# Anchored Generation: Controllable Scene Variation via 3D-Aware Object Latents

Donghoon Ahn[*1] Mohammadhossein Momeni[*1] Taesung Park[2] Alexei A. Efros[1]
[1]UC Berkeley [2]Reve

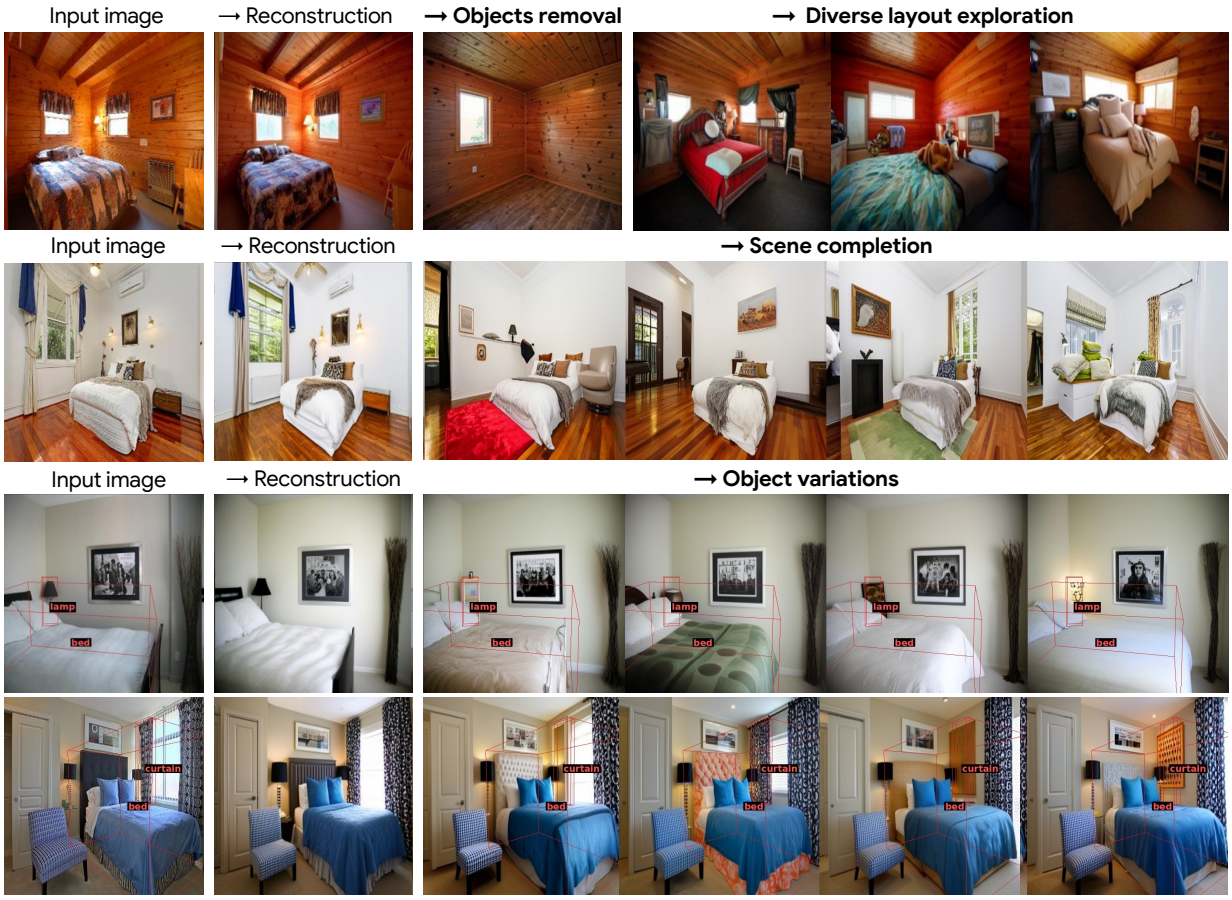


Figure 1. **Anchored generation. Top:** Starting from a reference image, users can preserve selected objects or layout elements, remove others, and explore diverse alternative scene layouts. **Mid:** Given only part of a scene, our method completes the remaining content while preserving the anchored objects and maintaining global consistency. **Bottom:** Our method can resample selected objects while keeping the rest of the scene fixed, enabling iterative exploration of plausible object variations.

## Abstract

*We introduce a framework that represents images as editable and disentangled scene latents and learns a generative model over this latent space. This allows users to resample only the latent factors they wish to change or explore, such as a subset of objects, the background, or camera parameters, while keeping the remaining factors fixed to obtain consistent scene variations. Within this framework, a user can encode an existing image into scene latents and iteratively reshape it through a sequence of natural operations such as moving, removing, or inserting objects. The same latent-space sampling mechanism also enables scene completion, object variation, and camera resampling, making exploratory generation a natural workflow.*

[*]Equal contribution.

