# Learning Generative 3D Scene Layouts from a Single Image

Linan Zhao*
Stanford University

Zeqing Yuan*
Stanford University

Yunzhi Zhang
Stanford University

Shangzhe Wu
Stanford University

Jiajun Wu
Stanford University

## Abstract

*What is a scene, conceptually? It can be decomposed into multiple objects, their spatial arrangement, and the background. While recent works have pushed the boundary on modeling 3D objects, the scene layout indicating how objects are arranged in 3D space remains under-explored. In this work, we build a generative model that learns the 3D scene layout distribution from a single 2D image, such as a photo of a parking lot containing several cars. We first retrieve the object geometry from segmented instances. Next, we build a permutation-equivariant model to generate layout parameters, which, combined with geometry, render scene images. We then leverage a patch-based discriminator on 2D images along with auxiliary losses to guide layout learning. Experiments demonstrate that our model successfully learns a wide range of layout distributions, each from a single Internet image. Our method achieves superior results on multiple downstream tasks, including extrapolating on number of instances and transferring learned layout to other objects. Project page at* https://sinlayout.github.io/.

## 1. Introduction

A scene can be conceptually disentangled into objects, their spatial layout and the background. Recent advances in generative models have significantly improved object modeling [12, 14, 17, 27]. However, composing multiple objects into a coherent layout remains an unresolved challenge. There are certain layout distributions of how objects could be oriented and located in our 3D space, originating from either nature or human fabrications.

[1, 23] have made strides in implicitly modeling object layouts in 3D-aware feature space but still rely on 2D image generator. [3] employ a large language model for planning scene layouts, but language falls short in describing scene layout compared to visual inputs. None of the previous works have explicitly learned the underlying 3D layout parameters from image inputs.

When we, as humans, perceive a scene with various objects, we are able to parse their spatial arrangement and map it to an abstract mental representation, which enables us to imagine different variations. This ability is crucial for generating diverse and plausible 3D scenes with controllable spatial composition and for understanding human visual cognition in scene comprehension. Nevertheless, this ability is non-trivial in computer vision. For instance, considering how object pose estimation [29] has been a long sought-after task, learning the distribution of spatial layout from a single image is particularly hard.

In this work, our goal is to build a pipeline that captures such 3D scene layout distribution from a single image and to use this model to generate faithful and diverse 3D scenes, which supports not only rendering under novel view points and illumination conditions, but also extrapolation to varying number of instances.

This task is challenging for the following four reasons. First, this is a zero-shot setting with only one single image as input, which marks the difference from typical generative tasks relying on a large dataset to fit the distribution. Second, we do not assume knowledge of the input image's ground-truth pose parameters, which must be learned from pixel signals. Third, the input image may have occlusion, making it harder for pose perception. Finally, the layout information we aim to learn is probabilistic, requiring the generator to understand reasonable interpolation and extrapolation. Compared to deterministic tasks such as 6D object pose estimation, this poses great difficulties.

To address these challenges, we propose an permutation-equivariant layout generator and an adversarial pipeline with an inductive bias that all instances are from the same

*Equal contribution; each reserves the right to be listed first.

category to avoid interference with inter-class semantic relations. Here, we define that two images containing multiple instances of the same object have the same layout if their distribution of randomly sampled local patches is similar. As we will see, this definition is baked into our pipeline, which uses patch-based discriminators with auxiliary losses for guidance. The generator can synthesize diverse 3D layout with varying number of instances. Along with obtained geometry, scene images can be rendered under any view point and illumination condition.

To demonstrate the effectiveness of our method, we conduct experiments on synthetic and Internet images, including interpolation on input noise and extrapolation to varying number of instances. The result show that our model is capable of generating faithful and diverse layouts.

The contributions of this paper are as follows:

1. We propose the task of learning 3D scene layout from a single image containing multiple similar instances.

2. We build a generative pipeline fulfilling this task and show the effectiveness of our method on Internet images with interpolation and extrapolation results.

# 2. Related Work

## 2.1. Scene Layout Modeling

Works on scene image generation like BlobGAN [1] and BlobGAN-3D [23] disentangle individual objects and implicitly model object layouts in feature space by using approximate blobs as representations. The representations are projected into a feature map, serving as inputs to image synthesis networks to generate 2D scene images based on image priors. The generative process remains inherently 2D, though adopting 3D-aware feature representations. Our method explicitly represents object layouts in 3D space, which naturally manages problems such as occlusion and shading. It also enables rendering from any viewpoint and under various lighting conditions.

On the other hand, LayoutGPT [3] leverages a large language model (LLM) as scene layout planner. The language model outputs layout in the form of bounding box parameters. In comparison to image input, this approach is hampered by the limited ability of language to describe visual layouts and LLM's insufficient comprehension on 3D visual information.

