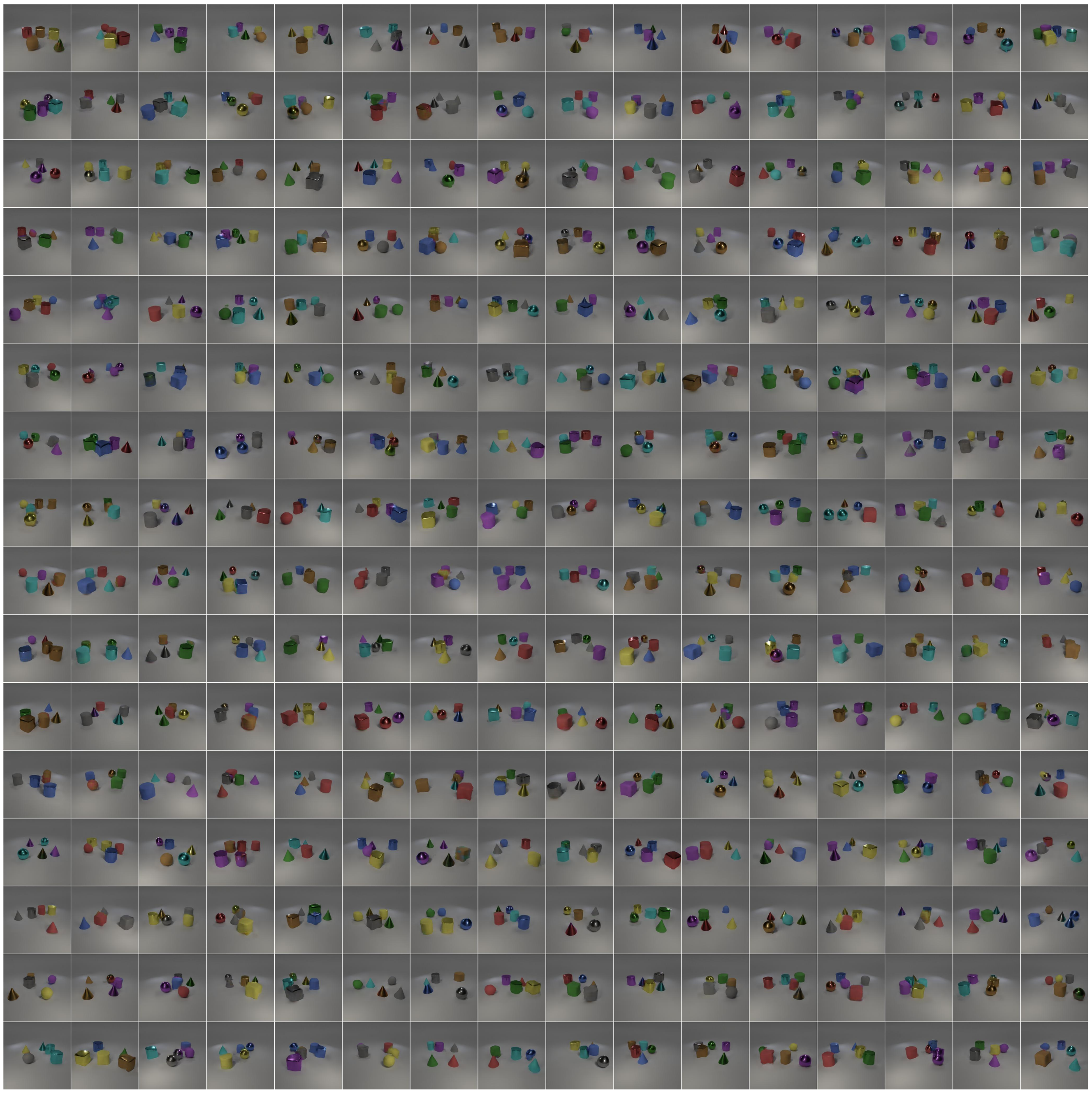


Figure 21: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *Clevr-5*.

