3INGAN: Learning a 3D Generative Model from Images of a Self-similar Scene

Animesh Karnewar
1 University College London
Oliver Wang
2 Adobe Research
Tobias Ritschel
1 University College London
Niloy J. Mitra
1, 2 University College London

Abstract

We introduce 3INGAN, an unconditional 3D generative model trained from 2D images of a single self-similar 3D scene. Such a model can be used to produce 3D “remixes” of a given scene, by mapping spatial latent codes into a 3D volumetric representation, which can subsequently be rendered from arbitrary views using physically based volume rendering. By construction, the generated scenes remain view-consistent across arbitrary camera configurations, without any flickering or spatio-temporal artifacts. During training, we employ a combination of 2D, obtained through differentiable volume tracing, and 3D Generative Adversarial Network (GAN) losses, across multiple scales, enforcing realism on both its 2D renderings and its 3D structure. We show results on semi-stochastic scenes of varying scale and complexity, obtained from real and synthetic sources. We demonstrate, for the first time, the feasibility of learning plausible view-consistent 3D scene variations from a single exemplar scene and provide qualitative and quantitative comparisons against two recent related methods. Code and data for the paper are available at https://geometry.cs.ucl.ac.uk/group_website/projects/2022/3inGAN/.

1. Introduction

In the context of images, unconditional generative models, such as GANs, learn to map latent spaces to diverse yet realistic high resolution images – notable architectures include StyleGan [36] and BigGAN [9]. Furthermore, these models have been shown to contain high-level semantics in their latent space mappings, allowing powerful post-hoc image editing operations, such as changing the appearance and expression of a generated person [1, 56, 22]. One key question therefore, is whether it is possible to learn similar generative models for 3D scenes.

There are, however, two key challenges. First, 3D generation suffers from data scarcity as obtaining large and diverse datasets for 3D data, both geometry and appearance, is significantly more challenging than for 2D data, where one can simply scrape images from the internet. Second, generative models struggle as the domain complexity increases, as well as when datasets are not prealigned. This problem is even more severe in 3D, due to the added scene complexity, both in terms of scene structure and object appearance, and the fact that 3D models and scans often come with their own coordinate systems and/or scaling.

In this work, we propose 3INGAN, a solution to address both problems in a restrictive setting: an unconditional generative model for 3D scenes that works on a per-scene basis for
self-similar configurations. As our method does not require a large 3D dataset during training, it could be used across a wide range of real world domains. By restricting the data domain to a single 3D scene, we simplify the unconditional generation problem space to one with limited domain complexity, allowing us to learn a high-quality generator. We were inspired by similar approaches proposed for 2D images, e.g., SinGAN [55]. However, extending such approaches to 3D is nontrivial, as in 3D, one must be able to generate arbitrary views in a way that is multi-view consistent. We handle this by directly generating a 3D scene representation that is, by definition, multiview consistent. In particular, we generate a regularly sampled nonlinearly-interpolated-voxel grid [31] due to its simplicity, local (feature) influence, rendering efficiency, and its amenability to the prevailing convolutional generator and discriminator architectures in 2D/3D. In order to achieve realism, both in terms of 3D structure and 2D image appearance, we simultaneously use 3D feature patches and 2D image patches to obtain gradients from the 3D and 2D discriminators, respectively. We link the 3D and 2D domains by using a differentiable volume rendering module to interpret the feature grid as RGB images. Note that recent neural 3D generators (e.g., BlockGAN [48], GRAF [54], π-GAN [10]) are trained on image/shape collections, and are not easily applicable in our setup (see Section 4 for comparison). For example, Fig. 1 shows a sampling of “remixed” results generated from images of a school of fish.

In summary, ours is the first work to introduce an unconditional generative model from a single 3D scene. In particular, we investigate scenes with some degree of stochastic structure, which are suitable to shuffling or “remixing” the scene content into a new 3D scene that makes sense. In addition, we make a further simplifying assumption that we drop the view-specific effects and reconstruct Lambertian scenes. We evaluate our proposed method on a series of synthetic and real scenes and show that our approach outperforms baselines in terms of quality and diversity.

