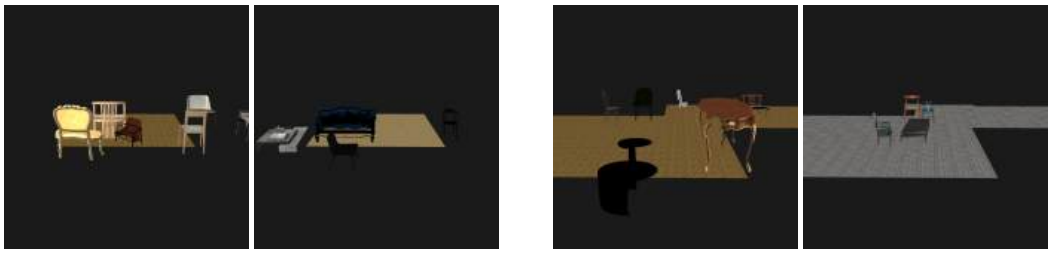
Vibescene: Learning to Generate 3D Scenes by Training on 2D Data

Anthony Baez
MIT
acbaez@mit.edu

Lucas De Bonet
MIT
ldebonet@mit.edu

Mehek Gosalia
MIT
mehek@mit.edu



Trained on 3D Data Baseline (50 3D scenes)

Trained on Mixed 2D-3D with Different Noise Schedule (50 3D scenes, 400 2D-derived scenes)

Figure 1: Top-down and orthographic renders of generated 3D scenes from Vibescene

Abstract

The diversity and quality of 3D indoor scenes constructed by generative models are often hindered by the scarcity of comprehensive 3D training data. In contrast, 2D images of scenes are abundant but carry less spatial and visual information. To bridge this gap, we introduce Vibescene, a multi-step framework that uses both abundant 2D image data and limited 3D scene data to train a robust 3D scene generation model. Our approach treats features derived from 2D scenes as a form of corrupted, incomplete 3D data. To this end, we introduce a novel conditional latent Denoising Diffusion Probabilistic Model (DDPM), named Embed2PC, for translating 2D semantic and stylistic features into 3D geometric latents (encoded point clouds). We then use a modified version of DiffuScene to decode and denoise this representation into a rendered 3D scene. We also explore different methods to mix 2D "noisy" data and 3D "clean" data in the training set to improve the quality of the 3D scenes. We found that Embed2PC was able to create meaningful encoded point clouds and that Vibescene using a 90-10 2D/3D data split and different noise schedule for 2D and 3D data performed the best on FID and KID compared to a dataset of only 3D data that contained the same number of 3D scenes. This demonstrates the potential of this framing and model architecture.

1 Introduction & Background

The creation of rich, 3D scenes is critical in fields such as gaming, virtual reality, and architectural design, yet is traditionally extremely labor-intensive and computationally expensive. Consequently, automated scene synthesis has become an important area of research in computer graphics and vision, aiming for more efficient and controlled methods to visualize 3D spaces. Early generative approaches for this task include Generative Adversarial Networks (GANs) [24] and Variational Autoencoders (VAEs) [18]. More recently, diffusion models [8] have emerged as the state-of-the-art, owning to their ability to produce high-quality and diverse generative outputs.

1.1 Denoising Diffusion and Latent Diffusion Models

Denoising Diffusion Probabilistic Models (DDPMs) learn a data distribution $q(x_0)$ by reversing a gradual noising process [8]. The "forward process" incrementally adds Gaussian noise to data x_0 over T steps, defined by $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$. At any step t, x_t can be sampled directly as $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ and $\epsilon \sim \mathcal{N}(0, I)$. The "reverse process" $p_\theta(x_{0:T})$ generates data by starting from Gaussian noise $p(x_T)$ and iteratively sampling $p_\theta(x_{t-1}|x_t)$ using a neural network $\epsilon_\theta(x_t, t)$ that predicts the added noise ϵ. A common training objective is

$$L_{simple}(\theta) = \mathbb{E}_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2],$$

which trains the network to denoise inputs and recover the original data distribution. Repeated application of these denoising steps enables the generation of new samples starting from x_T.

While effective, DDPMs applied directly to high-dimensional data (like raw 3D scene data or point clouds) can be computationally expensive. Latent Diffusion Models (LDMs) address this by performing diffusion in a lower-dimensional latent space learned by a pretrained autoencoder [21]. First, an autoencoder (encoder $\mathcal{E}$, decoder $\mathcal{D}$) compresses data x into a latent code $z = \mathcal{E}(x)$. Then, a diffusion model is trained in this latent space, with the objective:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}[||\epsilon - \epsilon_\theta(z_t, t)||^2]$$

where $z_t = \sqrt{\bar{\alpha}_t}\mathcal{E}(x) + \sqrt{1-\bar{\alpha}_t}\epsilon$. The denoised latent can then be decoded to produce a sample $\hat{x} = \mathcal{D}(\hat{z})$, approximating the original data distribution. This significantly reduces computational costs for training and inference.

In this project we employ latent diffusion in two different ways: for the generation of 3D scene arrangements, and also for generation of point-cloud latents used to retrieve 3D assets to place in the scenes.