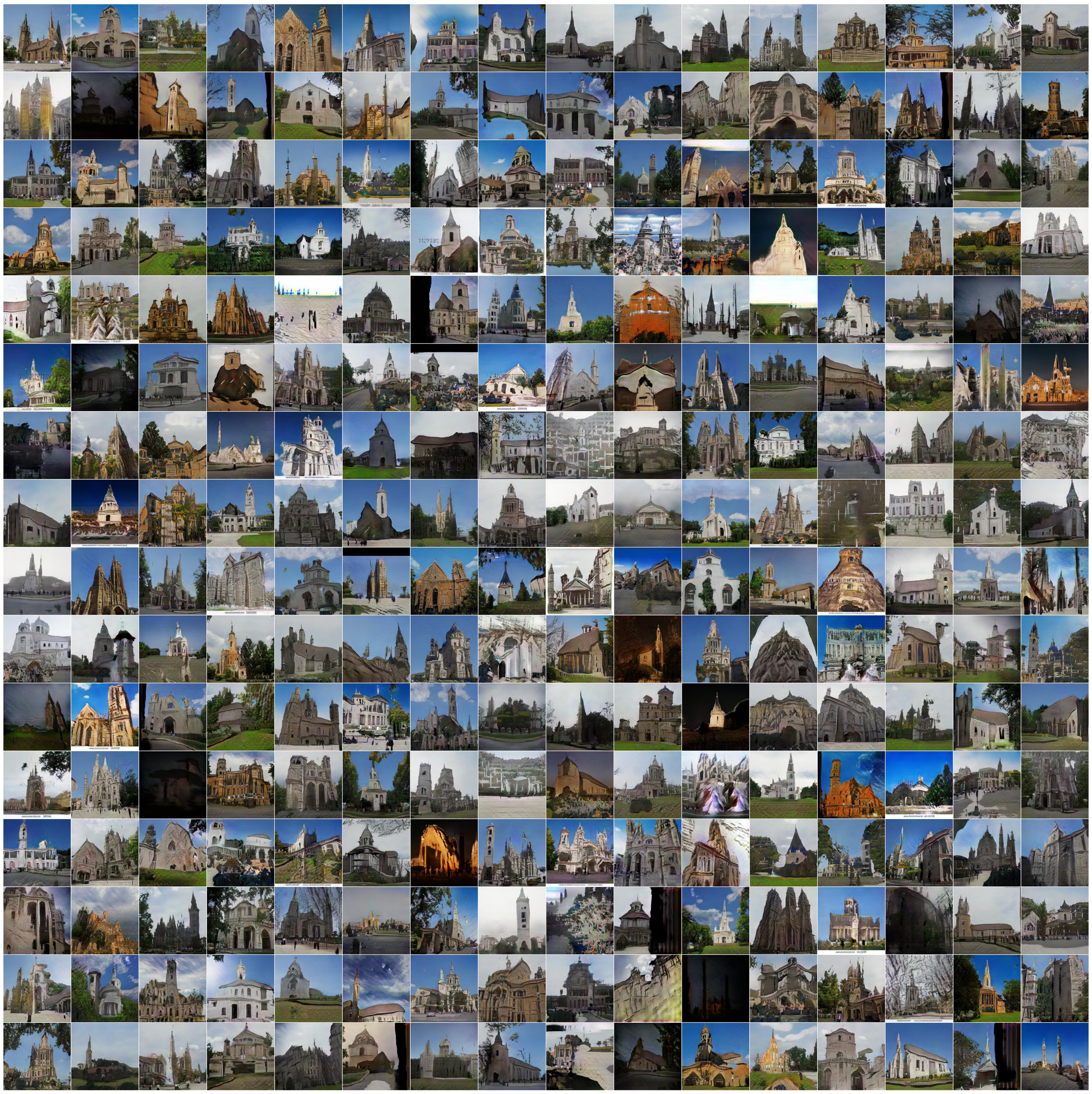


Figure 22: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *Churches*.