2. Related Work

2D generative models. Generative modeling for image synthesis learns a distribution of colour values over the pixels of an image, and has seen tremendous progress recently, with GANs being the de-facto standard for synthesizing realistic looking images [33, 36, 37, 9, 32]. Recent works such as [34] further improve result quality in data-sparse regimes, while [35] applies signal processing techniques to the generator architecture to correct for aliasing errors. Other contenders for generative modelling include VAEs [60, 2, 40, 26], flow-based models [39, 15, 14], noise-diffusion based models [57, 58, 29, 4], and even hybrids of these methods [51, 4, 18, 41]. Key to training such methods is the availability of large scale image collections, and often times such works are used on specific target domains, such as portrait photos.

More similar to ours, Single Image GANs (SinGANs) [55, 27] learn to generate the distribution of patches of a single image, in a progressive coarse-to-fine manner so as to generate plausible variations from that one single image. Such approaches avoid the problem of needing a large dataset, while still enabling useful applications such as retargetting. However, they are restricted to repeated, or stochastic-like patch-based variations. Ours shares similar advantages, as well as restrictions in terms of scene type as SinGAN. In Section 4 we show that a naïve extension to 3D volumes does not produce reasonable results.

3D generative models. Due to the lack of large scale real-world datasets, much of the research in 3D generative modeling has stayed in the synthetic realm such as modeling only 3D shapes [62, 45, 8, 7, 12, 16], or materials [21]. Other methods use synthetic datasets for predicting scene structure and use differentiable rendering for end-to-end training [66, 38, 50, 30]. Recently, methods that directly model 3D scenes, either using explicit or implicit neural scene representations, have been gaining popularity [25, 47, 10, 54, 49, 13, 67, 19]. Another successful line of work [47, 49] uses a neural renderer to render features from a volumetric grid, followed by per-image 2D CNNs used for upsampling, and as such are not multiview consistent. In contrast, [10, 54] and [13] are view-consistent by design since they use an implicit neural representation and explicit-implicit neural representation respectively for the 3D structure, using a physically based rendering equation for obtaining the final 2D images. Concurrent works such as [67, 19] produce impressive results performing 3D synthesis with multi-view consistency. Such approaches are designed to be trained only through 2D (image) supervision, and work best in cases where large datasets can be obtained for limited domains, such as faces or cars. Our method is also 3D view-consistent by design, and can be applied on a long-tail of real world scenes, provided they are self-similar.

Differentiable rendering. Differentiable rendering [59] enables neural networks to be trained by losses on the resulting rendered images of a 3D representation, by allowing the gradient to be back-propagated through the network. This has shown to be a highly effective tool, especially for learning 3D representations that allow for novel view synthesis. Multiple works [28, 46, 3, 61, 43, 68] have focused on designing neural-networks, either convolutional or otherwise, to go from spatial features to rendered pixels. Lately, methods that use volume tracing have been favored due to the advantage of flicker-free 2D rendering by design. Neural Radiance Fields (NeRFs) [44] were the first to introduce this way of using differentiable rendering while using a neural 3D scene representation. Subsequently, many extensions of this have been proposed, to increase quality, robustness to scene
Figure 2: **3INGAN setup.** Overview of our approach with two parts: an initialization of a reference 3D feature grid (top) and a stage-wise learning of a generative model (bottom). Input to the system is a set of 2D images seen on the top left. From these, optimization using differentiable rendering for known views produces the reference feature grid, which is the input to the next step. The rows below (“Level”) denote levels of training the generator, a 2D discriminator, and a 3D discriminator. The 3D discriminator (right) gets random 3D patches from the reference or generated 3D grid, while the 2D discriminator (right) gets random 2D patches from reference or from generated renderings.

