Learning Generative 3D Scene Layouts from a Single Image

Linan Zhao* Stanford University

Zeqing Yuan*† Zhejiang 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 advancements in generative models have significantly improved object modeling [11,13,17,27]. 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. [2] employ a large language model for planning scene layouts, but language falls short in describing scene layout compared to visual inputs.

Given a single image containing multiple instances of the same category, human can capture the layout distribution by inferring steady poses and layout patterns. This ability is vital for generating diverse and plausible 3D scenes and comprehending human visual cognition in scene understanding. Nevertheless, this ability is non-trivial in computer vision. Considering 6D object pose estimation [29] being a long sought-after task, learning from a single image the distribution of spatial layout 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 does not assume knowing the ground-truth pose parameters of the input image, which is necessary for typical diffusion pipeline to learn layout distribution in our setting. Third, the input image may include occlusion, making it hard for pose perception. Finally, the layout information we aim to learn is probabilistic, requiring the generator to understand reasonable interpolation and extrapolation. This poses great difficulties compared to deterministic task such as 6D object pose estimation.

To address these challenges, we propose an permutation-equivariant layout generator and an adversarial pipeline with an inductive bias that all instances within the image shares the same category in order to avoid the interference of inter-class semantic relation. Our pipeline obtains the instance geometry and use patch-based discriminators along

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

†Work done during UGVRI program at Stanford University.

with auxiliary losses to guide the training process. The generator can synthesize diverse 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 real 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 problem of learning 3D scene layout distribution from a single image, and thus decompose zero-shot 3D scene generation into object modeling, layout learning and background inpainting.

2. We build a generative pipeline fulfilling this task.

3. We demonstrate the effectiveness of our method through experiments on both synthetic and real images, as well as extrapolation study on varying number of instances.

2. Related Work

2.1. Scene Layout Modeling

Works on scene image generation like BlobGAN and BlobGAN-3D [1,23] 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 generating 2D scene images based on image priors. The generative process remains inherently 2D, though adopting 3D-aware feature representation. Our method explicitly represents object layouts in 3D space, thereby naturally managing problems such as occlusion and shading. It also enables rendering from any viewpoint and under various lighting conditions.

On the other hand, LayoutGPT [2] leverages a large language model 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 the comprehension of large language model on 3D visual information, leading to issues with controllability and diversity.

3inGAN [7] 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 information. Besides, it assumes the input scene image to be stochastic and self-similar, which we do not.

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

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 accurately fit specific distributions. However, recent works such as SinGAN and SinDiffusion [4, 15, 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 extends the focus towards 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 [8] 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 Figure 3. 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.

