

# Infinite Mobility: Scalable High-Fidelity Synthesis of Articulated Objects via Procedural Generation

Anonymous ICCV submission

Paper ID 3835



Figure 1: We design probabilistic programs to generate 22 common articulated objects. We demonstrate the motion sequence of the generated articulated objects. Ours generated articulated objects bear accurate geometry, realistic textures, and reasonable joints.

## Abstract

Large-scale articulated objects with high quality are desperately needed for multiple tasks related to embodied AI. Most existing methods for creating articulated objects are either data-driven or simulation based, which are limited by the scale and quality of the training data or the fidelity and heavy labour of the simulation. In this paper, we propose Infinite Mobility, a novel method for synthesizing high-fidelity articulated objects through procedural generation. User study and quantitative evaluation demonstrate that our method can produce results that excel current state-of-the-art methods and are comparable to human-annotated datasets in both physics property and mesh quality. Furthermore, we show that our synthetic data can be used as training data for generative models, enabling next-step scaling up.

## 1. Introduction

The scaling law has demonstrated its efficacy across diverse machine learning domains, including computer vision, natural language processing, and reinforcement learning. Yet, scaling the training of embodied AI-related tasks[37, 27, 29] remains challenging, largely due to the scarcity of high-quality training environments. Sim-to-real transfer[30] offers a promising solution by enabling training in simulation and then transferring the results to reality. Articulated objects, such as cabinets with drawers and fridges with openable doors, are common in the real world. They can depict objects' fine-grained structures and the corresponding manipulation operations, making them crucial for simulating realistic action chains in virtual environments. Although these objects are vital for the sim-to-real paradigm, there are limited datasets of them, and most existing datasets [37, 21, 18, 12, 20] are relatively small in scale.

To create sufficiently realistic digital twins of articulated

108 objects for sim-to-real transfer, it is essential to have both  
109 precise physical properties of joints and high-quality 3D assets.  
110 It's natural to try to create a real-to-sim-to-real loop by  
111 reconstructing articulated digital twins from real world ob-  
112 jects. But such methods[21, 11, 16, 18, 20] need manually  
113 collected data which is labor-intensive and time-consuming.  
114 Quality of 3D assets from these reconstructed objects are  
115 also not guaranteed. Some try to articulate existing 3D  
116 assets[37, 33, 38]. However, most of such methods rely  
117 on existing or self-made datasets for priors of articulaion  
118 information and thus are still limited by data scale. In re-  
119 cent years, generative models[14, 17] have been adopted to  
120 generate articulated objects. These models leverage popular  
121 diffusion models[8] as generation backbones to capture the  
122 training dataset distribution, but their performance remains  
123 suboptimal as most datasets contain only dozens of samples  
124 per category. It remains a challenge to obtain large amounts  
125 of high quality articulated 3D objects in a efficient way.  
126

127 In this paper, we introduce Infinite Mobility, a pro-  
128 cedural pipeline for synthesizing large-scale articulated ob-  
129 jects. For each object, we represent its articulation as a  
130 tree structure analogous to a URDF, where nodes corre-  
131 spond to links and edges represent joints. Our approach  
132 leverages a tree-growing strategy for articulation structure  
133 generation. Starting from a root node, we iteratively grow  
134 the tree by attaching new nodes to existing ones. Based  
135 on the semantic characteristics of each node, we have de-  
136 vised a set of growth rules that decide whether to attach a  
137 new part and specify which component to add if needed.  
138 This strategy ensures that essential components are gener-  
139 ated while non-essential parts are incorporated gradually,  
140 facilitating the reliable and diverse generation of articula-  
141 tion structures. In contrast to above methods which rely on  
142 noisy model inference for joint information, our proce-  
143 dural paradigm provides full control over the semantics and  
144 geometry of each part and joint, thereby ensuring the ac-  
145 curacy of physical properties. By configuring the pipeline  
146 with specific parameters, our programming rules can gen-  
147 erate highly sophisticated articulation trees. With no upper  
148 limit on tree size, our pipeline is capable of constructing  
149 objects that extend beyond the scope of existing datasets or  
150 reconstruction from real scenes. Beyond these, our hybrid  
151 asset pipeline seamlessly integrates procedurally generated  
152 meshes with curated dataset assets during the assembly of  
153 articulated objects, ensuring both the quality and diversity  
154 of the final mesh.

155 To demonstrate the superiority of our pipeline, we con-  
156 duct evaluations in both physical property and mesh qual-  
157 ity, comparing our results with human-annotated datasets  
158 and state-of-the-art generative methods. Physical properties  
159 can only be evaluated in simulation, so we use Sapien[37]  
160 to record the movement of each joint and invite human  
161 annotators to rate the fidelity of videos. Regarding mesh

162 quality, we rendered both normal maps and RGB images  
163 for each articulated object, constructing paired samples for  
164 comparison. We leveraged vision language models (VLMs)  
165 as evaluators[36] to facilitate large-scale testing. Results  
166 show that our pipeline provides results slightly better than  
167 datasets and defeats state-of-the-art methods in all aspects.  
168 Furthermore, we utilized our synthetic data to train genera-  
169 tive models, thereby facilitating subsequent scaling efforts.  
170

In summary, our contributions are three-folded:

- We propose Infinite Mobility, a novel pipeline for syn-  
thesizing high-fidelity articulated objects through pro-  
cedural generation boosted by 3D datasets annotated  
with part information.
- We demonstrate that our method can produce re-  
sults that excel current state-of-the-art methods and  
are comparable to human-annotated datasets in both  
physics property and mesh quality.
- We show that our synthetic data can be used as training  
data of existing generative models, enabling next-step  
scaling.

