3DCoder: Turning Images into Structured, Executable 3D

Wenjing Bian and Andreas Geiger

University of Tübingen, Tübingen AI Center

Abstract. We present 3DCoder, a framework that converts a single image into a structured and executable 3D representation suitable for editing and physics-based simulation. Rather than predicting geometry alone, 3DCoder decomposes an object or scene into a part-level structural graph that models connectivity, articulation, materials, and semantic relations between parts. Each part is associated with a parent in a kinematic hierarchy, articulation parameters, material attributes, and typed inter-part relations. The underlying geometry is produced by either a code-based or vision-based reconstruction backend and grounded to the symbolic graph through a colour-keyed segmentation representation. To support this formulation, we additionally introduce a unified dataset that jointly annotates part geometry, kinematic hierarchies, articulation parameters, materials, and non-tree relations across articulated objects, procedural single-object assets, and multi-object scenes. Experiments show that 3DCoder outperforms prior approaches on both articulation prediction and multi-object structural reasoning, while demonstrating good quality and interpretability for downstream applications.

1 Introduction

When constructing a 3D model, traditional workflows typically start with high-level structures and then assign content and properties to individual components, gradually producing a detailed and editable representation. Modern 3D reconstruction methods can bypass many of these intermediate steps and directly generate high-quality 3D models from observations. However, the absence of an explicit intermediate structure makes subsequent editing and manipulation significantly more difficult. Moreover, attaching additional properties beyond geometry and appearance (such as articulation, material attributes, or physical constraints) remains challenging, even when dense supervision is available. This limitation restricts downstream applications such as articulation, physical simulation, and semantic editing.

Recent advances in large language models (LLMs) and vision-language models (VLMs) have introduced new opportunities for structured 3D understanding. One line of work leverages LLMs as agents to iteratively generate or manipulate 3D content using their broad semantic priors and reasoning capabilities. While flexible, these multi-stage pipelines are often computationally expensive,

time-consuming, and lack precise pixel- or part-level alignment with visual observations. Another line of research fine-tunes language models for specialised 3D tasks. For example, methods such as MeshCoder [6] predict executable Blender programs from point cloud inputs, while PhysX-Anything [3] generates tokenised voxel geometry and articulated object structures in URDF form. Although effective for specific applications, these approaches are often tightly coupled to predefined representations or downstream tasks, limiting their flexibility and generalisation across object categories and simulation settings.

In this work, we instead propose a bottom-up approach that builds structured understanding from detailed geometry toward higher-level semantics and object structure. At the first stage, we leverage different reconstruction methods to estimate dense part-level 3D geometry in different 3D representations, which are unified into a common representation for subsequent structural reasoning. We then leverage a fine-tuned VLM to estimate part-level materials, relations between objects and parts, including physical constraints such as connectivity and articulation, as well as semantic relations such as symmetry and repeated structures. The output structured 3D model can then be utilised in downstream applications such as 3D editing and simulation.

To facilitate learning of structured 3D representations, we create a new dataset based on Infinigen [20], containing single and multiple data with articulated and static objects. The dataset provides object- and part-level masks together with structural annotations, enabling supervised learning of semantic and physical relationships. Experiments demonstrate that the proposed dataset substantially improves performance on structure understanding.

We summarise our main contributions as follows:

- We introduce a novel structured 3D representation that explicitly decouples geometric and structural components, unifying physical and semantic constraints to model both inter-object and intra-object relationships.
- We propose a flexible pipeline that maps diverse 3D representations into this unified structured form, enabling compatibility across different input formats and supporting downstream applications such as 3D editing and simulation.
- We present a new dataset for learning structured 3D representations and conduct extensive experiments demonstrating the effectiveness of our approach in producing accurate, interpretable, and simulation-ready results.

2 Related Work

2.1 3D Reconstruction for Articulation and Simulation

Early works on articulated object understanding primarily focus on articulation identification from visual observations [8, 30]. More recent methods aim to recover articulated 3D objects from diverse input modalities, including RGB images or videos [3, 5, 12, 15, 17], RGB-D observations [24], graph structures [16], and 3D representations such as point clouds [10, 13, 14] or bounding boxes [19].