# 1. Introduction

Modern generative models for visual synthesis have achieved remarkable success by modeling images in observation space, namely pixels. However, humans rarely seek to manipulate pixels directly. Instead, we infer, reason about, and edit the latent factors [3–5, 8, 12] that compose a scene, such as objects, appearance, and spatial relationships. Despite this, image generation is still dominated by text conditioning, where control is provided as descriptions of observations rather than editable representations of the underlying scene factors.

This reliance on text makes control indirect and nondeterministic. Natural language lacks a native coordinate system, making it difficult for users to specify exact spatial transformations. As a result, users must rely on repeated trial and error, iteratively adjusting prompts in the hope that the model's internal interpretation aligns with their intended layout. This makes precise, deterministic control difficult and limits the user's creative agency.

To address the lack of spatial control, layout-to-image methods such as GLIGEN [9] condition diffusion models on explicit spatial grounding signals. While these approaches provide stronger spatial control, they require the user to specify a complete layout before generation can begin. This leaves a gap in the creative workflow: the model cannot suggest plausible layouts or support progressive coexploration of the scene together with the user. Moreover, layout is not represented as an editable latent state that can be consistently revised over multiple steps. As a result, modifying a single bounding box often requires rerendering the full image, which may also change unrelated content and make iterative, variable-by-variable exploration difficult.

Rather than requiring the user to fully specify a scene upfront, our framework supports an iterative exploratory workflow in which users progressively discover and refine what they want to generate, receiving immediate visual feedback at each step. This supports a broad range of operations: varying individual objects or groups, adjusting global layout, swapping the background, changing camera parameters, or completing a partially specified scene. Together, these operations give a user a flexible vocabulary for reconstructing and discovering the scene they have in mind.

To support stable resampling across these operations, we use 3D-aware object tokens that encode a more complete object prior than visible image crops alone, performing object inference during encoding rather than at render time.

# 2. Modeling the Latents

## 2.1. General Framework

**Scene latents and object tokens.** We propose a framework that represents an image $\mathbf{x}$ by a structured set of scene latents $\mathcal{Y}$. In this work, we instantiate $\mathcal{Y}$ with object-level tokens and optional scene-level context variables:

$$\mathcal{Y} = \left(\mathbf{g}, \{\mathbf{o}^i\}_{i=1}^{N_{\max}}\right), \qquad \mathbf{o}^i = (\mathbf{p}^i, \mathbf{f}^i, z^i). \tag{1}$$

Each object token $\mathbf{o}^i$ corresponds to a meaningful element in the scene. Here, $\mathbf{p}^i$ denotes spatial attributes such as location and size, $\mathbf{f}^i$ denotes appearance, $z^i$ indicates whether the slot is occupied, and $N_{\max}$ denotes the maximum number of object slots. The scene-level variable $\mathbf{g}$ contains global factors shared across objects, such as camera or background information.

**Encoding an image into scene latents.** Given an input image, we extract scene latents using pretrained perception models: $\mathcal{Y} = E_\psi(\mathbf{x})$. Object detectors or segmentors provide object slots and spatial attributes, while vision models provide appearance features. Scene-level context variables are extracted when needed.

**Editing operations.** Because $\mathcal{Y}$ is structured, editing operates on latent variables rather than pixels. For example, one can remove or move an object, or vary the background while preserving the foreground.

**Decoding scene latents into images.** We train a decoder that renders an image from the scene latents:

$$\hat{\mathbf{x}} = D_\theta(\mathcal{Y}). \tag{2}$$

In our framework, $D_\theta$ is instantiated as a diffusion model conditioned on $\mathcal{Y}$. Given the latents extracted from an image, the decoder is trained to reconstruct the corresponding image, such that $\hat{\mathbf{x}} \approx \mathbf{x}$.

**Modeling the scene latents.** After training the decoder, we learn a generative prior $G_\phi$ over the same latent space $p_\phi(\mathcal{Y})$. This prior enables unconditional generation by sampling the full scene latent set, as well as conditional editing by fixing some variables and resampling others:

$$\mathcal{Y}_\mathcal{R} \sim p_\phi\left(\mathcal{Y}_\mathcal{R} \mid \mathcal{Y}_\mathcal{F}\right), \tag{3}$$

where $\mathcal{Y}_\mathcal{F}$ and $\mathcal{Y}_\mathcal{R}$ denote fixed and resampled subsets of scene latents, respectively. The completed latent set is then rendered by $D_\theta$.

