Artiverse: A Diverse and Physically Grounded Dataset for Articulated Objects

Supplementary Material

A. Additional Statistics

In this section, we provide additional analysis of the data.

A.1. Kinematic Graph Diversity

Besides the part and joint counts presented in the main paper, another important diversity dimension is the diversity of kinematic graphs. Kinematic graphs can act as a proxy of the data complexity, for the categories that allow for various graphs.

We start by measuring a number of unique graphs, judging by part connectivity, semantic labels and motion types. As the absolute counts are biased by overall and per category data scale, we opt to compute the probability of the occurrence of each graph and measure the entropy of this distribution.

We proceed to compute the effective number (perplexity) of the graphs in the Artiverse and PartNet-Mobility distributions. We find that the effective number of kinematic graphs in Artiverse is 1.5 times higher. Overall, besides the categories that limit the graph diversity (i.e., scissors), Artiverse features more complex part arrangements.

A.2. Model Selection

Our dataset construction begins by sourcing high-quality, articulatable models across diverse categories. We collect and filter models from multiple repositories through a hybrid retrieval workflow that combines automated feature analysis with human oversight. For well-structured datasets with fine-grained taxonomy, such as HSSD [24] and 3DComPaT200 [2], we directly remap the objects at the category level. For large repositories with noisy or inconsistent metadata, we adopt a retrieval-by-example strategy. We first manually select a small set of models for each sub-category with the help of metadata (e.g., data tags, descriptions, category). Then we run k-means algorithm on DuoDuoCLIP [28] embeddings to find representative *anchor* models that serve as reference exemplars to expand more models in a semi-automated way. These anchors serve as visual exemplars that guide the retrieval of similar shapes from the model pool. We render multi-view turntable images for every candidate model and extract visual embeddings using DuoDuoCLIP. Similarity scores between candidates and anchors are computed to rank potential matches. To ensure category correctness and asset quality, we employ a custom web interface for human-in-the-loop verification of the top-k retrieved candidates. Models that align with the intended sub-category and exhibit realistic geometry and textures are retained. The resulting curated collection forms a high-quality starting pool for the downstream annotation pipeline.

B. Semi-automated Annotation Pipeline

In this section, we supply more implementation details of different steps in our data collection and annotation pipeline.

B.1. Data Preprocessing

After category-level selection, we preprocess all models to ensure normalization, consistent orientation, metric scale, and geometric uniqueness across the dataset.

To standardize orientation, we convert all assets into a shared y-up coordinate frame and make them zero-centered. Most objects naturally follow a consistent front-facing direction due to common modeling conventions. For the remaining minority, we apply axis-aligned rotations to align the canonical front-facing direction.

For metric scaling, we preserve the physical dimensions provided by the original data sources whenever available. When scale information is missing, we infer a plausible size by sampling from a range of physical dimensions per subcategory. These ranges are obtained using GPT-4o reasoning and then validated against the objects whose physical sizes are known, ensuring the predicted ranges remain realistic and category-appropriate.

To remove redundant models, we perform near-duplicate detection using DuoduoCLIP [28] feature similarity. We manually assess models with cosine similarity greater than 0.99 in the specialized interface. Additionally, we exclude models already present in PartNet-Mobility [53] to prevent overlap with existing articulated datasets.

B.2. Part Segmentation Proposal

Taking the selected and preprocessed models from previous steps, we annotate functional parts guided by the per-category template (see an example template design in Figure 10). Our method combines few-shot multi-view segmentation with geometry processing algorithms to produce high-quality part instance proposals that are subsequently verified by human annotators. The key steps are shown in Figure 7.

Few-shot semantic segmentation. For each category or group of similar categories (e.g., furniture), we manually annotate 5-15 representative objects with functional parts labels. Using these exemplars, we train the few-shot adaptive segmentation model ASIA [41] on 16 rendered views per object. Each input image is overlaid with projected 3D

part masks sorted by visibility, enabling the model to learn category-specific functional semantics. Once trained, ASIA is applied to the same 16 views for all objects within the category scope to obtain dense 2D semantic predictions.

