

PhysX-3D: Physical-Grounded 3D Asset Generation

– *Supplementary material* –

Ziang Cao¹ Zhaoxi Chen¹ Linag Pan² Ziwei Liu^{1*}
¹Nanyang Technological University ²Shanghai AI Lab
<https://physx-3d.github.io/>

1 Implementation details

Structure of PhysXGen. In this section, we present the architectural details and implementation specifics of PhysXGen. To maintain consistency with the established pre-trained geometry space, our geometrical decoder preserves the original hyperparameter configuration from [6], ensuring effective utilization of pre-trained weights. For the physical processing components, we implement structurally symmetric encoder-decoder pairs (as detailed in Table 1). Notably, our physical generator employs a streamlined transformer architecture with 14 processing blocks rather than the conventional 24-block configuration to achieve satisfactory performance with lower computational overhead.

Texture retrieval for PhysXGen training. While the 3D objects in PartNet [3] inherently lack surface texturing data, we retrieve compatible UV texture coordinates from ShapeNet [1]. For instances where no corresponding texture exists in ShapeNet, we employ the grey color as their texture information.

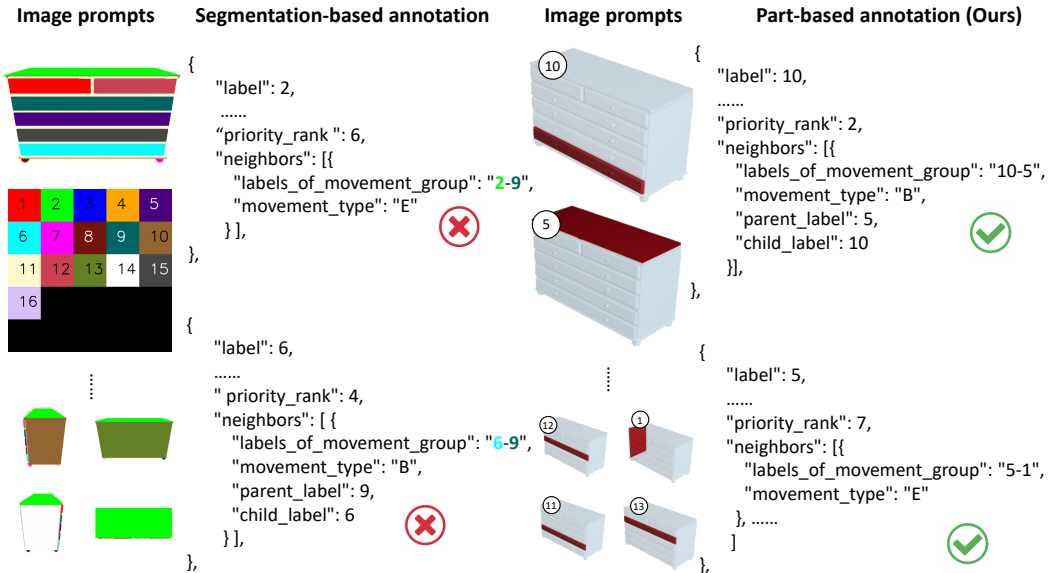


Figure 1: Qualitative comparison of different annotation setting.

*Corresponding author, ziwei.liu@ntu.edu.sg

Table 1: Hyper-parameters of main modules.

Model	resolution	model channel	latent channels	num_blocks	num_heads	mlp_ratio	window_size
Geometry decoder	64	768	8	12	12	4	8
Physics decoder	64	2048	8	4	16	4	8
Physical encoder	64	768	8	4	12	4	8

Table 2: Quantitative comparison against GPT-based method.

Methods	Geometry			Absolute scale ↓	Material ↑	Affordance ↑	Kinematic parameters		Description ↑
	PSNR ↑	CD ↓	F-Score ↑				COV ↑	MMD ↓	
TRE [6] + PartField [2] + GPT	24.31	13.2	76.9	8.81	7.95	6.73	0.09	0.24	14.31
PhysXGen	24.53	12.7	77.3	7.24	13.01	11.30	0.33	0.08	10.11

2 Details of human-in-the-loop annotation pipeline

In this section, we detail the technical configuration of our 3D annotation pipeline. For part-aware geometric annotation, two distinct methodologies were evaluated: segmentation-based and part-based approaches. The segmentation-based method employs multi-view projective rendering to establish inter-part spatial relationships in 2D projections. To avoid the occlusion caused by the number label in the rendered images, we input the index image for reference. While effective for macro-structural analysis, this approach demonstrates limitations in capturing occluded components and accurately resolving fine geometric details that fall below the effective pixel resolution threshold. Conversely, the part-based paradigm demonstrates superior robustness in occluded and micro-part annotations. However, this methodology introduces scalability challenges when processing complex assemblies with high part counts, as it requires rendering a separate image for each individual component - a process that incurs increasing expense with increasing part number.