## 2. Related Work

Both procedural generation and articulated objets have  
been in public view for decades. In this section, we com-  
pare the unique focuse of our work with existing procedural  
generators and reveal the vulnerability of other articulated  
objects data sources.

**Procedural Generation For Machine Learning** Proce-  
dural generation has been broadly used in providing training  
data and building evaluation environments. Since  
its debut in the 1980s, procedural generation quickly be-  
came a popular trick for game developing. Agents[2]  
playing such games as *Rogue*, *Minecraft*, *Diablo III* and  
*Civilisation VI* have attracted much attention. Procgen  
Benchmark[5] provides 16 procedrally generated game en-  
vironments for multiple tasks. In reinforcement learning,  
domain randomization[30, 34] is a simple form of proce-  
dural generation and one way to mitigate overfitting in ma-  
chine learning. To prepare more complex scenes for embed-  
ded AI, ProcTHOR [6] built a procedural system with more  
than 1000 static objects. Pushing the boundary of proce-  
dural generation, Infinigen[25, 26] use purely programming  
rules with layout constraint solvers and achieve remarkable  
results. All those works are mostly focused on generating  
scenes with static objects but ignoring more fine-grained in-  
teractive details of objetc.

**Articulated Object Datasets** Articulated object datasets  
are a category of datasets that contain deformed mesh  
parts annotated with fine-grained hierarchical articulation

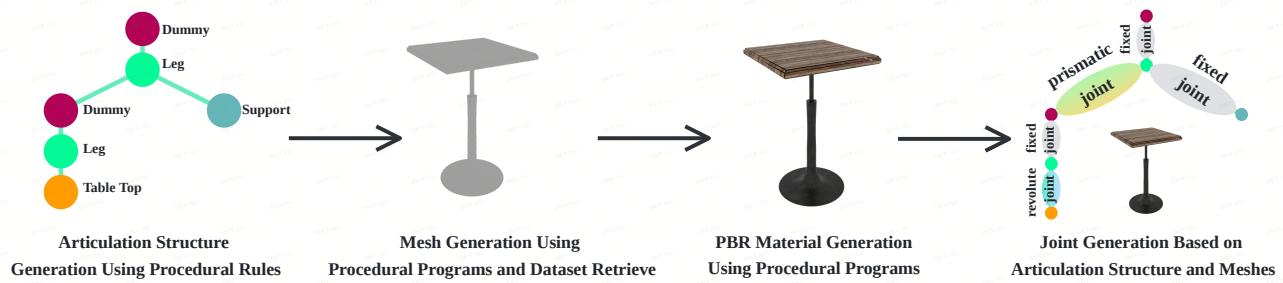


Figure 2: The whole pipeline can be divided into four parts: articulation tree structure generation, geometry generation, material generation and joint generation.

information. Various datasets have been proposed to provide assets in similar formats[3, 22, 21, 9, 15, 37, 18, 12, 23, 20]. ShapeNet[3] and PartNet[22] are two datasets that pioneered to provide relatively large-scale part annotations either labelled in a combined pipeline or manually and PartNet[22] has begun to offer hierarchically semantic information. Based on these two datasets, PartNet-Mobility[37] selected 46 categories of articulated objects and endowed them with high-quality articulation information. Extended from PartNet-Mobility, AO-Grasp[23] focused on grasping interaction with articulated objects and labelled a subset of PartNet-Mobility[37] with related data. Those datasets, through some introduced annotation algorithms, still require massive labor for annotation and are limited in scale. Instead of being derived from existing asset datasets, MultiScan[20], AKB48[18], ParaHome[12], RoCap[15] and RBO[21] consider from a real-to-sim perspective. The first three use real-world scans or images to reconstruct meshes with articulation information, while the latter document interactions with interactable objects for articulation. However, those digital twins that were reconstructed are not necessarily equipped with satisfactory mesh topology and articulation information. Apart from this, the real-to-sim paradigm can hardly supply objects with sophisticated inner articulation structures or have not been made in reality.

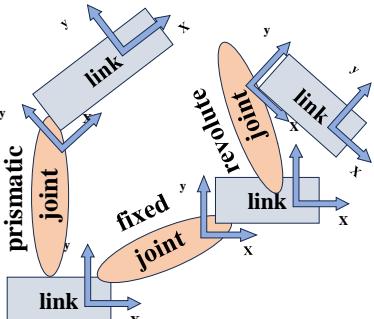
**Articulated Object Generation.** Some approaches to creating articulated objects are quite similar with those used to create datasets. Part of real-to-sim methods[4, 16, 19, 13, 24] reconstruct whole object mesh and deploy segmentation algorithms to obtain part with single functionality and rely on articulation perception algorithms for articulation. Others[31, 35, 7, 32] reconstruct object as implicit representation and retrieve desired parts after articulation perception. These approaches need accurate articulation perception models for high-fidelity results. Earlier ones[4, 16, 31, 35, 7, 32] rely on their own models which depend on data-driven training and the newest ones[19, 24, 13] resort to VLMs or LLMs which are not specialized in this

task for results. Attempts have also been made to generate articulated objects from real-world simulation and virtual simulators. Those routes[11, 28, 39, 10] usually use videos or object in different poses to infer articulation information. However, the heavy labour and the noise in the real-world or virtual simulation data mired those ideas. Generative[14, 17] models have also been used in this area. Built upon ScrewNet[10], NAP[14] design a new articulation tree parameterization and then apply a DDPM[8] as generation backbone. CAGE[17] continues to deploy DDPM[8] but with a more novel graph attention as network model and inject control signals for better usages. Nonetheless, limited training data still hinders generation with high standards.

