arXiv:2504.12684v1 [cs.GR] 17 Apr 2025

SOPHY: Generating Simulation-Ready Objects with PHYsical Materials

Junyi Cao Evangelos Kalogerakis
Technical University of Crete

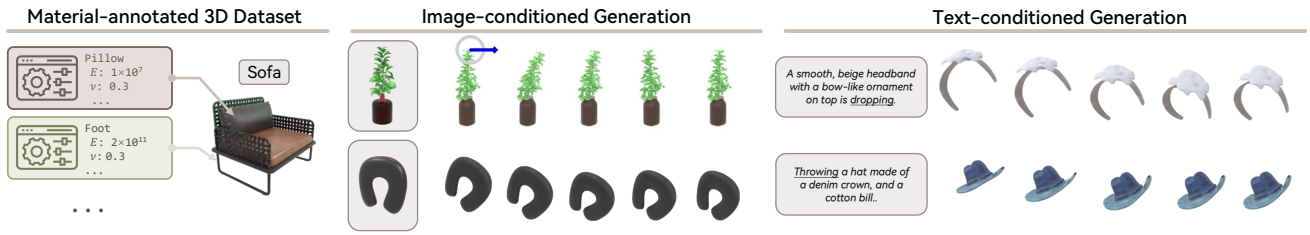

Figure 1. *Left:* We introduce a dataset of 3D objects enriched with physical material properties, *e.g.*, Young's modulus (E) and Poisson's ratio (ν), to capture their physical behavior under forces and interactions with other objects. *Middle*: We also introduce a physics-aware generative model of objects that can be used for 4D generation from a single image along with forces in an environment (*e.g.*, an external force for the plant shown in blue, collision with the ground for the pillow. *Right*: We also show text-to-4D generation as another application.

Abstract

We present SOPHY, a generative model for 3D physics-aware shape synthesis. Unlike existing 3D generative models that focus solely on static geometry or 4D models that produce physics-agnostic animations, our approach jointly synthesizes shape, texture, and material properties related to physics-grounded dynamics, making the generated objects ready for simulations and interactive, dynamic environments. To train our model, we introduce a dataset of 3D objects annotated with detailed physical material attributes, along with an annotation pipeline for efficient material annotation. Our method enables applications such as text-driven generation of interactive, physics-aware 3D objects and single-image reconstruction of physically plausible shapes. Furthermore, our experiments demonstrate that jointly modeling shape and material properties enhances the realism and fidelity of generated shapes, improving performance on generative geometry evaluation metrics. Our project page is available at: https://xjay18.github.io/SOPHY.

1. Introduction

Generating high-quality 3D assets that can be incorporated in interactive virtual worlds or manufactured into functional 3D objects is a fundamental challenge in generative AI and digital content creation. Unfortunately, current generative models of 3D shapes are limited to generating only static geometry and textures. Although there is significant effort in building 4D generative models for both geometry and motion synthesis, current approaches are limited to generating fixed animations and are incapable of producing motions resulting from complex, dynamic object interactions.

In this work, we introduce a diffusion-based generative model for physics-aware 3D object synthesis, capable of generating 3D assets with detailed shape, texture, and, most importantly, physical material attributes governing kinematic object deformations (*e.g.*, elastic deformations, plastic softening) and frictional interactions. Our approach produces physics-aware, interactive 3D objects, which are helpful for downstream applications such as physically based simulation, robotic interaction, and manufacturing. There are various challenges to developing such a generative approach. First, current 3D and 4D datasets do not contain physical material attributes for objects, making it hard to train, or even fine-tune such physics-aware generative models. Second, it is unclear how to jointly model the interplay of shape and material in 3D asset generation. Clearly, material attributes are not independent of geometric shapes *i.e.*, certain materials are associated with certain geometric structures, as also repeatedly observed in prior works of physical material prediction from shapes [1, 26, 28, 48].

Our work aims to address the above challenges. First, we introduce a semi-automatic pipeline for annotation, which combines VLM guidance with iterative expert (mechanical engineer) feedback for more efficient object annotation with material properties. Second, we introduce a diffusion-based generative model that builds upon recent advancements in latent diffusion models, yet incorporates novel insights on representing 3D objects as compact latent codes capturing shape geometry, albedo color, and material properties re-

lated to elasticity and plasticity, as well as capturing their interdependence through cross-attention blocks. We observe that modeling shape and material properties together enhances the realism and fidelity of the geometry in the generated shapes. Overall, our contributions are:

- We introduce a 3D dataset consisting of 3K objects and 15K parts annotated with physical material properties.
- We propose a novel autoencoder for representing physics-aware 3D objects into compact latent codes, along with a latent diffusion-based generative model that jointly synthesizes geometry, texture, and physical materials.
- We demonstrate applications of our generative framework on the synthesis of simulation-ready objects conditioned on text prompts or images.

2. Related Work

3D asset generation. Over the recent years, we have witnessed an explosion of 3D generative models capable of generating detailed geometry and texture. The advances have been driven by expressive neural representations, such as NeRFs [35, 41], Gaussian splats [20], signed distance fields [8, 39], occupancy fields [34], and numerous other representations for capturing shape or/and texture [11, 13, 14, 18, 27, 36, 40, 46, 59, 62] to name a few. We also refers readers to the recent surveys of 3D generative model [25, 56]. Unfortunately, the vast majority of existing 3D generative models neglect the physical material properties of the generated 3D objects, limiting their applicability to interactive and simulation-based environments. To address this gap, recent research has begun incorporating physical priors and constraints into the generative pipeline. For example, DiffuseBot [55] evaluates generated 3D robot models based on their simulation performance and refines the sampling distribution to favor more successful designs. Atlas3D [7] and Phys-Comp [12] enforce static equilibrium constraints during shape optimization, ensuring that generated objects remain stable under gravity. Despite these advances, all these prior works assume uniform or limited material properties, restricting their ability to generate diverse and physically intricate 3D assets. In contrast, our approach jointly optimizes 3D shape, texture, and material properties with a network that captures their natural dependencies.

Material-annotated 3D datasets. The advances in 3D generative models have also been led by the development of influential 3D datasets such as ShapeNet [5] and Objaverse [10]. More recent datasets, including ABO [9], Matsynth [52], and BlenderVault [31], incorporate surface texture information to train models for 3D generation and more plausible, photorealistic rendering. Despite this progress, a critical gap remains in the availability of detailed physical material information essential for accurately modeling material properties for physics-based simulations. While datasets such as ShapeNet-Mat [28], 3DCoMPaT [26], and its extensions [1, 48] attempt to address this limitation by providing part-level material labels, their annotations are coarse and unsuitable for direct use in physical simulators. To bridge this gap, we introduce a new dataset consisting of 3K objects spanning 12 shape categories, each annotated with precise part-level physical material properties.

4D content generation. 4D generation aims to create dynamic 3D content that aligns with input conditions such as images, text, or videos. Existing approaches primarily follow a data-driven pipeline, leveraging off-the-shelf image or video diffusion models to generate dynamic scenes. For example, DreamGaussian4D [44] adopts Stable Video Diffusion [3] to animate 3D Gaussian splats (3DGS) [20] reconstructed from a single image. STAG4D [60] initializes multi-view images using image-to-image diffusion [47] anchored on input video frames from a text-to-video module. However, these methods rely on pre-trained generative models that lack physical understanding, often resulting in physically unrealistic motions [2]. To address this, PhysGaussian [58] and PhysDreamer [63] integrate the Material Point Method (MPM) [17] with 3DGS to produce physics-aware dynamics. Building on these efforts, several concurrent studies have emerged recently, focusing on image-conditioned physical dynamics generation. Phy124 [29] obtains 3DGS from a single image, feeds them into an MPM simulator, and then renders the dynamic content. In PhysMotion [50], the visual quality is further enhanced with a diffusion-based video refinement module. However, these approaches often rely on manually assigned or predefined homogeneous material properties, failing to capture the diverse and heterogeneous nature of real-world materials. They also follow a two-stage pipeline, first generating 3D shapes and then assigning material properties, which can lead to inconsistencies between shape and materials. In contrast, our method introduces an end-to-end approach for simulation-ready object generation. By jointly modeling shape geometries and material properties, our method ensures greater physical realism and geometric coherence.

