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

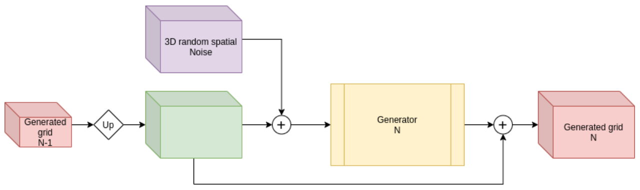

Figure 1: Architecture of the Generator G^N at a stage 'N'.

In this supplementary, we provide additional details about the architectures used for the Generator, the Discriminator, and about the stage-wise training schedules used in our method. We also provide additional ablation experiments on the use of the fixed-seed reconstruction noise, as mentioned in the main paper.

Finally, we plan to make the code, pre-trained models and the newly acquired 3D scenes available, under appropriate licences, to the research community.

1. Architecture Details

Generator. We employ a residual generator architecture similar to the one used by SinGAN [7] in the image case. As shown in Fig. 1, the input to the generator at any particular stage N is the output of the previous stage $(N-1)$, upsampled appropriately (in order to match the size at current stage exactly) and perturbed using 3D spatial random noise sampled from a standard Gaussian distribution $\mathcal{N}(0, I)$. The N-th stage generator G^N then models residual details that are added back to the upsampled output of previous stage $N-1$. For stage $N=1$, we consider the output of the previous stage to be *zero*, thereby making the generation purely from the random noise. The generator itself consists of 5 convolution layer blocks, each containing a *Conv* operation followed by a *BatchNorm* and a *LeakyReLU*. We pad the input to the generator before inputting so that the *Conv*s in the generator do not have to apply any additional padding.

Discriminator. We employ a combination of two discriminators: (a) the 3D feature patches D^N_{3D} and (b) the 2D image patches D^N_{2D} in our full method `Ours`.

As shown in Fig. 2 (a), the 3D discriminator D^N_{3D} for a certain stage N is a 3D convolutional block, which has a receptive field of $(11 \times 11 \times 11)$ and outputs a *real/fake* score for each 3D-feature patch extracted from all possible overlapping 3D feature patches in either the reference-grid or the generated grid.

As shown in Fig. 2 (b), for the 2D discriminator, we first render random views of the reference-grid or the generated-grid for camera poses sampled on the top-hemisphere surrounding the grid such that each camera always points to the center of the hemisphere which coincides with the center of the grid. Subsequently, we extract a random set of (11×11) patches from these views as input to the 2D discriminator D^N_{2D}. For efficiency, we *only* render the rays for which random patches are to be extracted instead of rendering the full image in order to avoid unnecessary computation. The 3D discriminator D^N_{3D} mirrors the architecture of the generator G^N while the 2D discriminator also has the same architecture containing 2D *Conv* operations instead of 3D.

2. Training Details

Most of our input 3D scenes are represented on a grid size of $(128 \times 128 \times 64)$. Hence, in order to strike a good balance between quality (realism) and diversity in the "remix"ed scenes, we train two versions of our full model `Ours` and `Ours_6`. The `Ours` version has 7 stages where each stage is upsampled by a factor of 1.33, similar to [7], in order to allow modelling of the frequency content at all stages of the pyramid. In contrast, the version `Ours_6` has 6 stages which improves the diversity at the cost of a slight loss in perceptual realism of the "remix"ed versions of the scene. Please refer to the supplementary webpage to explore this *diversity versus quality tradeoff*. We refer to (sec. 4) for visual results.

Training proceeds from the coarsest stage (lowest grid resolution) to the the finest stage (highest grid resolution). Once a stage is complete, we freeze the weights of the generator so that the subsequent training is less likely to diverge. Note that we tried lowering the learning rate at the earlier stages, similar to Hinz et al. [3], but still observed divergence in training, and hence decided to use the freezing approach. We do note that the freezing approaches requires to use bigger (more features) generator models at higher resolutions in order to model the high-frequency content and flag it as a