1.2 Latent Diffusion Models for 3D Scene Denoising

Latent diffusion models can be applied to scene generation by casting the problem of organizing the contents of the scenes as denoising [22]. Rather than generating 3D models to place in the scene from scratch, scene denoising involves denoising the placement of existing 3D models within a space. The problem is framed as denoising the relative positions of the assets, with high noise corresponding to cluttered and unrealistic layouts and low noise corresponding to organized and realistic layouts. DiffuScene, in particular, used latent diffusion models to denoise autoencoded visual features of the assets, and then use these denoised features to select which assets are placed in the scene.

1.3 2D and 3D Scene Data

Despite the success of these diffusion models, a primary challenge in applying these models to 3D scene generation is their reliance on large-scale, high-fidelity 3D datasets. The curation of such datasets is difficult and costly due to the need for high levels of artistic expertise and detail, with visual quality being significantly impacted by available computational resources [5]. This presents a major bottleneck to progress within the field.

2D image data of such scenes, however, is abundant and can be easily produced by various means, including drawings and photographs of real-life spaces. Additionally, 2D scene data can be easily generated by rendering already existing 3D scenes, and can be used to generate different images by rendering from different viewing angles.

Our work addresses how to mitigate this scarcity of data: we explore how abundant and easily obtainable 2D image data can be leveraged to train robust 3D scene generation models. We posit that information gleaned from 2D scenes, even if imperfect or incomplete from a 3D perspective, can still enhance the training of latent diffusion models for 3D environments, potentially surpassing the capabilities of models trained on limited 3D data alone.

We bridge the informational gap from 2D to 3D with the application of two techniques. The first is the use of scene-derived features. This provides necessary structure to the model to be able to learn this conversion. We also incorporate semantic information about the 2D scene to inform the

generation of the 3D scene. This is what leads to the use of latent diffusion. The second is the framing of our problem as learning to generate clean data (3D scenes) from corrupted data (2D scenes), due to information loss when generating a 2D scene from a 3D one. To address learning from corrupted examples, we apply a different denoising schedule to our 3D and 2D data to handle the corruption.

Our contributions are summarized as follows:

- We introduce Vibescene, a novel multi-step method to generate 3D scenes by training on 2D images of scenes
- We create Embed2PC, a latent diffusion model to convert semantic 2D information into 3D autoencoded pointclouds for placing textures of objects into scenes
- We frame the conversion of 2D to 3D scenes as a 'data corruption' problem, and apply known methods to address this corruption

2 Related Work

2.1 3D Scene Generative Models

Among the early generative approaches explored for 3D scene generation were Generative Adversarial Networks (GANs) [17] [1]. GAN methods stood out for their ability to create high quality, easy to sample representations, but also faced weaknesses in mode collapse and diversity. Transformers have been applied in 3D scene generation. LegoNet, used a transformer to iteratively denoise the arrangement of furniture in a scene [23]. They model the scene and its contents using features such as the floor plan, spatial information, and class labels.

Diffusion models, however, have recently become some of the best performing models at 3D scene generation, owing to their strengths of stability, mode coverage, and sample quality. MiDiffusion, a mixed continuous-discrete diffusion-based model, used 2D floor plans and geometric and semantic attributes of assets to generate and denoise 3D scenes [10]. To generate furniture for a floor plan, the model uses learned associations between floor plans and furniture at training. DiffuScene, another diffusion-based model, takes as input 3D features that describe the contents and arrangement of a room and generates a denoised 3D scene [22]. They frame the creation of a 3D scene as scene arrangement as 'scene denoising', as the input contains an unordered set of objects. CommonScenes uses a denoising diffusion model conditioned on a scene graph, a graph that describes the spatial relationships between the objects in the room [25]. They use a latent diffusion model conditioned on the layout to learn the spatial information of the objects, but do not include semantic or style information.

Conditional scene generation can also occur where the objects, layout, and entire scene is generated by a diffusion model. This can be conditioned on a 2D reference, eliminating the need for 3D object references or assets [11]. This previous work relies on the 2D images in order to conditionally generate scenes, whereas we seek to use 2D images as training data from which we can learn to unconditionally generate 3D scenes. Additionally, diffusion models can generate complex room layouts in an unconditioned manner, including the walls and objects in them [12]. Our method differs from previous methods in that we train a diffusion model conditioned on the semantic information of 2D objects to retrieve the 3D objects for the 3D scene. We also directly incorporate semantic information into our modified scene denoising model. We also only use 2D images as our condition, not specific features such as layout, scene graph, or labels.

2.2 Learning Clean Distributions from Corrupted Data

Training generative models when only corrupted or incomplete data is available presents a significant challenge. Ambient Diffusion [4] introduces a method to learn a clean distribution $p_0(x_0)$ from corrupted samples $y_0 = Ax_0$, where A is a generic corruption process. The approach involves applying a further random corruption $\tilde{A}$ to Ax_0. A diffusion model h_θ is then trained on this noised version $\tilde{A}x_t$ (where x_t is x_0 plus diffusion noise $\sigma_t\eta$) to predict x_0, but the loss $J^{corr}(\theta) = \frac{1}{2}\mathbb{E}[||A(h_\theta(\tilde{A}, \tilde{A}x_t, t) - x_0)||^2]$ evaluates the prediction against the initially corrupted Ax_0. Because the model cannot distinguish between the original corruption A and the further corruption $\tilde{A}$ at input time, it is incentivized to reconstruct all parts of the clean data x_0.