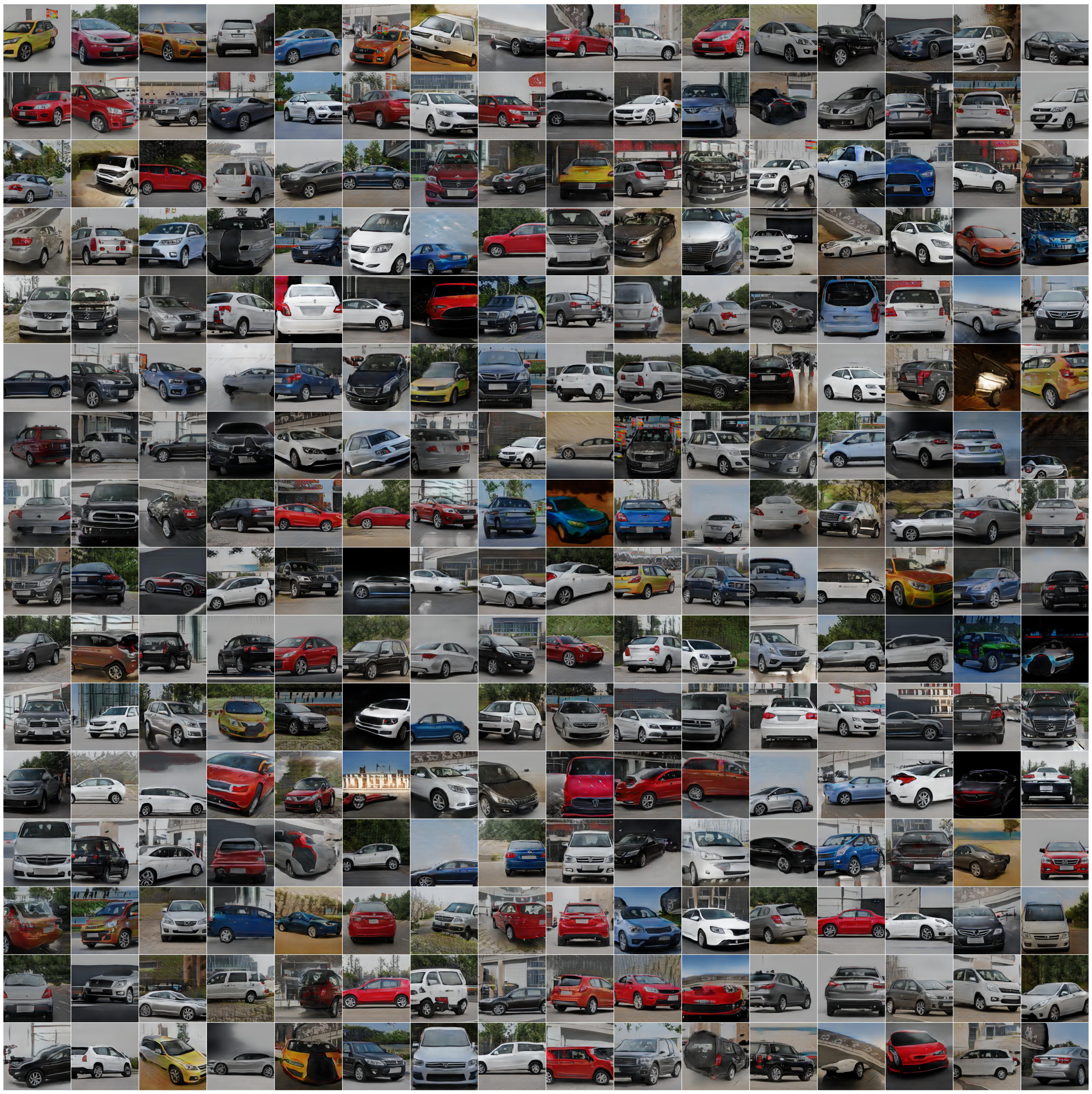


Figure 23: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *CompCars*.