### 3. Methodology

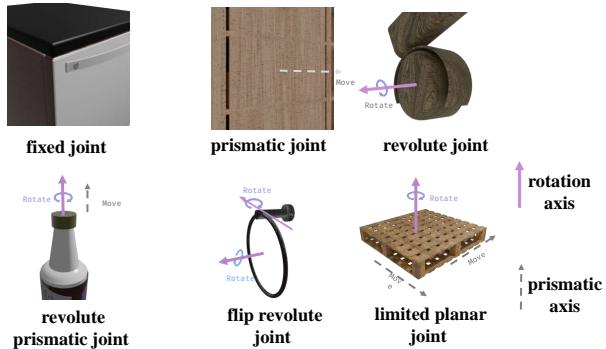
#### 3.1. Preliminary

Articulated objects constitute a class of objects that consist of multiple rigid bodies connected by various joints. We refer to URDF (Unified Robot Description Format) as the standard format for describing articulated objects. Each rigid body is described as a link, and each joint is a connection between two links. With such structure, an articulated object can be described as a tree structure, where each node represents a link and each edge represents a joint. A simple structure described by URDF is shown in Figure 3. In URDF, each link is described by its shape which can be either one of its predefined shapes or path to meshes stored in common formats. Joints in URDF are described by their origin, type, axis, limits, and dynamics properties. With regularly supported joint types like revolute, prismatic, continuous and fixed URDF is capable to portray simple movements with low DOFs. To describe more complex movements, URDF allows for the definition of compound joints, which are joints that are composed of multiple simple joints and dummy links with no shape specified. Such compound joints with simple joints connected one by one enable the description of complex movements with high DOFs. Joints



324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
Figure 3: Structure of the URDF file. Each link is a part of the object, which is represented as a textured mesh in our case. Each joint connects two links and describes the articulation structure between them.

we implemented are illustrated in Figure 4. Little work has been done to generate articulated objects with compound joints which are common in daily objects.



355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
Figure 4: We implement 6 kinds of joints in our articulated objects. First three are simple joints, and the last three are compound joints.

### 3.2. Articulated Object Generation

For the construction of articulated objects, both geometric and articulation data must be synthesized. Contrary to real-to-sim methods[21, 11, 16, 18, 20] and approaches using given 3D asset datasets[37, 33, 38] which extract articulation information out of an given object, our pipeline is much like those generative models for articulated objects[14, 17] which build the articulation structure first and then fill each node with meshes. Our work generates all articulation structures through a collection of purposefully designed programs that utilize randomly sampled input parameters. The codebase employed for result generation is systematically organized into category-specific generators, thereby enhancing usability and facilitating more efficient implementation. With most shape related parameters controllable by users and joint related parameters sampled from range calculated out of shape parameters, we can ensure both highly flexible mesh control and high-fidelity articu-

lation structure generation to be achieved by our pipeline. Workflow of our pipeline is summarized in Figure 2.

**Articulation Tree Structure Generation** In our pipeline, articulation tree structures are formulated as adjacency lists. With each node connected to only one parent joint, we can easily constraint a tree structure to be formed. For each daily-life category, there exists a general understanding of proper part attachments. Leveraging this understanding, we can establish a category-specific growth strategy for articulation trees. For every node with certain semantics, we define its requested children and plausible branches that can be attached to it. Those requested children depict the basic structure of the object and are necessary for the object to be complete like the legs and table top of a table. Plausible branches are optional and can be added to the object to increase its diversity like buttons of microwave. Commencing from a root node, which signifies the object’s origin, nodes can be recursively appended to the tree structure. For nodes amenable to attachments, the addition of new branches is dictated by discrete random parameters. Through recursion, a plausible tree structure is derived for each sample.

**Geometry Generation** Our geometry generation process is implemented via two approaches: procedural mesh generation and mesh retrieval with procedural refinement. For both methods, we first determine the bounding box of each node based on its semantics and generated meshes of other nodes. We will introduce our mesh retrieve approach in detail later in section 3.3. For procedural mesh generation, our pipeline is based on Infinigen[25, 26] augmented with our elaborate mesh creation programs.

**Material Generation** Materials used in our work are based on Blender shader node system. The material generation process is based on Infinigen[25, 26] boosted by collected crafted materials. To add more randomness, we modify diffuse, specular, roughness and normal maps of materials with random noise. For exportation, we resort to API created by Infinigen to save materials and textures in a format that can be easily imported into simulation tools.

**Joint Generation** Once the articulation tree structure is generated, the joint type for each edge is determined based on its parent and child nodes. The physical properties of each joint, including its axis, position, and motion range, are derived from its type and the overall generated meshes. This approach enables the generation of articulation trees that are both topologically diverse and physically high-fidelity. Result samples are shown in Figure 5.

### 3.3. Procedural Mesh Retrieve

As previously, meshes used in our pipeline can be retrieved from our curated dataset. This section outlines the methodology behind our dataset construction and details the mesh retrieval process.