3inGAN [8] focuses on a similar task of generating 3D scenes from image input by a multi-level pipeline using 3D feature grids. However, it relies on multiple images with pose information as input, whereas we use a single image without pose annotation to explicitly learn the underlying layout parameters.

SinGRAV [25] also adopts an adversarial pipeline for scene generation but learns to generate neural radiance volumes from multi-view observations of a single scene. It leverages a multi-scale framework with convolutional networks that emphasize spatial locality bias.

[2] leverages a pretrained diffusion model to optimize a 3D scene composed by a set of layout parameters. It rely on the diffusion prior to align the scene with input text prompt.

To the best of our knowledge, this work is the first to learn generative 3D scene-level layouts from a single image without ground-truth layout annotations or geometry.

## 2.2. Generative Modeling on a Single Image

Generative models traditionally require extensive datasets to fit specific distributions accurately. However, recent works such as SinGAN and SinDiffusion [5, 16, 19, 20, 24] utilize patch-based approaches to leverage diversity within regional crops, facilitating the training of GANs and diffusion models with just a single image. These methods have demonstrated the potential of single-image generative modeling. However, they predominantly focus on 2D aspects and do not explicitly address the complexities of 3D space.

Further, [27] have made strides in this area by developing an adversarial pipeline that recovers object intrinsics, including 3D shape and albedo, from singular images featuring multiple similar instances. Our work focuses on the intrinsics of 3D scenes, particularly spatial layouts.

Additionally, the significance of heavy augmentation in single-image training has been highlighted by [22] in image shape manipulation tasks. In line with this, ADA [9] introduces differentiable data augmentation techniques effective in limited-data settings.

# 3. Method

Given a single RGB image $I$ containing $n$ instances of the same object arranged in a patterned way, our goal is to learn its associated underlying layout pattern. Examples of such patterns include parallel lines, circular patterns, and objects being up-right, as shown in the left column of Figure 2. For all object instances in the scene, there exists a ground truth location and rotation of the object relative to the camera. We coin an object instance's location and rotation as its layout parameter, denoted by $p_i \in \mathrm{SE}(3)$ for $i = 1, 2, \cdots, n$. Thus, the layout of the input image $I$ can be expressed as the set $\{p_i\}_{i=1}^n$ since the order of constituting objects do not matter in a scene. Further, we view the input image's layout as one sample from a larger ground truth layout distribution $P$. This distribution $P$ captures all layout patterns that are structurally and semantically similar to the input image's $\{p_i\}_{i=1}^n$. Specifically, $P$ should be a subset of $\cup_{m\in\mathbb{N}}\{\mathrm{SE}(3)\}_{i=1}^m$ and any sample from $P$ should contain a set of $m$ layout parameters which form a scene similar to that of $I$. Our goal is to learn this underlying distribution from a single sample through adversarial training.