Propagating 2D semantic regions to 3D. We precompute mesh over-segmentation based on triangle connectivity, and project 2D semantic predictions back to these segments. For each view, we iterate over visible segments and record the semantic labels and their pixel counts. Each segment is assigned the semantic label with the highest aggregate pixel support across views. For the occluded segments that are invisible to all rendered views, we find its nearest labeled segment and inherit the corresponding semantic label. This proximity-based assignment ensures that interior or shadowed regions receive consistent labels without requiring exhaustive rendering.

Clustering segments into part instances. To derive instance-level parts, we cluster the over-segments within a subset for each semantic label, based on the distance between them. For each semantic label, we consider all pairs of segments within that label and compute the shortest triangle-triangle distance between them. These pairwise checks are highly parallelizable. Segments are considered connected if their minimum distance is below a threshold (e.g., 1 mm) We then apply a union-find algorithm to obtain disjoint connected subsets, each corresponding to a functional part instance.

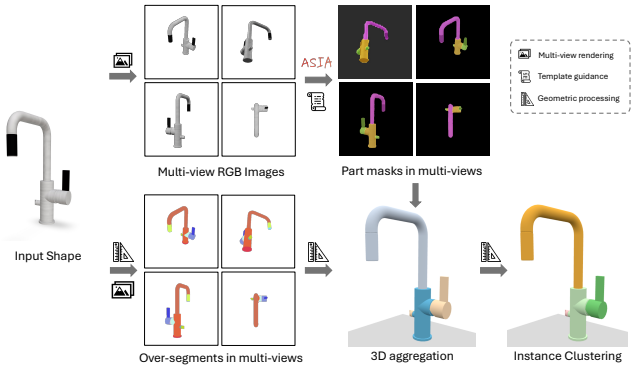

Figure 7. Key steps of proposing functional part segmentation. Given an input mesh, we 1) few-shot an adaptive segmentation model to predict masks on multi-view images for the parts predefined in the template; 2) propagate 2D semantic regions from multi-views to over-segments by assigning labels to mesh over-segments, including occluded regions, through visibility voting and proximity reasoning; and 3) cluster part instances into final part instances based on point-wise distanceb between over-segments.

B.3. Articulation Proposal

To generate articulated motion proposals, we design a set of geometric heuristics that infer joint parameters directly from 3D geometry. Our goal is to build a general, reusable framework that works across diverse object categories while requiring minimal per-class specialization.

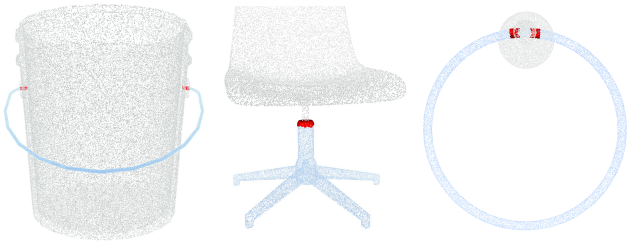

Figure 8. Illustration of the contact region (highlighed in red points) we detected between two kinematically dependent parts (colored in grey and blue) to infer the motion joints.

Motion type. In many cases, there is a single motion type for a particular semantic part. In the case of multiple possible motion types, as determined by the templates, we either use geometric heuristics to disambiguate the case if possible or simply go for the one we expect to be the most commonly present in the distribution.

Motion axis. For motion axis, we derive its origin and direction by either relying on the coarse geometric cues (e.g, part centroid, bounding boxes) or from the fine-grained contact region between two kinematically dependent parts, depending on the motion and part type.

For some articulated components (e.g., doors, drawers, buttons), the motion axis is naturally aligned with a characteristic dimension or face of the part bounding box. To robustly capture these cues, we compute three types of bounding boxes: 1) *AABB* – axis-aligned bounding box obtained from minimum and maximum points of the part geometries; 2) *POBB* – PCA-oriented bounding box whose axes follow principal directions; 3) *GOBB* – gravity-aligned bounding box where the vertical axis fixed to gravity direction and horizontal axes from PCA on the part projected to the ground plane. We select the bounding box with the smallest volume as the primary geometric descriptor, except when the GOBB volume is within 5% of the minimum. In that case, we prefer GOBB because it typically aligns better with human-centric object orientation conventions.