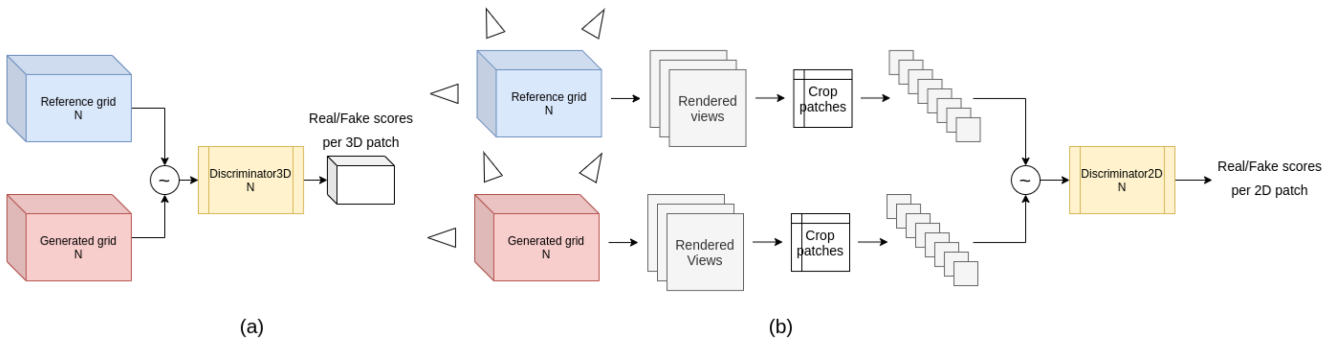

Figure 2: Architecture of the (a) 3D Discriminator D^N_{3D} and (b) the 2D Discriminator D^N_{2D}.

Table 1: Raw unscaled values for the Quality Metric computed for all 8 scenes for all 8 methods under comparison.

	Fish	FishR	Balloons	Dirt	Forest	Plants	Blocks	Chalk
PiGAN	1.04E-05	5.92E-06	1.07E-05	2.39E-05	7.86E-06	1.89E-05	2.16E-05	1.74E-06
Graf	1.74E-04	7.36E-06	2.41E-05	4.22E-05	9.97E-06	2.07E-05	1.73E-05	9.19E-07
OursPlatoGAN	3.06E-04	1.10E-04	1.32E-05	1.64E-05	4.15E-06	4.05E-05	4.32E-05	4.59E-05
OursSinGAN3D	5.91E-06	6.49E-06	7.08E-05	1.51E-04	1.26E-05	2.89E-05	7.84E-05	7.28E-05
OursPlatonicNoRecon	6.90E-05	1.71E-04	1.05E-04	3.73E-05	8.38E-05	5.20E-05	3.41E-05	5.16E-05
OursSinGan3DNoRecon	7.10E-05	5.12E-05	1.06E-05	4.47E-05	8.90E-05	1.11E-04	4.19E-05	3.65E-04
OursNoRecon	1.22E-04	1.44E-04	8.52E-05	1.28E-05	1.70E-04	1.75E-05	3.01E-05	2.16E-04
Ours	6.58E-07	2.40E-06	7.40E-06	2.49E-05	5.90E-06	3.16E-06	2.37E-05	8.54E-05

Table 2: Raw unscaled values for the Diversity Metric computed for all 8 scenes for all 8 methods under comparison.

	Fish	FishR	Balloons	Dirt	Forest	Plants	Blocks	Chalk
PiGAN	1.01E-07	2.53E-07	1.01E-07	1.99E-07	7.42E-08	1.26E-07	8.30E-07	1.74E-08
Graf	1.38E-07	5.85E-07	2.03E-06	3.32E-07	1.26E-07	3.14E-07	1.27E-06	9.15E-08
OursPlatoGAN	8.79E-07	1.44E-06	2.05E-06	9.92E-08	1.39E-07	6.99E-08	1.24E-06	2.53E-08
OursSinGAN3D	1.28E-07	9.77E-08	3.52E-07	1.24E-07	6.89E-08	1.58E-07	1.22E-06	1.00E-08
OursPlatonicNoRecon	2.19E-06	-3.73E-20	3.65E-06	2.20E-06	-3.73E-20	2.62E-06	3.44E-06	2.35E-08
OursSinGan3DNoRecon	4.17E-08	4.14E-07	6.91E-07	3.14E-07	1.11E-07	4.40E-09	6.48E-07	8.92E-07
OursNoRecon	5.24E-07	1.62E-06	1.54E-06	2.70E-07	1.41E-06	4.15E-07	5.85E-07	3.76E-07
Ours	3.85E-07	5.75E-07	4.25E-06	7.54E-07	1.57E-07	4.85E-07	3.71E-06	2.86E-07

Table 3: 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 $\underline{\text{underlined}}$.