3. Our Approach

**Motivation.** As input, we require an exemplar self-similar scene, which is provided as a set of posed 2D images. We desire that our method produces a plausible 3D scene structure that matches the exemplar scene on a patch basis, and realistic 2D image renderings that are consistent across the space of all views of that scene.

Given this goal, we choose to directly generate a 3D representation, so that we are guaranteed view consistency when rendering 2D images from it. The first question is, **what 3D representation should we use?** One choice would be coordinate-based MLPs, which have been shown to be compact scene representations able to generate very high quality novel views [44]. However, such a representation is ill-suited for a generative setup, as the costly evaluation makes rendering volumetric and image patches in the training loop infeasible, and the global and distributed nature of the MLP representation makes GAN training challenging in our patch-based setting. Another option is to operate directly on discretized RGBA volumes but, as demonstrated in Karnewar et al. [31], such an approach results in limited quality (blurry) results. Instead, we build our approach using their proposed grid-based ReLU Field representation. This representation is detailed in brief in section 3.1.

A naïve extension of SinGAN [55] to 3D fails to produce good results for multiple reasons: (i) only using a 3D discriminator doesn’t have the notion of free-space in the volume and also tries to replicate the inside regions of the occupied space that may contain random values. When such a model is rendered, the results have minor shape distortions and chromatic noise; (ii) only using 2D discriminators on the rendered images is challenging. Specifically, when trained on too small patches the discriminator becomes too weak to inform the generator, and when trained on too large patches, the generator simply memorizes the initial reconstruction. Our solution involves two main ingredients: the
Algorithm 1 Our 3tINGAN training. Function sample(...) samples all distributions provided as arguments; up(a, b) ADAM-updates the parameters a by the gradient of expression b with respect to a.

\begin{algorithm}
\caption{3tINGAN training.}
\begin{algorithmic}[1]
\STATE 1: repeat \Comment{3D representation building}
\STATE 2: \hspace{1em} \{I, C\} := sample(I, C)
\STATE 3: \hspace{1em} V := up(V, \|R(V, C) - I\|_2^2)
\STATE 4: \hspace{1em} until converged.
\STATE 5: repeat \Comment{Generator training}
\STATE 6: \hspace{2em} \{z, C\} := sample(N, D)
\STATE 7: \hspace{2em} P := P_{2D}(R(G^0(z), C))
\STATE 8: \hspace{2em} \phi := up(\phi, D_{2D}^\phi(P = \text{fake}))
\STATE 9: \hspace{2em} C := sample(D)
\STATE 10: \hspace{2em} P := P_{2D}(R(V, C))
\STATE 11: \hspace{2em} \phi := up(\phi, D_{2D}^\phi(P = \text{real}))
\STATE 12: \hspace{2em} z := sample(N)
\STATE 13: \hspace{2em} P := P_{3D}(G^0(z))
\STATE 14: \hspace{2em} \psi := up(\psi, D_{3D}^\psi(P = \text{fake}))
\STATE 15: \hspace{2em} P := P_{3D}(V)
\STATE 16: \hspace{2em} \psi := up(\psi, D_{3D}^\psi(P = \text{real}))
\STATE 17: \hspace{2em} \{z, C\} := sample(N, D)
\STATE 18: \hspace{2em} \theta := up(\theta, D_{3D}^\theta(P_{2D}(R(G^0(z), C)) = \text{real}))
\STATE 19: \hspace{2em} \theta := up(\theta, D_{3D}^\theta(P_{3D}(G^0(z)) = \text{real}))
\STATE 20: until converged.
\end{algorithmic}
\end{algorithm}

The use of ReLU-Fields representation [31] instead of the standard RGBA volumes for inherently inducing the notion of free space in the 3D-grid; and a mixture of 2D and 3D discriminators, along with multi-scale training, that robustly produce consistent and high quality reconstructions.