To avoid the expensive annotation of part-based annotation and build a robust and efficient annotation framework, we implement the following preprocessing pipeline: First, we normalize the 3D object’s spatial coordinates to the $[-1, 1]$ range through proportional scaling and translation. Subsequently, we perform geometric simplification by filtering and merging insignificant components based on dual criteria: surface fragments with area ≤ 0.2 , or those simultaneously satisfying face count ≤ 100 and area ≤ 0.06 , are systematically merged with their topologically adjacent regions. After removing the unnecessary parts, we perform the part-based annotation. Figure 1 shows the qualitative comparison of two annotation paradigms. The segmentation-based annotation pipeline exhibits a higher propensity for generating inconsistent structural interpretations. A representative case involves Part 9, which demonstrates translational movement relative to Part 2 (annotation B) instead of maintaining the expected rigid connection (annotation E). Besides, Part 6 can move relative to the base of the drawer (Part 2, Part 10, Part 13, or Part 14) rather than Part 9. Finally, we adopt the part-based annotation pipeline due to its robustness. Furthermore, we show the system prompt for part-based annotation (see Listing 1). By annotating from global to local, we can get better annotations.

3 Procedural generation in PhysXNet-XL

To facilitate robust and diverse physical 3D generation, we devise a set of procedural generation rules aimed at synthesizing a broad spectrum of physically plausible 3D assets. These rules are categorized into two types: a) intra-category procedural generation and b) cross-category procedural generation. To ensure the performance of procedural generation, we choose the parts that typically exhibit similar physical properties. For a), we target object classes with structural variability, including cabinets, tables, bottles, faucets, chairs, ovens, showers, knives, and laptops. For b), we identify drawers and doors as modular components that can be flexibly integrated into different object types to enhance compositional diversity. Figure 2 shows the workflow of our procedural generation method. Specifically, we identify the connected regions between the original object and the target part. To ensure structural and physical consistency, we adapt the scale of the new component to align it appropriately with the geometry of the base structure. Finally, there are more than 6 million physical 3D objects in our PhysXNet-XL. We will try to extend more categories in our future work.

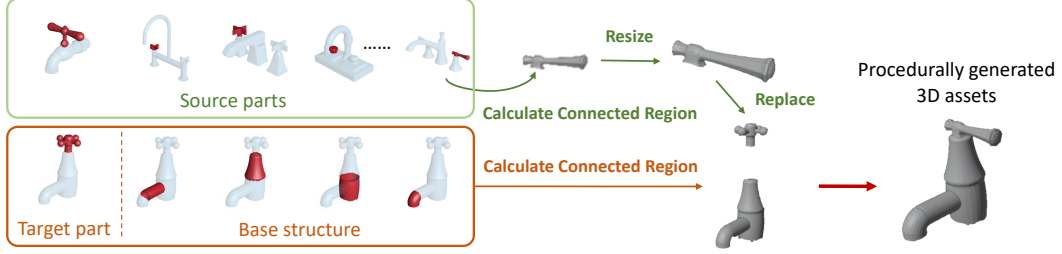


Figure 2: **Workflow of our procedural generation method.** Leveraging procedural generation within PhysXNet, we automatically generate over 6 million physically plausible 3D assets, forming an extended dataset denoted as PhysXNet-XL.