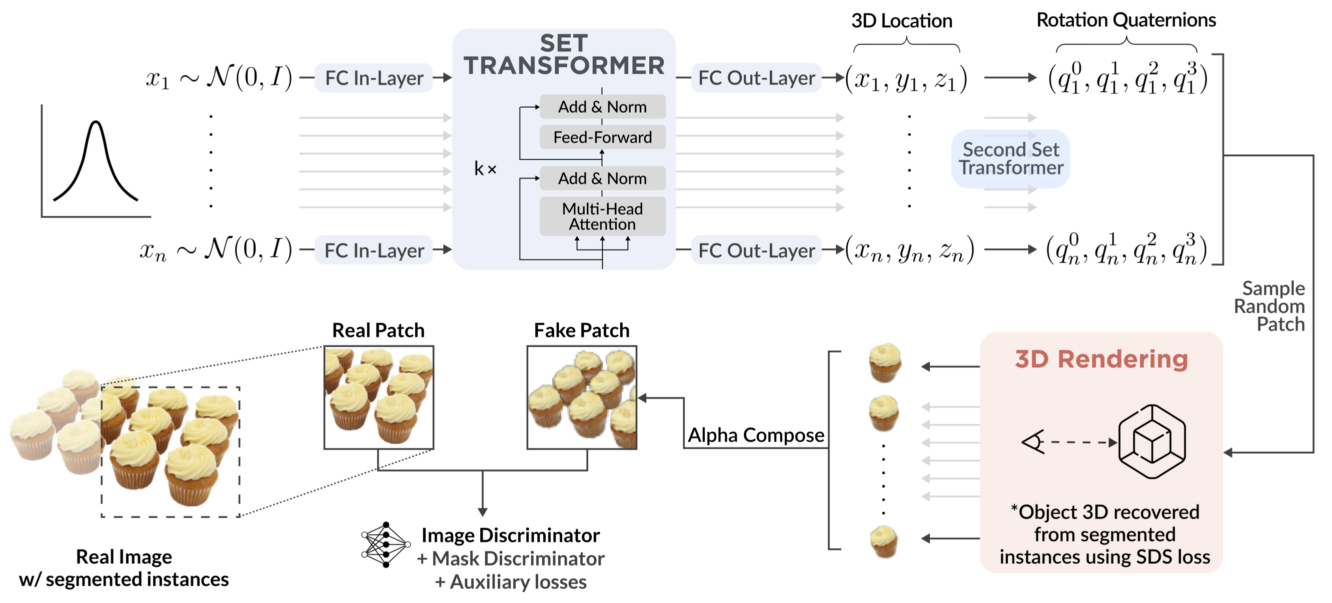

Figure 1. Model overview. We propose a generative model that recovers the layout of multiple objects within a scene from a single input image. We disentangle the geometry of objects from the layout parameters, which include location and rotation parameterized by quaternions. We first segment out individual instances in the input image and use score distillation sampling (SDS) loss with pre-trained diffusion models like Zero-1-to-3 [11] 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 [12] 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 [11, 13].

One approach we use to recover the 3D geometry and texture of the object is through score distillation sampling (SDS) guidance from Zero-1-to-3 [11]. Essentially, Zero123 is a fine-tuned diffusion model that aims to generate novel views of an object given any relative camera viewpoint conditional on one input image of the object. Given a single RGB image L_1 and camera extrinsics $R \in \mathbb{R}^{3\times3}, T \in \mathbb{R}^3$, Zero123 synthesizes novel views under the camera transformation $\hat{x}_{R,T} = f(x, R, T)$. In particular, let its denoiser be $\hat{\epsilon}_\phi(z_t; t, c(x, R, T))$, where z_t is the diffused image of $x_{(R,T)}$ and $c()$ is the embedding of input view and camera extrinsics being conditioned on.

To recover the 3D model of the object, we first initialize a

NeRF $f_\xi(R, T)$ parameterized by ξ, which renders images of the object from camera pose (R, T). Given a sampled (R, T), we can render the generated view $\hat{x}$ from our NeRF $f_\xi(R, T)$. We can then diffuse $\hat{x}$ and use SDS guidance to update the parameters ξ. The SDS loss is defined as in [16]:

$$\nabla_\xi \mathcal{L}_{\text{SDS}}(\phi, \hat{x} = f_\xi(R, T)) = \mathbb{E}_{t,\epsilon}[w(t)(\hat{\epsilon}_\phi(z_t; t, c(x, R, T)) - \epsilon)\frac{\partial \hat{x}}{\partial \theta}]. \tag{1}$$

Intuitively, this loss provides an update direction that moves the generated image to higher-density regions following the score function of the diffusion model. Once the NeRF is trained, we export it to a textured mesh for faster differentiable rendering [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_{θ_1} and g_{θ_2}. The set transformer design is drawn from [10], 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_{θ_2} generates the rotation quaternions conditional on the location vectors generated by g_{θ_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^{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 [3] to update the layout generator in pixel space. That is, we train an image discriminator D_η 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 [4, 15, 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 an image crop I_{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 size of the crop 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_{real}. The discriminator's goal is to distinguish I_{fake} from I_{real}. In our experiments, we found that a relatively large crop, i.e. $s = 95\%$, works best. Also, we fix n to be the number of instances in the input scene during training.

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_η, we add an auxiliary pose prediction task as regularization. This auxiliary loss is de-

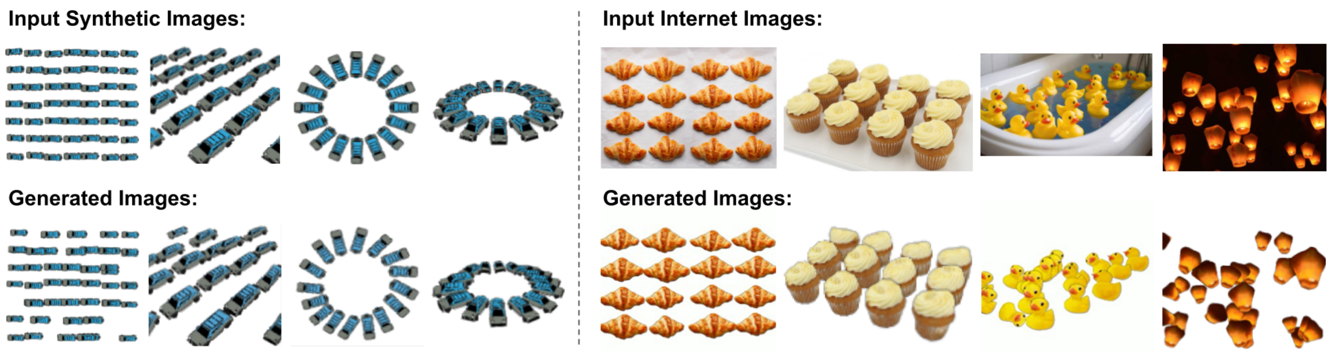

Figure 2. Learning from both synthetic (left) and real-world (right) images. Given a single 2D image containing similar objects arranged in a coherent layout, our proposed method can learn the underlying layout distribution in 3D space. The top row represents the input images whereas the bottom row are generated images using reconstructed geometry and learned layout parameters. At test time, our model can generate diverse layout parameters consistent with the input scene.

