GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields

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

Abstract

Deep generative models allow for photorealistic image synthesis at high resolutions. But for many applications, this is not enough: content creation also needs to be controllable. While several recent works investigate how to disentangle underlying factors of variation in the data, most of them operate in 2D and hence ignore that our world is three-dimensional. Further, only few works consider the compositional nature of scenes. Our key hypothesis is that incorporating a compositional 3D scene representation into the generative model leads to more controllable image synthesis. Representing scenes as compositional generative neural feature fields allows us to disentangle one or multiple objects from the background as well as individual objects' shapes and appearances while learning from unstructured and unposed image collections without any additional supervision. Combining this scene representation with a neural rendering pipeline yields a fast and realistic image synthesis model. As evidenced by our experiments, our model is able to disentangle individual objects and allows for translating and rotating them in the scene as well as changing the camera pose.

1. Introduction

The ability to generate and manipulate photorealistic image content is a long-standing goal of computer vision and graphics. Modern computer graphics techniques achieve impressive results and are industry standard in gaming and movie productions. However, they are very hardware expensive and require substantial human labor for 3D content creation and arrangement.

In recent years, the computer vision community has made great strides towards highly-realistic image generation. In particular, Generative Adversarial Networks (GANs) \cite{goodfellow2014generative} emerged as a powerful class of generative models. They are able to synthesize photorealistic images at resolutions of \(1024^2\) pixels and beyond \cite{pix2pix, larsen2015autoencoding, kulkarni2015deep, Bansal2017}. Despite these successes, synthesizing realistic 2D images is not the only aspect required in applications of generative models. The generation process should also be controllable in a simple and consistent manner. To this end, many works \cite{park2019semantic, mao2017least, karras2018progressive, kumar2019neural, wang2019Localized, stutz2019unsupervised, nine2019generative} investigate how disentangled representations can be learned from data without explicit supervision. Definitions of disentanglement vary \cite{alemi2016deep, kingma2018glow}, but commonly refer to being able to control an attribute of interest, e.g. object shape, size, or pose, without changing other attributes. Most approaches, however, do not consider the compositional nature of scenes and operate in the 2D domain, ignoring that our world is three-dimensional. This often leads to entangled representations (Fig. 2) and control mechanisms are not built-in, but need to be discovered in the latent space a posteriori. These properties, however, are crucial for successful applications, e.g. a movie production where complex object trajectories ...
In this work, we introduce GIRAFFE, a novel method for generating scenes in a controllable and photorealistic manner while training from raw unstructured image collections. Our key insight is twofold: First, incorporating a compositional 3D scene representation directly into the generative model leads to more controllable image synthesis. Second, combining this explicit 3D representation with a neural rendering pipeline results in faster age synthesis. We advocate to model the formation process directly in 3D, ignoring the three-dimensional structure of our world. In this work, we consider the synthesis process can be controlled at the object-level [3, 4, 7, 18, 19, 26, 45, 86, 90]. While achieving photorealistic results, all aforementioned works model the image formation process in 2D, ignoring the 3D representations can be incorporated as inductive bias into generative models [21, 29–32, 46, 55, 63, 64, 75, 77]. While many approaches use additional supervision [2, 10, 87, 88, 99], we focus on works which are trained on raw image collections like our approach.

Henzler et al. [32] learn voxel-based representations using differentiable rendering. The results are 3D controllable, but show artifacts due to the limited voxel resolutions caused by their cubic memory growth. Nguyen-Phuoc et al. [63, 64] propose voxelized feature-grid representations which are rendered to 2D via a reshaping operation. While achieving impressive results, training becomes less stable and results less consistent for higher resolutions. Liao et al. [46] use abstract features in combination with primitives and differentiable rendering. While han-

Figure 2: **Controllable Image Generation.** While most generative models operate in 2D, we incorporate a compositional 3D scene representation into the generative model. This leads to more consistent image synthesis results, e.g. note how, in contrast to our method, translating one object might change the other when operating in 2D (Fig. 2a and 2b). It further allows us to perform complex operations like circular translations (Fig. 2c) or adding more objects at test time (Fig. 2d). Both methods are trained unsupervised on raw unposed image collections of two-object scenes.