3. Material Annotation

We first describe the procedure for creating a dataset of 3D objects annotated with detailed physical material properties. Specifically, we annotate shapes with the material parameters used in the Material Point Method (MPM) [15, 17, 49], a popular simulator known for its effectiveness in handling complex simulations of behaviors and inertia effects inherent in solids [4, 24, 58, 63], including but not limited to elastic and plastic deformation of solids and frictional interactions. The material parameters include: (a) Young's modulus, (b) Poisson's ratio, (c) yield stress, (d) friction angle, and (e) material behavior type (also known as material model), including pure elastic deformation [57], plastic

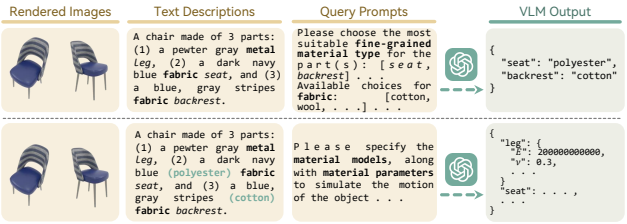

Figure 2. **VLM-guided material annotation of shape parts.**

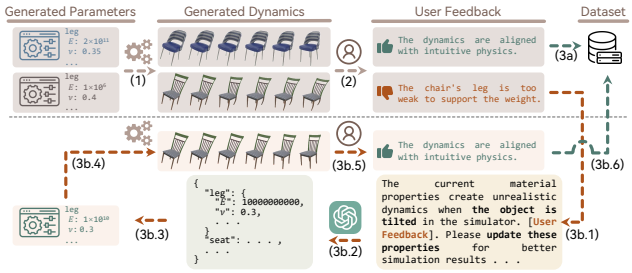

Figure 3. **Expert verification of material annotations.**

deformation with softening (damage) [53], plastic deformation without softening [54], and granular deformation [22].

Unfortunately, no datasets of 3D objects exist with such detailed material properties available. The most related dataset is 3DCoMPaT [26] along with its subsequent versions (3DCoMPaT++ [48], 3DCoMPaT200 [1]), which provide a rather coarse labeling of materials for 3D shape parts, such as "plastic", "fabric", "metal", "wood", and so on. While these classifications provide some insight into the general physical characteristics of objects, they are often too broad for accurate simulation in physics engines. For example, a "plastic" part made out of rigid PVC is much stiffer (*i.e.*, has a much higher Young's modulus) than a flexible plastic material (*e.g.*, LDPE). Unfortunately, such properties can be provided only by domain experts or manufacturers, and are difficult to interpret or estimate for annotators without a strong physics background and experience on commonly used material types in objects. Hiring experts to label material properties for every single component of every 3D object in a large database would be extremely cumbersome. Thus, we devised a semi-automatic pipeline that combines material property annotation by VLMs, followed by iterative expert feedback and verification.

3.1. VLM-guided Material Property Proposals

VLMs have recently become prominent in physical reasoning [6, 23, 30, 32, 64, 65] due to their extensive knowledge bases built on multi-modal data. Thus, we leveraged their zero-shot reasoning ability to give an initial estimation of material properties for input 3D shapes. We started by choosing 12 shape categories from 3DCoMPaT200 [1], which contained a variety of material compositions and behaviors capable of elastic or plastic deformation, such as bags, pillows, and chairs. We skipped categories where objects behave rigidly (*e.g.*, cabinets, faucets). As shown in Figure 2(top), we provided a popular VLM (ChatGPT-4o [37]) with (a) two automatically generated, rendered views of each textured object in the selected categories, (b) a textual description of the shape, including its object category, part tag, coarse material label, and color, as provided by 3DCoMPaT200, (c) a list of available fine-grained material categories *e.g.*, for "plastic", we include sub-types such as polypropylene, rigid or flexible PVC, and so on. We provide this fine-grained list in Section 7.1 (supplement). Our text prompt asks the VLM to identify the most plausible fine-grained material category per part. Then for each part, in a second round, we further prompted the VLM to provide the most likely material models and parameters (*e.g.*, Young's modulus, Poisson's ratio, yield stress, and friction angle) based on a similar textual description, including this time the fine-grained material category as shown in Figure 2(bottom). This two-round prompt provided better material property prediction based on our expert verification.

3.2. Expert Verification and Feedback

Although VLMs often offer reasonable initial estimates of material properties, they are not always reliable. To address this issue, we develop a pipeline that generates simulation videos of objects based on the material properties provided by the VLM, then ask experts with mechanical engineering backgrounds to assess their physical plausibility.

Test scenarios. Specifically, we created five test scenarios to simulate object dynamics: (1) *Drop*ping an object from a certain height; (2) *Throw*ing an object in a certain direction; (3) *Tilt*ing an object; (4) *Drag*ging an object; (5) Applying a short-term, time-variant (*e.g.*, *wind*) force. We render objects photorealistically under these scenarios using a particle-based simulator based on `warp-mpm` [66] that we extended it to model heterogeneous materials per object, since our dataset often contains objects whose components are made of different materials. Then we used PhysGaussian [58] to render the simulation videos. After collecting these videos, we enlisted five mechanical engineer graduates to evaluate the physical plausibility of each video and instructed them to reach a consensus on their judgment. As shown in Figure 3(2)-(3a), for simulated objects deemed as realistic by this group, we store their material properties in our dataset. For objects deemed to have unrealistic motion, we ask the group to provide feedback to the VLM, specifying which part of the object seems to move in an implausible manner and why. This allows us to re-query VLMs for material parameters, as demonstrated in Figure 3(3b.1)-(3b.5), and generate new simulation videos automatically based on the VLM's updated parameters. This process is iterative until satisfactory simulation results are achieved.

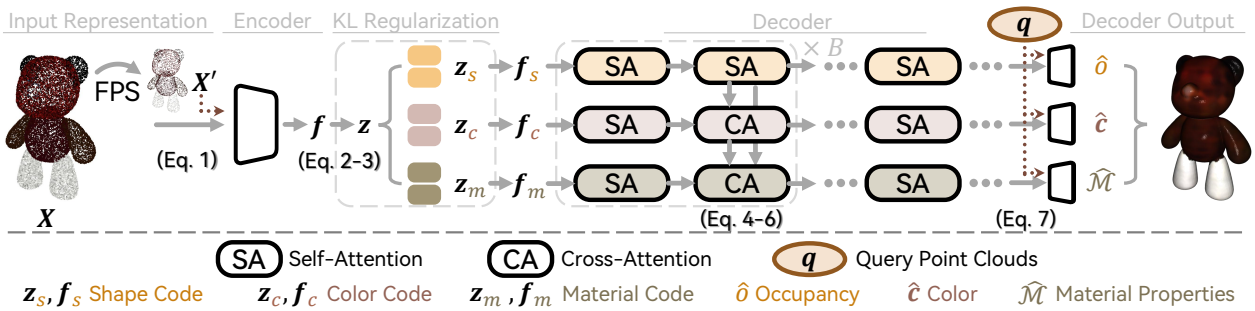

Figure 4. **SOPHY's autoencoder architecture for compressing 3D objects into compact shape, texture, and material-aware codes.**

3.3. Dataset Summary

After the above verification stage, we obtained $3,004$ 3D models in 12 object categories originating from the 3DCoMPaT200 [1] dataset. Every shape part is labeled with physical material properties and detailed material categories, resulting in a total of $15,575$ labeled parts. We split our dataset into training, validation, and test sets, following the 3DCoMPaT200 splits – the total number of samples in each split are 2462, 180, and 362 respectively. More information about our dataset and samples of simulation sequences can be found in Section 8 of our supplement.

4. Generative Model

Our proposed generative model, SOPHY, is based on the idea of Stable Diffusion [45], which generates data by denoising a compressed feature space (latent space). To extract this latent space, we process our shapes along with the color and physical material parameters through a variational autoencoder (Section 4.1). Then a latent diffusion model is trained to jointly model the distribution of object geometries, colors, and material properties in this latent space (Section 4.2). The rationale behind jointly modeling these properties together is that they are strongly interdependent on each other *i.e.*, certain shapes are strongly correlated with the use of specific materials and colors *e.g.*, thin cantilever-shaped chair bases are associated with the use of metal and have a metallic gray color appearance. Modeling these properties independently of each other would yield unlikely or impossible materials for sampled geometries. After training our diffusion model, it can be sampled to generate new latent codes, which are then decoded to generate shapes and compatible colors and materials.

4.1. Shape, Texture & Material Autoencoder

The goal of our autoencoder is to compress input 3D objects into compact latent codes that encode shape, color (albedo texture) and physical material properties. The autoencoder architecture is shown in Figure 4. We discuss its components in the following paragraphs.