fined as

$$\mathcal{L}_{\text{pose}}(\eta) = \mathcal{L}_{\text{cham}}(g_{\text{GS}}(\hat{R}), g_{\text{GS}}(R)), \tag{2}$$

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

$$\mathcal{L}_{\text{cham}}(A, B) = \sum_{a \in A} \min_{b \in B} ||a - b||_2^2 + \sum_{b \in B} \min_{a \in A} ||a - b||_2^2, \tag{3}$$

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

In addition to the image discriminator D_η, 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) [8] to augment I_{real} and I_{fake}. In particular, we found that adding random noise with a somewhat large probability (e.g. 0.5) works well.

Training Objective. Similar to [14, 27], we use a binary cross entropy loss as the GAN training objective, with a regularization term on the gradient of the discriminator:

$$\begin{aligned}\mathcal{L}_{\text{adv}}(\theta, \eta, I) = {} & \mathbb{E}_{z, c \sim C}[h(D_\eta(f(g_\theta(z), c)))] \\ & + \mathbb{E}_C[h(-D_\eta(I_{\text{real}})) - \lambda_{reg}||\nabla D_\eta(I_{\text{real}})||^2],\end{aligned} \tag{4}$$

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 directly supervise location generation:

$$\mathcal{L}_{\text{loc}}(\theta_1) = \mathcal{L}_{\text{cham}}(\{x_i, y_i\}_{i=1}^n, \{\hat{x}_i, \hat{y}_i\}_{i=1}^n). \tag{5}$$

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 of five terms:

$$\begin{aligned}\mathcal{L}(\theta, \eta, I) = {} & \mathcal{L}_{\text{adv}}(\theta, \eta, I) + \lambda_{\text{mask}}\mathcal{L}_{\text{adv}}(\theta, \eta_{\text{mask}}, I) + \\ & \lambda_{\text{pose}}\mathcal{L}_{\text{pose}}(\eta) + \lambda_{\text{loc}}\mathcal{L}_{\text{loc}}(\theta_1) + \lambda_{\text{off-scene}}\mathcal{L}_{\text{off-scene}}(\theta_1).\end{aligned} \tag{6}$$

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 (6) are specified as $\lambda_{\text{reg}} = 10, \lambda_{\text{mask}} = 0.1, \lambda_{\text{pose}} = 1, \lambda_{\text{loc}} = 10$, and $\lambda_{\text{off-scene}} = 10$. The input noise has dimension 3, as it matches with the 3-dimensional x, y, z coordinates. The transformer architecture is adopted from [26] with 256 dimension and 4 attention heads. The backbone for both discriminators is adapted from GIRAFFE [14]. To provide a better initialization, we pretrain the first set transformer $g_{\theta_1}(z)$ using Chamfer loss similar to Equation (5). This avoids heavy occlusion at the start of training. We use an

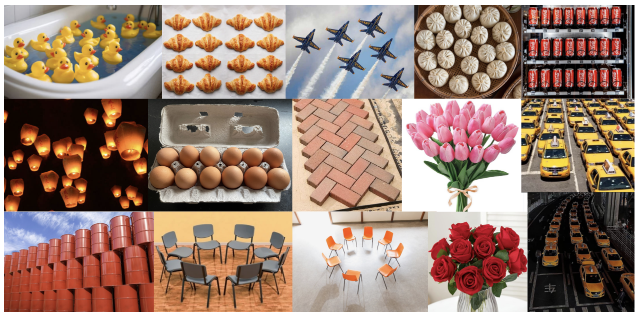
Figure 3. Examples of in-the-wild images from the Internet with diverse layouts and objects.