Additionally, in instances where training data is corrupted by Gaussian noise (i.e., $X_{t_n} = X_0 + \sigma_{t_n} Z$), Daras et al. [3] introduce a framework highly relevant to our Vibescene approach. Their method uses a "Double Tweedie" formula for noise levels $t > t_n$: $\mathbb{E}[X_0|X_t = x_t] = \frac{\sigma_t^2}{\sigma_t^2 - \sigma_{t_n}^2}\mathbb{E}[X_{t_n}|X_t = x_t] - \frac{\sigma_{t_n}^2}{\sigma_t^2 - \sigma_{t_n}^2} x_t$, and a consistency loss for levels $t \leq t_n$. This handles the case where some data (such as our 2D-derived features) can be viewed as corrupted versions of the target data, as long as we can assume that the noise is gaussian-like. Their approach suggests using different noise schedules: full schedule for clean data, and higher noise levels ($t > t_n$) for corrupted data—a strategy we adopt when combining our limited 3D data with abundant 2D-derived features [1].

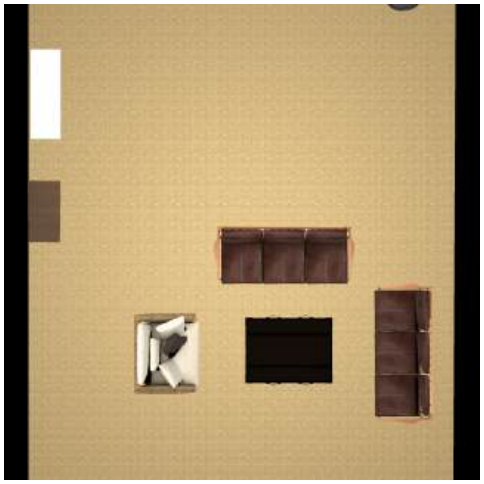
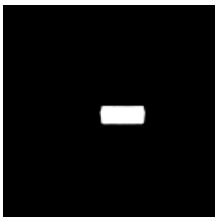


Figure 2: 2D scene generated from top-down render of living room from 3D-FRONT dataset

(a) Segment of sofa



(b) Mask of sofa

3 Method

Vibescene contains two parts, the 2D-to-3D scene conversion and the 3D scene denoising process. Within the 2D-to-3D scene conversion, we use a novel conditional latent diffusion model, which we call Embed2PC, to convert 2D semantic information to an autoencoded point cloud of the 3D object assets. The 3D denoising model is a DiffuScene model modified and retrained to also include visual data in the feature vectors of the scene. We call this model DiffuScene+. In our method, we first train DiffuScene+ on a small amount of actual 3D scenes. We then train on a dataset of 2D scenes. We then extract features from the 2D scenes to convert them to 3D features, then generate 3D scenes by training DiffuScene+ on these created 3D features.

3.1 3D Scene Feature Representation

We use the 3D-FRONT dataset for our 3D scenes. 3D-FRONT is a synthetic dataset that contains 18K textures for rooms, including dining rooms and living rooms, and 13K textures for furniture objects [5]. The scenes in the 3D-FRONT dataset are themselves arrangements of 3D furniture assets from the 3D-FUTURE dataset [6]. We use a subset of 900 dining and living rooms to be our ground truth 3D scene data and to generate our 2D scenes and features.

For each furniture item O_i within a scene, we extract a set of features that can be categorized as spatial, categorical, or semantic:

- **Centroid** $\mathbf{l}_{3D,i} \in \mathbb{R}^3$: The 3D coordinates of the object's center.
- **Size** $\mathbf{s}_{3D,i} \in \mathbb{R}^3$: The dimensions (e.g., width, height, depth) of the object, typically representing its bounding box extents from the centroid.

[1] This idea was also presented to us in Lecture 4 on training on corrupted data

- **Z-Axis Rotation Angle** $\mathbf{a}_{3D,i} \in \mathbb{R}$: The orientation of the object represented by a single angle of rotation around the vertical (Z) axis. The combination of $\mathbf{l}_{3D,i}, \mathbf{s}_{3D,i}$, and $a_{3D,i}$ defines the object's 3D bounding box.
- **Object Class** $\mathbf{c}_{3D,i} \in \{0,1\}^{22}$: A one-hot vector representing the type of furniture. These class categories are predefined and derived from the 3D-FUTURE dataset asset classifications.
- **Encoded Point Cloud** $\mathbf{p}_i \in \mathbb{R}^{64}$: An autoencoded representation of the object's 3D point cloud. When decoded, this feature reconstructs the object's external surface geometry, capturing its visual 3D form.
- **CLIP Embedding** $\mathbf{T}_{3D,i} \in \mathbb{R}^{512}$: A semantic feature vector generated by encoding a top-down render of each furniture object using CLIP [19].