**2. Related Work**

**GAN-based Image Synthesis:** Generative Adversarial Networks (GANs) [24] have been shown to allow for photorealistic image synthesis at resolutions of 1024² pixels and beyond [6, 14, 15, 39, 40]. To gain better control over the synthesis process, many works investigate how factors of variation can be disentangled without explicit supervision. They either modify the training objective [9, 40, 71] or network architecture [39], or investigate latent spaces of well-engineered and pre-trained generative models [1, 16, 23, 27, 34, 78, 96]. All of these works, however, do not explicitly model the compositional nature of scenes. Recent works therefore investigate how the synthesis process can be controlled at the object-level [3, 4, 7, 18, 19, 26, 45, 86, 90]. While achieving photorealistic results, all aforementioned works model the image formation process in 2D, ignoring the three-dimensional structure of our world. In this work, we advocate to model the formation process directly in 3D for better disentanglement and more controllable synthesis.

**Implicit Functions:** Using implicit functions to represent 3D geometry has gained popularity in learning-based 3D reconstruction [11, 12, 22, 59, 60, 65, 67, 69, 76] and has been extended to scene-level reconstruction [8, 13, 35, 72, 79]. To overcome the need of 3D supervision, several works [50, 51, 66, 81, 92] propose differentiable rendering techniques. Mildenhall et al. [61] propose Neural Radiance Fields (NeRFs) in which they combine an implicit neural model with volume rendering for novel view synthesis of complex scenes. Due to their expressiveness, we use a generative variant of NeRFs as our object-level representation. In contrast to our method, the discussed works require multi-view images with camera poses as supervision, train a single network per scene, and are not able to generate novel scenes. Instead, we learn a generative model from unstructured image collections which allows for controllable, photorealistic image synthesis of generated scenes.

**3D-Aware Image Synthesis:** Several works investigate how 3D representations can be incorporated as inductive bias into generative models [21, 29–32, 46, 55, 63, 64, 75, 77]. While many approaches use additional supervision [2, 10, 87, 88, 99], we focus on works which are trained on raw image collections like our approach.
duling multi-object scenes, they require additional supervision in the form of pure background images which are hard to obtain for real-world scenes. Schwarz et al. [77] propose Generative Neural Radiances Fields (GRAF). While achieving controllable image synthesis at high resolutions, this representation is restricted to single-object scenes and results degrade on more complex, real-world imagery. In contrast, we incorporate compositional 3D scene structure into the generative model such that it naturally handles multi-object scenes. Further, by integrating a neural rendering pipeline [20, 41, 42, 49, 62, 80, 81, 83, 84], our model scales to more complex, real-world data.

3. Method

Our goal is a controllable image synthesis pipeline which can be trained from raw image collections without additional supervision. In the following, we discuss the main components of our method. First, we model individual objects as neural feature fields (Sec. 3.1). Next, we exploit the additive property of feature fields to composite scenes from multiple individual objects (Sec. 3.2). For rendering, we explore an efficient combination of volume and neural rendering techniques (Sec. 3.3). Finally, we discuss how we train our model from raw image collections (Sec. 3.4). Fig. 3 contains an overview of our method.

3.1. Objects as Neural Feature Fields

Neural Radiance Fields: A radiance field is a continuous function $f$ which maps a 3D point $x \in \mathbb{R}^3$ and a viewing direction $d \in S^2$ to a volume density $\sigma \in \mathbb{R}^+$ and an RGB color value $c \in \mathbb{R}^3$. A key observation in [61, 82] is that the low dimensional input $x$ and $d$ needs to be mapped to higher-dimensional features to be able to represent complex signals when $f$ is parameterized with a neural network. More specifically, a pre-defined positional encoding is applied element-wise to each component of $x$ and $d$:

$$
\gamma(t, L) = \begin{pmatrix} 
\sin(2^0 t \pi), \cos(2^0 t \pi), \ldots, \sin(2^L t \pi), \cos(2^L t \pi) 
\end{pmatrix}
$$

where $t$ is a scalar input, e.g. a component of $x$ or $d$, and $L$ the number of frequency octaves. In the context of generative models, we observe an additional benefit of this representation: It introduces an inductive bias to learn 3D shape representations in canonical orientations which otherwise would be arbitrary (see Fig. 11).