**Method overview.** Input to our method is a set of \( n_{2D} \) 2D images \( I := \{I_1, \ldots, I_{n_{2D}}\} \) taken from one real world or synthetic self-similar scene. Fig. 2 presents a graphical overview and Alg. 1 a pseudo-code version. Our method consists of two main steps. First, we convert the 2D image set into a 3D feature grid \( V \) (Sec. 3.1). Then, we train a generative model \( G \) of 3D scenes from this 3D feature grid and its 2D rendered images (Sec. 3.2). This generative model \( G(z) \) then converts spatial random latent grids \( z \) into 3D feature grids containing remixes of the exemplar scene, which can be consistently rendered from arbitrary views.

**3.1. Representation**

**Foreground ReLU Field [31].** Foreground is bound by a user-provided Axis-Aligned Bounding Box (AABB) covered by a volumetric grid, \( V \), of fixed resolution \( n_{V_x} \times n_{V_y} \times n_{V_z} \) to contain feature values in the \([-1, 1]\) range. These values correspond to the raw-features that are stored on the voxel-grid. In order to obtain continuous density field, the trilinearly interpolated values of these raw features are passed through a single channel-wise Rectified Linear Unit (ReLU) to convert them to \([0, 1]\) range which can be physically interpreted as the density values. We do not model view-dependent appearance, \( i.e., \), we approximate the scene with Lambertian materials.

**Background.** The background is assumed to be constant black for synthetic scenes. For real scenes, we model the background of the scene using an implicit neural network \( B \), similar to NeRF++ [65], but without using the inverted-sphere parametrization of the scene. Our goal is not to model the entire scene perfectly, but rather to provide appropriate inductive bias to the reconstruction pipeline to do the foreground-background separation correctly. This allows us to reconstruct real-world scenes without the requirement of additional segmentation masks.

**Optimization.** Let the camera pose (extrinsic translation and rotation as well as intrinsics) for each input image be \( C := \{C_1, \ldots, C_{n_{2D}}\} \) and assume they are known, \( e.g., \), by using structure-from-motion (we use ColMap [53]). Further, we denote the rendering operation to convert the feature grid \( V \) and the camera pose \( C \) into an image as \( R(V, C) \).

Specifically, we use emission-absorption raymarching [42, 25]. See supplementary for more details for the rendering. We can then directly optimize the feature grid’s photometric loss, given the pose and the 2D images, as

\[
\arg\min_V \sum_{i=1}^{n_{2D}} \|I_i - R(V, C_i)\|_2^2. 
\]

We minimize with batched optimization over 2048 random rays out of all the rays for which we know the 2D input image pixel value. Input images are of size \( 512 \times 512 \). Further, instead of directly optimizing for the full-resolution volume \( V \), training proceeds progressively in a coarse-to-fine manner. Initially, the feature grid is optimized at a resolution where each dimension is smaller by factor 16. After seeing 20k batches of input rays, the feature grid resolution is multiplied by two and the feature grid tri-linearly upsampled.

**3.2. Generation**

Training the generative model makes use of the 3D feature grid \( V \) trained in the previous section 3.1, which we denote as the reference grid herein. We look into the generator details first, before explaining the losses used to train it: 2D and 3D discriminators, and a 2D and 3D reconstruction loss.

**Generator.** Recall, that the model \( G \) maps random latent codes \( z \) to a 3D feature grid \( G(z) \) at the coarsest stage. While adds fine residual details to previous stage’s outputs at the rest of the stages similar to SinGAN [55]. The generator is a 3D CNN that stagewise decodes a spatial grid of noise
vectors $z$ of size $n_z$ (we use $seedDimension = 4$) into the grid of the desired resolution.

**Training.** We train the architecture progressively: the generator first produces grids of reduced resolution. Only once this has converged, layers are added and the model is trained to produce the higher resolution. Note that we freeze the previously trained layers in order to avoid the GAN training from diverging. We employ an additional reconstruction loss that enforces one single fixed seed $z^*$ to map to the reference grid. We supervise this fixed seed loss via an MSE over the 3D grids and with 2D rendered patches.