Input representation. We represent an object as a dense surface point cloud $\boldsymbol{P} = \{\boldsymbol{p}_j\}_{j=1}^{N}$, where $\boldsymbol{p}_j$ is a 3D point position ($N = 2048$ in our implementation). The input to our autoencoder is a color- and material-augmented point cloud $\boldsymbol{X} = \{\boldsymbol{x}_j\}_{j=1}^{N}$, where each entry $\boldsymbol{x}_j$ contains the following per-point information:

- the point's 3D position $\boldsymbol{p}_j$.
- the point's RGB color $\boldsymbol{c}_j$.
- a material property vector $\boldsymbol{m}_j$ with the following 8 entries: (a) the logarithm of Young's modulus E_j (a scalar), (b) Poisson's ratio ν_j (a scalar), (c) logarithm of yield stress σ_j (a scalar), (d) friction angle ϕ_j (a scalar), and (e) a 4-dimensional learnable embedding $\boldsymbol{\mu}_j$ of material behavior type (*i.e.*, 4 different embedding vectors for 4 material behavior types used in our MPM simulator).

Note that we use a logarithmic scale for Young's modulus and yield stress due to their extremely wide value ranges. For example, Young's modulus may vary from 10^{-3} GPa for very soft materials to 10^{3} GPa for very stiff materials. We represent material information per point rather than per part to accommodate the general case where a semantic part is not made of a homogeneous material.

Encoder. To aggregate the per-point information from the augmented point cloud, we design a set-to-set network inspired by 3DShape2VecSet [62] as our encoder. We first sub-sample a smaller point cloud $\boldsymbol{P}'$ with $M = 512$ fewer points from the original point cloud through furthest point sampling (FPS) [42], and augment it with color and material information to obtain $\boldsymbol{X}' = \{\boldsymbol{x}'_i\}_{i=1}^{M}$. Then, we use cross attention to produce the object's latent representation:

$$\{\boldsymbol{f}_i\}_{i=1}^{M} = \mathrm{CrossAttn}\left(g\left(\{\boldsymbol{x}'_i\}_{i=1}^{M}\right), g\left(\{\boldsymbol{x}_j\}_{j=1}^{N}\right)\right), \quad (1)$$

where g is a linear layer projecting the augmented point cloud into the embedding space $\mathbb{R}^C$ ($C = 512$ in our implementation) and $\{\boldsymbol{f}_i\}_{i=1}^{M}$ is a set of latent codes.

KL regularization. The above object's representation is quite high-dimensional. Thus, we seek to compress it towards a more compact latent code. Following Stable Diffusion [45] and 3DShape2VecSet [62], we adopt a variational autoencoder (VAE) regularized with the KL-divergence to achieve this effect. We first use two fully connected (FC) layers to project each latent code $\boldsymbol{f}_i$ to mean and variance:

$$\boldsymbol{f}_i^{\mu} = \mathrm{FC}_{\mu}(\boldsymbol{f}_i), \ \boldsymbol{f}_i^{\sigma} = \mathrm{FC}_{\sigma}(\boldsymbol{f}_i), \quad (2)$$

where $\boldsymbol{f}_i^{\mu}, \boldsymbol{f}_i^{\sigma} \in \mathbb{R}^{C_0}$ and $C_0 \ll C$ ($C_0 = 24$ in our implementation). Then, we use the reparameterization sampling and obtain a smaller latent code $\boldsymbol{z}_i \in \mathbb{R}^{C_0}$:

$$\boldsymbol{z}_i = \boldsymbol{f}_i^{\mu} + \epsilon \cdot \boldsymbol{f}_i^{\sigma}, \ \epsilon \sim \mathcal{N}(0, 1), \tag{3}$$

which enables us to train diffusion models on a lower-dimensional space later. Finally, we project $\boldsymbol{z}_i$ back into the original embedding space $\mathbb{R}^C$ with another FC layer.

Decoder. One possible decoder design follows the approach of 3DShape2VecSet [62], where latent codes are first processed through a series of self-attention blocks, then for a query point $\boldsymbol{q} \in \mathbb{R}^3$, its occupancy is determined by interpolating the latent codes based on the query position and transforming the result through a fully connected layer. This design could be naturally extended to also decode color and material information per query point. However, we observed worse performance using this approach, as color and material information are not meaningful for non-occupied query points (*i.e.*, points outside the shape volume).

Empirically, we achieved better results with an alternative decoder design. In this approach, we first decode geometry in terms of occupancy values, then decode color, and finally decode material properties, but only for query points classified as occupied. Performance was further improved when we explicitly split the latent codes $\boldsymbol{z}_i$ into three sub-codes: the shape code $\boldsymbol{z}_{i,s}$ for occupancy decoding, the color code $\boldsymbol{z}_{i,c}$ for texture decoding, and the material code $\boldsymbol{z}_{i,m}$ for material property decoding. Each sub-code has a shape of $M \times \frac{C_0}{3}$. To decode these components, we introduce three dedicated branches, each responsible for processing one of the sub-codes (Figure 4). We also apply cross-attention layers to the color and material branches, enabling them to attend on information from previously decoded latents. This models the natural dependency of color on geometry and material properties on both geometry and color (Figure 4). Our decoder design is inspired by the workflow commonly used in 3D asset creation, where artists typically start by defining the object's shape, then apply textures, and finally assign physical attributes. Specifically, the cross-attention in our decoder is formulated as follows:

$$\{\boldsymbol{f}_{i,s}^{(l)}\} = \mathrm{SelfAttn}\big(\{\boldsymbol{f}_{i,s}^{(l-1)}\}\big), \tag{4}$$

$$\{\boldsymbol{f}_{i,c}^{(l)}\} = \mathrm{CrossAttn}\big(\{\boldsymbol{f}_{i,c}^{(l-1)}\}, \{\boldsymbol{f}_{i,s}^{(l-1)}\}\big), \tag{5}$$

$$\{\boldsymbol{f}_{i,m}^{(l)}\} = \mathrm{CrossAttn}\big(\{\boldsymbol{f}_{i,m}^{(l-1)}\}, \{[\boldsymbol{f}_{i,s}^{(l-1)}, \boldsymbol{f}_{i,c}^{(l-1)}]\}\big) \tag{6}$$

where l is the layer index, and $[\cdot, \cdot]$ denotes concatenation.

Decoder output. Given a query point $\boldsymbol{q} \in \mathbb{R}^3$, the occupancy values are decoded by interpolating the geometry sub-codes from the last layer (layer L) and processing them through a fully connected (FC) block:

$$O(\boldsymbol{q}) = \mathrm{FC}\Big(\sum_{i}^{M} \mathrm{Softmax}\big(\frac{q(\boldsymbol{f}_{\boldsymbol{q}}) \cdot k(\boldsymbol{f}_{i,s}^{(L)})}{\sqrt{C}}\big) v(\boldsymbol{f}_{i,s}^{(L)})\Big), \tag{7}$$

where $\boldsymbol{f}_{\boldsymbol{q}}$ is a feature vector obtained by processing the query point through our encoder described in Equation (1). The functions $q(\cdot), k(\cdot), v(\cdot)$ are the query-key-value linear transformations used in attention [51]. By disentangling the latent codes, we ensure that occupancy is determined solely based on the shape sub-codes relevant to this task.

Color and material properties are decoded similarly, each using their own dedicated FC block and interpolation over the corresponding color and material sub-codes. For color prediction, we apply a normalization operation after the FC block to ensure the output falls within the range $[0, 1]$. Young's modulus and yield stress are always positive, so the FC blocks predict their values on a logarithmic scale. The friction angle has a range of $[0, \pi/2]$, so we apply a sigmoid activation after the FC block and scale the output by $\pi/2$. The categorical material behavior type is predicted using a softmax activation after the FC block. We store the material properties only for query points predicted as occupied. For color, we only predict the values for query points sampled on the mesh surface obtained by marching cubes [33].

Autoencoder training. While training our variational autoencoder, we jointly optimize a combination of loss functions involving 3D occupancies, colors, and material properties. Specifically, we minimize a weighted sum of the following losses: (a) occupancy loss $\mathcal{L}_o$ (binary cross-entropy), (b) color loss $\mathcal{L}_c$ (ℓ_1 norm), (c) Young's modulus loss $\mathcal{L}_E$ (ℓ_1 norm applied to logarithmic values), (d) Poisson's ratio loss $\mathcal{L}_\nu$ (ℓ_1 norm), (e) yield stress loss $\mathcal{L}_\sigma$ (ℓ_1 norm applied to logarithmic values), (f) friction angle loss $\mathcal{L}_\phi$ (ℓ_1 norm), (g) material model loss $\mathcal{L}_M$ (cross-entropy), (h) a regularization loss $\mathcal{L}_r$ imposing a KL penalty towards a standard normal distribution on the latent codes, as typically used in VAEs [21]. We note that the color and material losses are computed only for training query points on or inside the object's surface. For off-surface training points, we assign color and material properties by copying them from their nearest surface points. The combined loss function is:

$$\mathcal{L} = \mathcal{L}_o + \sum_{\omega} \lambda_\omega \mathcal{L}_\omega, \quad \omega = \{c, E, \nu, \sigma, \phi, M, r\}, \tag{8}$$

where λ_ω are weight coefficients listed in Section 9.3. To further enhance training, we leverage the pretrained 3DShape2VecSet model [62], which was trained on ShapeNet [5] (a larger dataset containing 55K shapes) using occupancy supervision alone. Specifically, we initialize all network layers shared with 3DShape2VecSet, including the cross-attention weights on point positions in our encoder, the self-attention weights on shape sub-codes, and the occupancy decoder weights, using their pretrained values. This provides a small boost compared to training our model from scratch on our smaller 3K-shape dataset.