Certain joints require finer geometric reasoning, e.g., rotational joints located off-center (e.g., chair casters), or hinges defined by narrow attachment regions. For these cases, we identify the contact region between a part and its parent by sampling points from each part surface, computing pairwise distances between the two point sets, and take the nearest points as the contact region. Figure 8 shows some examples of the contact region we detected and used for motion inference across different categories. If multiple contact regions are detected (e.g., two hinge mounts), the motion axis is inferred from the geometric relationship

between these regions, for instance, by connecting region centroids. This strategy allows reliable axis placement even for small, highly localized joints.

Motion range estimation. Parts vary in pose and may be partially colliding due to modeling artifacts, making range inference non-trivial. We therefore simulate joint motion and monitor collision changes with some tolerance to identify the joint limit at two ends.

For rotational joints, we incrementally rotate the part in both directions. At each step, we record the number of collision points. A valid collision event is detected when the count shows a sustained ramp-up over multiple steps and exceeds the initial collision count by a threshold factor. We take the two collision bounds and derive a safe operational subrange (e.g., 80% of collision-to-collision span).

For translational joints, we first detect the detachment distance by translating the part until collisions disappear. Then we sweep in the opposite direction to find the first collision boundary. These detachment-collision distances are post-processed with category rules to match realistic physical limits.

Kinematic graph construction. After estimating motion parameters, we construct the kinematic graph describing parent-child relationships between articulated parts. For each part, we compute pairwise distances between sampled surface points and assign the closest valid parent according to the template constraints. This results in a complete and consistent kinematic hierarchy that reflects the articulation structure of the object.

B.4. Interior Completion

Many static 3D repositories provide only the outward appearance of objects, leaving internal structures either partially modeled or entirely missing. To ensure functional completeness and realistic physical simulation, we perform interior completion for categories where internal geometry is essential for the following three scenarios.

Partial articulated parts. Some articulated components are modeled only partially in static datasets. A common example is a drawer, where the front panel exists but the drawer box is missing—leading to unrealistic behavior once articulation is introduced. For such cases, we complete the missing geometry using a heuristic adapted from prior work S2O [19]: we cast rays behind the drawer's front panel to infer the enclosed volume and procedurally insert side panels, a bottom panel, and a back panel consistent with the enclosing cavity.

Missing articulated parts. Large appliances often rely on internal movable components that are not modeled in the original mesh but are essential for functional interactivity, e.g., baskets in dishwashers, turntables in microwaves, or containers in refrigerators. For categories requiring such internal articulated elements, we collect set of representative instances from the existing assets to define the expected interior parts. Next, when processing other objects sharing the same category, we detect whether such elements already exist from upstream segmentation; if not, we procedurally insert interior articulated components by randomly retrieving geometry templates from annotated exemplars or synthesizing simple parametric versions aligned with the internal cavity.

Missing interaction affordance parts. Many furniture and storage objects lack internal shelving or structural dividers, leading to hollow cavities that break realism and limit utility for more interactivity. For targeted categories (e.g., cabinets, wardrobes, refrigerators), we check segmentation results to determine whether interior components are present. If the interior is empty, we procedurally generate shelves, racks, or rails based on the object's internal bounding volume. These elements are aligned with the geometry of the enclosing body and follow category-specific placement patterns (e.g., horizontal shelves for wardrobes, horizontal racks for armoires, additional vertical dividers for cabinets).

This structured interior completion pipeline ensures that objects are not only visually consistent but also functionally complete, enabling realistic articulated behavior and physically plausible interaction in downstream simulations.

B.5. Manual verification details

Each data sample undergoes a total of 4 verification stages. Namely, segmentation correction, segmentation verification, motion correction, and final verification. We dedicate 2 steps to segmentation as we find it the most prone to human errors. Correction is performed in the 3D annotation tool directly, while verification is performed in specialized UI where the expert annotators are being presented with a video for each part, with the part of interest highlighted and other parts semi-transparent, with the camera revolving around that part. Motion correction is also performed in 3D motion annotation tool. In the final verification, annotators are presented with the videos of each single part moving, both highlighted and textured. These objects also include the completed interiors, and material proposals, which users verify in this stage only.