To support physically realistic simulation, several approaches additionally estimate physical properties together with geometry. PhysGen3D [4] leverages language models to infer object-level physical parameters, while other methods [2, 3, 11] learn physical attributes jointly with geometry through supervised training. In contrast, our method introduces a unified structured representation that jointly models articulation, material properties, and inter-object relations at the part level, while remaining decoupled from the underlying geometry representation. This design enables more flexible downstream applications, including multi-object reasoning, editing, and simulation.

2.2 LM-based 3D Generation and Reconstruction

The success of large language models (LLMs) and vision-language models (VLMs) has recently motivated a growing body of work on language-driven 3D generation and reconstruction. One line of research employs off-the-shelf LMs as agents that progressively or iteratively construct 3D assets [7,9,18,22,26,28,29,31,32]. These methods benefit from the semantic reasoning capabilities of language models for understanding object and scene relationships, but often require multiple interaction rounds and may struggle to preserve fine-grained visual details from the input observations. Another line of work focuses on adapting language models directly to 3D generation tasks through supervised fine-tuning. Examples include MeshCoder [6], which predicts Blender code for 3D reconstruction in a single forward pass, and CAD-MLLM [25], which generates CAD representations from multimodal inputs. However, these approaches are typically tied to specific geometric representations and downstream applications, limiting their ability to model more general structural relationships. Proc3D [21] introduces an engine-agnostic graph representation for editable 3D structures, though with limited geometric expressiveness. Our approach differs from prior work by introducing a structured 3D representation that explicitly decouples geometry from structural reasoning. This formulation enables flexible integration with different geometry backends while supporting richer modeling of articulation, materials, and inter-object relationships.

3 Method

Given an input image, our goal is to recover a structured 3D representation that describes the object(s) in the scene through geometry, appearance, kinematic structure, hierarchical connectivity, semantic relations, and material properties. Such a representation enables a wide range of downstream applications, including interactive editing, robotics, and physical simulation.

In the following, we first introduce the proposed structured representation in Section 3.1. We then describe the pipeline used to convert images into this representation in Section 3.2, followed by the dataset constructed for training and evaluation in Section 3.3.

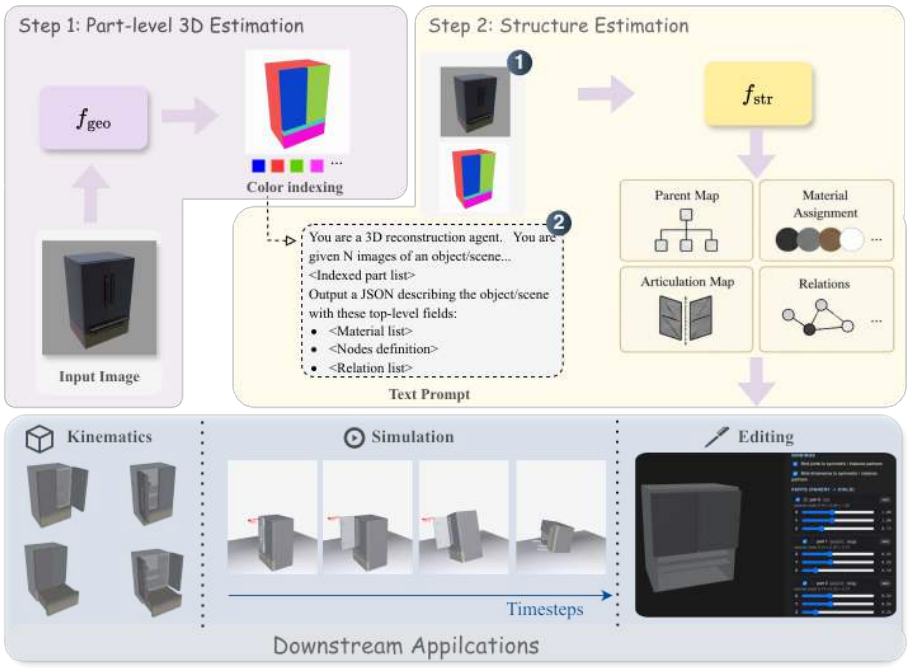

Fig. 1: 3DCoder Pipeline. Given an input image, we first recover part-level 3D geometry using a geometry reconstruction model f_{geo}, and render a coloured part segmentation map from the reconstructed geometry. The input image and segmentation map are then provided to a VLM-based structure inference model f_{str}, together with a text prompt. The model predicts hierarchical parent-child connectivity, articulation structure, material assignments, and semantic relations between parts, which can subsequently be used for various downstream applications.