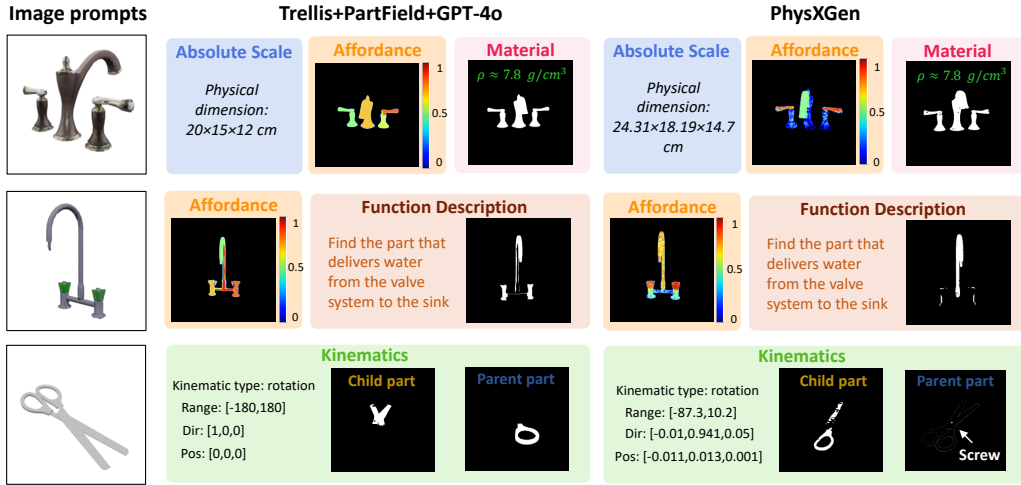


Figure 3: **Qualitative comparison of different methods.** Compared with the existing method, our method achieves robust performance in generating physical 3D assets.

4 More experimental results

4.1 Comparison with GPT-based baseline

To evaluate the capabilities of our proposed method, PhysXGen, in generating physically-grounded 3D assets, we conduct comprehensive qualitative and quantitative comparisons against a GPT-based baseline pipeline comprising Trellis [6], PartField [2], and GPT-4o. Under this benchmark framework, given an input image prompt, Trellis first generates textured 3D meshes with complete geometric and appearance representations. These assets are subsequently processed by PartField to perform fine-grained part segmentation, followed by a GPT-based physical property assignment module that predicts material parameters and dynamic attributes for each identified part. As shown in Table 2, our method exceeds the GPT-based method in geometry and most physics metrics. Across the four evaluation dimensions of **absolute scale**, **material**, **kinematics**, and **affordance**, PhysXGen demonstrates significant performance gains over the GPT-based baseline, achieving relative improvements of **24%**, **64%**, **28%**, and **72%**, respectively. In **function description**, our PhysXGen performs reduced robustness compared to GPT-4o, primarily attributable to its training on a relatively smaller dataset, *i.e.*, PhysXNet. Furthermore, we visualize the generated results of the GPT-based baseline and our PhysXGen in Fig. 3. The qualitative evaluations prove the impressive performance in physical-grounded 3D asset generation, especially in **kinematics** and **affordance**.