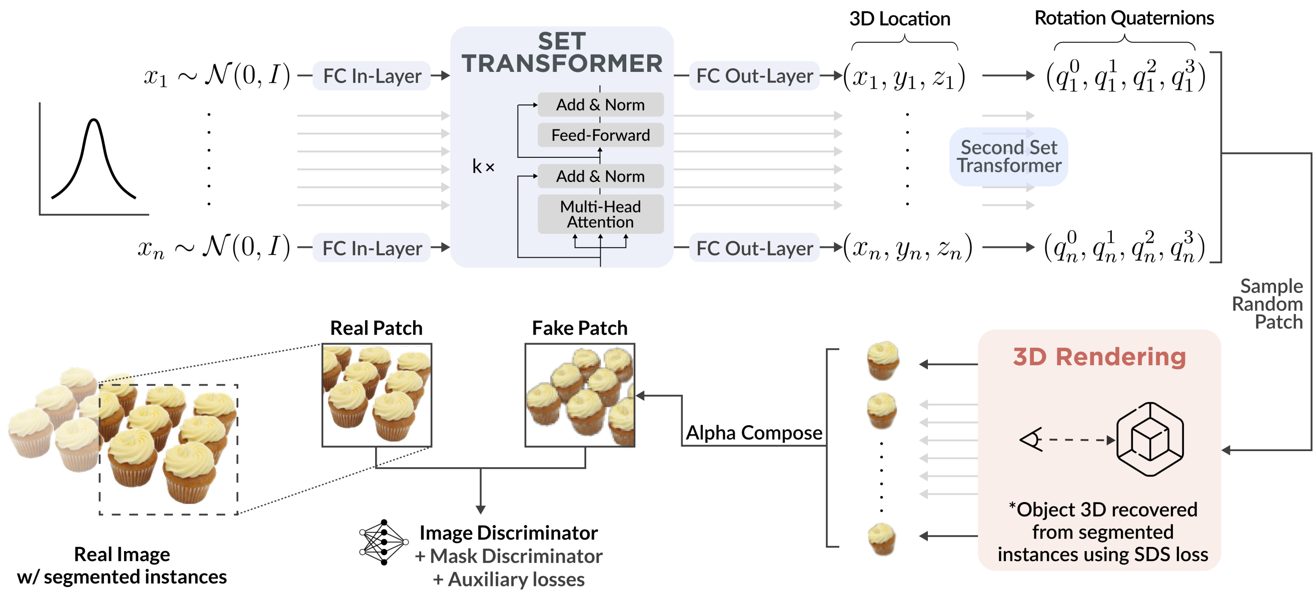


Figure 1. Pipeline overview. We propose a generative model that recovers the layout of multiple objects within a scene from a single input image. We disentangle the geometry of objects from the layout parameters, which include location and rotation parameterized by quaternions. We first segment out individual instances in the input image and use multi-view diffusion models like Wonder3D [14] to recover the object's 3D geometry. Then, we model the layout generator as a two stage permutation-equivariant set transformer. The first set transformer generates locations from sampled Gaussian noises and the second set transformer generates rotation quaternions conditional on locations. Combined with the reconstructed 3D geometry, we can render a patch of the generated scene. The whole pipeline is trained with patch-based discriminators and auxiliary losses.

Here, note that we do not restrict $m$ to be the same as the number of instances $n$ of the input scene. This is because, for example, we regard a tray of 12 cupcakes arranged in 3 rows and 4 columns and a tray of 8 cupcakes arranged in 2 rows and 4 columns to share the same overall layout distribution $P$. The two arrangements are two separate draws from the same layout distribution. Further, we define $P$ as a distribution over sets as we assume order invariance of object instances. The set of objects form a coherent scene but the relative order bears no significance in our setting.

Figure 1 illustrates an overview of our training pipeline, which involves recovering object geometry, modeling layout, and learning from patch-based adversarial loss. Section 3.1 details how we recover the 3D geometry and texture of an average object from segmented instances $I_i$ in the input image $I$. With the recovered object geometry, Section 3.2 explains the permutation-equivariant design of our layout generator. Together, the model can generate scenes of the objects, which is trained with a patch-based adversarial scheme. The training objective and details are presented in Section 3.3 and Section 3.4. Through our pipeline, the layout generator learns to match the distribution of generated layout parameters to $P$.

### 3.1. Geometry Modeling

In this section, we illustrate how we infer an average object geometry from the input instance observations. Given the input image $I$, we first use Grounded SAM [13] to segment out the individual instances $I_1, I_2, \cdots, I_n$ as well as the background, assuming there are $n$ instances in the input image scene. Going forward, we can assume that $I$ has no background. Since we do not assume the scene has no occlusion, some object instances in $\{I_i\}_{i=1}^n$ may be incomplete. As such, we cannot easily run textual-inversion-based methods like [17] and resort to single-image-based methods like [12, 14].

One approach we use to recover the 3D geometry and texture of the object is through the multi-view diffusion model Wonder3D [14]. Wonder3D finetunes Stable Diffusion on the LVIS subset of the Objaverse dataset [**?**] to generate the color images and normal maps from six views conditional on an input image. They use multi-view self-attention to ensure view consistency and cross-domain self-attention to ensure information flow between the color images and normal maps. Given the generated multi-view color images and normal maps, Wonder3D extracts 3D geometry via neural implicit signed distance fields. Once trained, we export the 3D representation to a textured mesh for faster differentiable rendering using [18].

Now, to render a scene of $n$ objects with layout parameters $p^n$ in $\mathrm{SE}(3)^n$, we first use the recovered mesh to render individual objects. Then, given the objects' $z$ coordinate ordering, we alpha-compose all the individual objects into the same scene. This approach has the advantage of faster rendering, especially when we only need to render a patch of the scene. For simplicity, the rendering process is modeled as $f(p^n, c)$, where $p^n \in \mathrm{SE}(3)^n$ is the location and rotation of all $n$ instances, and $c = (w_0, w_1, h_0, h_1)$ defines the crop of the whole scene we want to render.

### 3.2. Layout Modeling

Now that we can render a scene of objects given any combination of layout parameters, we wish to generate layout that are similar to the input image's. Specifically, we wish to train a generator $g_\theta(z) : \mathcal{N}(0, I)^n \to \mathrm{SE}(3)^n$ to generate a distribution $\hat{P}$ that is similar to $P$. Note that we can further decompose $\mathrm{SE}(3)^n$ in terms of translation vectors and rotation matrices. We can also parameterize rotation matrices in $\mathbb{R}^{3\times3}$ as quaternions $a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} \in \mathbb{H}$. This allows for a more compact representation and a potentially more continuous parameter space. Thus, any $p \in \mathrm{SE}(3)$ can be interpreted as $(R, T)$ with $R \in \mathbb{H}$ and $T \in [0, 1]^3$, where we assume location in a finite-size scene is normalized to $[0, 1]^3$.