**2D discriminator.** A 2D loss discriminates 2D patches of renderings of the generated feature grid to 2D patches rendered from the feature grid $V$. To render the 3D grids we need to model another distribution of poses, denoted by $D$, that uniformly samples camera locations to point at the center of the hemisphere and where focal length is varied stagewise linearly, where the value at the final stage corresponds to the actual camera intrinsics.

Further, let $P_{2D}()$ be an operator to extract a random patch from a 2D image, with discriminators,

$$ p^{2D}_F = P_{2D}(R(G(z), D)) \quad \text{and} \quad p^{2D}_R = P_{2D}(R(V, D)). \quad (2) $$

Note, that we did not define $p^{2D}_R = P_{2D}(I)$, as this would limit ourselves to use real samples only from the limited set of known 2D image patches. “Trusting” our reference 3D feature grid has been extracted properly, we can instead sample it from arbitrary views and get a much richer set.

**3D discriminator.** The 3D discriminator compares 3D patches from the generated feature grid to 3D patches of the reference feature grid. Let $P_{3D}(V)$ be an operator to extract a random patch from a 3D feature grid $V$. The distributions to discriminate are,

$$ p^{3D}_F = P_{3D}(G(z)) \quad \text{and} \quad p^{3D}_R = P_{3D}(V). \quad (3) $$

Finally, we use Wasserstein GAN [5] to both distributions as well as the reconstruction losses (in both 2D and 3D),

$$ \mathcal{L} = \gamma_{2D} \cdot \text{wgan}(p_{R}^{3D}, p_{F}^{3D}) + \gamma_{3D} \cdot \text{wgan}(p_{R}^{2D}, p_{F}^{2D}) + \rho_{2D} \cdot \mathbb{E}_{C \in D}[\|R(G(z^*), C) - R(V, C)\|_{2}^{2}] + \rho_{3D} \cdot \|G(z^*) - V\|_{2}^{2}, \quad (4) $$

weighted by two pairs of two factors $\gamma_{2D}$, $\gamma_{3D}$ and $\rho_{2D}$, $\rho_{3D}$. In practice, we use $\gamma_{2D} = \gamma_{3D} = 1.0$ and as well as $\rho_{2D} = \rho_{3D} = 10$.

4. Evaluation

We perform quantitative and qualitative evaluations of our approach on a number of different scenes, on which we compare to baseline methods, and evaluate design choices via an ablation study.

| Table 1: Comparisons and ablations. We enumerate the different methods based on how they make use of 2D versus 3D information, and if they operate on a single scene or multiple scenes. |
| --- | --- | --- | --- | --- |
| Disc. | Recon. | Single scene |
| 2D | 3D | 2D | 3D | |
| PiGAN [10] | ✓ | ✓ | ✓ | ✓ | ✓ |
| Graf [54] | ✓ | ✓ | ✓ | ✓ | ✓ |
| OursPlatoGAN | ✓ | ✓ | ✓ | ✓ | ✓ |
| OursSinGAN3D | ✓ | ✓ | ✓ | ✓ | ✓ |
| Ours | ✓ | ✓ | ✓ | ✓ | ✓ |

**Comparisons and ablations.** We compare to four methods (Tab. 1). Besides our full method SinGAN (Ours), we study two prior approaches and two ablations.

As there is no existing method for 3D single scene remixing, we instead compare to two recent methods that were designed to learn a 3D generative model for classes of objects, trained on a dataset of images/renderings where each image corresponds to a different instance and view, PiGAN [10] and Graf [54]. In these baselines, we test how well such methods work when given instead, many rendered views of a single scene. In both cases, we use the code provided by the authors and their recommended parameter settings. We mark whether the original approach was designed for a single scene or not in the last column in Tab. 1.