3.1 Structured 3D Representation

We represent an object or scene as a typed graph that decomposes visual content into independently editable components, including part decomposition, hierarchical connectivity, articulation, material assignment, and additional non-tree relations. The representation is designed around two key principles. First, each component should remain independently addressable, enabling downstream operations such as re-posing a single joint, modifying a material, or duplicating a specific part without reconstructing the entire object. Second, the representation should admit a deterministic mapping to standard asset formats, including URDF/MJCF for physical simulation, Blender Python for content editing, and glTF for visualisation. This allows seamless integration into existing 3D pipelines without requiring an additional learned decoder.

Formal definition. Given an input image I and a corresponding part-segmentation map $S : \Omega \to \{1, \ldots, N\}$ with N regions, we represent the object as a tuple

$$\mathcal{O} \;=\; \big(T,\, \Phi_J,\, \Phi_M,\, \mathcal{R}\big), \tag{1}$$

where $\mathcal{P} = \{1, \ldots, N\}$ is the set of parts (one per segmentation region; node ids and pixel labels coincide, so part identities are visually grounded), and the four fields are:

- a parent map $T : \mathcal{P} \to \mathcal{P} \cup \{\perp\}$ whose acyclic restriction defines a forest of kinematic trees on the parts;
- a partial articulation map $\Phi_J : \mathcal{P} \rightharpoonup \mathcal{J}$ that assigns some children a joint

$$j \;=\; \big(\tau,\ \hat{a},\ x,\ [\theta_-, \theta_+]\big) \;\in\; \mathcal{J}, \tag{2}$$

  with type $\tau \in \{\textit{hinge}, \textit{slide}\}$, unit axis $\hat{a} \in S^2$, pivot $x \in \mathbb{R}^3$, and range $[\theta_-, \theta_+]$; absence of $\Phi_J(p)$ means that the attachment $p \to T(p)$ is rigid;
- a material assignment $\Phi_M : \mathcal{P} \to \mathcal{M}$ drawn from a shared dictionary $\mathcal{M}$ of material entries;
- a set of typed editing relations $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P} \times \mathcal{T}_R$, where $\mathcal{T}_R = \{\textit{instance}, \textit{symmetric}\}$.

A material entry $m \in \mathcal{M}$ is represented as a pair (*family*, *desc*), where *family* is selected from a fixed vocabulary of ten material categories: *Wood*, *Metal*, *Plastic*, *Glass*, *Fabric*, *Leather*, *Ceramic*, *Stone*, *Rubber*, and *Other*. The *desc* field is a free-form textual description (e.g., "brushed black metal") that preserves the open-vocabulary expressive capability of the backbone model while maintaining a compact and structured prediction space. Such material descriptions can further be mapped to approximate physical properties during inference.

Editing relations. The relation set $\mathcal{R}$ stores second-order facts that do not belong on the kinematic tree, one atomic edge per fact, so that any single relation can be added or deleted without touching the rest of $\mathcal{O}$. We propose two commonly used types: `symmetric`(a, b, π) with $\pi \in \{xy, yz, xz\}$ (mirror modifier) and `instance`(a, b) (linked duplicate).

Geometry. Fine-grained geometry is not stored directly within $\mathcal{O}$. Instead, each part $p \in \mathcal{P}$ is associated with an explicit triangular mesh g_p generated by a separate geometry reconstruction stage. The correspondence between symbolic part identifiers and geometric components is recovered through the visually grounded mapping between part ids and segmentation regions. We describe two alternative geometry generation pipelines in Section 3.2.

Multi-object scene extension The same structured representation naturally extends to multi-object scenes by representing each object as a sub-tree under a shared virtual scene root. Scene-level `instance` relations capture repeated or semantically equivalent objects, while parent pointers in the kinematic tree encode both intra-object part hierarchies and inter-object support relationships.