These individual features are concatenated to form a comprehensive feature vector $\mathbf{o}_i$ for each object O_i:

$$\mathbf{o}_i = [\mathbf{l}_{3D,i}, \mathbf{s}_{3D,i}, \mathbf{a}_{3D,i}, \mathbf{c}_{3D,i}, \mathbf{p}_i, \mathbf{T}_{3D,i}] \tag{1}$$

Thus, each object O_i is represented by a vector $\mathbf{o}_{3D,i} \in \mathbb{R}^{3+3+1+22+64+512} = \mathbb{R}^{605}$. An entire 3D scene, consisting of N objects, is then represented as an unordered set of these feature vectors:

$$\mathbf{S}_{3D} = \{\mathbf{o}_{3D,1}, \mathbf{o}_{3D,2}, \ldots, \mathbf{o}_{3D,N}\} \tag{2}$$

This set-based representation is processed by our diffusion model.

3.2 2D Data Generation and Feature Representation

Our 2D scene data is generated by rendering top-down orthographic views of the 3D scenes from the 3D-FRONT dataset. Each object within these 2D renderings is then described by a feature vector. Each of these features is extracted from only the 2D image using segmentation to isolate individual furniture items for feature extraction.

For each object O_i identified in a 2D scene, its feature vector $\mathbf{o'}_i$ consists of spatial, categorical, and semantic components:

- **2D Centroid** $\mathbf{l}_{2D,i} \in \mathbb{R}^2$: The (X, Y) coordinates of the object's center in the top-down view.
- **2D Size** $\mathbf{s}_{2D,i} \in \mathbb{R}^2$: The planar dimensions (e.g., width, depth) of the object in the top-down view.
- **Z-Axis Rotation Angle** $\mathbf{a}_{2D,i} \in \mathbb{R}$: The object's orientation in the 2D plane, corresponding to its rotation around the Z-axis in the original 3D scene.
- **Object Class** $\mathbf{c}_{2D,i} \in \{0,1\}^{22}$: A one-hot vector representing the type of furniture (e.g., chair, table, sofa) out of 22 predefined categories relevant to our indoor scenes.
- **2D Semantic Embedding** $\mathbf{T}_{2D,i} \in \mathbb{R}^{896}$: A rich semantic descriptor derived from the object's 2D appearance and textual description. This itself is a concatenation of:
  - *CLIP Image Embedding* $\mathbf{E}_{\text{CLIP},i}$: An embedding generated using CLIP [19] from the 2D image of the isolated object.
  - *SBERT Text Embedding* $\mathbf{E}_{\text{SBERT},i}$: An embedding generated using SBERT [20] from a natural language description of the object (the description itself is generated by processing the object's 2D image with GPT-4o).

The concatenated 2D semantic embedding is $\mathbf{T_{2D}}_i = [\mathbf{E}_{\text{CLIP,i}}, \mathbf{E}_{\text{SBERT,i}}]$. The complete 2D feature vector for object O_i is then:

$$\mathbf{o}_i = [\mathbf{l}_{2D,i}, \mathbf{s}_{2D,i}, \mathbf{a_i}_{2D,i}, \mathbf{c}_{2D,i}, \mathbf{T}_{2D,i}] \tag{3}$$

Thus, each 2D object O_i is represented by a vector $\mathbf{o}_i \in \mathbb{R}^{2+2+1+22+896} = \mathbb{R}^{923}$.

To populate these 2D feature vectors:

1. **From 3D Ground Truth:** For training our 2D-to-3D feature conversion model Embed2PC, we generate "ground truth" 2D feature vectors directly from the 3D features. This is so the model can learn how to convert 2D to 3D data using corresponding 2D-3D pairs. The 3D

spatial features (centroid $\mathbf{l}$, size $\mathbf{s}$) are projected to 2D by taking their X and Y components to yield $\mathbf{l}_{2D,i}$ and $\mathbf{s}_{2D,i}$; the Z-axis rotation $a_{2\text{D},i}$ is retained. The object class $\mathbf{c}_{2D,i}$ is known from the 3D asset. Semantic features $\mathbf{T}_{2\text{D},i}$ are derived from the rendered 2D appearance of the corresponding 3D object.

2. **From Novel 2D Images:** To process arbitrary 2D scene images, we first segment individual objects using GroundingDINO [14]. For each segmented object, its 2D bounding box (providing position and size, which are then mapped to the same coordinate system as the projected features) is extracted using Segment Anything (SAM) [13]. The object class $\mathbf{c}_{2D,i}$ is obtained by performing zero-shot classification on the segmented object's image using GPT-4o against our predefined 22 furniture classes. The semantic features $\mathbf{T}_{\mathbf{2D}i}$ are generated from these 2D segments as described above (CLIP from the segment, GPT-4o description of the segment followed by SBERT).

Using these methods, we created corresponding 2D features for the 900 dining and living rooms and approximately 3,000 objects processed from the 3D-FRONT dataset.

