

SOPHY: Generating Simulation-Ready Objects with PHYSical Materials

Supplementary Material

7. Additional Analysis

7.1. Conditioning on images from other datasets

In this subsection, we investigate how well SOPHY generalizes to conditioned images sourced from different datasets. We examine both synthetic and real-world scanned objects. Specifically, we choose the Objaverse [17], a widely used synthetic dataset in previous 3D generation research, along with the Google Scanned Objects [18], which is a high-quality dataset featuring scanned everyday items. We obtain 2D renderings of several 3D objects from them to serve as input conditional signals for our method. After completing the physics-aware 3D generation, we simulate the generated assets using the pipeline described in Section 3.2 of the manuscript, finally obtaining dynamic videos. We present the results in Figure 7 and Figure 8. Even though our method was not specifically trained on these datasets, it can still generate 3D assets with plausible material properties and physical simulations.

7.2. Conditioning on text prompt variations

Here, we study whether SOPHY can generate different 3D objects aligned well with the prompts based on edited text conditions. Specifically, we qualitatively demonstrate changes in the generated results by changing object parts or categories in the input prompt. The results are shown in Figure 9. Starting with the text shown in Figure 9(a), we add a description of “a polyester pillow” in (b). The resulting generations accurately reflect this change in the descriptions. Furthermore, we alter the object category from “sofa” in (a) to “chair” in (c). As shown in Figure 9(c), SOPHY can generate similar shapes for the unedited parts in the prompt, such as “cushion” and “armrest”, while the rest of the shape changes to reflect the edited category.

8. Implementation Details

8.1. Denoiser Architecture

In Figure 10, we present the architecture of the denoising network used in our method. We alternate self-attention (SA) and cross-attention (CA) layers. The cross attention is used to inject the condition signal c (image or text prompt) into the latent representation.

8.2. Conditioning Signals for training

In the image-conditioned experiment, we use 10 rendered images from randomly sampled viewpoints with a size of 256×256 for training. In the text-conditioned experiment, each object is accompanied by 5 text descriptions. 4 of

these descriptions are automatically generated using meta-data from the 3DCoMPaT200 [1], as shown in the code snippet below. The fifth description is created by querying ChatGPT-4o [57] for a text description of the object based on two rendered views.

```
1 # Conditional Signals for Text
2 # 1. with color, coarse material type, and part label
3 text_1 = f"A {shape_name} made of"
4 for i, (part, mat, mat_fine) in enumerate(part_descriptions, start=1):
5     mat_color = MAT_COLORS[mat].lower()
6     if i != len(part_descriptions):
7         text_1 += f" a {mat_color} {mat} {part},"
8     else:
9         text_1 += f" and a {mat_color} {mat} {part}."
10
11 # 2. with fine material type and part label
12 text_2 = f"A {shape_name} made of"
13 for i, (part, mat, mat_fine) in enumerate(part_descriptions, start=1):
14     if mat_fine is None:
15         mat_name = mat
16     else:
17         mat_name = mat_fine
18     if i != len(part_descriptions):
19         text_2 += f" a {mat_name} {part},"
20     else:
21         text_2 += f" and a {mat_name} {part}."
22
23 # 3. with part label
24 text_3 = f"A {shape_name} composed of"
25 for i, (part, mat, mat_fine) in enumerate(part_descriptions, start=1):
26     if i != len(part_descriptions):
27         text_3 += f" a {part_name},"
28     else:
29         text_3 += f" and a {part_name}."
30
31 # 4. with category label
32 text_4 = f"A 3D shape of {shape_name}."
```

8.3. Data Preprocessing

For object autoencoding and generation, we mainly follow the preprocessing pipeline of 3DShape2VecSet [94] to obtain point-based shape representations. First, each 3D shape is converted into a watertight mesh using Manifold-Plus [27] and is normalized to its bounding box. Then we sample a dense surface point cloud of 150K points. To train the network, we randomly sample 300K points in 3D space—each annotated with occupancy, color, and material properties—along with an additional set of 300K points from the near-surface region with the same attributes. We apply the frequency embedding [54, 94] to each point’s position and color before forwarding it to the encoder. We note that color for off-surface points 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 interior shape points, we query their 5 nearest points and assign the label via a majority vote. Once part labels are obtained, we map corresponding material attributes based on part annotations in our material-augmented dataset.