3.2 Two-Stage Prediction Pipeline

As shown in Figure 1, we factor the inference process of predicting $\mathcal{O}$ jointly with full per-part geometry from a single image into two stages,

$$I \;\xrightarrow{f_{\text{geo}}}\; \big(\{g_p\}_{p\in\mathcal{P}},\ S\big) \;\xrightarrow{f_{\text{str}}}\; \mathcal{O}, \tag{3}$$

where f_{geo} produces the part decomposition together with one triangle mesh g_p per part and a flat-colour segmentation map S rendered from the same viewpoint as I, and f_{str} predicts the symbolic graph $\mathcal{O}$ from (I, S). The structure model f_{str} is fine-tuned from Qwen3-VL-4B on the dataset of Section 3.3; the geometry stage admits two alternative implementations, described below, with different application senorios. The factoring mirrors the representation itself (Section 3.1), in which fine geometry is not stored in $\mathcal{O}$ but is recovered through the visually-grounded id–region correspondence.

Stage 1 – Part-level 3D Estimation. Stage 1 produces a triangle mesh g_p per part of the object visible in I, together with the segmentation map S that links each part back to its image region. We provide two interchangeable backends and choose between them depending on the downstream application.
(i) Code-based backend. We apply a VLM-based approach to predict Blender Python code for reconstructing the object shown on the input image. We adopt a similar code representation format as MeshCoder [6], which constructs each part of the object explicitly with pre-defined Blender functions. The resulting decomposition has clean, non-overlapping boundaries and exact part–mesh correspondence, which makes the segmentation map S obtainable directly by rendering each part with a flat colour. In practice, we finetune a Qwen3-VL-4B model [1] on a code dataset adapted from MeshCoder [6].
(ii) Vision-based backend. An off-the-shelf pipeline that combines Hunyuan3D-2 [23] for image-to-mesh generation with XPart [27] for 3D part segmentation. Hunyuan3D produces a single high-resolution mesh of the whole object; XPart partition this mesh into individual parts $\{g_p\}$, which simultaneously yields the per-part labelling needed to render S. This backend requires no fine-tuning and produces visibly richer surface detail, at the cost of less reliable part boundaries when neighbouring parts share materials.

The two backends therefore trade detail for topology: the vision-based backend generates the more visually-detailed geometry, while the code-based backend yields cleaner topology and segmentation, which are suitable for different downstream tasks. Once $\{g_p\}$ are obtained, the parts are placed in a common canonical frame and rendered into a flat-colour segmentation map S from the same viewpoint as I, with each region carrying a unique colour drawn from a per-object deterministic palette permutation.

Stage 2 – Structure Estimation. The structure prediction model f_{str} takes as input the RGB image I, the flat-colour segmentation map S produced in Stage 1, and a text prompt, and autoregressively predicts the structured representation $\mathcal{O}$ as a single JSON document. The prompt serves two purposes. First, it enumerates the segmented parts by associating each integer region identifier $p \in \mathcal{P}$ with its corresponding colour in S and the coarse bounding box (c_p, e_p) supplied by Stage 1 backend. Second, it defines the target schema by specifying the fields of $\mathcal{O}$, including `materials`, `nodes`, relations, where each node contains its parent assignment, material label, and optional articulation parameters. For articulated objects, the prompt additionally provides the number of joints to predict.

We fine-tune f_{str} from Qwen3-VL. Since the part inventory and colour correspondences are explicitly provided, the model focuses on reasoning about structural relationships between visible parts, including connectivity, articulation, symmetry, and repetition patterns. The correspondence between symbolic part identifiers and fine-grained geometry is recovered through the colour association with S, and consequently with the Stage 1 geometry predictions $\{g_p\}$.

Associating parts with the output format. Each maximal connected region with a unique colour in the segmentation map S defines a part and is assigned a stable integer identifier p. The prompt contains a colour legend that establishes a bijection between part identifiers and segmentation colours, and the rendered segmentation map is generated using the same palette by construction. The structured representation $\mathcal{O}$ uses the identical identifier set: the `nodes` dictionary is indexed by part ids, and all symbolic references in the representation, including parent assignments $T(p)$, relation endpoints, and material associations, are expressed using these identifiers. Under this formulation, predicting $\mathcal{O}$ becomes a structured reasoning problem over a fixed set of observed parts. The language model predicts connectivity, articulation, materials, and semantic relations, while the dense geometry $\{g_p\}$ is not generated directly, but instead recovered through the identifier-to-colour-to-geometry correspondence defined by S. This design keeps the symbolic representation compact while maintaining explicit grounding to the underlying geometry (Section 3.1).