3.3 Converting 2D Features to 3D Features

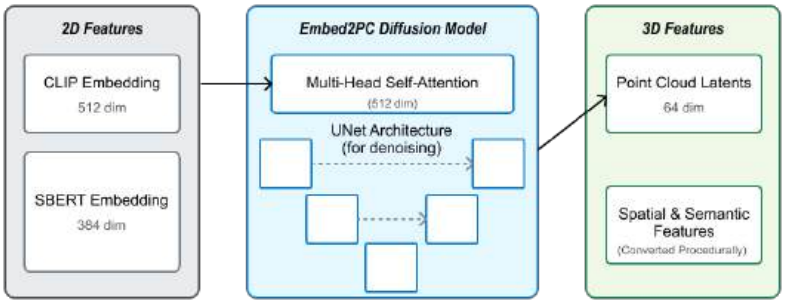


Figure 3: Embed2PC architecture, which generates 3D point cloud latents for use in the 3D features via conditional diffusion based on the stylistic embeddings from the 2D data.

2D render of ground truth 3D scene Decoded output from Embed2PC

Figure 4: Comparison of input and output of 2D render with Embed2PC after performing decoding and nearest neighbor search for the best asset

To convert the 2D bounding box $[\mathbf{l}_{2\text{D},i}, \mathbf{s}_{2\text{D},i}, \mathbf{a}_{\mathbf{i}2\text{D},i}]$ into a 3D bounding box $[\mathbf{l}_{3\text{D},i}, \mathbf{s}_{3\text{D},i}, \mathbf{a}_{3\text{D},i}]$, we tried to train a neural network to learn the conversion of the 2D to 3D bounding boxes conditioned on the 2D semantic information, but this method also proved ineffective. We also tried direct calculating using a projection from the 2D to 3D coordinates for the x and y axis, but we found that the 3D bounding boxes available in 3D-FRONT were not consistent with the actual location of the object. Additionally, the 3D and 2D bounding boxes used different coordinate systems that varied based

on the room size, since the scale for the 2D bounding boxes was in pixel-space. This prevented any consistent conversion to be learned between the 2D and 3D bounding boxes. In the end, we decided to pass in the ground truth 3D bounding boxes.

We also created a conditioned latent DDPM, which we call Embed2PC, to learn how to generate the the autoencoded point cloud $\mathbf{p}_i$ for each 3D feature by conditioning on the 2D semantic features $\mathbf{T}_{2D,i}$. A diffusion model is a natural choice for this task due to its ability to operate in latent spaces and preserve structural relationships between features that correspond to different points in space. Diffusion models have already been used to generate non-autoencoded 3D point clouds [15]. The autoencoded point clouds are used by DiffuScene+ to select the appropriate 3D models to placed in the scene by contrastive matching rather than complete reconstruction [22].

Embed2PC first uses a multi-head self-attention encoder to refine the passed in CLIP and SBERT embeddings. We found that applying this encoder to the concatenated embeddings with performed better than encoding them separately. It also used classifier-free guidance to improve the response to the presence of the text conditions [9]. The denoising model was a UNet that contained three up and down blocks and residual connections. Embed2PC also used a cosine noise schedule to improve stability during sampling. The condition was applied at each residual connection. More details about the training regime can be found in the Appendix.

3.4 3D Scene Denoising Latent Diffusion Model

The latent diffusion model we use to denoise 3D scenes is a modified DiffuScene [22] model we call DiffuScene+. Compared to DiffuScene, DiffuScene+ contains an additional component in the 3D feature vector input. We modified it with the addition of CLIP-embedded visual information representing each furniture object in the passed in feature vectors. We then retrained DiffuScene with our new feature vectors according to the original training regime.

Experimentally, we noticed that the 3D scene vectors produced by our 2D to 3D pipeline do not match the quality of the ground truth 3D scenes, and this is to be expected as the pipeline was itself only trained on a subset of an already small dataset of 3D scenes. Thus, we treat this 2D data as corrupted with Gaussian-like noise equivalent to the noise at timestep t_n. In training DiffuScene+, we only apply noise schedule steps $t_n < t < T$, where t is the current timestep and T is the last timestep, to this corrupted data [3]. This allows us include corrupted scene data along with clean scene data in the training set without training the model to produce corrupted scenes.

After training, the objects in the 3D scene are rendered by retrieving the most similar human-made asset to each of the feature vectors by performing nearest-neighbor search on cosine similarity on the point cloud and semantic embeddings. They are then placed in the scene based on the spatial components of the feature vector. The visualization is then created by a 3D render of this scene. Top-down rendered images of these generated 3D scenes are then compared to the ground truth 3D scenes by calculating FID and KID scores.

4 Experiments

Our experimental setup was designed to evaluate the efficacy of training a 3D scene generation model using 2D-derived data. We utilized a total of 900 dining room scenes from the 3D-FRONT dataset [5].

4.1 Embed2PC Training

The 900 scenes were divided into two distinct sets of 450 scenes each.

1. **3D Ground Truth Set (450 scenes):** Embed2PC was trained exclusively on the 3D and corresponding 2D features derived from this set. This separation ensures that Embed2PC does not remember the 3D scenes it will later convert from 2D scenes for training DiffuScene+.
2. **Simulated 2D Source Set (450 scenes):** This was the 2D dataset we used to evaluate our entire Vibescene framework. It was produced by taking 2D top-down renders of 3D scenes and used as the sole source of information about a scene.