378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431

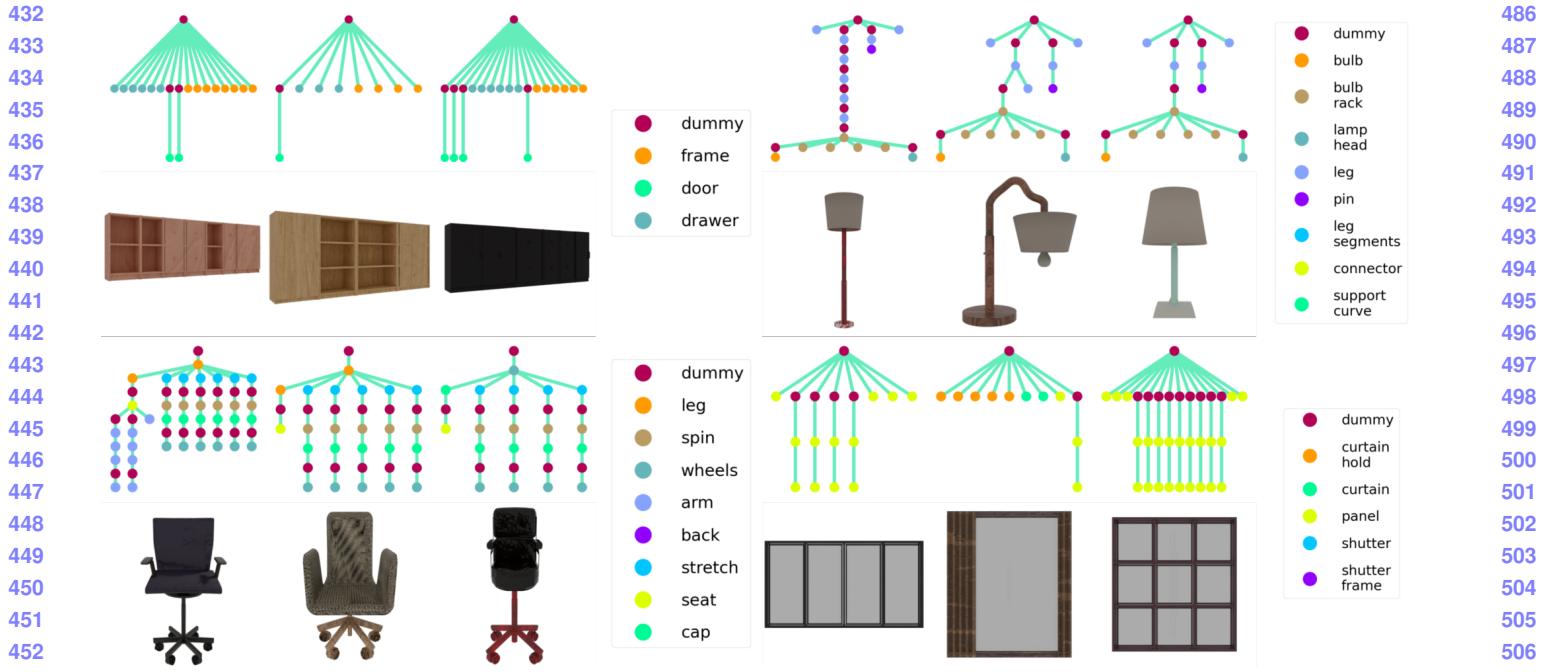


Figure 5: Examples of our generated cabinets, chairs, lamps and windows. For each object, we display the textured mesh with the corresponding articulation tree above. The generated objects are diverse in both shapes and articulation structures.

**Dataset Construction** 3D asset datasets like ShapeNet[3] and PartNet[22] use segmentation algorithms to obtain part meshes and label them with various annotations. However, those meshes in such datasets are not consistent in quality and part with single functionality is often segmentated into many pieces, resulting in bad visual fidelity and obstacles when applying new materials. We concentrate on 22 part labels which are most commonly interacted when agents are trained to perform tasks and query those parts according to labels. By inspecting and fixing those meshes by human labor, we obtain a curated dataset containing hundreds of meshes with requisite information. Comparisons between meshes from our dataset and meshes in the original dataset are shown in Figure 7.

**Mesh Retrieve with Procedural Refinement** According to the articulation tree generated, we can determine the semantics of every node and thus retrieve mesh from dataset mentioned above. For most existing methods, retrieved meshes are only transformed to fit bounding boxes. However, if new meshes are placed directly into required bounding boxes, there would be parts with complex shapes that either suspend in the air or penetrate into other parts. To avoid such situations, we only use bounding box for initial placements and then define each part label with procedurally defined critical supporting points. By aligning meshes with those points, we can ensure that the final mesh is physically plausible and visually appealing. Example of such situation and our fix is shown in Figure 8. Qualitative re-

sults are shown in Figure 6.

### 3.4. Ensure Physical Plausibility

Creating fascinating meshes combined in a tree structure with proper semantics is able to ensure the basic fidelity of the generated objects. However, to adopt those objects in simulation tools, a series of amendments need to be adopted to conquer following problems we observed.

**Problem: Ill-designed parts lead to unreasonable collisions with rigid ground.** Some artists make models different from reasonable real world objects for better appearance and simplicity. This phenomenon happens frequently when it comes to the base part of an object. Objects created this way suffer from problems when parts in lower positions are articulated with joints moving downward.

**Modification:** For those categories, we either build a base to position the articulated part further from the ground, or we modify the articulated parts by shrinking their sizes and limiting their motion ranges to a proper number.