However, relying only on RGB colours in a single-view segmentation map can lead to ambiguities, especially when different parts are assigned visually similar colours or when certain parts are occluded under the viewpoint. We therefore augment the prompt with the centroid $c_p \in \mathbb{R}^3$ and extent $e_p \in \mathbb{R}^3_+$ of each part to provide additional geometric cues. Both quantities are normalised into the unit object coordinate frame with $\max_{p,i}(e_p)_i = 1$. These values are computed directly from the segmentation and included in the prompt together with the colour legend:

$$\texttt{prompt}(I, S) \;\supseteq\; \big\{\, (p,\ \textit{colour}_p,\ c_p,\ e_p) \;:\; p \in \mathcal{P} \big\}. \tag{4}$$

3.3 Dataset Construction

Since no publicly available dataset jointly provides the part-level geometry, kinematic hierarchy, articulation parameters, material annotations, and non-tree relations required by the representation in Section 3.1, we construct a new dataset procedurally by combining multiple complementary data sources within a unified processing pipeline, converting all assets into the structured schema of $\mathcal{O}$.

Data sources. We combine several data sources during training as described below, which together provide a diverse set of articulated and rigid objects and multi-object scenes.

(i) Single-object 3DCoder Data. We generate 27 categories of furniture and household objects using the procedural factories from Infinigen [20]. Hierarchi-

cal part trees, articulated joints, and per-part PBR materials are extracted directly from the procedural scene graph. To improve supervision for repeated structures, we additionally include four instance-heavy factories (`cell_shelf`, `large_shelf`, `simple_bookcase`, and `triangle_shelf`), increasing the diversity of `instance` relations relative to `symmetric` ones.

(ii) Multi-object 3DCoder Data. Beyond single objects, we procedurally generate multi-object scenes based on Infinigen assets. A category-aware placement system samples support–supported object relationships and instantiates repeated objects across scene surfaces. Scenes are exported using both part-level and object-level segmentation granularities.

(iii) PhysX-Mobility and PhysXNet [3]. We incorporate the articulated objects from the PhysX-Mobility and PhysXNet datasets and convert their articulated hierarchies into the schema of $\mathcal{O}$. Non-tree relations are further computed from the provided part-level 3D annotations.

Rendering. Each object or scene contributes a canonical RGB image (I) rendered with Cycles using Infinigen's procedural shaders, together with a segmentation map (S) rendered from the same camera viewpoint using flat-colour shading at identical image resolution. In addition, we render multi-view observations from randomly sampled viewpoints for use during training. For articulated objects, each additional view is rendered after randomly selecting and perturbing one articulated joint, exposing the model to varying articulation states during training. For multi-object scenes, segmentation maps are generated using a mixture of object-level and part-level annotations, with corresponding structure definitions reflected in the ground-truth JSON. This allows the model to jointly learn object-level and part-level structure prediction using only the input image and segmentation map as supervision signals.

Permuted colour palette. Using a fixed colour assignment for segmentation maps would allow the model to associate colours directly with part indices, reducing the need for visual grounding. To avoid this shortcut, we deterministically permute the segmentation colour palette for each object using a hash seed computed from the object category and identifier, $h(\text{cat} + \text{obj})$. Both the rendered segmentation map and the corresponding prompt colour legend are generated using the same permuted palette, preserving consistency while preventing statistical correlation between semantic part roles and rendered colours across the dataset.

Coordinate normalisation. To account for the inherent scale ambiguity in monocular 3D reconstruction, all centroids, joint pivots, and articulation axes in $\mathcal{O}$ are represented within a normalised object coordinate frame. Specifically, objects are translated such that the object centre is located at the origin and uniformly scaled so that the maximum spatial extent equals one. The bounding-box centroid and extent pairs (c_p, e_p) used in the prompt representation (Eq. 4) are computed within the same normalised coordinate system.

Relation extraction. For each pair of sibling parts sharing the same parent node $T(p)$, we compute a rotation- and reflection-invariant geometric signature consisting of sorted bounding-box extents, coarse vertex-count statistics, and part volume. Part pairs with matching signatures are classified as `symmetric` when their centroids exhibit mirror symmetry with respect to one of the principal coordinate planes, and as `instance` relations otherwise. To remove ambiguity in relation representation, each `instance` group is converted into a star-anchored structure using the part with the smallest integer identifier as the anchor node.