4.2 Generation of Corrupted 3D Data

The 2D feature vectors derived from the Simulated 2D Source Set were then processed by inference by Embed2PC. This produced a dataset of 450 "corrupted" 3D scene features, representing the data obtained by converting from 2D.

4.3 DiffuScene+ Training

We then evaluated DiffuScene+ under four distinct training conditions using these datasets:

1. **3D Data Baseline:** DiffuScene+ was trained exclusively on 50 ground truth 3D scenes from the 3D Ground Truth Set. This acts as the baseline to determine whether the addition of 2D "corrupted" scene data improves 3D scene generation.

2. **2D Data:** DiffuScene+ was trained exclusively on the 450 "corrupted" 3D scene feature sets derived from the Simulated 2D Source Set.

3. **Mixed 2D-3D Data:** DiffuScene+ was trained on a dataset containing a naive combination 50 ground truth 3D scenes from the 3D Ground Truth Set and 400 "corrupted" 3D scenes to make a 90%-10% split.

4. **Mixed 2D-3D Data with Different Noise Schedule:** DiffuScene+ was trained on a dataset containing a naive combination 50 ground truth 3D scenes from the 3D Ground Truth Set and 400 "corrupted" 3D scenes to make a 90%-10% split. However, we apply noising time steps $t = 0$ to $T = 1000$ to the "clean" 3D data and noising time steps $t_n = 500$ to $T = 1000$ to the "clean" 3D data [4].

4.4 Evaluation

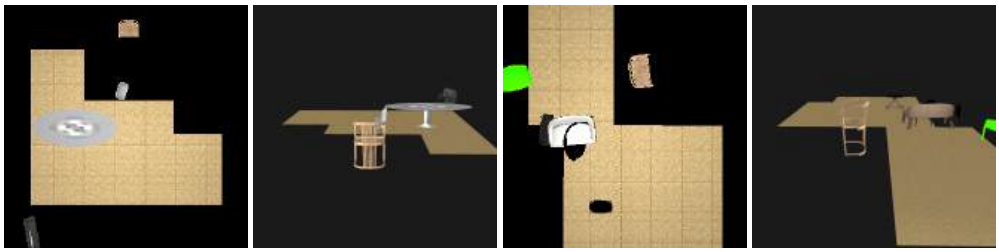
The denoised 3D scenes generated by DiffuScene+ were evaluated using Fréchet Inception Distance (FID) [7] and Kernel Inception Distance (KID) [2] to assess their quality and similarity to the distribution of ground truth 3D scenes. FID measures the distance between the feature distributions of real and generated samples assuming Gaussianity, while KID computes a kernel-based MMD score that remains unbiased even with limited sample sizes.

5 Results

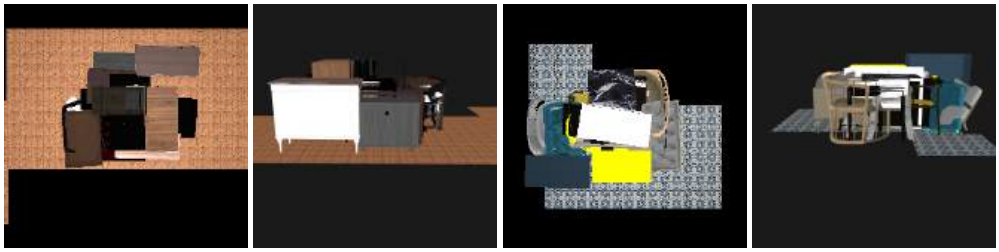
Dataset	Time Steps	FID ↓	KID ↓
3D Data Baseline	1000 All	144.14	0.0394
2D Data	1000 All	266.70	0.178
Mixed 2D-3D Data	1000 Corrupted/1000 Clean	202.67	0.103
Mixed 2D-3D Data	500 Corrupted/1000 Clean	**124.10**	**0.0249**

Table 1: Quantitative evaluation on the 3D-FRONT dataset comparing mixtures of 2D and 3D data and the effect of noise schedules.

Looking at Table 1, we can compare the performance of different mixtures of 2D and 3D data on FID and KID. In the order of worst to best performance, the 2D Data model, which was trained solely on 2D-derived data, achieved an FID of 266.70 and KID of 0.178; Mixed 2D-3D Data with the same noise schedule resulted in an FID of 202.67 and KID of 0.103; the pure 3D baseline achieved an FID of 144.14 and KID of 0.0394; and mixing 2D and 3D data with different noise schedules yielded the best performance, with an FID of 124.10 and KID of 0.0249.