## 2.2. 3D Object Tokens

We represent each object using a 3D-aware token extracted from a pretrained image-to-3D model, so that the decoder conditions on a complete object representation rather than inferring missing structure at render time.

**Implementation.** We instantiate the object component of $\mathcal{Y}$ with 3D-aware object tokens. For each detected object slot $i$, we define

$$\mathbf{o}^i_{\text{3D}} = \left(\mathbf{p}^i_{\text{3D}}, \mathbf{f}^i_{\text{3D}}, z^i\right), \tag{4}$$

where $z^i$ is the presence variable, $\mathbf{p}^i_{3D}$ denotes 3D spatial attributes, and $\mathbf{f}^i_{3D}$ denotes a 3D-aware appearance representation. The spatial attribute is represented by a 3D layout descriptor relative to the camera:

$$\mathbf{p}^i_{3D} = \left(\mathbf{t}^i, \mathbf{q}^i, \mathbf{s}^i\right), \tag{5}$$

where $\mathbf{t}^i \in \mathbb{R}^3$, $\mathbf{q}^i \in \mathbb{R}^4$, and $\mathbf{s}^i \in \mathbb{R}^3$ denote translation, quaternion rotation, and anisotropic scale, respectively.

For appearance, we use the intermediate object representation from SAM3D. We do not condition the decoder on the final explicit 3D output, such as a mesh or Gaussian splats. Instead, we use the internal representation that SAM3D decodes into these outputs, since it contains rich information about the object's 3D shape and appearance while remaining suitable as a compact conditioning signal.

Concretely, given an object mask or crop $\mathbf{m}^i$, SAM3D produces object-level latent features

$$\mathbf{H}^i_{3D} = \mathrm{SAM3D}(\mathbf{x}, \mathbf{m}^i), \tag{6}$$

which we compress into $K$ appearance tokens via attention pooling:

$$\mathbf{f}^i_{3D} = \mathrm{AttnPool}\left(\mathbf{H}^i_{3D}\right). \tag{7}$$

These tokens encode a more complete object-level representation than visible image crops alone, since they are inferred from a 3D generative prior.

**Scene-level context.** In addition to object tokens, the full scene latent set $\mathcal{Y}$ includes scene-level context variables

$$\mathbf{g} = (\mathbf{c}, \mathbf{b}), \tag{8}$$

where $\mathbf{c} = (f_x, f_y)$ denotes camera intrinsics and $\mathbf{b}$ denotes a global background token. The camera intrinsics are needed because 3D object tokens specify pose and scale in camera coordinates. The background token is a learned embedding that cross-attends to DINOv2 patches.

Thus, in our 3D instantiation, the full latent set is

$$\mathcal{Y}_{3D} = \left(\mathbf{g}, \{\mathbf{o}^i_{3D}\}_{i=1}^{N_{\max}}\right). \tag{9}$$

### 2.3. Structured Latent Prior

Given the 3D scene latent representation $\mathcal{Y}_{3D}$, we train a transformer prior $G_\phi$ to model $p_\phi(\mathcal{Y}_{3D})$. Since $\mathcal{Y}_{3D}$ contains both object latents and global context, the prior learns the distribution over object layout, object appearance, camera, and background, enabling joint control over all scene factors during anchored generation. Samples from this prior can be directly rendered by the diffusion decoder.

**Hybrid diffusion prior.** For the prior $G_\phi$, we use a hybrid discrete–continuous diffusion process over the scene latents. For each continuous variable $\mathbf{u}$, including object layout, appearance, and scene-level context, we use a rectified-flow forward process [1, 10, 11]

$$\mathbf{u}_\tau = \tau \mathbf{u}_0 + (1-\tau)\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I). \tag{10}$$

For the discrete presence variable $z$, we use an absorbing masking process [2, 6, 7, 13]

$$z_\tau = \begin{cases} z_0, & \text{with probability } \tau, \\ [\mathsf{M}], & \text{with probability } 1-\tau. \end{cases} \tag{11}$$

To support arbitrary conditioning, we follow a diffusion-forcing-style training scheme and assign each latent variable its own random diffusion time $\tau_j \sim \mathcal{U}(0,1)$. This exposes the model to mixed clean and noisy latent configurations during training, allowing the same prior to support both full-scene generation and conditional resampling of selected latent variables at inference time.

**Anchored generation as a unified view of generation and editing.** At inference time, generation and editing are both controlled by the initial amount of noise injected into each latent variable. Let $\alpha_j \in [0,1]$ denote the initial time for variable $j$: $\alpha_j = 1$ keeps the variable fixed, while $\alpha_j = 0$ resamples it from noise. Sampling then follows

$$\tau_j(\rho) = \alpha_j + (1-\alpha_j)\rho, \qquad \rho \in [0,1]. \tag{12}$$

Setting all $\alpha_j = 0$ yields unconditional generation, while editing fixes a subset of variables and resamples the rest. The latent set is rendered as $\hat{\mathbf{x}} = D_\theta\left(\mathcal{Y}_\mathcal{F} \cup \mathcal{Y}_\mathcal{R}\right)$. We refer to this unified procedure as anchored generation, which preserves selected scene factors while resampling the others.

