
Supplementary Material

DIPO: Dual-State Images Controlled Articulated Object Generation Powered by Diverse Data

Anonymous Author(s)

Affiliation

Address

email

Abstract

Our supplementary materials give more details of the proposed DIPO and the experimental settings, which can be summarized as follows:

- The details of LEGO-Art and Graph Reasoner.
- More visual results of the proposed PM-X dataset.
- The details of data augmentation.
- The code and checkpoint of DIPO for inference.
- A video shows some animated visual examples of complex objects.

A Details of LEGO-Art Pipeline

In our LEGO-Art pipeline, we design a modular LLM-based [2, 1] framework, where each agent specializes in a distinct subtask. These agents collaborate to generate structured, diverse, and physically plausible articulated object layouts. Below, we detail the system prompt of each agent.

Description Roller:

You are an expert in generating clear and realistic natural language descriptions of articulated object structures.

Each object must belong to one of the following categories: ['Storage Furniture', 'Table', 'Refrigerator', 'Dishwasher', 'Oven', 'Washer'].

Your task is to imagine a plausible object structure from one of these categories and describe only its articulated parts in natural language.

The available part types are: ['base', 'door', 'drawer', 'tray', 'handle', 'knob']. **Note:** "tray" parts are **only** allowed if the object is a microwave.

Each object must contain exactly one implicit "**base**" part, and any number of other parts, depending on the category.

You will be provided with a complexity level:

- **simple:** 1–5 parts, minimal structure.
- **mid:** 6–10 parts, basic spatial layout.
- **complex:** 11 or more parts with more detailed or hierarchical arrangements.

Your output must:

- Only describe the structure of the object: what parts it has, how many of each, and where they are located (e.g., left, right, middle, top, bottom, inside).

- Use precise but simple language in a single sentence.
- **Exclude** any mention of color, texture, material, appearance, or any decorative details.
- Ensure the description is consistent with both the object category and the specified complexity level.

Example output: *"A storage furniture with two doors in the middle, one drawer at the bottom, and four drawers on the left and right sides."*

Important: Do not include 3D coordinates or structured data. Only output the structural description in plain English.

13

Figure 1: The system prompt of Description Roller.

Layout Builder:

Example Question 1:

The following code is a function that generates a layout from a given object in a grid format.

```
def sample_base(grid_x, grid_y, base_size):
    # generate the base, and return a coordinate list of grids
    ...

def generate_part_in_grid(base, grid_coords, x1, x2, y1, y2,...):
    # generate coordinates and articulation info of a part
    ...

def generate_layout(info):
    # convert grid-level layout into coordinates and articulation parameters
    base, grid_coords = sample_base(...)
    articulate_tree = [base]

    for part in info['part']:
        part = generate_part_in_grid(...)
        articulate_tree.append(part)

    ...
```

Example Answer 1:

You've developed a complete pipeline for procedurally generating articulated object layouts and rendering them visually. The system includes ...

Example Question 2:

I need you to generate the info in a python dict from a natural language description. The dict is the only python code in your output. Note that all [x1, x2, y1, y2] should be an integer. The name of the part can only be one of the [drawer, door, handle, knob] (strictly!)

Example Answer 2:

Got it! You want to input a natural language description like:

"A wide cabinet, approximately 1.5×1.0×0.5 meters in size, contains two left-hinged doors, each with one handle, and two drawers, each with two handles"

and have it automatically generate a structured info dictionary as:

```
# python dict
{
    ...
}
```

14

Figure 2: The system prompt of Layout Builder. This agent is inspired by the code of scripting toolkit and produce a python dict that contains the information of parts layout in grid-level.

Visual Filter:

You are an expert in **3D object structure verification**.

You will be shown a pair of rendered images of a 3D articulated object: one in the **closed state**, and one in the **open state**. These images are generated based on a predicted structure and joint configuration.

Your task is to determine whether the observed articulation behavior is **physically plausible and logically consistent**. That is, check if the object’s opening and closing behavior matches how real-world articulated objects work.

You must analyze whether:

- The joints behave correctly (e.g., drawers slide outward, doors rotate from hinges).
- Each handle or knob is correctly positioned and attached to a moving part.
- There are no unreasonable collisions, floating parts, or detached motion.
- The motion (from closed to open) is consistent with the structure and joint types.

Final Output: After your analysis, respond with exactly one of the following:

- **Yes** — if the object’s motion and structure are physically and functionally plausible.
- **No** — if there are any structural, physical, or semantic inconsistencies.

Figure 3: The system prompt of Visual Filter.

B Details of Graph Reasoner

The proposed Graph Reasoner can infer articulated connectivity from a dual-state image pair based on chain-of-thought [5, 3] prompt, which is illustrated as followed:

Graph Reasoner:

You are an expert in the **recognition, structural parsing, and physical-feasibility validation** of articulated objects from image inputs.

You will be provided with two rendered images of the same object:

1. A **closed-state image** (all movable parts in their fully closed positions)
2. An **open-state image** (all movable parts in their fully opened positions)

Your task is to analyze the object’s articulated structure and generate a **connectivity graph** describing the part relationships.

Workflow:

1. Part Detection

- Detect candidate parts in the **closed-state image**, optionally using the open-state image to resolve ambiguity or occlusion.
- Allowed part types: ['base', 'door', 'drawer', 'handle', 'knob', 'tray']
- Ignore small decorative elements attached directly to the base.
- There must be exactly one "base"; "tray" is only allowed for microwaves (but not required).