In addition, instead of generating all the location and rotation parameters all at once, i.e. learning the joint distribution of $P(R, T)$, we find that modelling $P(R|T)P(T)$ is easier. Thus, we use two generators, the first one $g_{\theta_1}(z) : \mathcal{N}(0, I)^n \to ([0, 1]^3)^n$ maps Gaussian noises to location parameters, and the second one $g_{\theta_2}(T) : ([0, 1]^3)^n \to \mathbb{H}^n$ maps generated location to rotation quaternions. Thus, $g_\theta(z)$ is equivalent to $(g_{\theta_1}(z), g_{\theta_2}(g_{\theta_1}(z)))$.

As shown in Figure 1, we use two permutation-equivariant set transformers to model $g_{\theta_1}$ and $g_{\theta_2}$. The set transformer design is drawn from [11], with the intention of generating a set instead of generating an ordered list. Specifically, we sample $n$ independent $d$-dimensional Gaussian noise vectors, and pass them through the same fully-connected in-layer to be projected into $n$ higher dimensional vectors. These vectors are then passed into a transformer architecture without positional embeddings, before being projected down to $n$ $[0, 1]^3$ location vectors by another fully-connected layer and sigmoid layer. Similarly, the second set transformer $g_{\theta_2}$ generates the rotation quaternions conditional on the location vectors generated by $g_{\theta_1}$. The transformer backbone follows the common structure in [21] as indicated in Figure 1.

The advantage of the set transformer is that the attention mechanism can readily grasp the complex interactions between set elements. It is expressive enough to capture the relationship among objects' layout parameters, which is exactly the layout distribution we want to learn. The permutation equivalence property ensures that as long as the input noises are the same as a set, the final scene would look the same. This makes sure that the $i^{\text{th}}$ generated layout parameter does not correspond to a fixed object in the scene and over-fits to that object's location and pose. Moreover, this design allows for the generation of an arbitrary number of layout parameters - $n$ can be any positive integer and does not have to be the exact number of instances in the input scene. This feature grants the ability to extrapolate on the number of instances to create more diverse scenes.

### 3.3. Adversarial Training

Since we do not know object instance locations and poses, we cannot directly train the generator in layout parameter space. Instead, we leverage a generative adversarial (GAN) framework [4] to update the layout generator in pixel space. That is, we train an image discriminator $D_\eta$ that discriminates on image crops from real and fake scenes. This patch-based design is inspired by recent works in single-image generative models like [5, 16, 19, 20, 24].

**Image Crops.** Specifically, in each iteration, we sample $n$ Gaussian noises and generate the corresponding $g_\theta(z)$ layout parameters $p^n$. Given these layout parameters, we can render image crops $I_{\text{fake}}$ of the fake scene using $f(p^n, c)$, where $c = (w_0, w_1, h_0, h_1)$ is the coordinates of a randomly sampled crop. The crop size is a fixed proportion $s$ of the input image's dimensions. We further denote $c \sim C$, where $C$ is the uniform distribution of all possible such crops. Similarly, we randomly sample a crop out of $C$ for the real image $I$ and denote it as $I_{\text{real}}$. The discriminator's goal is to distinguish $I_{\text{fake}}$ from $I_{\text{real}}$. Our experiments found that using two discriminators with different crop sizes works best. We choose a small crop $s = 40\%$ for the first discriminator $D_{\eta_1}$ and a larger crop $s = 80\%$ for the second discriminator $D_{\eta_2}$. During training, we sample multiple crops from the same generated image so that the gradient flows back to most of the image regions.

**Discriminator Design.** The discriminator uses a convolutional neural network architecture followed by a linear projection layer. To stabilize training and avoid early overfitting of the discriminator $D_{\eta_2}$ that distinguishes the larger crop, we add an auxiliary pose prediction task as regularization. This auxiliary loss is defined as

$$\mathcal{L}_{\text{pose}}(\eta) = \mathcal{L}_{\text{cham}}(g_{\text{GS}}(\hat{R}), g_{\text{GS}}(R)), \tag{1}$$

where $\mathcal{L}_{\text{cham}}(A, B)$ is the Chamfer loss between two sets of vectors defined as

$$\mathcal{L}_{\text{cham}}(A, B) = \sum_{a\in A} \min_{b\in B} ||a-b||_2^2 + \sum_{b\in B} \min_{a\in A} ||a-b||_2^2, \tag{2}$$

$R, \hat{R}$ are the rotation matrices used to generate the fake scene and the predicted pose by the discriminator, respectively, and $g_{\text{GS}}$ maps $\text{SO}(3)$ rotations to 6D embeddings following [28].