**Problem: Insufficient gap between parts cause unstable movements of articulated joints.** It's common to build meshes as a whole with no gaps between parts for convenience when modeling. However, this makes it difficult for simulation to handle the collisions when such meshes are directly divided into parts and imported. When meshes are imported into simulation, different simulators have different strategies for collision detection. But those methods hardly use original meshes as the real colliders but resort to



Figure 6: We use two methods to obtain the geometry of each part of an object. 1) Blender python API created meshes. 2) Parts retrieved from our carefully curated and processed dataset. For each group of objects, the left one is generated by Blender python API, while the right 3 objects are obtained by retrieving parts from curated PartNet-Mobility. Parts obtained from both methods can be seamlessly joined, and it improves the diversity of generated shapes by adopting both methods to obtain part geometry.

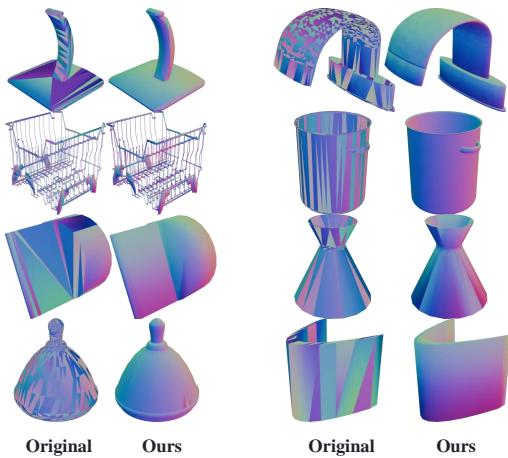


Figure 7: Original parts from PartNet and ShapeNet bear many back faces and thus their normals are highly irregular. We flip these back faces in Blender using recalculate normal function followed by human repair and ensure the meshes bear consistent outward-facing normals.

other approximations for efficiency. Such practice usually brings wrong collisions into the simulation of articulated joints even if those meshes are not overlapping.

**Modification:** For any parts articulated by joints, we additionally bring in gaps of 2% of original scales of meshes.

We present simulation results of one example and our amendment result in Figure 9. It's worth mentioning that problems mentioned above are quite frequent in articulated objects from other sources and appear in commonly used datasets like PartNet-Mobility[37].

## 4. Experiment

Our pipeline constitutes a high-quality data source similar to existing datasets like PartNet-Mobility[37] and sur-

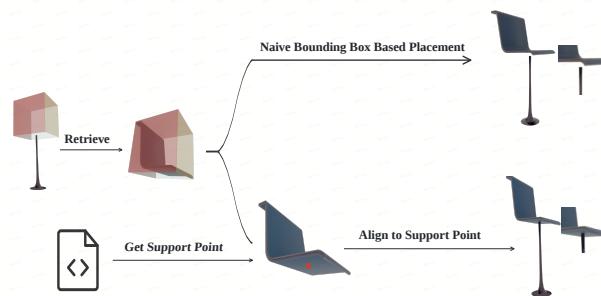


Figure 8: We adopt support point-based placement to position the retrieved part on the object. Naive bounding box-based placement may create a gap between the retrieved part and the object. Our approach guarantees a seamless connection.

pass generative model like NAP[14] and CAGE[17]. In this section, we evaluate the quality of our generated results, train a generative model on our results and verify the application of our results in embodied AI.

### 4.1. Evaluations of Articulated Objects

We first assess the quality of our articulated objects by comparing them against the PartNet-Mobility dataset[37] and samples produced by state-of-the-art generative models [14, 17]. To evaluate joint fidelity, we conduct human evaluations in which 10 participants review 50 randomly selected object pairs from each category, judging joint performance based on movement videos. One sample pair is displayed in Figure 10. In addition, visual language models (VLMs) are employed to assess mesh quality. Following a recent evaluation paradigm [36], we utilize GPT-4v as an evaluator by feeding it normal maps and RGB images, thereby enabling large-scale, unbiased testing that rates both the geometry and texture of the meshes. We present the results in Table 1

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

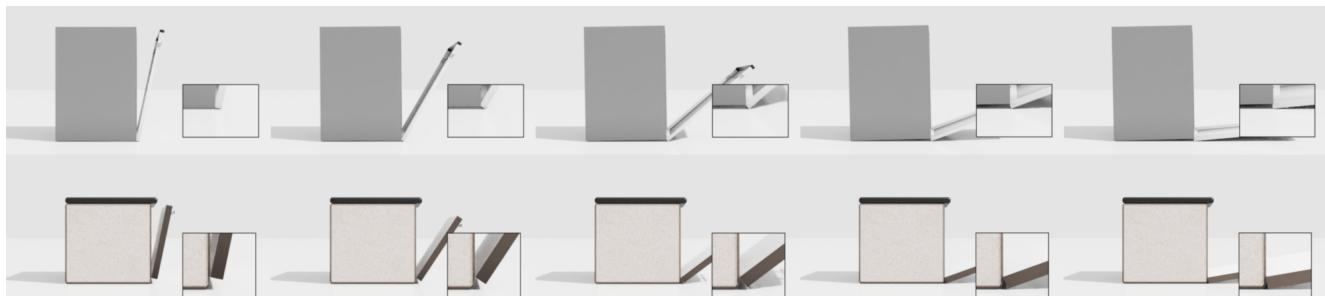


Figure 9: We carefully position each part and optimally configure the joint parameters to guarantee that no collisions occur either among the parts themselves or between the object and the ground during its articulation. The top row is a dishwasher from PartNet-Mobility. When the dishwasher door is opened, its base collides with the ground, forcing the main body to tilt upwards. While the dishwasher generated by our method in the bottom row does not have this problem.

and Table 2.