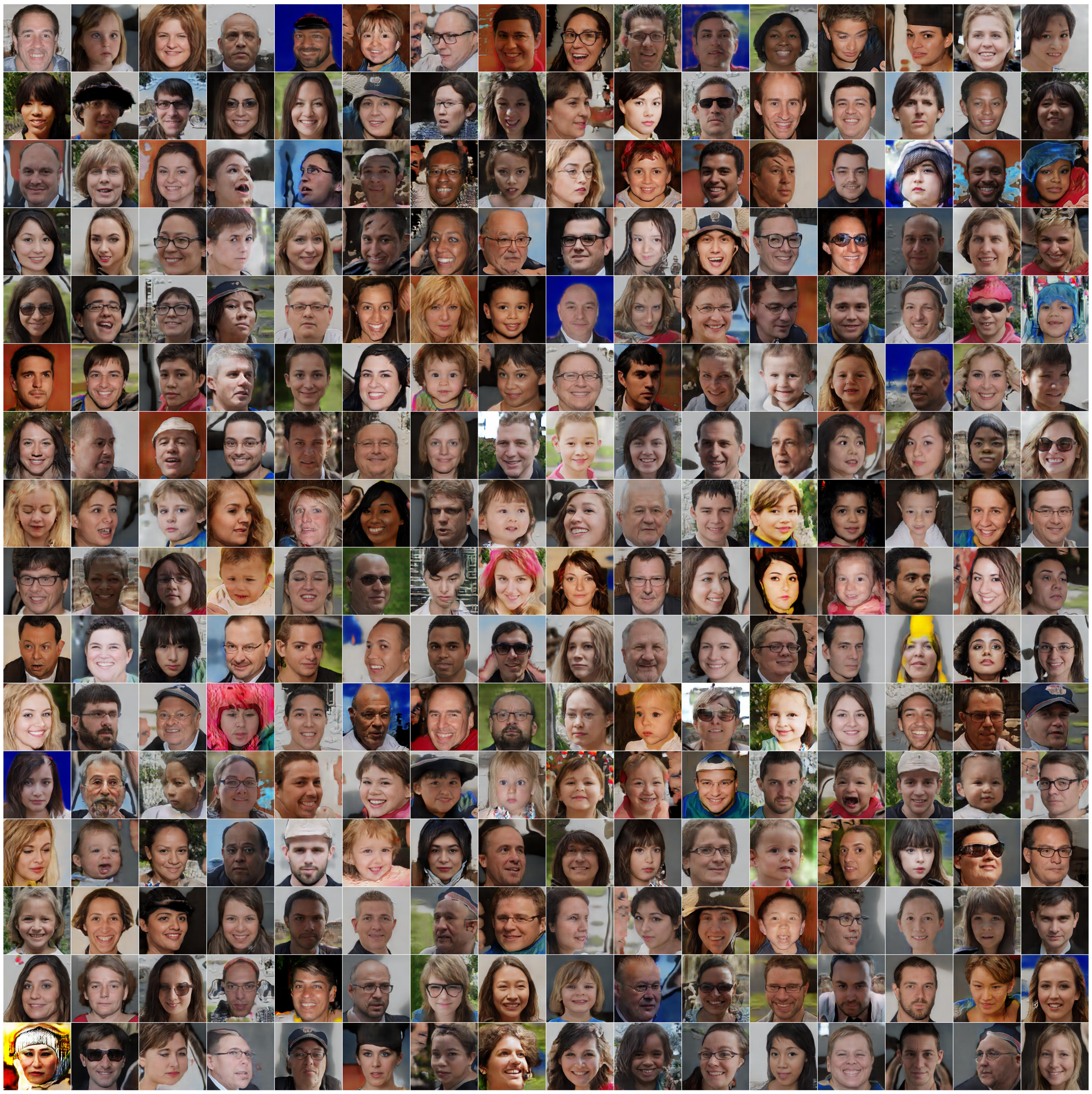


Figure 24: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *FFHQ*.