In the first ablation, our 2D only ablation, we evaluate the importance of the 3D discriminator and 3D reconstruction seed losses by running a version of our method with those losses removed, i.e. ($\rho_{3D} = \gamma_{3D} = 0$ in Eq. 4). We refer to this method as OursPlatoGAN, as its GAN loss setting, which is only on the rendered 2D images, is similar to the setup used in Henzler et al. [25], while being applied on only a single scene.

In the second ablation, our 3D only ablation, we evaluate the importance of the 2D discriminator and 2D reconstruction seed loss, which we refer to as OursSinGAN3D. This approach is a na"ıve extension of SinGAN [55] to 3D. In other words, it is our approach without any differentiable rendering, i.e., with the 2D discriminator and 2D reconstruction seed losses removed ($\gamma_{2D} = \rho_{2D} = 0$). Please see the supplementary material for further ablations of the reconstruction seed losses.

**Scenes.** We consider a mix of synthetic and real scenes, with various levels of stochasticity (a requirement for patch-based remixing of scenes). These scenes include a synthetic scene rendered from Blender showing fishes with the same orientation (FISH) as well as with random orientations (FISHROT), a scene composed of 3D balloons (BALLOONS), inspired by [55]. We also use four semi-synthetic scenes that were real scenes reconstructed from
images using photogrammetry and then cleaned by an artist and sold on SketchFab: a pile of dirt (DIRT), a log pile (LOGS), and a bush (PLANTS). We also make a fully synthetic (FOREST) scene which has a ground plane so as to resemble most real-world settings. Finally, we include two real scenes with background for which we have no ground truth 3D available, which both show a random arrangement of geometric toys (BLOCKS) or pieces of colored chalks (CHALK). Note that in all the cases, regardless of the source, our method only accesses 2D renderings/images of the scene, not the 3D scene.

**Evaluation metric.** We evaluate our method along two axes – *visual quality*, and *scene diversity*. With traditional 2D GANs, these are most commonly evaluated using Frechet Inception Distance (FID). However, this metric is typically used over two datasets of images, whereas our situation is slightly different; we have only one ground truth scene, and a diverse distribution of generated scenes. We extend Single Image FID (SIFID) [55] to 3D, and also explicitly separate quality and diversity to separately compare along each axis.

*Visual quality* is measured as the expectation of SIFID scores between the distribution of exemplar 2D images and the distribution of rendered generated 2D images for a fixed camera over multiple seeds. We compute this expectation by taking a mean over a number of camera-views. This is similar in sprit to SIFID, except we compute it over images rendered from different views of the 3D scenes. Lower distance reflects better quality.

Unfortunately, in the single scene case, we cannot meaningfully compute FID scores between the exemplar patch distribution and the distribution of all generated patches across seeds as it would make diversity appear as a distribution error. This is different from a typical GAN setup where we are given real images as samplings of desirable distribution of both quality and diversity. Instead, we measure *scene diversity* as the variance of a fixed patch from a fixed view over random seeds, which is a technique used to study texture synthesis diversity [24]. Larger diversity is better.