	`OursSinGan3DNoRecon`		`OursPlatonicNoRecon`		`OursNoRecon`		`Ours`	
	Qual. ↓	Div. ↑	Qual. ↓	Div. ↑	Qual. ↓	Div. ↑	Qual. ↓	Div. ↑
FISH	$\underline{104.86}$	**5.69**	107.88	0.11	185.32	$\underline{1.36}$	**1.00**	1.000
FISHR	71.14	0.00	$\underline{21.35}$	0.72	60.22	**2.81**	**1.00**	$\underline{1.000}$
BALLOONS	14.22	$\underline{0.86}$	$\underline{1.44}$	0.16	11.51	0.36	**1.00**	**1.000**
DIRT	1.50	**2.91**	1.79	0.42	**0.51**	0.36	$\underline{1.00}$	$\underline{1.000}$
FOREST	$\underline{14.20}$	0.00	15.08	0.71	28.89	**9.00**	**1.00**	$\underline{1.000}$
PLANTS	16.47	**5.40**	35.02	0.01	$\underline{5.54}$	0.86	**1.00**	$\underline{1.000}$
BLOCKS	1.44	$\underline{0.93}$	1.77	0.17	$\underline{1.27}$	0.16	**1.00**	**1.000**
CHALK	**0.60**	0.08	4.27	**3.12**	2.53	$\underline{1.31}$	$\underline{1.00}$	1.000

limitation of our current method.

We optimize for 2000 steps in the GAN training per stage while doing 3 updates for the generator and the discriminator per step. We use the *Adam* optimizer [5] with the *beta* values set to $(0.5, 0.999)$ and a learning rate of 0.0005. The value of the learning rate is reduced to 0.00005 after 1600 steps for applying fine optimization.

3. Additional Ablations

As shown in Table 3, we present three more ablations of our full method `Ours`. In `OursSinGan3DNoRecon`, we use only a 3D discriminator D_{3D} for training without the fixed-seed reconstruction loss. In `OursPlatonicNoRecon` and `OursNoRecon`, we apply only the 2D discriminator loss D_{2D} without the reconstruction loss and use only the two discriminators D_{3D} and D_{2D} without either of the reconstruction losses respectively. As evident from Table 3, `Ours` still produces the best quality for most of the scenes.

As for diversity, the numbers are higher for other methods in most scenes because they produce broken incoherent structures which accounts for diversity in the generations, but at unacceptable quality. We refer to the qualitative samples for more information on these.

We also provide the raw unscaled values for the *Quality* and the *Diversity* metrics in table 1 and table 2 respectively.

4. Storage requirements

The trained CNN based generator models are quite memory efficient, requiring $\sim$12MB for storage. The generated grids, on the other hand, require the following sized PyTorch tensors in GPU memory during the forward pass for our 7-stage models: 0.10MB, 0.23MB, 0.54MB, 1.20MB, 2.84MB, 6.75MB and 16.0MB, respectively. All the results were rendered at a resolution of 512×512 while our biggest volumetric grids are of resolution $128\times128\times64$.

5. Potential negative impact of the work.

Our work contributes a method for generating novel static 3D scenes with view-consistent camera control. This could be used to generate synthetic (fake) content, or to edit existing 3D content. Similar to what we have seen in the 2D domain, e.g., with DeepFakes, once photorealistic quality is obtained, such methods have the potential to be used to spread disinformation. Future work on media forensics, as well as provenance may be needed to moderate these risks.

6. Result Quality

We want to highlight that our current results are not photorealistic quality; they contain visual artifacts such as "splotchyness", blur and blemishes. But it can be observed that even with these artifacts our results look much better compared to the baselines, namely PiGAN [1], GRAF [6] and PlatonicGAN [2]. This indicates that the specific problem of creating a 3D generative model from a single 3D exemplar scene is particularly challenging. We believe that our proposed approach makes substantial strides towards solving this challenging problem. Often the first work attempting a new problem domain has low resolution or low quality results (e.g., 2D GANs) but serve to inspire follow up works that resolve the issues.