Input Images

Dynamic Generation



Figure 7. Generated results by conditioning our method on images from Objaverse [17] shown on the left.

8.4. Perception-based Baseline

In Section 5.2, we introduced the perception-based baseline (*i.e.*, “B. Perc.”). The baseline uses perceptual models to estimate material properties based on an off-the-shelf 3D generation model. Here, we present more details of how this baseline is implemented. Specifically, we adopt TRELIS [86], a state-of-the-art 3D generation model, to generate textured 3D shapes given image or text conditions. To obtain the material properties of each generated shape, we then leverage an open-vocabulary 3D part segmentation model, Find3D [52], to get the part labels for the sampled surface points. Note that the query part names we provide to Find3D are derived from the set of part labels in our dataset, which comes from 3DCoMPaT200 [1]. Finally, we

leverage ChatGPT-4o [57] by providing it with two renderings of the generated 3D object and asking it to estimate the material properties for each part of the object retrieved by Find3D. The prompts we employed follow the one used in our dataset collection pipeline, as described in Section 10.2. After we acquire the material information for each part of the 3D object, we propagate this information to the points sampled within the object’s volume using the same strategy outlined in Section 8.3. Finally, we simulate the generated object with the estimated material properties.

8.5. Training Details

For object autoencoding, we use a surface point cloud of 2,048 points as input to the autoencoder. At each iteration,

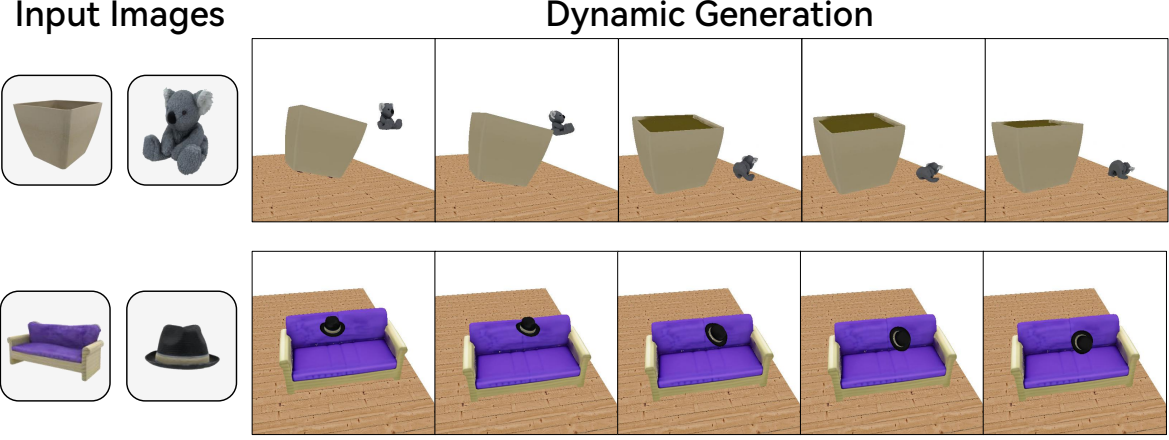


Figure 8. Generated results by conditioning our method on images from Google Scanned Objects [18] shown on the left.

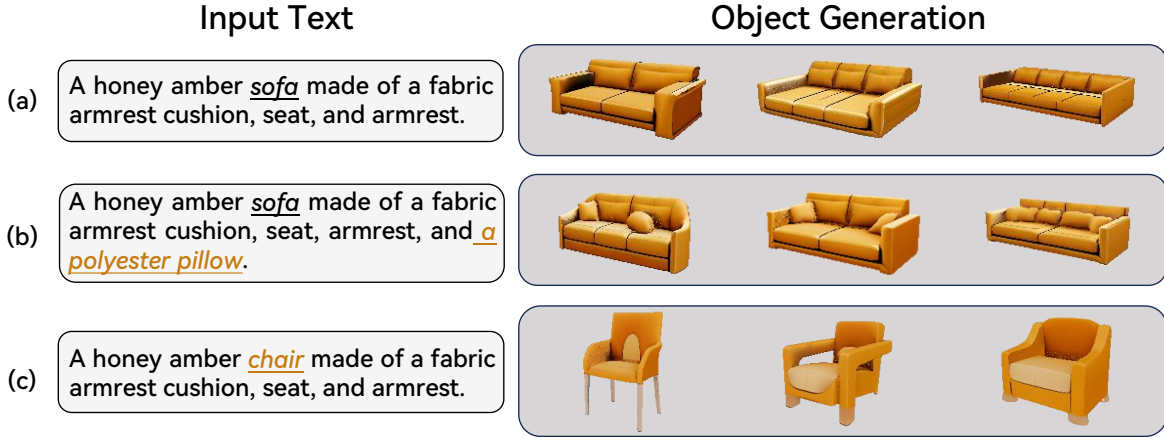


Figure 9. Changing parts or object categories in text prompts shown on the left. On the right, we show different samples of objects generated by our method based on the edited prompts.