Complexity and diversity of our results and PartNet-Mobility[37] are also measured through quantitative metrics. For complexity, we calculated average joint numbers of each category from both data sources. For diversity, we calculated variances of joint number and average Tree-Edit-Distance between each pair of articulation trees in a category. Results are shown in Table 3.

We further compare the generation time of our pipeline with NAP[14] and CAGE[17]. Results are shown in Table 4.

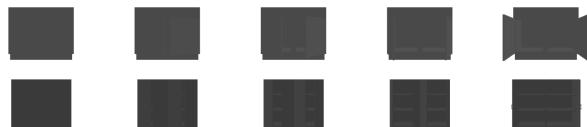


Figure 10: Human evaluators observe textureless videos of our generated articulated objects (lower row) and those from PartNet-Mobility (upper row), subsequently determining which exhibits superior motion structure.

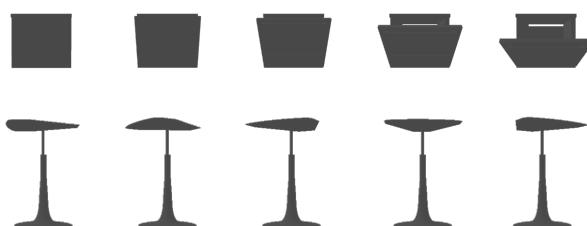


Figure 11: Examples of articulated objects generated by CAGE trained on our results. The generated object has accurate geometry and reasonable motion structure.

## 4.2. Training Generative Models

To further scale the generation of articulated objects, we train generative models using results from our pipeline. We use CAGE[17] as the generative model and train it with our generation results. Different from the original CAGE trained using PartNet-Mobility[37] which is small in scale, we used 1000 samples per category for training. We display two result samples in Figure 11.

## 4.3. Embodied AI

Our results can be imported in popular simulation environments like Sapien[37], Isaac Sim and Genesis[1]. In isaac sim, we use our results to annotate waypoints for articulated objects and train agents using motion-planning algorithms. We present samples in Figure 12.

## 5. Limitations

At this stage, Infinite Mobility still has some limitations which could be explored in future works. Since this version of Infinite Mobility is built on generators with largely human crafted rules, it requires considerable efforts to be extended to other categories. The automation of this process could probably be achieved by using LLMs with enough spatial intelligence and coding abilities. Joints parameters like type, axis and motion range have been created with high quality now, but properties like friction, damping and motor strength are still missing. Work could be done to infer those properties from meshes, materials and joint types.

## 6. Conclusion

In this paper, we present a procedural pipeline for generating high-fidelity articulated objects. Quality of procedurally generated data from our work beat those from other models and rival best dataset now in both appearances and articulation properties. Our work demonstrates the potential of procedural generation in creating not only appearance appealing but also structure diverse objects.

Category	Human Evaluation (Articulation)			GPT Evaluation (Geometry)			GPT Evaluation (Texture)			810 811 812 813 814 815 816 817 818 819 820 821 822 823 824
	Ours (%)	PartNet-Mobility (%)	Equal (%)	Ours (%)	PartNet-Mobility (%)	Equal (%)	Ours (%)	PartNet-Mobility (%)	Equal (%)	
All Categories	<b>8.76</b>	4.63	86.61	<b>64.18</b>	35.45	0.37	<b>84.81</b>	14.44	0.74	
Bottle	<b>9.64</b>	4.42	85.94	<b>78.00</b>	21.56	0.44	<b>91.56</b>	8.44	0.00	813
Dishwasher	<b>15.17</b>	4.99	79.84	<b>71.43</b>	28.57	0.00	<b>85.45</b>	14.55	0.00	814
Display TV	<b>8.20</b>	5.40	86.40	35.14	<b>64.86</b>	0.00	<b>58.86</b>	40.54	0.60	815
Door	<b>7.20</b>	3.20	89.60	<b>74.60</b>	25.08	0.32	<b>79.37</b>	20.32	0.32	816
Fridge	<b>3.00</b>	0.60	96.40	<b>69.79</b>	29.95	0.26	<b>87.34</b>	12.66	0.00	817
Lamp	7.40	<b>8.00</b>	84.60	<b>75.13</b>	24.11	0.76	<b>77.48</b>	21.62	0.90	818
Microwave	<b>7.39</b>	2.20	90.42	<b>62.39</b>	37.61	0.00	<b>64.10</b>	35.90	0.00	819
Oven	<b>8.76</b>	1.39	89.84	26.34	<b>73.66</b>	0.00	<b>56.84</b>	42.74	0.43	820
Pot	<b>4.20</b>	0.40	95.40	<b>54.67</b>	45.33	0.00	<b>82.22</b>	17.33	0.44	821
Table	<b>13.57</b>	4.79	81.64	<b>72.89</b>	26.89	0.22	24.67	<b>75.11</b>	0.22	822
Tap	<b>9.60</b>	0.20	90.20	<b>90.00</b>	10.00	0.00	<b>93.11</b>	6.89	0.00	823
Toilet	<b>22.86</b>	0.00	77.14	<b>59.33</b>	40.67	0.00	<b>87.56</b>	12.44	0.00	824
Cabinet	<b>9.40</b>	2.60	88.00	47.33	<b>52.67</b>	0.00	<b>93.65</b>	6.35	0.00	825
Window	3.21	<b>6.41</b>	90.38	<b>76.81</b>	23.19	0.00	<b>75.93</b>	24.07	0.00	826
Chair	1.99	<b>4.57</b>	93.44	<b>66.29</b>	32.20	1.52	<b>78.52</b>	21.48	0.00	827

Table 1: Compare articulated objects generated by our method and those from PartNet-Mobility in terms of kinematic articulation, geometric accuracy, and textural fidelity. We report the comparative success rates of our method versus PartNet-Mobility. “Equal” denotes that the human evaluator or GPT assess the quality of the two objects as equivalent.