In addition to the image discriminator $D_\eta$, we use a second discriminator $D_{\eta_{\text{mask}}}$ for masks. This discriminator receives the masks of the cropped fake image and the cropped real image for discrimination. We found that the separate mask discriminator helps to stabilize location generation and improves overall result.

**Augmentations.** In single-image GANs, it is very easy for the discriminator to over-fit the input image. As such, [22] points out that heavy augmentation is needed. Since the generated fake scene and the input image have no background, we use random color backgrounds to stabilize training, similar to [27]. Further, we use Adaptive discriminator augmentation (ADA) [9] to augment $I_{\text{real}}$ and $I_{\text{fake}}$. In particular, we found that adding random noise with a somewhat large probability (e.g. 0.5) works well.

**Training Objective.** Similar to [15, 27], we use a binary cross entropy loss as the GAN training objective, with a regularization term on the gradient of the discriminator:

$$\begin{aligned}\mathcal{L}_{\text{adv}}(\theta,\eta, I) = \mathbb{E}_{z,c\sim C}[h(D_\eta(f(g_\theta(z), c)))] \\ + \mathbb{E}_C[h(-D_\eta(I_{\text{real}})) - \lambda_{reg}||\nabla D_\eta(I_{\text{real}})||^2],\end{aligned} \tag{3}$$

where $h(t) = -\log(1 + e^{-t})$.

In addition, since we have masks for the segmented individual objects, we can use the average of the masks' $x$ and $y$ coordinates as a proxy for the object's location projected onto the image plane. Denote these locations as $\{x_i, y_i\}_{i=1}^n$. Similarly, we can project the generated location vectors $g_{\theta_1}(z)$ onto the image plane to obtain $\{\hat{x}_i, \hat{y}_i\}_{i=1}^n$. We can then use Chamfer loss to supervise location generation directly:

$$\mathcal{L}_{\text{loc}}(\theta_1) = \mathcal{L}_{\text{cham}}(\{x_i, y_i\}_{i=1}^n, \{\hat{x}_i, \hat{y}_i\}_{i=1}^n). \tag{4}$$

Similarly, we can use the detected object locations in the image plane to calculate the minimum pairwise distance among all objects $d$. This is a proxy measure for the amount of overlapping among objects in the input scene. Then, we can calculate the pairwise distances among all generated location vectors $g_{\theta_1}(z)$ that are projected onto the image plane and penalize any two objects that are too close to each other. We penalize the $L_2$ distance between any pair of objects whose distance is below $\lambda_d d$, where we choose $\lambda_d$ to be $0.75$. This auxiliary loss prevents generated instances from overlapping, which blocks the successful flow of gradient updates. We coin this loss $\mathcal{L}_{\text{overlap}}(\theta_1)$.

Lastly, since the space visible to the camera is a frustum and not a cube, we add a simple off-scene loss $\mathcal{L}_{\text{off-scene}}(\theta_1)$ that penalizes generated locations outside the viewing frustum. This loss encourages all generated objects to be within view, as objects not rendered in the fake scene will receive no gradient updates.

Combining all the losses above, the final training objective comprises six terms:

$$\begin{aligned}&\mathcal{L}(\theta, \eta, I) = \\ &\mathcal{L}_{\text{adv}}(\theta, \eta_1, I) + \lambda_{\eta_2}\mathcal{L}_{\text{adv}}(\theta, \eta_2, I) + \lambda_{\text{pose}}\mathcal{L}_{\text{pose}}(\eta_2) + \\ &\lambda_{\text{loc}}\mathcal{L}_{\text{loc}}(\theta_1) + \lambda_{\text{overlap}}\mathcal{L}_{\text{overlap}}(\theta_1) + \lambda_{\text{off}}\mathcal{L}_{\text{off-scene}}(\theta_1).\end{aligned} \tag{5}$$