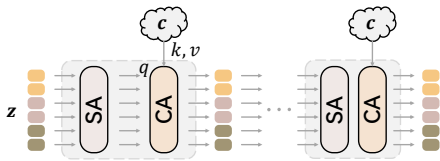


Figure 10. **Denoiser architecture.** We adopt self attention (SA) and cross attention (CA) blocks to learn denoised representations starting from latents (see Equation (3)). The cross attention incorporates conditions c (images/prompts).

we sample 1,024 query points from the bounding sphere of the object and another 1,024 points from the surface region for attribute prediction. Following [93], we ensure an equal distribution of query points between occupied and non-occupied regions. We set the weight coefficients described in Equation (6) as $\lambda_c = \lambda_v = \lambda_\phi = \lambda_\rho = \lambda_M = 0.1$, $\lambda_E = \lambda_\sigma = 0.01$, and $\lambda_r = 0.001$. The autoencoder is

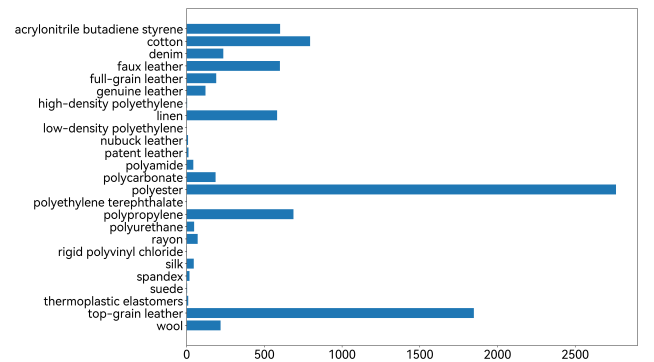


Figure 11. **Statistics of fine-grained material types in our dataset.**

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 3,000 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 [30]. During sampling, the final latent codes are obtained using only 18 denoising steps. All our training experiments are conducted on two NVIDIA L40s GPUs.

9. Dataset Details

In Table 4, we present the detailed composition of the proposed dataset, which covers 12 common categories with 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” respectively¹. The occurrence of parts labeled with these fine-grained materials is shown in Figure 11.

Furthermore, we present more dataset statistics in Figure 12. We begin by illustrating the number of rectification attempts made by experts for each object during the data verification phase outlined in Section 3.2. As presented in Figure 12(a), approximately one-third of the data has received at least one round of material parameter updates based on the experts’ feedback². This highlights the importance of human verification in correcting the initial proposals generated by the VLM. Furthermore, it is noteworthy that most verifications are completed within two iterations, which demonstrates the efficiency of our data annotation pipeline. Next, we provide the statistics of several material properties, including material behavior types in Figure 12(b), the logarithm of Young’s modulus in Figure 12(c), Poisson’s ratio in Figure 12(d), and the logarithm of yield stress in Figure 12(e). 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. These figures showcase the material diversity of our collected dataset.

In summary, 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 re-

¹ 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.

² Please refer to Figure 14, where we provide some screenshots of the web interface we used for verifying material annotations.

search and contribute to developing effective algorithms and techniques that improve our understanding of the physical world.

10. Full Prompts

In this section, we provide the detailed prompts we used in the VLM-guided material annotation stage described in Section 3.

10.1. 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 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].

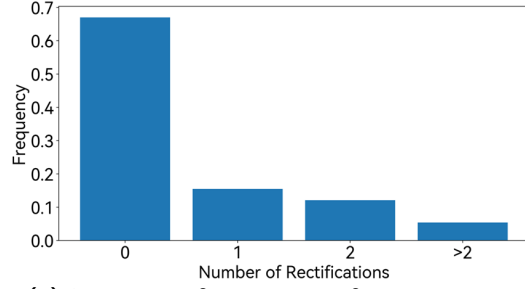
Given the appearance and your knowledge on material
composition, please choose the most suitable
fine-grained material type for the part(s): [A LIST
OF PART NAMES].