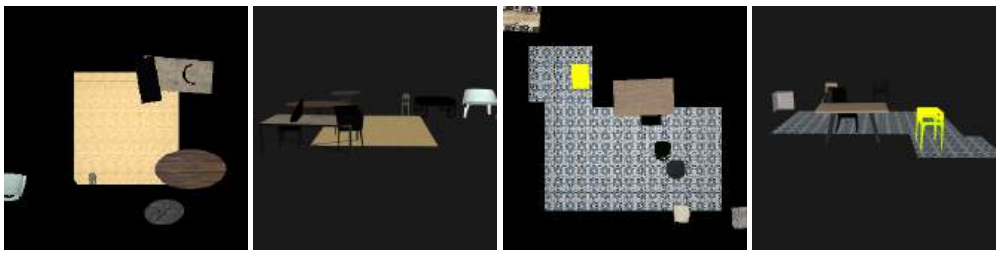


3D Data Baseline



2D Data

Mixed 2D-3D Data



Mixed 2D-3D Data with Different Noise Schedule

Figure 5: Top-down and orthographic renders of generated 3D scenes from Vibescene

6 Discussion

6.1 Quantitative Analysis

After analyzing the results, we can observe several important trends. First, the model trained solely on 2D-derived data performed the poorest, confirming our hypothesis that 2D data alone is insufficient for high-quality 3D generation. This suggests that 2D data lacks crucial spatial information in regards to recreating the 3D scene. When mixing 2D and 3D data with the same noise schedule (1000 time steps for both), we observed a substantial improvement, demonstrating the value of adding limited 3D training data. The next best performing data was the pure 3D baseline dataset. This was also also expected, as 3D scene data holds crucial information to the construction of 3D scenes. It also shows that the naive addition of 2D-derived scene data degrades the benefits achieved by the pure 3D data. Most significantly, applying different noise schedules to clean and corrupted data (500 time steps for corrupted 2D-derived data, 1000 for clean 3D data) yielded dramatically better results. This 39% reduction in FID and 76% reduction in KID compared to the same noise schedule validates our corrupted data framing and the effectiveness of the noise scheduling technique inspired by Daras et al. [3]. These results strongly suggest that treating 2D-derived features as corrupted 3D data and

applying appropriate noise schedules is an effective strategy for leveraging limited 3D data in scene generation, especially considering it outperforms generation on limited 3D scenes alone.

6.2 Visual Analysis

In the generations on DiffuScene+ trained with 2D-Only, the objects in the scene are crowded together and are overlapping in the middle of the scene, demonstrating a lack of ability to place the furniture in the scene properly. This likely is caused by the degree of corruption in the 2D-derived scene data. With mixed 2D-3D data, the generations have frequent overlapping objects, but happened less frequently than 2D-Only. This demonstrates a somewhat better spatial understanding of object locations and relationships. We attribute this to the small size of the dataset we used. The visual quality would likely benefit from scaling both the amount and quality of 2D and 3D data, such as using all the data in 3D FRONT. In the 3D data baseline, the furniture is not overlapping, but they are placed very sparsely. There are also fewer objects placed in the scene. This is likely due to having too few data points to learn proper spatial relationships. Finally, the Mixed 2D-3D Data with different noise schedules seem to have the most visually appealing layouts. The objects are neither overlapping nor very far away. In the rightmost images, there is also a chair placed at the desk in a realistic manner. However, like in all of the different data regimes, the layouts are not visually appealing or realistic.

6.3 Broader Implications

Our findings suggest that 2D data, when framed as corrupted 3D data and combined with appropriate noise scheduling, can effectively supplement limited 3D training sets. This approach has several implications for 3D scene generation. The improvements in FID and KID achieved by our mixed approach with differential noise scheduling demonstrate the viability of this method. Additionally, our results indicate that the integration between 2D and 3D data may be as important as the quantity of 3D examples. This is clear when incorporating actual 2D images into this pipeline as opposed to only 2D feature vectors, where the quality of segmentation is crucial to create useful forms of corrupted 3D data. The performance improvements observed using 50 3D scenes combined with 400 2D-derived scenes suggest a possible reduction in the data requirements for 3D scene generation. This approach could be relevant to other domains where high-dimensional data is limited but lower-dimensional projections are more accessible. As generative models continue to develop, our method offers one approach for integrating data across modalities with varying information densities.

6.4 Limitations and Future Work

We encountered various limitations in our method throughout the process, some of which made us have to reduce the scope of our method. When given top-down renderings of scenes, 2D bounding boxes of the segmented furniture can be inflated into 3D bounding boxes using a projection between different coordinate systems. In our specific training dataset, however, DiffuScene+ was trained using 3D bounding boxes that had slightly different spatial layouts than the actual outputted scenes, due to the scenes using additional 3D metadata about each furniture's scale and position when constructing the renders. Therefore, algorithmic reconstruction of the 3D bounding boxes from the ground truth 3D renders is not possible purely using the 2D data (without the 3D furniture metadata). We were limited on time to retrain DiffuScene+, so we had to use the ground truth 3D bounding boxes. With this modification, we are emulating the ideal case where a perfect mapping between 2D and 3D can be learned, which in reality would likely require both 2D bounding box data and 2D semantic information about each furniture piece to learn a predicted furniture height. Future work would include retraining DiffuScene+ with 3D data that would enable us to convert from 2D bounding boxes to 3D bounding boxes.

Another limitation in our method was the segmentation model. The segmentation model would sometimes combine two furniture objects into the same segment, or misclassify the type of furniture in a segment. Future work could include finetuning SAM on samples on our 3D-FRONT dataset or other similarly styled 2D scene images to improve the segmentation and labeling. This would significantly improve data quality, which would improve the performance of Vibescene.