### 3.4. Training Details

For different images, we use different scene resolutions for training. The scene resolution is roughly proportional to $\lceil\sqrt{n}\rceil$, where $n$ is the number of instances in the input scene. Weights of the loss terms in Equation (5) are specified as $\lambda_{\text{reg}} = 10, \lambda_{\eta_2} = 0.5, \lambda_{\text{pose}} = 1, \lambda_{\text{loc}} = 200$, $\lambda_{\text{overlap}} = 5.0$, and $\lambda_{\text{off}} = 100$. The input noise has dimension 3, which matches the 3-dimensional $x, y, z$ coordinates. The transformer architecture is adopted from [26] with 256 dimensions, 4 attention heads, and 3 layers. The backbone for both discriminators is adapted from GIRAFFE [15]. To provide a better initialization, we pretrain the first set transformer $g_{\theta_1}(z)$ using the three auxiliary losses $\lambda_{\text{loc}}\mathcal{L}_{\text{loc}}(\theta_1) + \lambda_{\text{overlap}}\mathcal{L}_{\text{overlap}}(\theta_1) + \lambda_{\text{off}}\mathcal{L}_{\text{off-scene}}(\theta_1)$. This avoids heavy occlusion at the start of training and provides better location initializations. We use an Adam [10] optimizer for the generator and RMSprop [6] optimizers for the two discriminators, with learning rates $2e^{-5}$ and $1e^{-4}$ respectively. Since 3D rendering takes significant VRAM space, we found that using gradient accumulation [7] to mimic a small batch of 32 is quite effective. In our experiments, training usually takes 100,000 iterations to converge, which takes approximately 3 hours on a Nvidia GeForce RTX 3090 GPU.

## 4. Experiment

We test our method on both synthetic data and real-world images and evaluate the generation quality extensively. Synthetic data allows us to test performance under specified common layouts while real-world images allow us to test the robustness of our methods. Experiments show that our proposed method can recover a diverse range of layouts from both synthetic and real data. We also demonstrate several downstream tasks that showcase the versatility of our learned generator.

**Dataset.** For the synthetic data set, we download the mesh of a car from the Internet. We then build custom scenes with well-defined layouts as shown in the left half of Figure 3. In particular, we focus on four cases – layout with or without uniform rotation and from top-down or slanted camera views. For instance, the left most image is a case of uniform rotation with top-down camera view.

For the real-life images, we curate a diverse set of images from the Internet. These images capture many different types of objects as well as many different forms of layout distributions. Note that we try to sample a wide variety