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

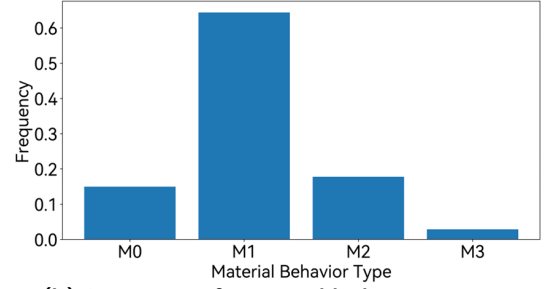
Please provide your answer in the following JSON
format:
'''
{
  "part_name": "most_suitable_material_type",
  ...: ... # other parts
}
'''
```

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

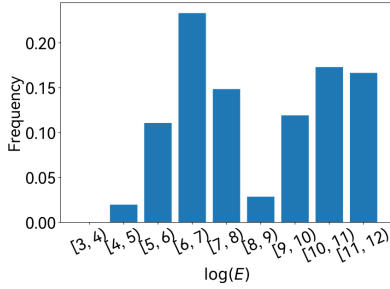
Table 4. Data composition of our proposed dataset.



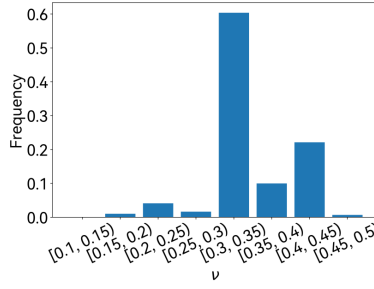
(a) Statistics of expert rectification times



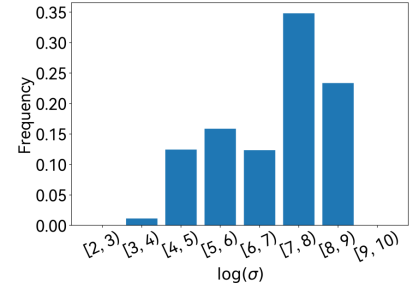
(b) Statistics of material behavior types



(c) Statistics of Young's modulus



(d) Statistics of Poisson's ratio



(e) Statistics of yield stress

Figure 12. More dataset statistics.

The output should **only** contain the dictionary.

10.2. 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



Figure 13. Sampled simulation results from our proposed dataset.

```
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.

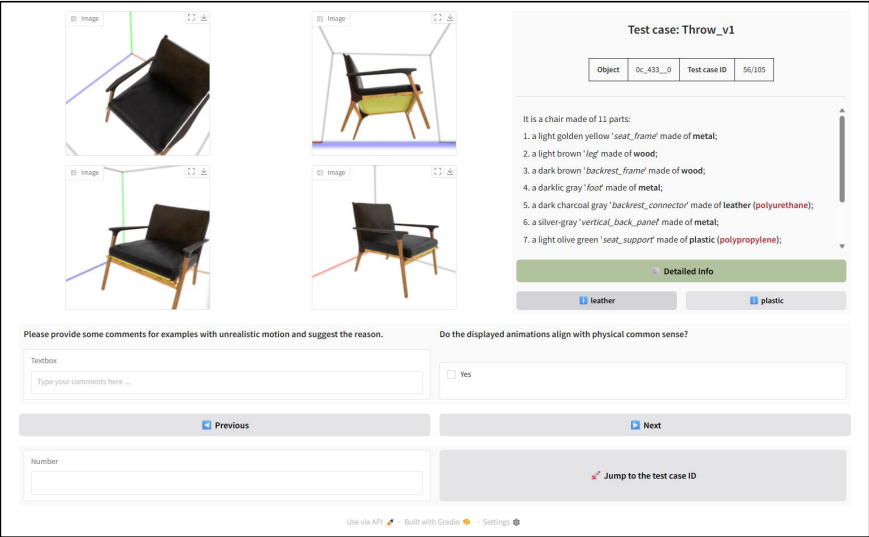
It is a [SHAPE NAME] made of [N_P] parts:
[PART-MATERIAL DESCRIPTION].

For each part, you need to specify **both** the elastic and the plastic material model and the material parameters, such as Young's modulus, Poisson's ratio, density, etc.