We were also limited on compute and training time. From our empirical findings, training DiffuScene+ for 20 hours unlocks maximal performance on the scene generation task. However, because we

sought to train on multiple data configurations and needed to iteratively improve our method, we had to reduce this training time significantly to 4 hours. With either more compute or more time, we could fully train and sample from Vibescene or increase the size of our dataset to get the best possible results. We could also evaluate our model on real or synthetic 2D scene data not from 3D-FRONT to better assess performance on out-of-distribution data.

One possible new direction could be training Vibescene on 2D images not taken from the top-down perspective. This would enable us to sample 3D scenes from 2D images from any angle, including angles more similar to a human's view. While this would complicate the conversion of the 2D to 3D bounding box, we could also explore more various different methods to successfully convert from 2D to 3D bounding boxes, such as with more specialized neural networks that are informed by work in that field[16].

6.5 Conclusion

In this work, we created Vibescene, a multi-step framework to generate 3D scenes using 2D scenes in training. We did this by deriving 2D features, which described the spatial and style information of the objects of the scene, and converted them into 3D features using various methods, including a novel conditioned latent diffusion model, Embed2PC. We framed these 2D-derived 3D features as "corrupted" examples, and applied techniques such as mixing corrupted and clean examples in the dataset and using a differential noise schedule to train DiffuScene+, a modified version of DiffuScene. We found that Vibescene trained with a mixture of 2D and 3D scene data using a differential noise schedule outperformed a training set of only 2D and of only 3D data with the same number of 3D data examples. This demonstrates the effectiveness and robustness of using 2D scene data to supplement 3D scene data in training 3D scene generative models. Further work, however, is needed to make Vibescene fully reliant on 2D images and could extend its applicability to a broader range of use cases.

References

[1] Sherwin Bahmani, Jeong Joon Park, Despoina Paschalidou, Xingguang Yan, Gordon Wetzstein, Leonidas Guibas, and Andrea Tagliasacchi. Cc3d: Layout-conditioned generation of compositional 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7171–7181, 2023.

[2] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gansf, 2021.

[3] Giannis Daras, Alexandros G Dimakis, and Constantinos Daskalakis. Consistent diffusion meets tweedie: Training exact ambient diffusion models with noisy data. *arXiv preprint arXiv:2404.10177*, 2024.

[4] Giannis Daras, Kulin Shah, Yuval Dagan, Aravind Gollakota, Alexandros G. Dimakis, and Adam Klivans. Ambient diffusion: Learning clean distributions from corrupted data, 2023.

[5] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Jiaming Wang Cao Li, Zengqi Xun, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3d-front: 3d furnished rooms with layouts and semantics, 2021.

[6] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture, 2020.

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[9] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[10] Siyi Hu, Diego Martin Arroyo, Stephanie Debats, Fabian Manhardt, Luca Carlone, and Federico Tombari. Mixed diffusion for 3d indoor scene synthesis. *arXiv preprint arXiv:2405.21066*, 2024.

[11] Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. Midi: Multi-instance diffusion for single image to 3d scene generation. *arXiv preprint arXiv:2412.03558*, 2024.

[12] Xiaoliang Ju, Zhaoyang Huang, Yijin Li, Guofeng Zhang, Yu Qiao, and Hongsheng Li. Diffindscene: Diffusion-based high-quality 3d indoor scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4526–4535, 2024.

[13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[14] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024.

[15] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021.

[16] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[17] Thu H Nguyen-Phuoc, Christian Richardt, Long Mai, Yongliang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *Advances in neural information processing systems*, 33:6767–6778, 2020.

[18] Pulak Purkait, Christopher Zach, and Ian Reid. Sg-vae: Scene grammar variational autoencoder to generate new indoor scenes. In *European Conference on Computer Vision*, pages 155–171. Springer, 2020.

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[20] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[22] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20507–20518, 2024.

[23] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. Lego-net: Learning regular rearrangements of objects in rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19037–19047, 2023.

[24] Ming-Jia Yang, Yu-Xiao Guo, Bin Zhou, and Xin Tong. Indoor scene generation from a collection of semantic-segmented depth images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15203–15212, 2021.

[25] Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graph diffusion. *Advances in Neural Information Processing Systems*, 36:30026–30038, 2023.

A Appendix

A.1 Embed2PC Training Setup

The hidden dimensions of the UNet blocks were 1024, 2048, and 4096, respectively. The dimension of the multi-head self-attention encoder was 512. The text condition shared the same hidden dimension at each UNet block. Embed2PC was trained for 100 epochs with the Adam optimizer and a learning rate of 0.0005 with a linear schedule. 250 time steps was used in both training and inference. A guidance scale of 7.5 was used.

A.2 DiffuScene+ Training Setup

The model for each dataset was trained for 1500 epochs using the Adam optimizer with a learning rate of 0.0002. A step-based learning rate schedule was used, with the learning rate decaying by a factor of 0.5 every 2000 steps. All other training parameters are the same as [12]