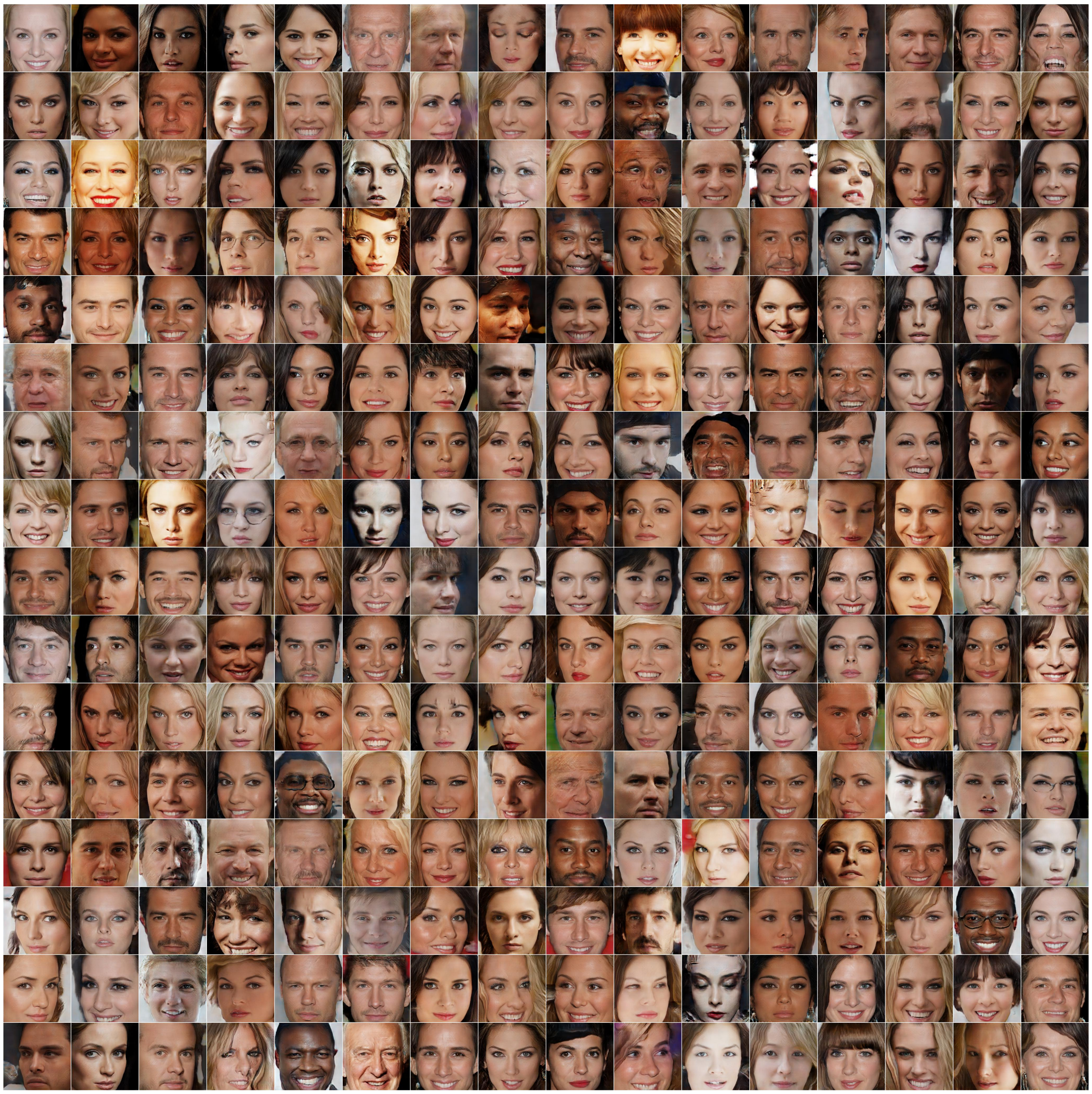


Figure 25: **Random Samples.** We show random samples for our method at $256^2$ image resolution on *CelebA-HQ*.

## References

[1] Robert A. Drebin, Loren C. Carpenter, and Pat Hanrahan. Volume rendering. In *ACM Trans. on Graphics*, 1988. 5

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 3

[3] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 3, 5

[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. 3

[5] Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 3, 4

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1

[7] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 3

[8] 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. 4

[9] 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. 4

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[11] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015. 3

[13] Yiyi Liao, Katja Schwarz, Lars Mescheder, and Andreas Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5, 6

[14] 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. 4

[15] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 3

[16] 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. 1, 2

[17] 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, 5

[18] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3, 5

[19] 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. 1

[20] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *Communications of the ACM*, 2018. 4

[21] William S. Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 9

[22] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2016. 1

[23] 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. 1, 3, 4

[24] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 2

[25] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. LR-GAN: layered recursive generative adversarial networks for image generation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. 5

[26] Jiaolong Yang and Hongdong Li. Dense, accurate optical flow estimation with piecewise parametric model. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4

[27] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv.org*, 1506.03365, 2015. 4

[28] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat head detection - how to effectively exploit shape and texture features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2008. 4