Following implicit shape representations [12, 59, 69], Mildenhall et al. [61] propose to learn Neural Radiance Fields (NeRFs) by parameterizing $f$ with a multi-layer perceptron (MLP):

$$
f_\theta : \mathbb{R}^{L_x} \times \mathbb{R}^{L_d} \rightarrow \mathbb{R}^+ \times \mathbb{R}^3
$$

where $\theta$ indicate the network parameters and $L_x, L_d$ the output dimensionalities of the positional encodings.

Generative Neural Feature Fields: While [61] fits $\theta$ to multiple poses of an image, Schwarz et al. [77] propose a generative model for Neural Radiance Fields (GRAF) that is trained from unposed image collections. To learn a latent space of NeRFs, they condition the MLP on shape and appearance codes $z_s, z_a \sim \mathcal{N}(0, I)$:

$$
g_\theta : \mathbb{R}^{L_x} \times \mathbb{R}^{L_d} \times \mathbb{R}^{M_s} \times \mathbb{R}^{M_a} \rightarrow \mathbb{R}^+ \times \mathbb{R}^3
$$

where $M_s, M_a$ are the dimensionalities of the latent codes.

In this work we explore a more efficient combination of volume and neural rendering. We replace GRAF’s formulation for the three-dimensional color output $c$ with a more generic $M_f$-dimensional feature $f$ and represent objects as

Generative Neural Feature Fields:

$$
h_\theta : \mathbb{R}^{L_x} \times \mathbb{R}^{L_d} \times \mathbb{R}^{M_s} \times \mathbb{R}^{M_a} \rightarrow \mathbb{R}^+ \times \mathbb{R}^{M_f}
$$

Object Representation: A key limitation of NeRF and GRAF is that the entire scene is represented by a single model. As we are interested in disentangling different entities in the scene, we need control over the pose, shape and appearance of individual objects (we consider the background as an object as well). We therefore represent each object using a separate feature field in combination with an affine transformation

$$
T = \{s, t, R\}
$$

where $s, t \in \mathbb{R}^3$ indicate scale and translation parameters, and $R \in SO(3)$ a rotation matrix. Using this representation, we transform points from object to scene space as follows:

$$
k(x) = R \begin{bmatrix} s_1 \\
 s_2 \\
 s_3 
\end{bmatrix} \cdot x + t
$$

In practice, we volume render in scene space and evaluate the feature field in its canonical object space (see Fig. 1):

$$
(\sigma, f) = h_\theta(k^{-1}(x), \gamma(k^{-1}(d)), z_s, z_a)
$$

This allows us to arrange multiple objects in a scene. All object feature fields share their weights and $T$ is sampled from a dataset-dependent distribution (see Sec. 3.4).

3.2. Scene Compositions

As discussed above, we describe scenes as compositions of $N$ entities where the first $N-1$ are the objects in the scene and the last represents the background. We consider
Figure 3: GIRAFFE. Our generator $G_\theta$ takes a camera pose $\xi$ and $N$ shape and appearance codes $z^i, \xi^i$ and affine transformations $T_i$ as input and synthesizes an image of the generated scene which consists of $N - 1$ objects and a background. The discriminator $D_\phi$ takes the generated image $\hat{I}$ and the real image $I$ as input and our full model is trained with an adversarial loss. At test time, we can control the camera pose, the shape and appearance codes of the objects, and the objects’ poses in the scene. Orange indicates learnable and blue non-learnable operations.

Using numerical integration as in [61], $f$ is obtained as

$$f = \sum_{j=1}^{N_s} \tau_j \alpha_j f_j \quad \tau_j = \prod_{k=1}^{j-1} (1 - \alpha_k) \quad \alpha_j = 1 - e^{-\sigma_j \delta_j}$$

(10)

where $\tau_i$ is the transmittance, $\alpha_j$ the alpha value for $x_j$, and $\delta_j = ||x_{j+1} - x_j||_2$ the distance between neighboring sample points. The entire feature image is obtained by evaluating $\pi_{\text{vol}}$ at every pixel. For efficiency, we render feature images at resolution $16^2$ which is lower than the output resolution of $64^2$ or $256^2$ pixels. We then upsample the low-resolution feature maps to higher-resolution RGB images using 2D neural rendering. As evidenced by our experiments, this has two advantages: increased rendering speed and improved image quality.