The recruited annotators are mostly grad and undergrad students recruited within the institution. Throughout all the annotation sessions, a total of 33 annotators have contributed. 18 of them, contributing more than 5 hours total. The overall annotation time across all the manual correction and verification steps is 400 hours.

B.6. Pipeline limitations

Most of the failure modes come from the geometry artifacts, such as: thin structures, duplicated surfaces, parts that are not modeled with separate connected components. Another limitation of the pipeline appears in the cases of complex

kinematic chains such as folding chairs. In these cases, we adopt simplified kinematic model which might ignore or remove some parts. Finally, our current representation does not allow modeling of deformable parts, which otherwise would contribute to the physical realism of the data.

C. Experiments

In this section we provide additional implementation details of the experiments and provide more results.

C.1. Implementation details

For FPNGroupMot, we first pre-process our data to merge even the interactable parts, such as handles, into the articulated parts. We extend the method with an extra axis prediction head that allows support for the universal joint. The metrics follow the same setup as in S2O [19], except we ignore semantic labels due to the mismatch between PM and Artiverse while aiming at cross-validation. We increase the batch size to 32 and the number of training epochs to 600, otherwise keeping the hyperparameters in accordance to the original paper. Training is conducted on 4 NVIDIA L40S GPUs for 20 hours on Artiverse and 13 hours on PM.

Due to the flaw of the original design of label representation, SINGAPO cannot be scaled well to the large number of parts. This fact significantly impacts the performance as semantic labels are used for retrieval. In order to setup a fair comparison, we enhance augment SINGAPO with generative models with strong image-to-3D prior at inference time. In particular, the image is first passed through TRELLIS [54] to obtain the initial 3D geometry. This geometry, combined with SINGAPO bounding boxes is then used to condition X-Part [55] for per-part generation. Furthermore, we extend SINGAPO with an extra axis and range attributes to support universal joints. We increase the batch size to 120 and perform training on 2 NVIDIA H200 GPUs for 18 hours.

C.2. Image to Articulated object generation

The qualitative comparison on image to articulated object task is presented in Figure 9. We find that Articulate Anything is heavily bottlenecked by retrieval, which is especially problematic on our diverse data. While SINGAPO combined with geometry generation excels both at structural understanding and geometry reconstruction. However, we find that SINGAPO is limited by occasional misalignment of TRELLIS geometry with SINGAPO bounding boxes that are used to condition XPart.

Figure 9. Qualitative results of Articulate Anything compared to SINGAPO + XPart on Artiverse. Articulate Anything is limited by retrieval quality while SINGAPO combined with XPart allows adapting to the input geometry, while predicting reliable object structures.

```
{
    "main_category": "fan",
    "gloss": "A household electrical appliance that produces airflow ...",
    "content": [
      {
        "name": "base",
        "gloss": "The bottom support structure of the fan that provides stability and
        ↪  houses electrical connections.",
        "kinematic": {
          "articulatable": false,
          "link_dependency": [],
          "joint_type": ["fixed"]
        }
      },
      {
        "name": "pole",
        "gloss": "The vertical support column connecting the base to the head ...",
        "kinematic": {
          "articulatable": true,
          "link_dependency": ["base"],
          "joint_type": ["cylindrical", "fixed", "revolute", "prismatic"]
        }
      },
      {
        "name": "head",
        "gloss": "The main assembly containing the motor and blades, mounted on the
        ↪  pole ...",
        "kinematic": {
          "articulatable": true,
          "link_dependency": ["pole"],
          "joint_type": ["revolute", "fixed", "universal"]
        }
      },
      {
        "name": "blade_set",
        "gloss": "A set of rotating aerofoil components attached to the motor shaft
        ↪  ...",
        "parts": [
          {
            "name": "blade",
            "gloss": "..."
          },
          {
            "name": "rotor",
            "gloss": "..."
          }
        ],
        "kinematic": {
          "articulatable": true,
          "link_dependency": ["head"],
          "joint_type": ["continuous"]
        }
      },
      ...
    ]
  }
```

Figure 10. An example of our designed template.

Figure 11. Preview of all Artiverse categories.