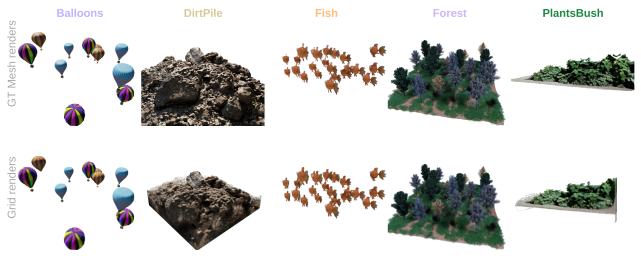

Figure 3: Qualitative comparison between the GT synthetic mesh renders and corresponding reconstructed grid renders.

Real scene Qualitative results Figure 4 provides different qualitative samples for the two real-captured scenes BLOCKS and CHALK. As seen in Table 2 (main paper), the results on the real-captures are of lower quality.

7. Evaluation on real scenes

We captured two scenes used for experimentation named BLOCKS and CHALK, reconstructed from real multi-image mobile-phone captures, calibrated using Colmap (for pose estimation) and reconstructed by our grid-based pipeline [4].

8. Stage-1 reconstruction quality:

The stage-1 of our pipeline, which reconstructs the 3D-grids from the posed-images, is robust and yields good quality reconstructions. We therefore think it is likely that lack in result quality is entirely from the stage-2 (GAN) of the proposed pipeline. Figure 3 shows the renders of our reconstructed 3D grids for visual inspection, and Table 4 summarizes the quantitative performance. The PSNR scores

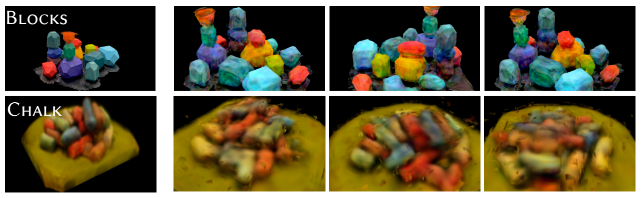

Figure 4: Qualitative results for BLOCKS and CHALK real scenes.

are reported on a hold-out test set of views not used in the training (reconstruction). In spite of our Lambertian approximation for the reconstructed scenes, all of them have > 20.0 PSNR except for DIRTPILE and PLANTSBUSH. As apparent from Figure 3, these two scenes have content outside of our defined AABB which affects the score. This is done to ensure that the scenes remain amenable to "remixing" instead of sticking to a fixed grid structure.

Scene	Balloons	DirtPile	Fish	Forest	PlantsBush
PSNR	25.96	9.853	26.76	25.88	14.88

Table 4: Quantitative evaluation of the grid reconstructions

9. *Rendering* Details

We indeed apply trilinear-interpolation for obtaining continuous feature values on the grid. As noted in Section 3.1 (main paper), we apply a *ReLU* to the interpolated feature-values to make them $[0 - 1]$ range compatible with the rendering algorithm. This can be viewed as a "post-activation" on the minimal neural feature-grid. We use the Emission-Absorption ray-marching algorithm for the rendering operation similar to PlatonicGAN [25], NeRF [43], and their variants. Please refer to [4] for more details.

10. Geometric Quality

Our 3D discriminator uses reconstructed density values as ground truth. At test time, in absence of actual ground truth this is the second best source of 3D priors in our setup. We already demonstrated the importance of using both the 2D and 3D discriminators in our ablation studies. Additionally, we compared our reconstruction quality to ground truth data for the synthetic scenes and found the quality to be satisfactory (e.g., similar accuracy as Planoxels, ReluField, etc.).

References

[1] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *IEEE CVPR*, pages 5799–5809, 2021. 3

[2] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *ICCV*, pages 9984–9993, 2019. 3

[3] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021. 1

[4] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy J. Mitra. ReLU fields: The little non-linearity that could. In *Proc. of SIGGRAPH*, volume 41, pages 13:1–13:8, 2022. doi: 10.1145/3528233.3530707. 3, 4

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[6] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020. 3

[7] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4570–4580, 2019. 1