Canonicalisation. We apply several deterministic post-processing steps to make the target representation $\mathcal{O}$ invariant to symbolic ambiguities. First, JSON dictionaries (e.g., `nodes` and `materials`) are serialized using ascending integer-key order, preventing the token-level objective from penalizing semantically equivalent dictionary permutations. Second, degenerate joints with zero motion range or invalid articulation axes, which are occasionally produced as placeholders in Infinigen, are removed. Third, each connected component of the hierarchy tree T is re-rooted at the part with the largest bounding-box volume, and all articulation directions are updated accordingly.

Decomposition augmentation. For each articulated training object, we additionally generate augmented samples using structure-preserving decomposition operations. Specifically, we introduce a *split* operation during preprocessing, where a randomly selected articulated leaf part is divided by a plane passing through its centroid. The resulting sub-parts are inserted as sibling nodes under the original parent while inheriting the original joint type, axis, position, and motion range. We further introduce a complementary *merge* operation, which selects two sibling leaf parts with compatible articulation properties, either both rigid or sharing similar articulation parameters within a predefined tolerance, and merges them into a single part with combined geometry. After augmentation, the segmentation map S is re-rendered to match the modified decomposition. As discussed in Section 4.3, these augmentations improve robustness to over- and under-segmentation and lead to improved performance at test time.

4 Experiments

Implementation details. We fine-tune Qwen3-VL-4B-Instruct [1] for three epochs on the structured objective of our proposed dataset with input context length 8192 tokens, effective batch size 16, and a cosine learning-rate schedule peaking at 10^{-5}. The vision encoder and MLP are kept frozen and only the LLM decoder is finetuned. All experiments use four NVIDIA A100 GPUs. Please refer to the supplementary for more details.

Metrics. We report **parse success** (the fraction of generations that yield a parsable structured JSON), and on the parsed subset: **parent accuracy** after star-anchor canonicalisation, a **edge** F_1 on the directed parent graph; **joint-presence** F_1 from a Hungarian matching between predicted and reference joints;

joint-type accuracy (revolute vs. prismatic on matched joints); **joint-axis accuracy** under the angular threshold of 0.25; **joint position error** in metres; **joint range IoU**; and **relation** F_1 on the symmetric- and instance-edge sets. Following [12], we report *articulation failure attribution*, where each failed joint is assigned to its first violated gate (type $\rightarrow$ axis $\rightarrow$ origin $\rightarrow$ limit). The **success** rate shows the fraction of matched joints that simultaneously pass the gates on type, axis, origin, and range. All metrics are computed after the canonicalisation described in Section 3.3, so a model is not penalised for predicting an equivalent star anchor or a different ordering of an unordered group.

Baselines. We compare against three baselines. **Qwen3-VL**(zero-shot) uses the same Qwen3-VL-4B-Instruct backbone without fine-tuning and receives the identical prompt, isolating the effect of supervised adaptation from the underlying backbone capability. **PhysX-Anything** [3] is a multi-stage VLM framework that predicts URDF structures from a single image. We separate its joint prediction stage from the downstream geometry generation components and denote this variant as **PA**. The predicted URDF representation is converted into our structured format using centroid-based part rebinding and coordinate frame correction. Additional implementation details are provided in the supplementary material. Finally, **Scene Language** [32] provides a program-synthesis baseline for compositional scene understanding, and is used as a baseline for the multi-object setting in Section 4.2.

4.1 Evaluation on Structure Estimation

We evaluate single-view conditioned structure estimation on the PhysX-Mobility [3] test split. As shown in Table 1, we compare against the zero-shot Qwen3-VL baseline [1] as well as PhysX-Anything [3] in terms of structure, joint and relations. Two variants of our model are reported. **Ours-PhysX** is trained on PhysX-Mobility training split [3] together with the PhysXNet articulated-object collection [3]. **Ours** additionally mixes in the 5,384 procedural rigid objects and 1,730 multi-object scenes of our proposed dataset, keeping every other hyperparameter fixed. Both variants supervise under the URDF-consistent joint convention native to PhysX-Mobility. More details can be found in the supplementary.