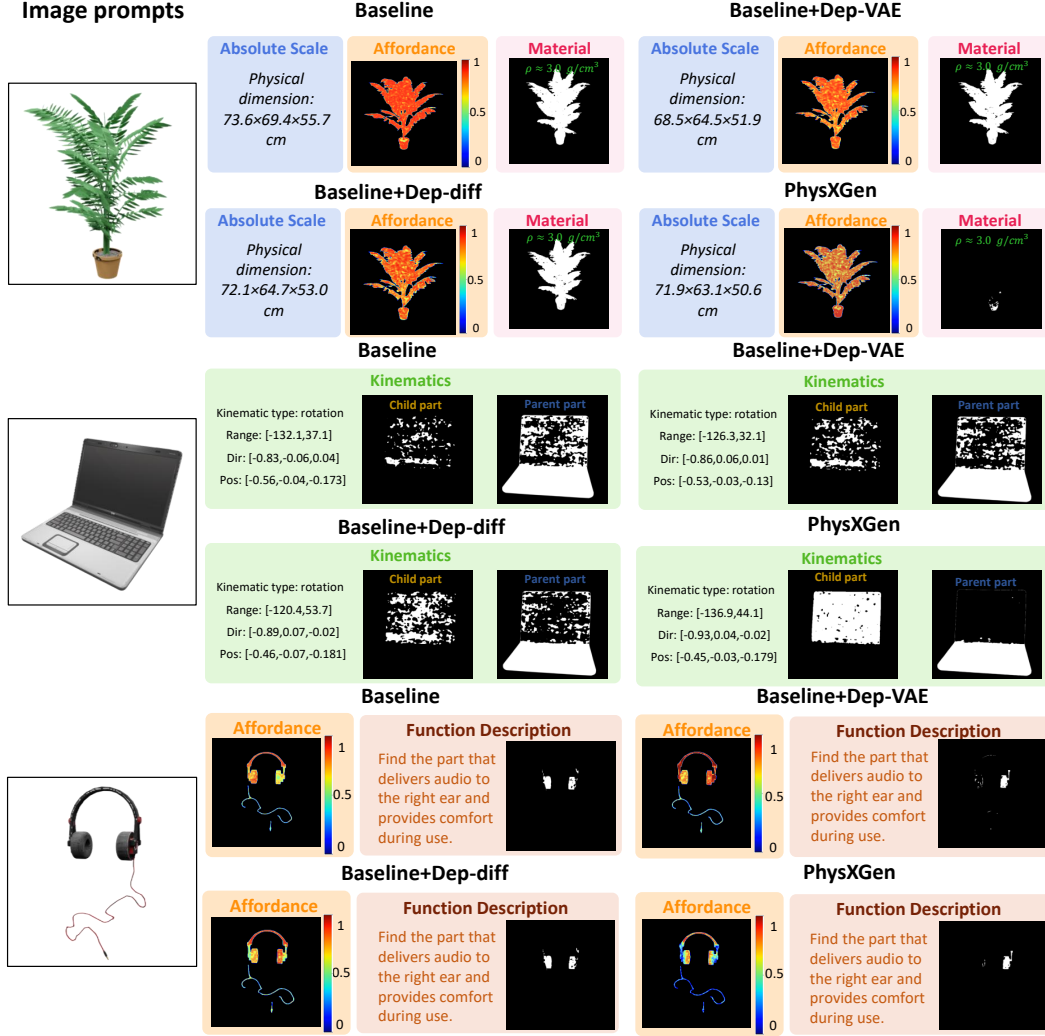


Figure 4: Qualitative comparison of different architectures.

4.2 Qualitative comparison among different architectures

Additionally, we implement the qualitative evaluations of the different architectures in ablation studies (see Figure 4). By integrating the correlation between geometry and physics in VAE and diffusion model, the generative performance of physical properties is improved gradually. In **material**, **kinematics**, and **affordance**, our PhysXGen is more stable and accurate in determining the target region with fewer artifacts.

5 Further analysis on challenges in physical-grounded 3D generation

In this section, we analyze the new challenges in physical-grounded 3D asset generation. For clarification, we summarize the special challenges in physical property generation.

Absolute scale: Our experimental results with PhysXGen reveal a constraint in absolute scale prediction: conventional normalization strategies prove inadequate for handling the inherent challenges of dimensional distribution. The absolute scale measurements exhibit a long-tailed distribution spanning three orders of magnitude (1-1000 cm), with a concentration of most samples below 300 cm. This long-tailed distribution makes linear normalization suboptimal due to its poor preservation of relative scale differences in the predominant sub-300 cm range. While logarithmic normalization presents a