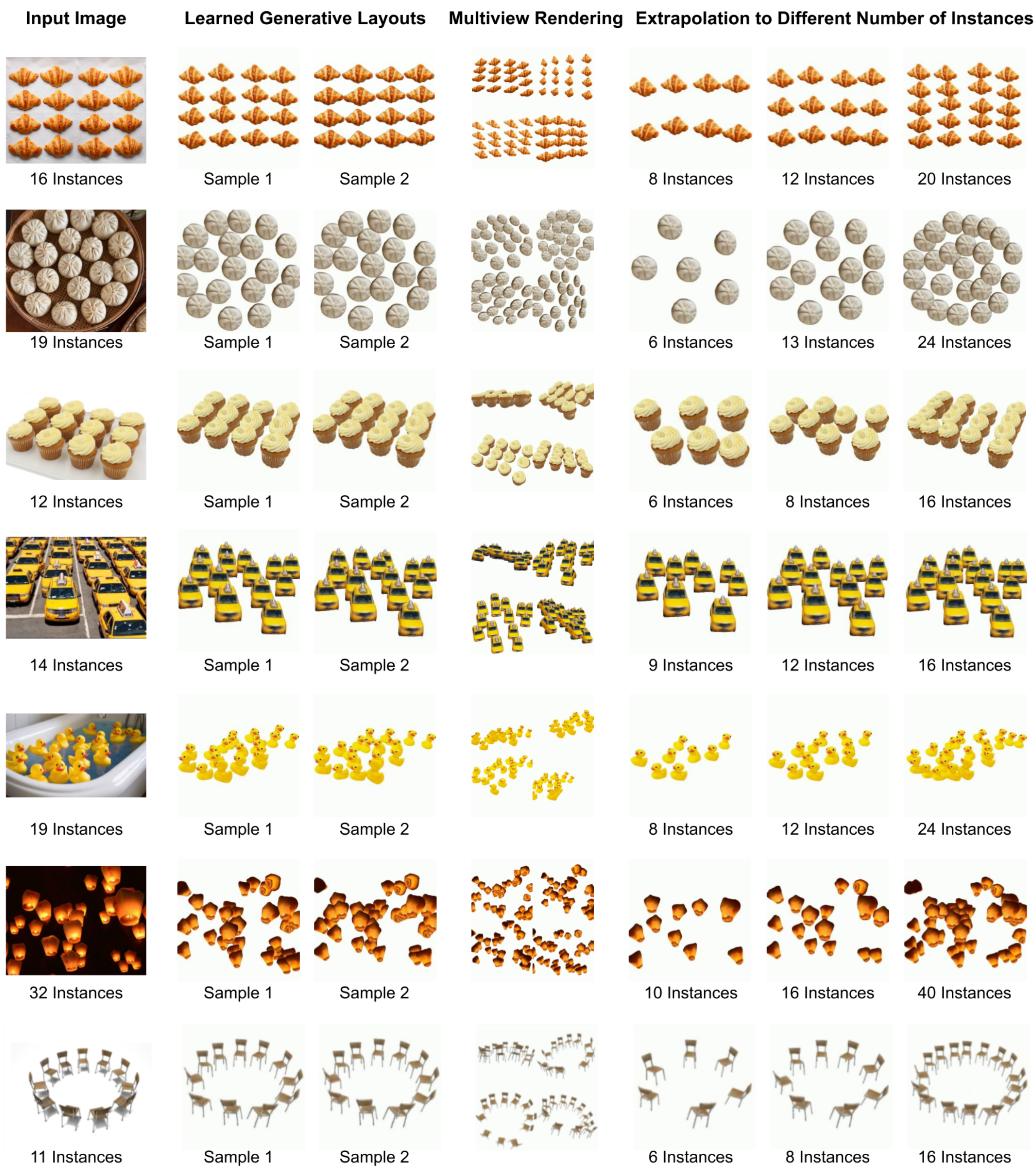


Figure 2. Main results. The left column shows the input image. The second and third columns show two samples of the learned layouts. The middle column shows four views (bottom, right, top, and left 30 degrees) of the learned layout in 3D space. The right three columns show generations with different numbers of instances.

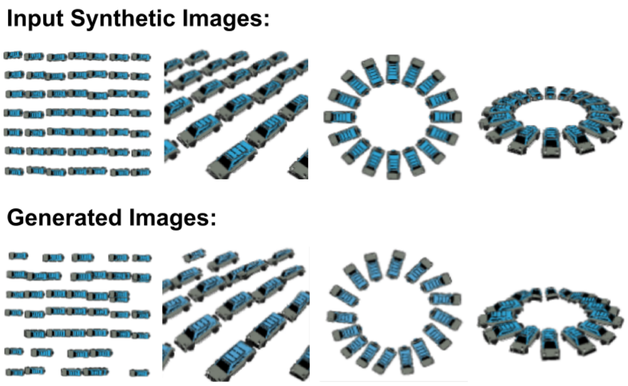


Figure 3. Learning from synthetic images. The top row represents the input images, whereas the bottom row are generated images using reconstructed geometry and learned layout parameters.

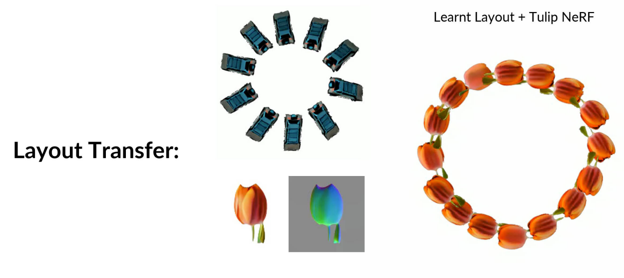


Figure 4. **Layout Transfer.** We can transfer the layout learnt from a circular parking lot to tulip geometry and form a flower wreath.

of images – those with or without occlusion, those with or without slight instance variations, and those with or without uniform texture.

**Results.** We are able to demonstrate promising results on synthetic images. In particular, we can solve any of the four cases mentioned above. Even when there are occlusions among objects and nonuniform poses, our layout generator can still capture the underlying layout distribution. In Figure 3, we see that our method can faithfully recover a diverse range of layout distributions.

Similar successes are evident on real images as well. We can recover uniform layout from top-down views and slanted views where there are significant amounts of occlusion. We can also learn non-uniform layout and capture the spatial pattern in more stochastic scenes. Notice that the reconstructed meshes in the real images are not exactly the same as the real objects and the lighting in the real-world images can also be quite complex. Thus, our method is reasonably robust to these variations and can learn layout in realistic situations.

**Novel View Rendering.** Since the objects are placed in 3D space, we can render multi-view images of the scenes generated from learned layouts. For each real-world image, four views are shown in Figure 2. The underlying pattern is captured by viewing from different angles as well.

**Diversity.** To demonstrate diversity, we can interpolate $z$ in the latent space and sample multiple layout parameters and scenes. Figure 2 illustrates two different samples for each real-world image. Layout parameters generated from different input noises are similar and largely adhere to the input image's layout distribution, with subtle distinctions contributing to a desirable diversity. However, some instabilities do exist due to stochastic GAN training and we are actively working on improving the results.

**Generalizability.** To demonstrate generalizability, we can extrapolate on the number of instances within the scene. This ability comes from the generator's permutation-equivariant design. Although the model is only trained on an image with a fixed number of instances, it can produce faithful layouts for different numbers of instances. During inference, we can simply expand or reduce the number of Gaussian noises being sampled to generate scenes with more or less number of instances. Figure 2 demonstrates three extrapolation examples from each input scene. Most notably, when we generate 6 instances or 16 instances in the circular chair layout case, the generated layout still forms a reasonable circle. And when we generate 8 or 12 instances of croissants, the generator synthesizes a two-by-four and a three-by-four layout respectively. Given the training image only contains the four-by-four uniform layout, this generalizing ability akin to human is quite impressive.