Metric	Baseline	w/o CA	Fused	SOPHY
M.B. Acc(%) ↑	71.04	92.77	93.23	**93.55**
MAE-log(E) ↓	1.18	0.50	0.47	**0.45**
MAE-ν($\times 10^{-2}$) ↓	4.10	3.06	**2.98**	3.06
MAE-log(σ) ↓	1.07	0.41	0.32	**0.29**
MAE-ϕ ($\times 10^{-2}$) ↓	5.45	1.89	**1.22**	1.28
Sim-CD ($\times 10^{-3}$) ↓	53.84	17.47	10.05	**8.72**
MAE-c($\times 10^{-2}$) ↓	8.75	8.47	8.35	**8.13**
IoU(%) ↑	90.69	90.75	90.66	**90.89**
CD($\times 10^{-4}$) ↓	3.51	3.34	3.30	**3.02**
F-Score(%) ↑	93.65	93.77	93.79	**93.85**

Table 1. **Quantitative comparison in the auto-encoding setting.** The upper, middle, and lower sections display the average metrics for predictions of material, color, and shape respectively.

4.2. Diffusion

Generating a simulation-ready object involves executing the reverse process of a diffusion that progressively transforms Gaussian noise in the latent space of objects into target latent codes: $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1}^{M}$. Sampling is performed by solving the stochastic differential equations from the EDM diffusion pipeline [19]. Once a latent code is sampled, we pass it through our trained decoder to extract occupancy values on a dense $\mathbb{R}^3$ grid. These values are then converted into a mesh using marching cubes, then color and material properties are decoded at densely sampled mesh points.

Training. To train the diffusion model, we use the loss:

$$\mathcal{L}_{\text{edm}} = \mathbb{E}_{\hat{\boldsymbol{Z}} \sim p_{data}} \mathbb{E}_{\boldsymbol{N} \sim \mathcal{N}(\boldsymbol{0}, \sigma_t^2 \boldsymbol{I})} \| D(\hat{\boldsymbol{Z}} + \boldsymbol{N}, \sigma_t, \boldsymbol{c}) - \hat{\boldsymbol{Z}} \|_2^2,$$

where D is the denoising network, $\hat{\boldsymbol{Z}}$ are training latent codes, $\boldsymbol{N}$ are added Gaussian noises, σ_t denotes the noise level, and $\boldsymbol{c}$ is a signal for conditioning. We experimented with two input conditions to our denoiser: (a) an RGB image of a target object, where $\boldsymbol{c}$ represents here the extracted features from a pre-trained image encoder ("DINOv2-ViT-B/14" [38]), (b) a text prompt, where $\boldsymbol{c}$ represents features from a pre-trained text encoder ("CLIP-ViT-L/14" [43]). The denoising network consists of alternating self-attention layers for processing the latent representations and cross-attention layers for incorporating the signals $\boldsymbol{c}$. Details on the denoiser are provided in Section 9.1 of the supplement.

5. Experiments

We evaluate SOPHY on its autoencoder effectiveness (Section 5.1), its generative capabilities for image-conditioned generation (Section 5.2) and text-conditioned generation (Section 5.3) of simulation-ready 3D objects. All our comparisons against alternatives were performed in the same test split of our dataset, described in Section 3.3.

5.1. Auto-encoding Evaluation

In terms of auto-encoding evaluation, our goal is to check how well we are able to recover a test 3D shape, along with its color and material properties, given its input representation. This evaluation is common in 3D latent-based generative models [62], since it is imperative for the autoencoder to capture latent spaces that can generalize to novel inputs.

Competing approaches. We stress that there is no existing generative model matching our setting of joint shape and physical material synthesis, thus, we compare here with alternative designs for our autoencoder:
(a) baseline is a model that excludes color and material properties from the generation process *i.e.*, it generates a 3D shape, then predicts color conditioned on the shape through a decoder, and then the material through another decoder. The choice of this baseline attempts to answer the question of whether there is any benefit of incorporating the physical materials in the generation process *i.e.*, whether it is simply better to generate the 3D shape first, then guess its most likely texture and material discriminatively. For this baseline, we use 3DShape2VecSet as the generative model, trained on the same split as our method. We use a texture decoder, which decodes its latent shape representation with a set of self-attention blocks to per-point colors. Then we use one more decoder to decode the latent shape representation to material properties, including a cross-attention block for conditioning on colors. The number of attention blocks and the number of parameters remain comparable to our model.
(b) w/o CA is a degraded variant of our proposed SOPHY. It discards the cross-attention blocks used in the color and material decoder branch. This variant is equivalent to decoding our latent sub-codes with three non-interacting branches for occupancy, color, and material predictions.
(c) Fused is another variant of SOPHY that uses a unified representation for the latent code, without separating it into the shape, color, and material sub-codes. The decoder infers the per-query occupancy, color, and material properties conditioned on this single latent code.

Metrics. For material and color prediction, we report the classification accuracy of the material behavior type (M.B. Acc.) and the mean absolute error (MAE) of all our material parameters. In addition, we report Chamfer distance (Sim-CD) between densely sampled points of the predicted shapes and ground-truth ones under the deformed states computed by our MPM simulator along the whole simulation trajectory for the dropping test scenario (Section 3.2). Finally, we report purely geometric measures for the rest state of the reconstructed test shape compared to the ground-truth one. We use the standard metrics of Chamfer Distance (CD), volumetric Intersection-over-Union (IoU), and F-score, as reported in prior 3D generative models [62]. We average all errors across the test cases of our dataset.

Metric	DG4D [44]	Baseline	SOPHY
VideoPhy z-score ↑	-0.09	-0.06	**0.16**
DINO Similarity ↑	0.73	0.76	**0.77**

Table 2. **Comparison on image-conditioned 4D generation.**

Metric	STAG4D [60]	Baseline	SOPHY
VideoPhy z-Score ↑	-0.82	0.30	**0.52**
CLIP Similarity ↑	**0.13**	0.12	**0.13**

Table 3. **Comparison on text-conditioned 4D generation.**

Results. Table 1 presents quantitative comparisons in the auto-encoding setting. Compared to the "baseline" model, jointly modeling shape, color, and material in a shared embedding space significantly improves material metrics, with additional, more modest gains in color and geometry evaluation metrics. Notably, the material behavior type classification is improved by more than 20%, while Sim-CD is reduced by a factor of 5x. This performance gain supports our hypothesis that geometry, color, and material attributes are strongly correlated and should be jointly modeled in the generative process. Compared to not using cross-attention in the decoder ("w/o CA") model, we see that our full model has 2x lower Chamfer distance during simulation, and still slightly better performance in all other metrics. Finally, we observe that the "Fused" variant, has a tiny edge over our model in Poisson's ratio and friction angle prediction. Yet, it performs significantly worse in terms of Sim-CD, and worse in all other measures. We suspect this behavior is due to an uneven network capacity allocation across occupancy, color, and material within a single fused latent code.

5.2. Image-to-4D

We now discuss applying SOPHY to generate 4D dynamics given a single input RGB image. Specifically, given a 3D object with physics material properties generated by our method conditioned on the input image, we plug it into a virtual 3D environment with other objects or primitives e.g., ground planes, walls, and so on, and animate it based on its material properties and interactions with the environment. To evaluate in this setting, we render a 2D image from each object of our test split, provide it to our trained diffusion model for generation, then simulate the generated object using the test scenarios of Section 3.2 to create 4D scenes.

Competing approaches. In the absence of published methods for generating physics-grounded 4D scenes from a single image[1], we first compare with our "baseline" model discussed in the previous section. We also compare with a popular image-to-4D method, DreamGaussian4D, or in short DG4D [44], which generates deforming 3D Gaussian splats (3DGS) over time conditioned on an input image, yet does not rely on any explicit physics-based representation.