2D Neural Rendering: The neural rendering operator

$$\pi_{\text{neural}} : \mathbb{R}^{H \times W \times M} \rightarrow \mathbb{R}^{H \times W \times 3}$$

(11)

with weights $\theta$ maps the feature image $I_{\text{vol}} \in \mathbb{R}^{H \times W \times M}$ to the final synthesized image $I \in \mathbb{R}^{H \times W \times 3}$. We parameterize $\pi_{\text{neural}}$ as a 2D convolutional neural network (CNN) with leaky ReLU [56, 89] activation (Fig. 4) and combine nearest neighbor upsampling with $3 \times 3$ convolutions to increase the spatial resolution. We choose small kernel sizes and no intermediate layers to only allow for spatially small refinements to avoid entangling global scene properties during image synthesis while at the same time allowing for increased output resolutions. Inspired by [40], we map the feature image to an RGB image at every spatial resolution, and add the previous output to the next via bilinear upsampling. These skip connections ensure a strong gradient flow to the feature fields. We obtain our final image prediction $\hat{I}$ by applying a sigmoid activation to the last RGB layer. We validate our design choices in an ablation study (Tab. 4).
The motivation for this choice is that in most real-world scenes, objects are arbitrarily rotated, but not tilted due to gravity. The observer (the camera in our case), in contrast, can freely change its elevation angle with respect to the scene.

For these entities in the scene, we parameterize the discriminator \(D_\phi\) as a CNN [73] with leaky ReLU activation.

During training, we sample the the number of entities in the scene \(N \sim p_N\), the latent codes \(z_i, z_a \sim N(0, I)\), as well as a camera pose \(\xi \sim p_\xi\) and object-level transformations \(T_i \sim p_T\). In practice, we define \(p_\xi\) and \(p_T\) as uniform distributions over dataset-dependent camera elevation angles and valid object transformations, respectively. The motivation for this choice is that in most real-world scenes, objects are arbitrarily rotated, but not tilted due to gravity. The observer (the camera in our case), in contrast, can freely change its elevation angle with respect to the scene.

We train our model with the non-saturating GAN objective [24] and \(R_1\) gradient penalty [58]

\[
V(\theta, \phi) = \mathbb{E}_z, \xi \sim N, T_i \sim p_T \left[ f(D_\phi(G_\theta(z^i, z^a, T_i, \xi))) \right] + \mathbb{E}_I \sim p_D \left[ f(-D_\phi(I)) - \lambda \| \nabla D_\phi(I) \|_2^2 \right]
\]

where \(f(t) = -\log(1 + \exp(-t))\), \(\lambda = 10\), and \(p_D\) indicates the data distribution.

3.5. Implementation Details

All object feature fields \(\{h^N_i\}_{i=1}^{N-1}\) share their weights and we parameterize them as MLPs with ReLU activations. We use 8 layers with a hidden dimension of 128 and a density and a feature head of dimensionality 1 and \(M_f = 128\), respectively. For the background feature field \(h^N_N\), we use half the layers and hidden dimension. We use \(L_X = 2 \cdot 3 \cdot 10\) and \(L_T = 2 \cdot 3 \cdot 4\) for the positional encodings. We sample \(M_s = 64\) points along each ray and render the feature image \(I_V\) at \(16^2\) pixels. We use an exponential moving average [93] with decay 0.999 for the weights of the generator. We use the RMSprop optimizer [85] with 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 set \(M_f = 256\) and half the generator learning rate to \(2.5 \times 10^{-4}\).

4. Experiments

Datasets: We report results on commonly used single-object datasets Chairs [68], Cats [95], Celeba [52], and CelebA-HQ [38]. The first consists of synthetic renderings of Photoshape chairs [70], and the others are image collections of cat and human faces, respectively. The data complexity is limited as the background is purely white or only takes up a small part of the image. We further report results on the more challenging single-object, real-world datasets CompCars [91], LSUN Churches [94], and FFHQ [39]. For CompCars, we randomly crop the images to achieve more variety of the object’s position in the image.\(^2\) For these datasets, disentangling objects is more complex as the object is not always in the center and the background is more cluttered and takes up a larger part of the image. To test our model on multi-object scenes, we use the script from [36] to render scenes with 2, 3, 4, or 5 random primitives (Clevr-N). To test our model on scenes with a varying number of objects, we also run our model on the union of them (Clevr-2345).