## 3. Experiments

We evaluate our framework from two complementary perspectives. First, we showcase applications of anchored generation for flexible scene manipulation. Second, we analyze how individual latent components affect the rendered image, showing that our representation provides control over scene attributes and a useful degree of disentanglement.

### 3.1. Implementation Details

We train our encoder, decoder, and prior on a ∼97K-image subset of LSUN Bedrooms. For the encoder $E_\psi$, we run the SAM3D pipeline to obtain, for each detected object, a 3D bounding box, a pooled appearance feature, and camera intrinsics $\mathbf{K}$. For the diffusion decoder $D_\theta$, we use FLUX.2-Klein-4B-base as the backbone and inject a cross-attention adapter after each MM-attention layer to condition on the scene tokens. For the latent prior $G_\phi$, we use a 12-layer Transformer with 118M parameters.

### 3.2. Anchored Generation

As shown in Fig. 1, a user starts from a reference image, which is first encoded into scene latents and reconstructed. From this starting point, different anchored generation operations can be applied. Removing objects and resampling the layout produces diverse scene arrangements while keeping the background consistent. Anchoring a subset of objects and resampling the rest enables scene completion, filling


(a) Object removal (b) Object replacement

(c) Object movement and rotation (d) Camera control

Figure 2. **Disentangled scene latents. (a) Object removal.** Removing an object token preserves the others and naturally removes associated lighting or reflections in previously occluded regions. **(b) Object replacement.** An object can be replaced by swapping its appearance feature and box size with those from another image. **(c) Object movement and rotation.** An object can be repositioned or rotated while leaving the rest of the scene largely unchanged, even without paired image-editing supervision. **(d) Camera control.** The decoder also conditions on the intrinsic matrix $K$, which can be resampled or manually controlled.

in plausible content consistent with the anchored context. Resampling individual objects produces coherent variations that blend naturally with the surrounding scene, including appropriate shadows and lighting.

### 3.3. Controllability and Disentanglement

We next study how individual latent components affect the rendered image. In particular, we vary one component at a time while keeping the others anchored. For example, we test object removal (Fig. 2) by setting the presence variable $z^i \leftarrow 0$. Interestingly, when we remove a lamp from the scene, the associated lighting effects also disappear. This suggests that the decoder captures object-scene interactions rather than treating each object independently. In contrast, inpainting would require manually specifying an editing region large enough to cover both the lamp and its lighting effects, which is less intuitive.

We also demonstrate object insertion by adding an object appearance feature $\mathbf{f}^i$ extracted from another image, as well as object swapping by replacing the current object's appearance with that of another object. In addition, we show object repositioning and control of camera intrinsics. Across these examples, modifying one latent component preserves the others and mainly affects the intended scene attribute.

We do not claim superiority in identity-preserving image editing. Rather, our claim is that the decoder naturally learns a plausibly disentangled object-centric latent space, even without paired image-editing supervision. This makes the representation well suited for exploratory generation, where users can vary layout, move, remove, or insert objects, or adjust camera intrinsics while preserving others.

## 4. Conclusion

In this work, we propose anchored generation, a framework for iterative scene exploration through a structured object-level latent space. Our framework first constructs a structured scene latent space with 3D-aware object tokens and global scene context, and then learns a prior over these latent variables. We believe this direction suggests a paradigm shift: rather than treating editing and generation as separate tasks, generative models can directly model controllable units, making exploratory generation a natural and mainstream workflow.

Our current framework has several limitations. The model is trained on LSUN Bedrooms, and it is unclear how well the approach generalizes to more diverse scenes. Reconstruction quality is imperfect, and the encoder inherits the detection and segmentation limits of the SAM3D pipeline. We hope this work opens the door to future exploration of perception-oriented encoders as active participants in generation, in which richer scene understanding leads directly to more controllable synthesis.

## References

[1] Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for

flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. 3

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021. 3

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2

[4] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $\beta$-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[5] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. 2

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022. 3

[7] Chunsan Hong, Sanghyun Lee, and Jong Chul Ye. Unifying masked diffusion models with various generation orders and beyond. *arXiv preprint arXiv:2602.02112*, 2026. 3

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[9] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22511–22521, 2023. 2

[10] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 3

[11] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 3

[12] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020. 2

[13] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024. 3