Table 2: Compare articulated objects generated by our method, CAGE and NAP in terms of kinematic articulation and geometric accuracy.

Category	Human Evaluation (Articulation)			GPT Evaluation (Geometry)		
	Ours (%)	CAGE (%)	Equal (%)	Ours (%)	CAGE (%)	Equal (%)
All Categories	<b>22.95</b>	5.60	71.45	<b>79.26</b>	20.74	0.00
Table	<b>19.04</b>	7.62	73.35	<b>87.50</b>	8.33	4.17
Cabinet	<b>11.82</b>	9.82	78.36	48.00	<b>51.33</b>	0.67
Dishwasher	<b>29.58</b>	2.01	68.41	<b>65.62</b>	34.38	0.00
Microwave	<b>28.69</b>	3.20	70.92	<b>65.28</b>	34.72	0.00
Oven	<b>22.24</b>	2.20	77.56	<b>69.79</b>	29.95	0.26
Fridge	<b>29.28</b>	1.99	68.73	<b>84.31</b>	15.69	0.00

Category	Human Evaluation (Articulation)			GPT Evaluation (Geometry)		
	Ours (%)	NAP (%)	Equal (%)	Ours (%)	NAP (%)	Equal (%)
All Categories	<b>58.36</b>	7.96	33.69	<b>74.81</b>	24.81	0.37

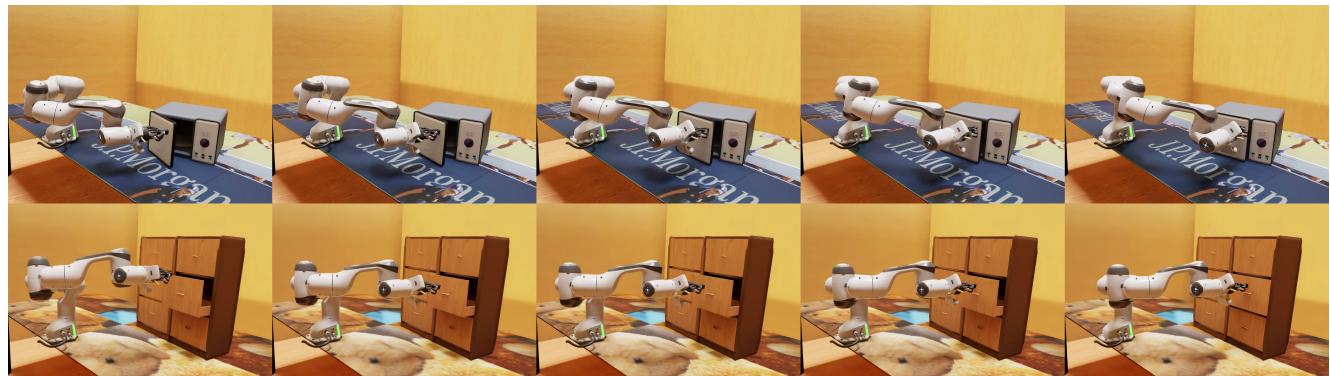


Figure 12: Kinematic sequences demonstrating a bimanual robotic manipulator interacting with our generated articulated objects within the Isaac Sim simulation environment.

Table 3: Compare articulated objects generated by our method and those from PartNet-Mobility in terms of joint numbers and diversity of tree structures.

Metrics	Ours	PartNet-Mobility
Average Joint Number	<b>12.32</b>	5.91
Variance of Joint Number	<b>659.02</b>	40.31
Tree Edit Distance	<b>78.62</b>	3.88

Table 4: Compare generation time of our method, NAP and CAGE.

Metrics	Ours	NAP	CAGE
Time(s / object)	<b>0.46</b>	2.04	1.96

864

## References

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

- [1] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. 7
- [2] Shaofei Cai, Zhancun Mu, Kaichen He, Bowei Zhang, Xinyue Zheng, Anji Liu, and Yitao Liang. Minestudio: A streamlined package for minecraft ai agent development. 2024. 2
- [3] 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. 3, 5
- [4] Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024. 3
- [5] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 13–18 Jul 2020. 2
- [6] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 2
- [7] Jianning Deng, Kartic Subr, and Hakan Bilen. Articulate your neRF: Unsupervised articulated object modeling via conditional view synthesis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2, 3
- [9] Ruizhen Hu, Wencho Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Learning to predict part mobility from a single static snapshot. *ACM Trans. Graph.*, 36(6), Nov. 2017. 3
- [10] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 13670–13677. IEEE Press, 2021. 3
- [11] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4
- [12] Jeonghwan Kim, Jisoo Kim, Jeonghyeon Na, and Hanbyul Joo. Parahome: Parameterizing everyday home activities towards 3d generative modeling of human-object interactions, 2024. 1, 3
- [13] Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Di-

- nesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In *The Thirteenth International Conference on Learning Representations*, 2025. 3
- [14] Jiahui Lei, Congyue Deng, Bokui Shen, Leonidas Guibas, and Kostas Daniilidis. NAP: Neural 3d articulated object prior. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2, 3, 4, 6, 7
- [15] Jiahao Nick Li, Toby Chong, Zhongyi Zhou, Hironori Yoshida, Koji Yatani, Xiang 'Anthony' Chen, and Takeo Igarashi. Rocap: A robotic data collection pipeline for the pose estimation of appearance-changing objects, 2024. 3
- [16] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. PARIS: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023. 2, 3, 4
- [17] Jiayi Liu, Hou In Ivan Tam, Ali Mahdavi-Amiri, and Manolis Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17880–17889, 2024. 2, 3, 4, 6, 7
- [18] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Yang Han, and Cewu Lu. Akb-48: A real-world articulated object knowledge base, 2022. 1, 2, 3, 4
- [19] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. In *The Thirteenth International Conference on Learning Representations*, 2025. 3
- [20] Yongsen Mao, Yiming Zhang, Hanxiao Jiang, Angel Chang, and Manolis Savva. Multiscan: Scalable rgbd scanning for 3d environments with articulated objects. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9058–9071. Curran Associates, Inc., 2022. 1, 2, 3, 4
- [21] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. The rbo dataset of articulated objects and interactions, 2018. 1, 2, 3, 4
- [22] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3, 5
- [23] Carlota Parés Morlans, Claire Chen, Yijia Weng, Michelle Yi, Yuying Huang, Nick Heppert, Linqi Zhou, Leonidas Guibas, and Jeannette Bohg. Ao-grasp: Articulated object grasp generation, 2024. 3
- [24] Xiaowen Qiu, Jincheng Yang, Yian Wang, Zhehuan Chen, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Articulate anymesh: Open-vocabulary 3d articulated objects modeling. *arXiv preprint arXiv:2502.02590*, 2025. 3
- [25] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings*

- 972        *of the IEEE/CVF Conference on Computer Vision and Pat- 1026*  
973        *tern Recognition*, pages 12630–12641, 2023. 2, 4 1027
- 974        [26] Alexander Raistrick, Lingjie Mei, Karhan Kayan, David 1028  
975        Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal 1029  
976        Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu 1030  
977        Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor 1031  
978        scenes using procedural generation. In *Proceedings of the IEEE/CVF 1032 Conference on Computer Vision and Pattern Recognition (CVPR)*, 1033  
979        pages 21783–21794, June 2024. 2, 4 1034
- 980        [27] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, 1035  
981        Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia 1036  
982        Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv 1037  
983        Batra. Habitat: A Platform for Embodied AI Research. In 1038  
984        *Proceedings of the IEEE/CVF International Conference on 1039 Computer Vision (ICCV)*, 2019. 1 1040
- 985        [28] Chaoyue Song, Jiacheng Wei, Chuan Sheng Foo, Guosheng 1041  
986        Lin, and Fayao Liu. Reacto: Reconstructing articulated ob- 1042  
987        jects from a single video. In *Proceedings of the IEEE/CVF 1043 Conference on Computer Vision and Pattern Recognition*, 1044  
988        pages 5384–5395, 2024. 3 1045
- 989        [29] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, 1046  
990        Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, 1047  
991        Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, 1048  
992        Vladimir Vondrus, Sameer Dharur, Franziska Meier, Woj- 1049  
993        ciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Ji- 1050  
994        tendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: 1051  
995        Training home assistants to rearrange their habitat. In *Ad- 1052  
996        vances in Neural Information Processing Systems (NeurIPS)*, 1053  
997        2021. 1 1054
- 998        [30] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Woj- 1055  
999        ciech Zaremba, and Pieter Abbeel. Domain randomization 1056  
1000        for transferring deep neural networks from simulation to the 1057  
1001        real world. In *2017 IEEE/RSJ International Conference on 1058 Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. 1, 2 1059
- 1002        [31] Wei-Cheng Tseng, Hung-Ju Liao, Lin Yen-Chen, and Min 1060  
1003        Sun. Cla-nerf: Category-level articulated neural radiance 1061  
1004        field. In *2022 International Conference on Robotics and Au- 1062  
1005        tomation (ICRA)*, page 8454–8460. IEEE Press, 2022. 3 1063
- 1006        [32] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. 1064  
1007        Template-free articulated neural point clouds for reposable 1065  
1008        view synthesis. *Advances in Neural Information Processing 1066 Systems*, 36, 2024. 3 1067
- 1009        [33] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qin- 1068  
1010        ping Zhao, and Kai Xu. Shape2motion: Joint analysis of 1069  
1011        motion parts and attributes from 3d shapes, 2019. 2, 4 1070
- 1012        [34] Lilian Weng. Domain randomization for sim2real transfer. 1071  
1013        *lilianweng.github.io*, 2019. 2 1072
- 1014        [35] Yijia Weng, Bowen Wen, Jonathan Tremblay, Valts Blukis, 1073  
1015        Dieter Fox, Leonidas Guibas, and Stan Birchfield. Neural 1074  
1016        implicit representation for building digital twins of unknown 1075  
1017        articulated objects. In *CVPR*, 2024. 3 1076
- 1018        [36] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, 1077  
1019        Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt- 1078  
1020        4v(ision) is a human-aligned evaluator for text-to-3d genera- 1079  
1021        tion. In *Proceedings of the IEEE/CVF Conference on Com-  
1022        puter Vision and Pattern Recognition (CVPR)*, pages 22227–  
1023        22238, June 2024. 2, 6