Baselines: We compare against voxel-based PlatonicGAN [32], BlockGAN [64], and HoloGAN [63], and radiance field-based GRAF [77] (see Sec. 2 for a discussion).

\(^1\)Details can be found in the supplementary material.

\(^2\)We do not apply random cropping for [32] and [77] as we find that they cannot handle scenes with non-centered objects (see supplementary).
suggests that our method disentangles objects of the methods. We further compare against HoloGAN and our method. We additionally report a ResNet-based 2D GAN for reference.

**Metrics:** We report the Frechet Inception Distance (FID) score [33] to quantify image quality. We use 20,000 real and fake samples to calculate the FID score.

### 4.1. Controllable Scene Generation

**Disentangled Scene Generation:** We first analyze to which degree our model learns to generate disentangled scene representations. In particular, we are interested if objects are disentangled from the background. Towards this goal, we exploit the fact that our composition operator is a simple addition operation (Eq. 8) and render individual components and object alpha maps (Eq. 10). Note that while we always render the feature image at $16^2$ during training, we can choose arbitrary resolutions at test time.

Fig. 5 suggests that our method disentangles objects from the background. Note that this disentanglement emerges without any supervision, and the model learns to generate plausible backgrounds without ever having seen a pure background image, implicitly solving an inpainting task. We further observe that our model correctly disentangles individual objects when trained on multi-object scenes with fixed or varying number of objects. We further find that unsupervised disentanglement is a property of our model which emerges already at the very beginning of training (Fig. 6).

We show renderings of our model on Clevr-2345 at 256$^2$ pixels after 0, 1, 2, 3, 10, and 100-thousand iterations. Unsupervised disentanglement emerges already at the very beginning of training.

**Generalization Beyond Training Data:** The learned compositional scene representations allow us to generalize outside the training distribution. For example, we can increase the translation ranges of objects or add more objects than there were present in the training data (Fig. 6). Note how our model synthesizes individual objects before spending capacity on representing the background.

**Controllable Scene Generation:** As individual components of the scene are correctly disentangled, we analyze how well they can be controlled. More specifically, we are interested if individual objects can be rotated and translated, but also how well shape and appearance can be controlled. In Fig. 7, we show examples in which we control the scene during image synthesis. We rotate individual objects, translate them in 3D space, or change the camera elevation. By modeling shape and appearance for each entity with a different latent code, we are further able to change the objects’ appearances without altering their shape.

**Network Parameter Comparison:** We report the number of generator network parameters in million.

<table>
<thead>
<tr>
<th>2D GAN</th>
<th>Plat. GAN</th>
<th>BlockGAN</th>
<th>HoloGAN</th>
<th>GRAF</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.69</td>
<td>381.56</td>
<td>4.44</td>
<td>7.80</td>
<td>0.68</td>
<td>0.41</td>
</tr>
</tbody>
</table>

### 4.2. Comparison to Baseline Methods

Comparing to baseline methods, our method achieves similar or better FID scores at both $64^2$ (Tab. 1) and $256^2$ (Tab. 2) pixel resolutions. Qualitatively, we observe that while all approaches allow for controllable image synthesis on datasets of limited complexity, results are less consistent for the baseline methods on more complex scenes.
Figure 7: **Controllable Scene Generation at 256^2 Pixel Resolution.** Controlling the generated scenes during image synthesis: Here we rotate or translate objects, change their appearances, and perform complex operations like circular translations.

Figure 8: **Generalization Beyond Training Data.** As individual objects are correctly disentangled, our model allows for generating out of distribution samples at test time. For example, we can increase the translation ranges or add more objects than there were present in the training data.

Figure 9: **Qualitative Comparison.** Compared to baseline methods, we achieve more consistent image synthesis for complex scenes with cluttered background at 64^2 (top rows) and 256^2 (bottom rows) pixel resolutions. Note that we disentangle the object from the background and are able to rotate only the object while keeping the background fixed.

With cluttered backgrounds. Further, our model disentangles the object from the background, such that we are able to control the object independent of the background (Fig. 9).