**Other Applications.** Further, our method can enable downstream applications. Since layout parameters and the geometry are disentangled, we can transfer learned layout to other geometries, shown in Figure 4. This allows us to creatively generate more different scenes with other assets. We can also directly edit the layout parameters. For instance, we can scale the location parameters by 2 to get a more dispersed scene. Further, since the rendering process is fully explicit and involves cameras and lights, we can render the same scene from any camera viewpoint and using any lighting. This demonstrates the potential of our setting as we can lift a single 2D image to a coherent 3D scene.

## 5. Conclusion

In this work, we introduce a novel task of learning generative 3D scene layouts from a single image, focusing on the rotation and location of objects from the same category. To tackle this problem, we propose a two-stage permutation-equivariant layout generator and an adversarial training pipeline. Through experiments, our method has demonstrated its effectiveness in capturing various layout distributions from a single Internet image, successfully generating diverse yet authentic layouts consistent with the input image.

## References


[1] Dave Epstein, Taesung Park, Richard Zhang, Eli Shechtman, and Alexei A. Efros. Blobgan: Spatially disentangled scene representations. *European Conference on Computer Vision*, 2022. 1, 2

[2] Dave Epstein, Ben Poole, Ben Mildenhall, Alexei A. Efros, and Aleksander Holynski. Disentangled 3d scene generation with layout learning, 2024. 2

[3] Weixi Feng, Wanrong Zhu, Tsu jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *arXiv preprint arXiv: 2305.15393*, 2023. 1, 2

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 4

[5] Niv Granot, Assaf Shocher, Ben Feinstein, Shai Bagon, and M. Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. *Computer Vision and Pattern Recognition*, 2021. 2, 4

[6] Alex Graves. Generating sequences with recurrent neural networks, 2014. 5

[7] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019. 5

[8] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 3inGAN: Learning a 3D generative model from images of a self-similar scene. In *Proc. 3D Vision (3DV)*, 2022. 2

[9] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12104–12114. Curran Associates, Inc., 2020. 2, 5

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[11] Adam R. Kosiorek, Hyunjik Kim, and Danilo J. Rezende. Conditional set generation with transformers. *CoRR*, abs/2006.16841, 2020. 4

[12] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. *ICCV*, 2023. 1, 3

[13] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2023. 3

[14] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint arXiv: 2310.15008*, 2023. 1, 3

[15] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields, 2021. 5

[16] Yaniv Nikankin, Niv Haim, and Michal Irani. Sinfusion: Training diffusion models on a single image or video. *arXiv preprint arXiv: 2211.11743*, 2022. 2, 4

[17] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. Dreambooth3d: Subject-driven text-to-3d generation. *arXiv preprint arXiv: 2303.13508*, 2023. 1, 3

[18] Nikhila Ravi, Jeremy Reizenstein, David Novotný, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *CoRR*, abs/2007.08501, 2020. 3

[19] Tamar Rott Shaham, Tali Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. *IEEE International Conference on Computer Vision*, 2019. 2, 4

[20] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and remapping the "dna" of a natural image. *arXiv preprint arXiv: 1812.00231*, 2018. 2, 4

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 4

[22] Yael Vinker, Eliahu Horwitz, Nir Zabari, and Yedid Hoshen. Image shape manipulation from a single augmented training sample. *IEEE International Conference on Computer Vision*, 2020. 2, 5

[23] Qian Wang, Yiqun Wang, Michael Birsak, and Peter Wonka. Blobgan-3d: A spatially-disentangled 3d-aware generative model for indoor scenes. *arXiv preprint arXiv: 2303.14706*, 2023. 1, 2

[24] Weilun Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Dong Chen, Lu Yuan, and Houqiang Li. Sindiffusion: Learning a diffusion model from a single natural image. *arXiv preprint arXiv: 2211.12445*, 2022. 2, 4

[25] Yujie Wang, Xuelin Chen, and Baoquan Chen. Singrav: Learning a generative radiance volume from a single natural scene. *arXiv preprint arXiv: 2210.01202*, 2022. 2

[26] Shangzhe Wu, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Dove: Learning deformable 3d objects by watching videos, 2022. 5

[27] Yunzhi Zhang, Shangzhe Wu, Noah Snavely, and Jiajun Wu. Seeing a rose in five thousand ways. *Computer Vision and Pattern Recognition*, 2022. 1, 2, 5

[28] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks, 2020. 5

[29] Yingzhao Zhu, Man Li, Wensheng Yao, and Chunhua Chen. A review of 6d object pose estimation. In *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 10, pages 1647–1655, 2022. 1