Metrics. Given rendered images of the 4D scenes generated by any of the above compared methods, and the ones rendered from the ground-truth test objects from the same viewpoints, we leverage VideoPhy [2], a state-of-the-art VLM trained based on human annotations to evaluate whether the generated videos align with real-world physics. The method produces a per-scene score that is uncalibrated for different scenarios (*i.e.*, it has different scales). Thus, following [50], we apply z-score normalization on the scores across all methods for each scene to calibrate them, allowing us to obtain a more meaningful average score across the test scenes. Positive z-scores mean that a method performs better than average, and negative means worse. Additionally, we assess the image similarity averaged across all frames and scenes using cosine similarity on DINO features from the pre-trained encoder of "DINOv2-ViT-B/14" [38]. Given that DG4D generates 3DGS, we also use 3DGS for rendering our generated shapes using the 3DGS renderer [20] and the splat fitting provided in PhysGaussian [58]. We note that the generation times are comparable for all methods (about 10 minutes per scene, including the feedforward pass, simulation, and rendering, as measured on a single L40s GPU).

Results. SOPHY outperforms the other methods, demonstrating positive z-scores for alignment with real-world physics based on VideoPhy. In terms of image similarity, our method is better aligned with the ground-truth. We also show qualitative results for a few sample frames for test scenes in Figure 5. We observe that DG4D struggles to generate noticeable deformations, even for soft objects *e.g.*, plants, due to its reliance on Stable Video Diffusion (SVD) [3], which is relatively hard to tune for deformation. The baseline model generates object geometry independently of physical materials. Thus, it often creates physically implausible objects or predicts inappropriate materials for the generated shapes. SOPHY produces more realistic dynamics aligning closely with the given conditions.

5.3. Text-to-4D

We also evaluate SOPHY for text-to-4D. The evaluation is the same as before with the only difference being the input condition (text instead of an image). We test on prompts, describing the object category, part tags, fine-grained material categories for our test objects, following the prompts of Section 3.1. We also include the test scenario in the prompt.

Competing methods & Metrics. We compare our method with the text-to-4D method of STAG4D [60] that generates deforming 3DGS driven by text prompts. We note that our method outputs 4D videos about 10x faster than STAG4D based on the same GPU and splat renderer. We also compare with our baseline. For evaluation metrics, we

[1] PhysMotion [50] is a concurrent work with no published code at the time of this submission – see related work for discussion of differences.

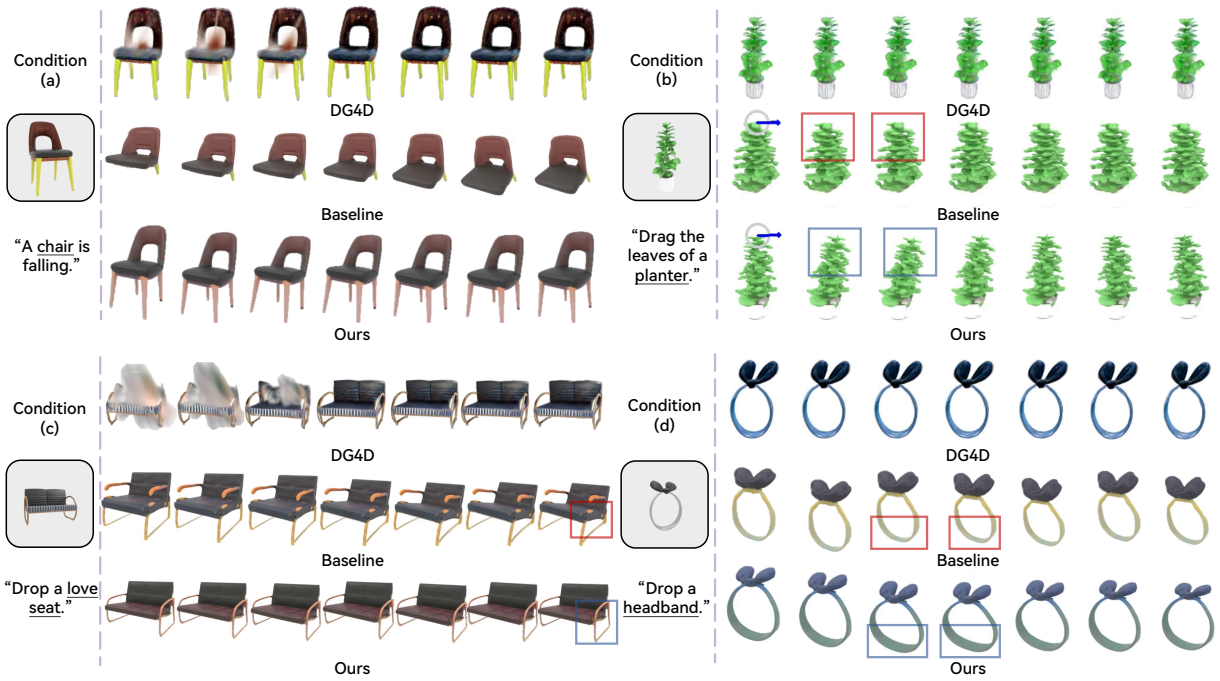

Figure 5. **Qualitative comparison in the image-to-4D setting.** We pick test scenarios described in Section 3.2 to create 4D videos. The description below the conditioned image illustrates the desired dynamics. The blue arrow shows the direction of the external force. In the dropping scenarios, the objects fall onto a ground plane (not shown here).

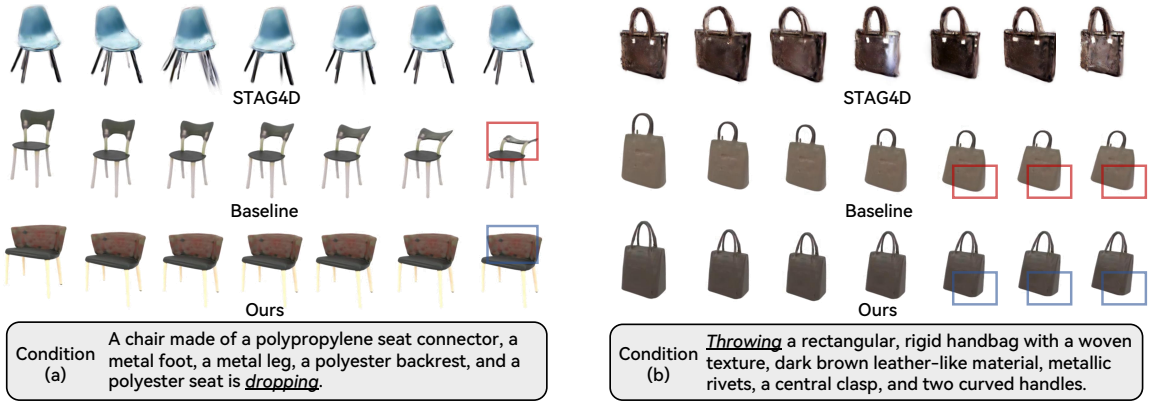

Figure 6. **Qualitative comparison in the text-to-4D setting.** The text prompts used as input condition are shown below.

use the z-normalized VideoPhy scores and CLIP similarity for measuring alignment of the rendered frames with the input text prompt averaged over all frames and test scenes.

Results. SOPHY achieves the highest performance in VideoPhy Score, demonstrating a large positive score in contrast to STAG4D's negative score. The CLIP score is comparable for all methods demonstrating similar alignment with text. We also present a visual comparison for the generated dynamics in Figure 6. We observe that STAG4D produces unrealistic object behavior, *e.g.*, in Figure 6(b), the size of the handbag fluctuates over time. Compared to the baseline, our method produces more geometrically coherent and physically plausible results. These findings highlight SOPHY's ability to generate simulation-ready 3D objects for diverse, physically realistic 4D content.

6. Discussion

We have presented a new generative model of 3D objects incorporating geometry, color, and physical material properties. Our experiments demonstrated significant benefits of the approach, including generating physically plausible 4D videos from images and text. Interesting avenues for future work include extending our generative model for synthesiz-

ing whole scenes and generating other phenomena *e.g.*, fluids or gases. More accurate simulators, *e.g.*, based on finite elements, could be used instead since our model generates the full objects' volume. Our annotated dataset is only a first step towards more physics-aware 3D datasets – enlarging it with more objects and material properties would be a fruitful direction.

Acknowledgements. This work has received funding from the European Research Council (ERC) under the Horizon research and innovation programme (No. 101124742).