We further note that our model achieves similar or better FID scores than the ResNet-based 2D GAN [58] despite fewer network parameters (0.41m compared to 1.69m).

This confirms our initial hypothesis that using a 3D representation as inductive bias results in better outputs. Note that for fair comparison, we only report methods which are
Table 4: **Ablation Study.** We report FID (\( \downarrow \)) on CompCars without RGB skip connections (-Skip), without final activation (-Act.), with nearest neighbor instead of bilinear image upsampling (+ NN. RGB Up.), and with bilinear instead of nearest neighbor feature upsampling (+Bi. Feat. Up.).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16.16</td>
<td>16.66</td>
<td>21.61</td>
<td>17.28</td>
<td>20.68</td>
</tr>
</tbody>
</table>

Figure 10: **Neural Renderer.** We change the background while keeping the foreground object fixed for our method at 256\(^2\) pixel resolution. Note how the neural renderer realistically adapts the objects’ appearances to the background.

(a) 0° Rotation for Axis-Aligned Positional Encoding [61]

(b) 0° Rotation for Random Fourier Features [82]

Figure 11: **Canonical Pose.** In contrast to random Fourier features [82], axis-aligned positional encoding (1) encourages the model to learn objects in a canonical pose.

Figure 12: **Dataset Bias.** Eye and hair rotation are examples for dataset biases: They primarily face the camera, and our model tends to entangle them with the object rotation.

4.3. Ablation Studies

**Importance of Individual Components:** The ablation study in Tab. 4 shows that our design choices of RGB skip connections, final activation function, and selected upsampling types improve results and lead to higher FID scores.

**Effect of Neural Renderer:** A key difference to [77] is that we combine volume with neural rendering. The quantitative (Tab. 1 and 2) and qualitative comparisons (Fig. 9) indicate that our approach leads to better results, in particular for complex, real-world data. Our model is more expressive and can better handle the complexity of real scenes, e.g. note how the neural renderer realistically adapts object appearances to the background (Fig. 10). Further, we observe a rendering speedup: compared to [77], total rendering time is reduced from 110.1ms to 4.8ms, and from 1595.0ms to 5.9ms for 64\(^2\) and 256\(^2\) pixels, respectively.

**Positional Encoding:** We use axis-aligned positional encoding for the input point and viewing direction (Eq. 1). Surprisingly, this encourages the model to learn canonical representations as it introduces a bias to align the object axes with highest symmetry with the canonical axes which allows the model to exploit object symmetry (Fig. 11).

4.4. Limitations

**Dataset Bias:** Our method struggles to disentangle factors of variation if there is an inherent bias in the data. We show an example in Fig. 12: In the celebA-HQ dataset, the eye and hair orientation is predominantly pointing towards the camera, regardless of the face rotation. When rotating the object, the eyes and hair in our generated images do not stay fixed but are adjusted to meet the dataset bias.

**Object Transformation Distributions:** We sometimes observe disentanglement failures, e.g. for Churches where the background contains a church, or for CompCars where the foreground contains background elements (see Sup. Mat.). We attribute these to mismatches between the assumed uniform distributions over camera poses and object-level transformations and their real distributions.

5. Conclusion

We present GIRAFFE, a novel method for controllable image synthesis. Our key idea is to incorporate a compositional 3D scene representation into the generative model. By representing scenes as compositional generative neural feature fields, we disentangle individual objects from the background as well as their shape and appearance without explicit supervision. Combining this with a neural renderer yields fast and controllable image synthesis. In the future, we plan to investigate how the distributions over object-level transformations and camera poses can be learned from data. Further, incorporating supervision which is easy to obtain, e.g. predicted object masks, is a promising approach to scale to more complex, multi-object scenes.

Acknowledgement

This work was supported by an NVIDIA research gift. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting MN. AG was supported by the ERC Starting Grant LEGO-3D (850533) and DFG EXC number 2064/1 - project number 390727645.
References


In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2020. 1, 2


[40] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 4


[55] Sebastian Lunz, Yingzhen Li, Andrew W. Fitzgibbon, and Nate Kushman. Inverse graphics GAN: learning to generate 3d shapes from unstructured 2d data. *arXiv.org*, 2020. 2


[82] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ra-