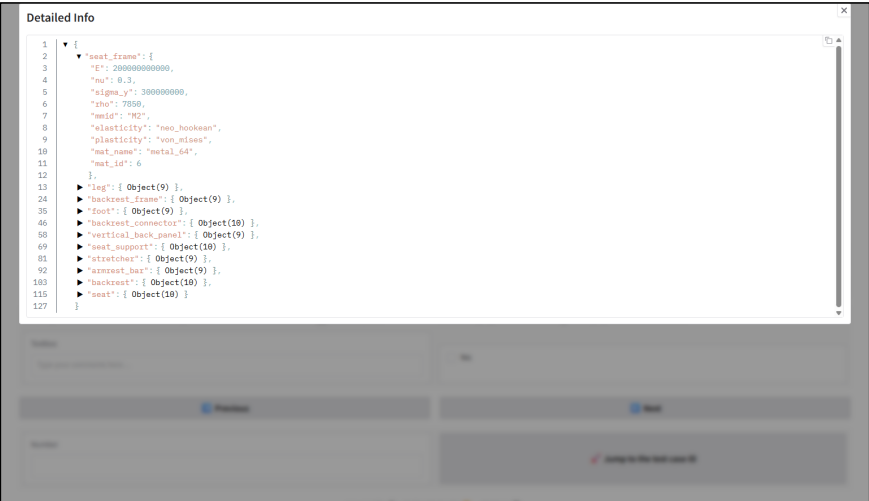
Please provide your answer in the following JSON format (For Young's modulus and yield stress, the unit is Pa. For density, the unit is kg/m³.):

```
'''
{
  "part_name": {
    "CID": "Mx", // Combination ID
    "E": youngs_modulus,
    "nu": poissons_ratio,
    "...": ..., // other parameters, e.g., yield
    stress ("sigma_y"), friction angle ("phi"),
    density ("rho")
  }
  ..., // other parts
}
'''
```

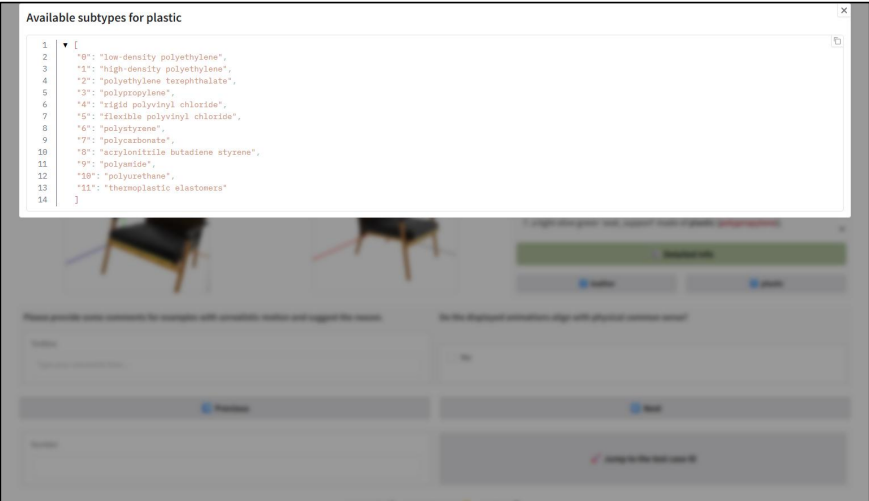
The output should **only** contain the dictionary.



(a) Screenshot of the main verification panel



(b) Screenshot of the detailed information for an object



(c) Screenshot of the available subtypes for a coarse material category

Figure 14. Screenshots of the interactive panel for verifying material annotations.

10.3. 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:

```
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.
```

11. Limitations

Our current dataset of annotated shapes includes 3K objects and 15.5K labeled parts. This size is relatively small, especially for training large-scale 3D or 4D foundation models. We believe that our semi-automatic pipeline can be enhanced in the future to generate significantly larger datasets. Artifacts in 3D shape and texture generation are sometimes visible in our method (*e.g.*, see the left, dark-shaded arm-rest of the first sofa shown in Figure 9). Generating alternative representations based on our physics-aware latents and denoiser, such as meshes [86] or marching tetrahedra [66], along with the generation of high-resolution textures and physically based rendering materials (PBR) [67], in an end-to-end manner could help reduce artifacts and further enhance the quality of the results.