References

[1] Mahmoud Ahmed, Xiang Li, Arpit Prajapati, and Mohamed Elhoseiny. 3DCoMPaT200: Language-grounded compositional understanding of parts and materials of 3D shapes. In *NeurIPS*, 2024. 1, 2, 3, 4

[2] Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. VideoPhy: Evaluating physical commonsense for video generation. *arXiv:2406.03520*, 2024. 2, 7

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv:2311.15127*, 2023. 2, 7

[4] Junyi Cao, Shanyan Guan, Yanhao Ge, Wei Li, Xiaokang Yang, and Chao Ma. NeuMA: Neural material adaptor for visual grounding of intrinsic dynamics. In *NeurIPS*, 2024. 2

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv:1512.03012*, 2015. 2, 5

[6] Annie S Chen, Alec M Lessing, Andy Tang, Govind Chada, Laura Smith, Sergey Levine, and Chelsea Finn. Commonsense reasoning for legged robot adaptation with vision-language models. *arXiv:2407.02666*, 2024. 3

[7] Yunuo Chen, Tianyi Xie, Zeshun Zong, Xuan Li, Feng Gao, Yin Yang, Ying Nian Wu, and Chenfanfu Jiang. Atlas3D: Physically constrained self-supporting text-to-3D for simulation and fabrication. In *NeurIPS*, 2024. 2

[8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[9] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. ABO: Dataset and benchmarks for real-world 3D object understanding. In *CVPR*, 2022. 2

[10] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *CVPR*, 2023. 2

[11] Yiangos Georgiou, Marios Loizou, Melinos Averkiou, and Evangelos Kalogerakis. Im2surftex: Surface texture generation via neural backprojection of multi-view images, 2025. 2

[12] Minghao Guo, Bohan Wang, Pingchuan Ma, Tianyuan Zhang, Crystal Elaine Owens, Chuang Gan, Joshua B Tenenbaum, Kaiming He, and Wojciech Matusik. Physically compatible 3D object modeling from a single image. In *NeurIPS*, 2024. 2

[13] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *ICLR*, 2024. 2

[14] Jingyu Hu, Ka-Hei Hui, Zhengzhe Liu, Ruihui Li, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3D shape generation, inversion, and manipulation. *ACM TOG*, 43(2): 1–18, 2024. 2

[15] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM TOG*, 2018. 2

[16] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. ManifoldPlus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv:2005.11621*, 2020. 4

[17] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *SIGGRAPH 2016 courses*. ACM, 2016. 2

[18] Heewoo Jun and Alex Nichol. Shap-E: Generating conditional 3D implicit functions. *arXiv:2305.02463*, 2023. 2

[19] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022. 6, 4

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):1–14, 2023. 2, 7

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 5

[22] Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 3

[23] Wenqiang Lai, Tianwei Zhang, Tin Lun Lam, and Yuan Gao. Vision-language model-based physical reasoning for robot liquid perception. In *IROS*, 2024. 3

[24] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. PAC-NeRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *ICLR*, 2022. 2

[25] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3D generation: A survey. *arXiv:2401.17807*, 2024. 2

[26] Yuchen Li, Ujjwal Upadhyay, Habib Slim, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, and Mohamed Elhoseiny. 3DCoMPaT: Composition of materials on parts of 3D things. In *ECCV*, 2022. 1, 2, 3

[27] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3D content creation. In *CVPR*, 2023. 2

[28] Hubert Lin, Melinos Averkiou, Evangelos Kalogerakis, Balazs Kovacs, Siddhant Ranade, Vladimir Kim, Siddhartha Chaudhuri, and Kavita Bala. Learning material-aware local descriptors for 3D shapes. In *3DV*, 2018. 1, 2

[29] Jiajing Lin, Zhenzhong Wang, Yongjie Hou, Yuzhou Tang, and Min Jiang. Phy124: Fast physics-driven 4D content generation from a single image. *arXiv:2409.07179*, 2024. 2

[30] Jiajing Lin, Zhenzhong Wang, Shu Jiang, Yongjie Hou, and Min Jiang. Phys4DGen: A physics-driven framework for controllable and efficient 4D content generation from a single image. *arXiv:2411.16800*, 2024. 3

[31] Yehonathan Litman, Or Patashnik, Kangle Deng, Aviral Agrawal, Rushikesh Zawar, Fernando De la Torre, and Shubham Tulsiani. Materialfusion: Enhancing inverse rendering with material diffusion priors. In *3DV*, 2025. 2

[32] Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. PhysGen: Rigid-body physics-grounded image-to-video generation. In *ECCV*, 2024. 3

[33] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. 5

[34] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 2

[35] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 4

[36] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019. 2

[37] OpenAI. GPT-4o system card. *arXiv:2410.21276*, 2024. 3

[38] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 6, 7

[39] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[40] Dmitry Petrov, Pradyumn Goyal, Vikas Thamizharasan, Vladimir Kim, Matheus Gadelha, Melinos Averkiou, Siddhartha Chaudhuri, and Evangelos Kalogerakis. GEM3D: Generative medial abstractions for 3D shape synthesis. In *SIGGRAPH*, 2024. 2

[41] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*, 2023. 2

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 4

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 6

[44] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4D: Generative 4D gaussian splatting. *arXiv:2312.17142*, 2023. 2, 7

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 4

[46] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3D shape synthesis. In *NeurIPS*, 2021. 2

[47] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: A single image to consistent multi-view diffusion base model. *arXiv:2310.15110*, 2023. 2

[48] Habib Slim, Xiang Li, Yuchen Li, Mahmoud Ahmed, Mohamed Ayman, Ujjwal Upadhyay, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, et al. 3DCoMPaT++: An improved large-scale 3D vision dataset for compositional recognition. *arXiv:2310.18511*, 2023. 1, 2, 3

[49] Deborah Sulsky, Zhen Chen, and Howard L Schreyer. A particle method for history-dependent materials. *Computer methods in applied mechanics and engineering*, 118(1-2): 179–196, 1994. 2

[50] Xiyang Tan, Ying Jiang, Xuan Li, Zeshun Zong, Tianyi Xie, Yin Yang, and Chenfanfu Jiang. PhysMotion: Physics-grounded dynamics from a single image. *arXiv:2411.17189*, 2024. 2, 7

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 5

[52] Giuseppe Vecchio and Valentin Deschaintre. Matsynth: A modern pbr materials dataset. In *CVPR*, 2024. 2

[53] Stephanie Wang. *A Material Point Method for Elastoplasticity with Ductile Fracture and Frictional Contact*. University of California, Los Angeles, 2020. 3

[54] Stephanie Wang, Mengyuan Ding, Theodore F Gast, Leyi Zhu, Steven Gagniere, Chenfanfu Jiang, and Joseph M Teran. Simulation and visualization of ductile fracture with the material point method. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(2):1–20, 2019. 3

[55] Tsun-Hsuan Johnson Wang, Juntian Zheng, Pingchuan Ma, Yilun Du, Byungchul Kim, Andrew Spielberg, Josh Tenenbaum, Chuang Gan, and Daniela Rus. DiffuseBot: Breeding soft robots with physics-augmented generative diffusion models. In *NeurIPS*, 2023. 2

[56] Zhen Wang, Dongyuan Li, and Renhe Jiang. Diffusion models in 3D vision: A survey. *arXiv:2410.04738*, 2024. 2

[57] Alan Wineman. Some results for generalized neo-hookean elastic materials. *International Journal of Non-Linear Mechanics*, 40(2-3):271–279, 2005. 2

[58] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-integrated 3D gaussians for generative dynamics. In *CVPR*, 2024. 2, 3, 7

[59] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3D: Paint anything 3D with lighting-less texture diffusion models. In *CVPR*, 2024. 2

[60] Yifei Zeng, Yanqin Jiang, Siyu Zhu, Yuanxun Lu, Youtian Lin, Hao Zhu, Weiming Hu, Xun Cao, and Yao Yao. STAG4D: Spatial-temporal anchored generative 4D gaussians. In *ECCV*, 2024. 2, 7

[61] Biao Zhang and Peter Wonka. LaGeM: A large geometry model for 3D representation learning and diffusion. In *ICLR*, 2025. 4

[62] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM TOG*, 42 (4):1–16, 2023. 2, 4, 5, 6

[63] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. PhysDreamer: Physics-based interaction with 3D objects via video generation. In *ECCV*, 2024. 2

[64] Haoyu Zhao, Hao Wang, Xingyue Zhao, Hongqiu Wang, Zhiyu Wu, Chengjiang Long, and Hua Zou. Automated 3D physical simulation of open-world scene with gaussian splatting. *arXiv:2411.12789*, 2024. 3

[65] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D vision-language-action generative world model. In *ICML*, 2024. 3

[66] Zeshun Zong, Xuan Li, Minchen Li, Maurizio M Chiaramonte, Wojciech Matusik, Eitan Grinspun, Kevin Carlberg, Chenfanfu Jiang, and Peter Yichen Chen. Neural stress fields for reduced-order elastoplasticity and fracture. In *SIGGRAPH Asia*, 2023. 3

SOPHY: Generating Simulation-Ready Objects with PHYsical Materials

Supplementary Material

7. Full Prompts

In this section, we provide the detailed prompts we used in the VLM-guided material annotation stage described in Section 3.1.