### 4.1. Qualitative Evaluation

In Fig. 5 and Fig 6, we present qualitative results of 3INGAN, prior methods, and ablations. Please see the supplementary material for video results that can better visually demonstrate the quality and diversity of reconstructed scenes. We can see that, by construction, our method produces variations of 3D scenes that are view consistent.
Table 2: Quality versus diversity. A good generative model should have a good mix of quality and diversity – excellent quality with no diversity or vice versa are both undesirable. Visual Quality and Scene Diversity for different methods (columns) and different data sets (rows). To simplify comparison, we normalize the numbers so that ours is always 1. The best for each metric on each dataset is **bolded** and second best is *underlined*. Please refer to the supplementary for unscaled numbers.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FISH</td>
<td>15.74</td>
<td>0.261</td>
<td>264.27</td>
<td>0.359</td>
<td>465.47</td>
</tr>
<tr>
<td>FISHR</td>
<td>2.47</td>
<td>0.440</td>
<td>3.07</td>
<td><strong>1.018</strong></td>
<td>46.02</td>
</tr>
<tr>
<td>BALLOONS</td>
<td><strong>1.44</strong></td>
<td><strong>0.024</strong></td>
<td>3.25</td>
<td>0.477</td>
<td>1.79</td>
</tr>
<tr>
<td>DIRT</td>
<td>0.96</td>
<td>0.264</td>
<td>1.69</td>
<td><strong>0.440</strong></td>
<td><strong>0.66</strong></td>
</tr>
<tr>
<td>FOREST</td>
<td>1.33</td>
<td>0.473</td>
<td>1.69</td>
<td>0.801</td>
<td><strong>0.70</strong></td>
</tr>
<tr>
<td>PLANTS</td>
<td>5.99</td>
<td>0.261</td>
<td>6.56</td>
<td>0.648</td>
<td>12.81</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>0.91</td>
<td>0.224</td>
<td><strong>0.73</strong></td>
<td><strong>0.342</strong></td>
<td>1.83</td>
</tr>
<tr>
<td>CHALK</td>
<td>0.02</td>
<td>0.061</td>
<td><strong>0.01</strong></td>
<td><strong>0.320</strong></td>
<td>0.54</td>
</tr>
</tbody>
</table>

Figure 5: Qualitative comparison. Comparison of visual quality for different methods (columns) for different scenes (rows).

Scene retargeting. In Fig. 7 we show results of changing the aspect ratio of generated scenes. As our generator is fully convolutional, this can be done simply by changing the shape of the input noise. We can see that the scene structure remains plausible, with the semi-stochastic content repeated to fill in the space.

4.2. Quantitative Evaluation

In Tab. 2, we quantitatively compare the different variations using our proposed metrics. Mirroring the qualitative observations, we can see that 3INGAN performs better than prior work and ablations in terms of visual quality. The numbers show the importance of having both the 2D and 3D discriminators as they help to improve the object appearance and geometric structure, respectively, of the generations. Our main advantage is the noticeable boost in diversity of the generated results, as also measured in the 2–5× boost in scene diversity score over the other methods. Note that as indicated by the relative Visual Quality scores, our results are not yet close to being photorealistic (indistinguishable from the reference distribution) but nonetheless we achieve a healthy gain in over quality/diversity over competing alternatives.

5. Discussion

Limitations and future work. Our approach has a number of limitations. For one, we note that the proposed method of generating a distribution (i.e., a generative model) from a single scene only makes sense in case where scenes contain stochastic structures that can be shuffled around. We do not expect our method to work on highly structured content, e.g., people. Furthermore, we observe that results can still exhibit...
artifacts, such as high frequency noise, blobby output, and floaters. We believe that this is because the scene distribution is being estimated from very limited data (i.e., a single scene), and is analogous to the “splotchiness” commonly seen in 2D GANs when trained in limited data settings or over complex distributions. In this case, it should be possible to improve quality by using data augmentation as recommended in [34], or by improving the training regime, as in [27], however in the latter case it is important to balance diversity to avoid mode collapse. Finally, in our implementation, we assume scenes to be Lambertian, i.e., without view-dependent effects. In the future, view-dependent specular effects could be modelled using SH components.

Conclusion. We have presented 3iNGAN, a method for training a generative model of remixes of a single 3D scene. Similar to SinGAN, our approach works on “long-tail” data, as it does not require aligned 3D datasets for training. Instead, we require only a set of posed images of a single self-similar scene as input. We believe that this method has a number of downstream applications, such as retargeting or harmonization [55], and furthermore, studies in generative models for 3D scenes may, one day, lead to realistic view-consistent scene generation on par with images. This would be practical not only for 3D content generation, but as a generative prior to be used in many 3D reconstruction and editing tasks.

Acknowledgements. The work was partially supported by the Marie Skłodowska-Curie grant agreement No. 956585, gifts from Adobe, and the UCL AI Centre.
References


[26] Irina Higgins, Loic Matthey, Arkia Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016. 2


[33] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability,