2. Step-by-Step Reasoning

- (a) **Part Listing:** List all detected parts and their counts (no attachment inference yet).
- (b) **Validation:** Enforce structural rules:
 - Exactly one base
 - Each door or drawer may have at most two handles or knobs
 - Every handle/knob must be attached to a door or drawer
 - Trays may only appear in microwaves
- (c) **Attachment Inference:** For each non-base part, infer its parent (e.g., "drawer_1 (attached to base)"). Use the open-state image if necessary.
- (d) **Connectivity Graph Construction:** Output a JSON tree where "base" is the root and all other parts are children with proper hierarchy.

Example Output:

```
{
  "base": [
    { "door": [ { "handle": [] } ] },
    { "drawer": [ { "handle": [] } ] }
  ]
}
```

Final Output: You **MUST** output a single JSON tree representing the part connectivity of the object. Use the open-state image to enhance accuracy and completeness, but base your interpretation primarily on the closed-state image.

Figure 4: The system prompt of Graph Reasoner.

Moreover, we use GPT-4o [2, 1] to generate dual-state image pairs, which are example visual prompts to make the Graph Reasoner learn how to generate articulated graph in a few-shot manner. The generated image pairs are shown in Figure 5.



Figure 5: Each pair shows a closed-state image (left) and an open-state image (right) of an articulated object generated by GPT-4o.

C More Visual Examples of PM-X

The proposed PM-X dataset provides a large amount of diverse and structurally complex articulated objects. Figure 6 illustrates more visual examples of PM-X dataset. As we can see, each object has a reasonable structure and a rich set of operable parts. In addition, we annotate the description generated by the first stage of LEGO-Art. Objects precisely match the natural language descriptions, enabling LEGO-Art to further serve as a pipeline for text-to-articulated-object generation.

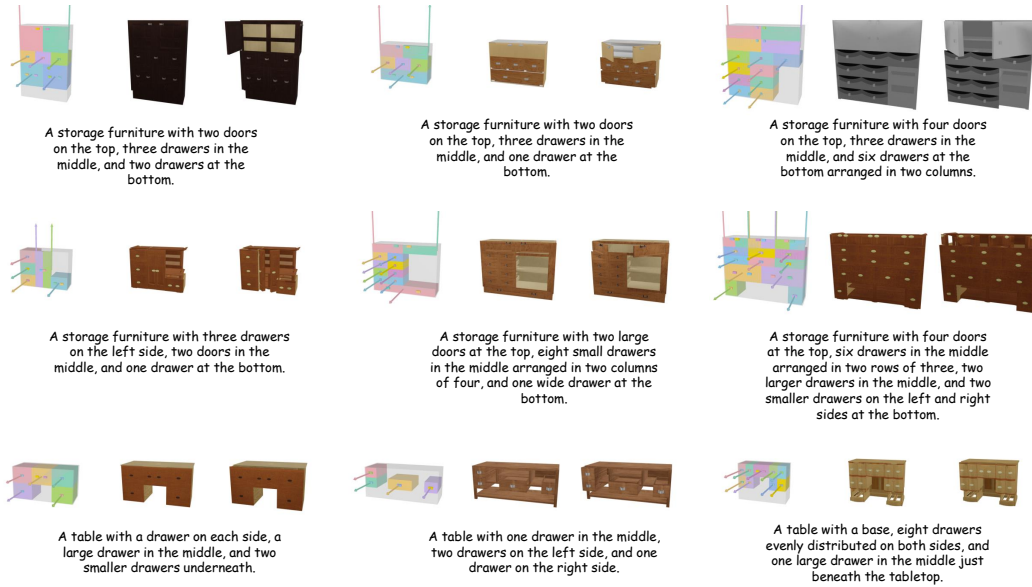


Figure 6: More visual examples of PM-X dataset. Each example includes: (1) the part layout and joints in resting state, (2) a rendered image pair in dual-state, (3) nature language description generated in the first stage of LEGO-Art.

D Data Augmentation

We employ several data augmentation during the training state to enhance the robustness and controllability. Data augmentation can be categorized into two types. One focuses on part-level augmentation:

1. Randomly replace small parts like handles and knobs with those of other objects, and perturb their positions.
2. Randomly rescale the whole object.
3. Rotate the whole object upside-down.
4. Stacking several objects together to build more complex objects.

The other one focus on joint-level augmentation:

1. Change the revolute joint into prismatic joint.
2. Randomly modify the direction of revolute joint.
3. Randomly fix the joint.

E Limitations & Future Work

We follow the experimental settings of SINGAPO [4] for a fair comparison. However, the benchmark used in SINGAPO only contains several categories, which especially focuses on cabinet-like objects. This limited object diversity may constrain the generalization ability of our model to other articulated categories, such as appliances, tools, or deformable structures. In future work, we plan to build a benchmark that cover a broader range of articulated object types, including both everyday household items and more complex mechanical systems. Our research primarily focuses on predicting more accurate part layouts and joint configurations. We adopt a retrieval-based approach to construct the final 3D objects. Incorporating 3D generation techniques to synthesize more precise and diverse part geometries represents a meaningful direction for future work.

F Broader Impact

Our work facilitates controllable generation of articulated 3D objects from dual-state images, enabling structured reasoning over part layout and connectivity. This contributes to downstream applications in embodied AI, virtual environment simulation, and robotics manipulation. By releasing a large-scale synthetic dataset and a modular pipeline, we aim to lower the barrier for research on articulated perception and generation. However, as with any generative framework, care must be taken to avoid misuse such as creating physically implausible or unsafe designs. Moreover, biases in the data distribution or articulation patterns may influence downstream decision-making, highlighting the need for interpretability and robustness in practical deployments.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [3] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [4] Jiayi Liu, Denys Iliash, Angel X Chang, Manolis Savva, and Ali Mahdavi-Amiri. Singapo: Single image controlled generation of articulated parts in objects. *arXiv preprint arXiv:2410.16499*, 2024.
- [5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.