7.1. Fine-grained Material Types

We pre-define a list of available fine-grained material types for coarse materials, *i.e.*, "fabric", "leather", and "plastic", from 3DCoMPaT200 [1]. This is necessary due to the significant variations in material characteristics within these broader classifications. Concretely, the available fine-grained materials for "fabric" include: cotton, wool, polyester, silk, denim, spandex, linen, and rayon. The available fine-grained materials for "leather" include: full-grain leather, top-grain leather, genuine leather, nubuck leather, suede, patent leather, bonded leather, and faux leather. The available fine-grained materials for "plastic" include: low-density polyethylene, high-density polyethylene, polyethylene terephthalate, polypropylene, rigid polyvinyl chloride, flexible polyvinyl chloride, polystyrene, polycarbonate, acrylonitrile butadiene styrene, polyamide, polyurethane, and thermoplastic elastomers.

When processing a part component that belongs to any of these coarse material types, we supply the VLM with the corresponding fine-grained material types to achieve a more precise result regarding its material composition, as shown below.

```
You are an intelligent AI assistant for computer
graphics, physical simulation, and material science.

Follow the user's requirements carefully and make
sure you understand them.

Keep your answers short and to the point.

Do not provide any information that is not required.

You are going to identify the most likely
fine-grained material type for one or more parts of
the object in the attached image(s).

The attached images describe a [SHAPE NAME] made of
[N_P] parts: [PART-MATERIAL DESCRIPTION].

Given the appearance and your knowledge on material
composition, please choose the most suitable
fine-grained material type for the part(s): [A LIST
OF PART NAMES].

The available options for the [COARSE-GRAINED
MATERIAL NAME] material type are: [A LIST OF
AVAILABLE FINE-GRAINED MATERIAL NAMES].

Please provide your answer in the following JSON
format:
```
{
  'part_name': 'most_suitable_material_type',
  ...: ... # other parts
}
```
The output should **only** contain the dictionary.
```

7.2. Material Models and Parameters

```
You are an intelligent AI assistant for computer
graphics, physical simulation, and material science.

Follow the user's requirements carefully and make
sure you understand them.

Keep your answers short and to the point.

Do not provide any information that is not required.

You are going to use the Material Point Method to
simulate the motion of the object shown in the
attached image(s).

To simulate the effect, you need to specify an
elastic and a plastic material model, along with
material parameters such as Young's modulus,
Poisson's ratio, and yield stress.

Note that the material parameters should be
reasonable for the object shown in the image(s).

More specifically, when the material parameters are
used to simulate dropping, throwing, or tilting the
object, the object should behave according to
physical common sense.


The available material models are list below.
# Available elastic material models (with parameters
required)
1. Neo-Hookean elasticity (Young's modulus, Poisson's
ratio);
2. StVK elasticity (Young's modulus, Poisson's ratio).
# Available plastic material models (with parameters
required)
1. Identity plasticity;
2. von Mises plasticity (Young's modulus, Poisson's
ratio, yield stress);
3. Drucker-Prager plasticity (Young's modulus,
Poisson's ratio, friction angle);

The available combinations of material models for
each material category are listed below. The leading
number is the **Combination ID**.
# ceramic
M1. Neo-Hookean elasticity, von Mises plasticity with
damage;
# fabric
M0. Neo-Hookean elasticity, Identity plasticity;
M1. Neo-Hookean elasticity, von Mises plasticity with
damage;
# leather
M0. Neo-Hookean elasticity, Identity plasticity;
M1. Neo-Hookean elasticity, von Mises plasticity with
damage;
# metal
M2. Neo-Hookean elasticity, von Mises plasticity;
# plant
M0. Neo-Hookean elasticity, Identity plasticity;
# plastic
M1. Neo-Hookean elasticity, von Mises plasticity with
damage;
# soil
M3. StVK elasticity, Drucker-Prager plasticity;
```

```
# wood
M1. Neo-Hookean elasticity, von Mises plasticity with
damage;

The attached image(s) describe the object you are
going to simulate.

It is a [SHAPE NAME] made of [N_P] parts:
[PART-MATERIAL DESCRIPTION].

For each part, you need to specify **both the elastic
and the plastic** material model and the material
parameters.

Please provide your answer in the following JSON
format (for Young's modulus and yield stress, the
unit is Pa.):
```
{
"part_name": {
   "CID": "Mx", // Combination ID
   "E": youngs_modulus,
   "nu": poissons_ratio,
   "...": ..., // other parameters, e.g., yield
   stress ("sigma_y"), friction angle ("phi")
}
..., // other parts
}
```
The output should **only** contain the dictionary.
```

7.3. Update with Expert Feedback

In this case, we send three messages with the role set by `user`, `assistant`, `user` in succession. The first `user` prompt is the same as the prompt for obtaining the initial material properties. The `assistant` prompt is the output of the original query from the VLM. The second `user` prompt is as follows.

```
The original output creates unrealistic dynamics when
the object [TEST CASE DESCRIPTION] in the simulator.

Specifically, [USER COMMENT].

Given this information, please update the material
parameters to make the object behave more
realistically.

The output should be formatted as the original
version.
```

8. Dataset Details

In Table 4, we present the detailed composition of the proposed dataset, which covers 12 common categories with detailed part-level material annotations for $3,004$ shapes. After the VLM-guided material annotation, we obtained 8, 7, 10 types of fine-grained materials for "fabric", "leather", and "plastic" respectively[2]. The occurrence of parts labeled with these fine-grained materials is illustrated in Figure 7. We also present the statistics of material behavior types in the proposed dataset in Figure 8. Specifically, M0 represents pure elastic materials using neo-Hookean elasticity and identity plasticity. M1 represents plastic materials with softening implemented with neo-Hookean elasticity and von Mises plasticity with damage. M2 is for plastic material without softening, which is supported by neo-Hookean elasticity and von Mises plasticity. Finally, M3 stands for granular material with StVK elasticity and Drucker-Prager plasticity as the material models.

[2]We did not obtain parts labeled with "bonded leather" for the "leather" type, and parts labeled with "flexible polyvinyl chloride", "polystyrene" for the "plastic" type from the VLM.

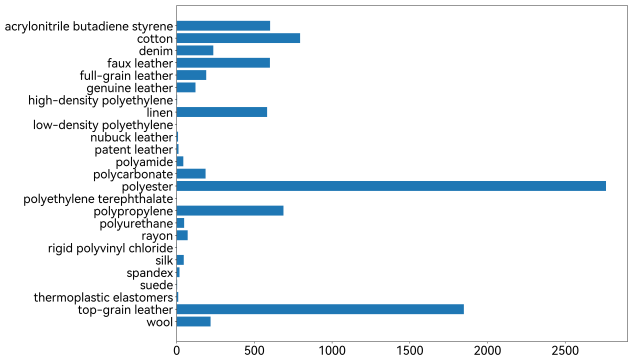

Figure 7. **Statistics of fine-grained material types in our dataset.**

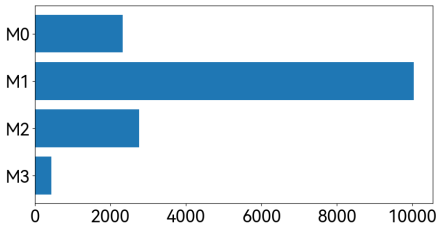

Figure 8. **Statistics of material models in our dataset.**

Our dataset offers detailed material properties, which serve as a first step for advancements in simulation-driven learning, physics-informed generative modeling, and interactive virtual environments. We hope that our dataset can expand the scope of 3D perception research and contribute to developing effective algorithms and techniques that improve our understanding of the physical world.