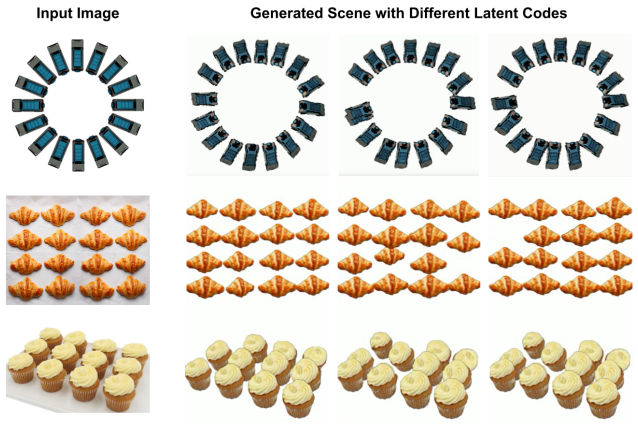

Figure 4. **Layout Interpolation.** Interpolation in the latent space allows us to generate diverse samples from the learned layout distribution. The leftmost column shows the input images and the right three columns show generated layout from different noises.

Adam [9] optimizer for the generator and RMSprop [5] optimizers for the two discriminators, with learning rates $3e^{-5}$ and $1e^{-4}$ respectively. Since 3D rendering takes significant VRAM space, we found that using gradient accumulation [6] to mimic a small batch of 16 is quite effective.

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 2. 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 shown in Figure 3. 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 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 2, 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.

Diversity. To demonstrate diversity, we can interpolate z in the latent space and sample multiple layout parameters and scenes. This allows us to see how the generated layout morphs from one to the other. Figure 4 illustrates three examples from both synthetic and real data. Layout parameters generated from different input noises are similar and largely adhere to the input image's layout distribution. However, some instabilities do exist due to stochastic GAN training and we are actively working on improving the results.

Extrapolation. To demonstrate generalizability, we can extrapolate on the number of instances within the scene. This ability comes from the permutation-equivariant design of our generator. Although the model is only trained on an image with a fixed number of instances, it can still produce faithful layout for different number 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 5 demonstrates four examples from different input scenes. Most notably, when we generate 6 instances or 20 instances in the synthetic circular layout case, the generated image still looks like a 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 lay-

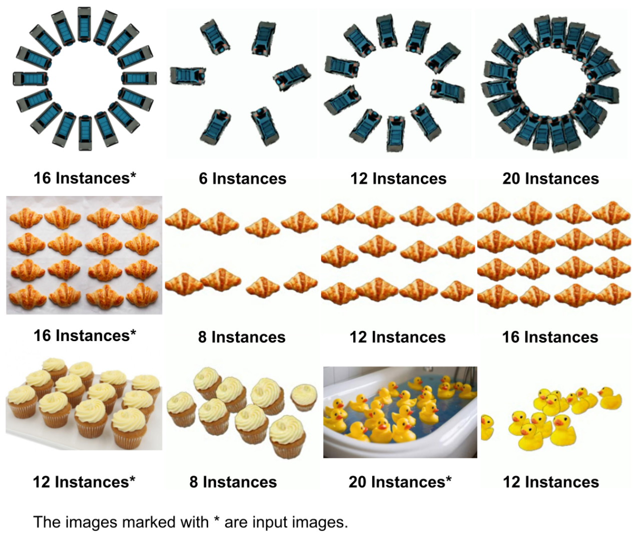

Figure 5. **Layout Extrapolation.** Extrapolating on the number of instances. Although the generator only learns from a single image, and hence a fixed layout, it is able to extrapolate to different numbers of instances while maintaining the layout patterns. Here, all images are marked with its number of instances in the scene, with input images marked with asterisk.

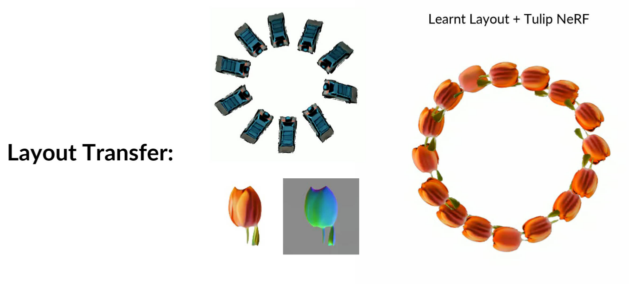

Figure 6. **Layout Transfer.** Transferring learned layout to other geometries. In this case, we can transfer a circle of cars to a flower wreath.

out, this generalizing ability is quite impressive.

Other Applications. Further, our method can have many downstream applications. Since layout parameters and the geometry are disentangled, we can transfer learned layout to other geometries, shown in Figure 6. This allows us to creatively generate many 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 importance and 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 3D scene layout distribution 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] 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

[3] 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

[4] 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

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

[6] 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. 6

[7] 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

[8] 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

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 6

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

[11] 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

[12] 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

[13] 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

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

[15] 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

[16] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022. 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. 4

[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