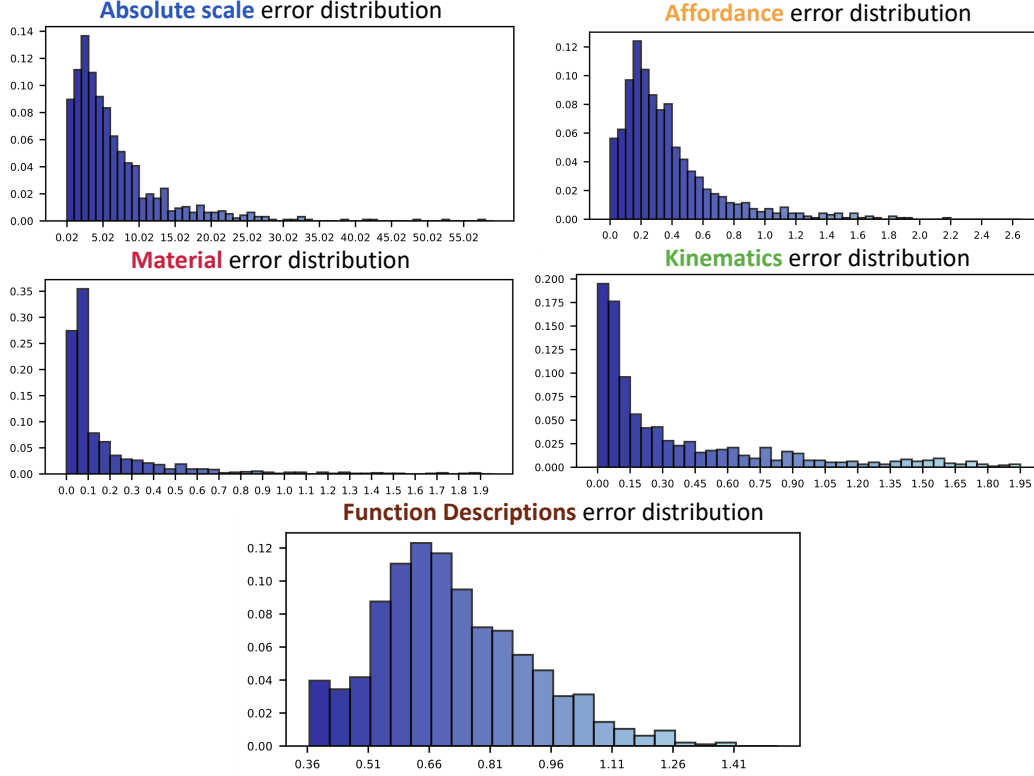


Figure 5: Error distribution of different physical properties.

compelling alternative for handling its span, direct implementation would disproportionately compress the feature space where most of the objects reside (1-300 cm range), potentially diminishing discriminative power within this critical operational regime. Figure 5 shows the error distribution in **Absolute scale**. Our PhysXGen is hard to maintain robustness in generating extremely large objects.

Material and **Affordance**: Our analysis further identifies analogous normalization challenges in material density prediction (0-10 g/cm³ range), though with diminished urgency compared to absolute scaling due to the constrained parameter space. However, a more critical limitation emerges in physical property coherence: both affordance estimation and material prediction exhibit spatial inconsistency artifacts as shown in Fig. 4. Besides, we report the error distribution in the two metrics in Fig. 5. As evidenced by the distribution figure, artifact-induced perturbations manifest as spatially scattered data points in the distribution. Furthermore, although morphological post-processing can enhance the generated results, the inconsistency in the physics space of neighboring regions may obstruct further improvement in physical-grounded 3D asset generation.

Kinematics: As the fine-grained physical properties, we split the kinematics into several parameters: **1) child part**; **2) parent part**; **3) movement type**: A. No movement constraints (like water in a bottle), B. Prismatic joints (like a drawer), C. Revolute joints (like a laptop), D. Hinge joint (like a hose in a shower system), or E. Rigid joint; **4) kinematic parameters** including rotation/movement direction, location of rotation axis, rotation/movement range. For challenges 1) and 2), the inherent difficulty in determining the number of parts during generation precludes effective implementation of classification-based loss. Consequently, in our method, our adoption of regression-based prediction inadvertently introduces artifacts in hierarchical part determination (parent-child relationships). More critically, the absence of explicit mapping between 3D coordinate systems and geometric structural features increases the difficulty in building kinematics space and inserting the correlation between physics and geometry shown in Fig 5.

Function description: Our framework leverages CLIP [4] for text embedding extraction, subsequently performing dimensionality reduction through 3D VAE to establish a compressed latent space. This architecture enables joint learning of all physical properties. However, the non-invertible

nature of CLIP’s encoder-only architecture fundamentally restricts embedding-to-prompt disentanglement, thereby constraining interpretability in downstream 3D semantic reasoning tasks. Meanwhile, compared with other physical properties, text embedding is more complex to learn and generate. As shown in Fig 5, the error in normalized function descriptions is larger than other properties. Furthermore, while encoder-decoder foundation models like T5 [5] theoretically permit decoding, their high-dimensional embedding spaces impose prohibitively expensive computational overhead for cross-domain alignment with physical properties.

6 Ethics statement

Human Annotator Ethical Concerns: All annotations used in this study were performed by the authors. No external participants were involved, and no personal or sensitive data were collected. According to our institution’s ethical guidelines, this study does not constitute human subjects research and therefore did not require IRB approval.

Clarification of dataset license: Since 3D data in PhysXNet and PhysXNet-XL are derived and modified from PartNet and ShapeNet, users are required to comply with the ShapeNet license terms².

Potential biases: While our dataset, PhysXNet, offers a new collection of 3D objects annotated with rich physical properties, we acknowledge that the dataset may contain representational biases, which we plan to address in future work. Moreover, we caution against the direct application of our methods in safety-critical domains (e.g., autonomous robotics or medical devices) without rigorous validation, as errors or inaccuracies in physical property predictions could result in adverse outcomes. Finally, since part of the annotation process involves vision-language models (VLMs), specifically GPT-4o (costs over \$1k), the dataset annotations may reflect biases inherent in these models despite subsequent human verification.