9. Implementation Details

9.1. Denoiser and Conditional Signals

In Figure 10, we present the denoising network used in our method. We alternatively use a self-attention (SA) layer and a cross-attention (CA) layer for image- and text-conditioned generation. The cross attention is used to inject the conditional signal $\boldsymbol{c}$ into the latent representation. In the image-conditioned experiment, we use 10 rendered images from randomly sampled viewpoints with a size of 256×256 as training data. In the text-conditioned experiment, each object is accompanied by 5 text descriptions. 4 of these descriptions are automatically generated using metadata from the 3DCoMPaT200 [1], as shown in the code snippet be-

Split	Bag	Bed	Chair	Crib	Hat	Headband	Love Seat	Pillow	Planter	Sofa	Teddy Bear	Vase	Total
Train	75	418	898	27	18	27	139	71	383	278	37	91	2462
Valid	10	26	52	5	2	5	21	11	21	18	6	3	180
Test	24	48	106	8	7	10	39	18	46	31	11	14	362

Table 4. **Data composition of our proposed dataset.**

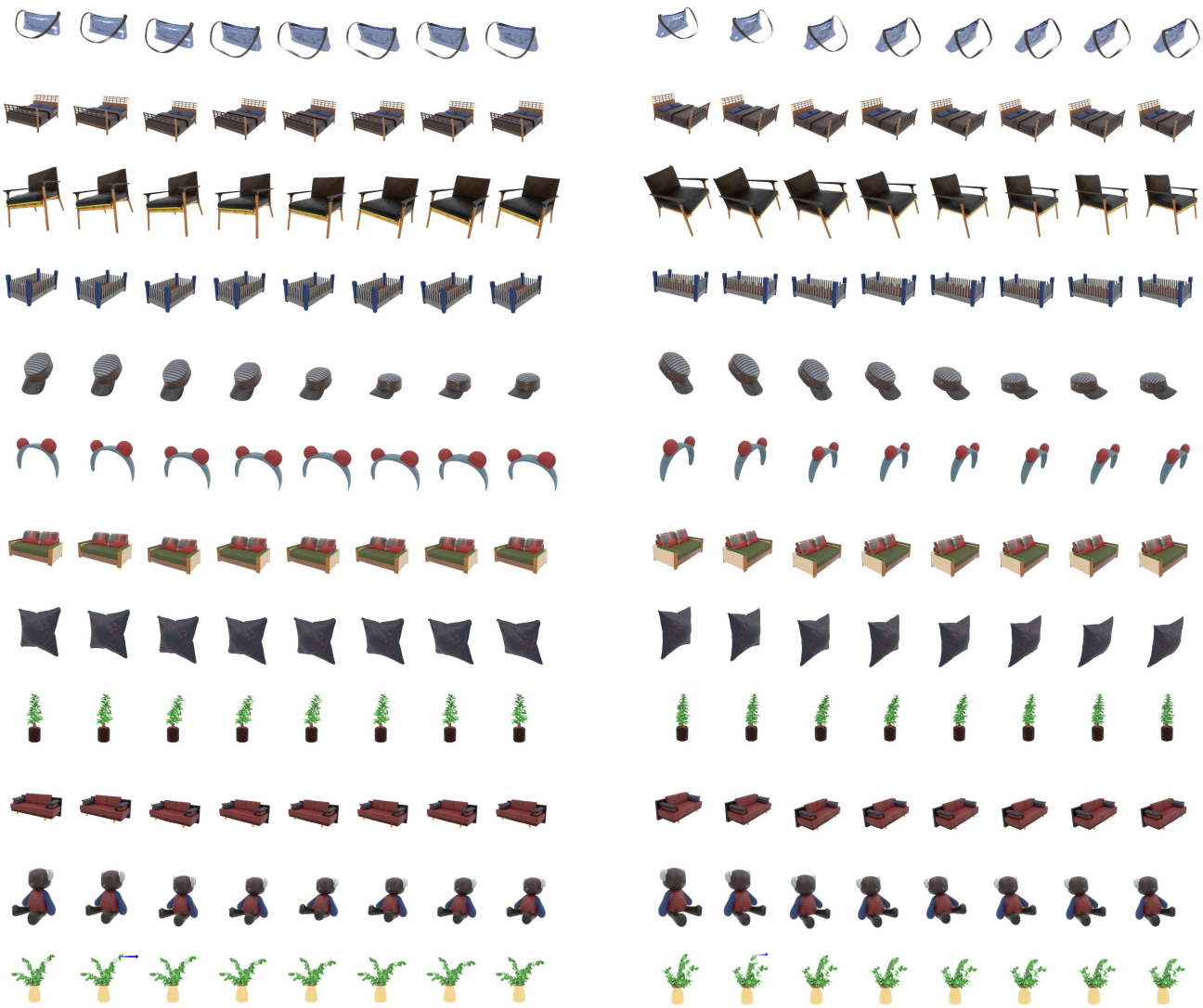

Figure 9. **Examples from our proposed dataset.**

low. The fifth description is created by querying ChatGPT-4o [37] for a text description of the object based on two rendered views.

```
# Conditional Signals for Text
# 1. with color, coarse material type, and part label
text_1 = f"A {shape_name} made of"
for i, (part, mat, mat_fine) in enumerate(part_descriptions, start=1):
    mat_color = MAT_COLORS[mat].lower()
    if i != len(part_descriptions):
        text_1 += f" a {mat_color} {mat} {part},"
    else:
        text_1 += f" and a {mat_color} {mat} {part}."

# 2. with fine material type and part label
text_2 = f"A {shape_name} made of"
for i, (part, mat, mat_fine) in enumerate(part_descriptions, start=1):
    if mat_fine is None:
        mat_name = mat
    else:
        mat_name = mat_fine
    if i != len(part_descriptions):
        text_2 += f" a {mat_name} {part},"
    else:
        text_2 += f" and a {mat_name} {part}."
```

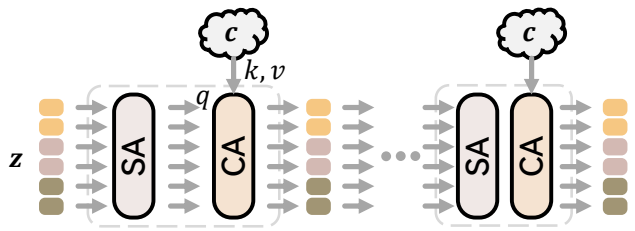

Figure 10. **Denoiser architecture.** We train the denoising network in the latent space (refer to Equation (3)) and adopt alternative self attention (SA) and cross attention (CA) to learn denoised representations. The cross attention is for incorporating conditional signal ($\boldsymbol{c}$) like images or text prompts.

```
# 3. with part label
text_3 = f"A {shape_name} composed of"
for i, (part, mat, mat_fine) in enumerate(part_descriptions, start=1):
    if i != len(part_descriptions):
        text_3 += f" a {part_name},"
    else:
        text_3 += f" and a {part_name}."

# 4. with category label
text_4 = f"A 3D shape of {shape_name}."
```

9.2. Data Preprocessing

For object autoencoding and generation, we mainly follow the preprocessing pipeline of 3DShape2VecSet [62] to obtain per-point information. First, each 3D shape is converted into a watertight mesh using ManifoldPlus [16] and normalized to its bounding box. From this, we sample a dense surface point cloud of $150,000$ points. To train the network, we randomly sample $300,000$ points in 3D space—each annotated with occupancy, color, and material properties—along with an additional $300,000$ points from the near-surface region with the same attributes. We apply the frequency embedding [35, 62] to each point's position and color before forwarding it to the encoder. We note that per-point color is assigned using a 1-NN approach, where each point inherits the color of its nearest surface point. Material properties are determined through a label propagation process. 3DCoMPaT200 [1] provides surface points with part labels for each object. To extend these labels to volumetric points, we apply a 5-NN algorithm, assigning labels via majority vote. Once part labels are obtained, we map corresponding material attributes based on shape part annotations in our material-augmented dataset.

9.3. Training Details

For object autoencoding, we use a surface point cloud of $2,048$ points as input to the autoencoder. At each iteration, we sample $1,024$ query points from the bounding sphere and another $1,024$ from the near-surface region for attribute prediction. Following [61], we ensure an equal distribution of query points between occupied and non-occupied regions. We set the weight coefficients described in Equation (8) as $\lambda_c = \lambda_\nu = \lambda_\phi = \lambda_M = 0.1$, $\lambda_E = \lambda_\sigma = 0.01$, and $\lambda_r = 0.001$. The autoencoder is trained with a batch size of 256 for 500 epochs, using a learning rate that linearly increases to 10^{-4} over the first 25 epochs before gradually decaying to 10^{-5} following a cosine schedule.

For object generation, we train our diffusion models with a batch size of 256 for $2,000$ epochs. The learning rate is linearly increased to 2×10^{-4} over the first 200 epochs and then gradually decays to 10^{-6} using a cosine schedule. We adopt the default hyperparameter settings of EDM [19]. During sampling, the final latent codes are obtained using only 18 denoising steps.