Listing 1: System prompt for part-based annotation (GPT)

```
You have a good understanding of the structure of an articulated
object. Your job is to assist the user in analyzing the
properties of it. Specifically, the user will give you
images of parts, and your task is to recognize the
articulated object and analyze the parts of that object.
You should find a similar physical 3D object in the real
world. Based on human knowledge of it, you should give
your answer about the information as follows:

Object-level:
(1) name, category, and dimension (length*width*height, in cm) of
the articulated object.

Part-level:
Part_1 (image_1):
(1) Label, name, material, density (g/cm^3) of the part.
(2) priority rank of being touched when using this object based
on human preference.
(3) labels of all neighboring parts.
(3.1) assign a movement type for each group between Part_1 and
its neighboring parts (A. merely touch and no movement
constraints, B. relative translationally move, C. rotation
about an axis, D. rotation about a point, or E. rigid
constraint). If the movement type is B, C, or D, output
the parent and child parts.
(3.2) assign a movement type for each group between Part_1 and
its neighboring parts (A. merely touch and no movement
constraints, B. relative translationally move, C. rotation
about an axis, D. rotation about a point, or E. rigid
constraint). If the movement type is B, C, or D, output
the parent and child parts.

...
```

²<https://shapenet.org/terms>

(4) summarize the basic information (including material, physical dimension, category, and name), functional, movement description, and priority of being grasped description.

Part_2 (image_2):

(1) Label, name, material, density (g/cm³) of the part.

(2) priority rank of being touched when using this object based on human preference.

(3) labels of all neighboring parts.

(3.1) assign a movement type for each group between Part_2 and its neighboring parts (A. merely touch and no movement constraints, B. relative translationally move, C. rotation about an axis, D. rotation about a point, or E. rigid constraint). If the movement type is B, C, or D, output the parent and child parts.

(3.2) assign a movement type for each group between Part_2 and its neighboring parts (A. merely touch and no movement constraints, B. relative translationally move, C. rotation about an axis, D. rotation about a point, or E. rigid constraint). If the movement type is B, C, or D, output the parent and child parts.

...

(4) summarize the basic information (including material, physical dimension, category, and name), functional, movement description, and priority of being grasped description.

For example:

```
{
  "object_name": "Rifle",
  "category": "ToyGun",
  "dimension": "80*10*25",
  "parts": [
    {
      "label": 1,
      "material": "Plastic",
      "density": "1.2 g/cm^3",
      "name": "Foregrip",
      "priority_rank": 2,
      "neighbors": [
        {
          "labels_of_movement_group": "1-8",
          "movement_type": "E",
        }
      ],
      "Basic_description": "It's a foregrip of a Rifle made of plastic.",
      "Functional_description": "It can control the ...",
      "Movement_description": "It cannot move normally...",
      "Grasped_description": "Most likely to be grasped or handled.",
    },
    {
      "label": 2,
      "material": "Plastic",
      "density": "1.2 g/cm^3",
      "name": "Stock",
      "priority_rank": 5,
      "neighbors": [
        {
          "labels_of_movement_group": "2-8",
          "movement_type": "B",
          "parent_label": 8,
          "child_label": 2
        }
      ],
      "Basic_description": "It's a foregrip of a Rifle classified as a gun. It is a big part of the object made of plastic.",
    }
  ]
}
```

```

    "Functional_description": "It can be grasped to control the
        object...",
    "Movement_description": "It cannot move normally...",
    "Grasped_description": "Less likely to be grasped.",
    ]
},
...
}

Remember:
(1) Do not answer anything not asked.
(2) You should base on the physical 3D object in the real world
    to analyze the properties and movement of the object.
(3) You should purely based on its function to determine the
    movement type of parts.
(4) You should prefer to analyze the rendered object as a real 3D
    object rather than a toy model.
(5) You should assign the priority rank of being grasped from 1
    to 10. The most likely part to be touched is 1.
(6) You should consider the function rather than the area or name
    of the target part to determine the priority rank of being
    grasped.
(7) The target part uses red color while the other parts use grey
    color.
(8) You should output full JSON including all parts.

```

References

- [1] 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 preprint arXiv:1512.03012*, 2015.
- [2] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. *arXiv preprint arXiv:2504.11451*, 2025.
- [3] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.
- [4] 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 *International conference on machine learning*, pages 8748–8763. PmLR, 2021.
- [5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- [6] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.