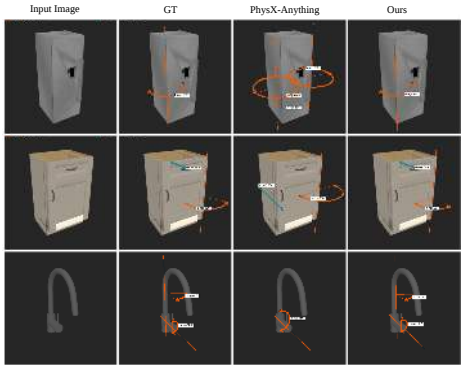

Fig. 2: Joint Estimation on PhysX-Mobility Dataset. We show the joint estimation results for our approach and PhysX-Anything against the ground truth. The joint axes are shown, along with an arrow indicating the joint range. Different colours represent different joint types.

Table 1: Structure estimation performance on the PhysX-Mobility dataset. Qwen3-VL denotes the pretrained backbone without fine-tuning. PA denotes PhysX-Anything. Ours-PhysX is fine-tuned only on PhysX-Mobility, while Ours is jointly fine-tuned on PhysX-Mobility and our dataset. Failure attribution assigns each failed joint to its first violated gate, so those five entries sum to 100 per row. All values are percentages, except joint position error (in metres).

	Structure			*Joint*				
	Parse↑	Parent↑	Edge F_1 ↑	Pres. F_1 ↑	Type↑	Axis↑	Pos.↓	Range IoU↑
Qwen3-VL	100.0	60.5	24.2	3.9	69.2	0.0	0.324	46.6
PA	100.0	51.6	17.5	73.4	92.0	88.7	0.210	23.8
Ours-PhysX	99.0	**86.2**	81.9	**91.5**	96.5	84.8	0.060	**71.7**
Ours	99.0	85.6	**85.4**	90.8	**96.3**	**89.5**	**0.052**	71.3
	Articulation failure attribution					*Relations*		
	Type↓	Axis↓	Origin↓	Limit↓	Success↑	Sym F_1 ↑	Inst F_1 ↑	Plane↑
Qwen3-VL	30.8	69.2	0.0	0.0	0.0	5.0	0.2	54.5
PA	8.0	6.1	10.3	0.7	74.9	2.3	1.0	83.3
Ours-PhysX	3.5	14.0	5.1	0.8	76.6	44.6	16.6	96.4
Ours	**3.7**	**8.1**	**5.5**	**0.3**	**82.4**	**54.2**	**19.4**	**98.4**

Discussion. The zero-shot backbone produces well-formed JSON and recovers a coarse part tree from the prompt-supplied bounding-box listing, but emits joints on only 3.9% of reference joints and defaults to a single canonical axis when it does, failing the axis gate on every matched joint. It suggests the limitation of the current VLM to recover articulation without supervision. PhysX-Anything attains a strong axis prediction, but is bottlenecked by the joint position and range estimation. Both proposed variants of our method outperform the baseline in terms of overall articulation success. Compared to **Ours-PhysX** which is finetuned on the PhysX-Mobility and PhysXNet datasets, adding our procedural rigid objects and multi-object scenes (**Ours**) yields a further $\sim$ 6 pp gain in articulation success and substantially improves instance-relation, confirming that exposing the model to rigid procedural objects and multi-object scenes sharpens relational reasoning without sacrificing articulation accuracy. The reason for both of our models encoutered failure on parse success is that we limit the max token of output to 8192 (same for training), while the dataset contains several samples of fine-grained keyboards which get cut off. Qualitative results on joint estimation are shown in Figure 2. Figure 4 shows an example of relation estimation, where the geometry is estimated by the code-based model.

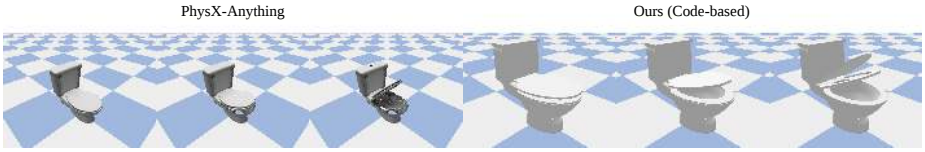

Fig. 3: Articulation Visualisation. Our code-based representation leverages clean part-level geometry, while the voxel-based representation used by PhysX-Anything leads to artefacts for overlapping surfaces and occluded regions.

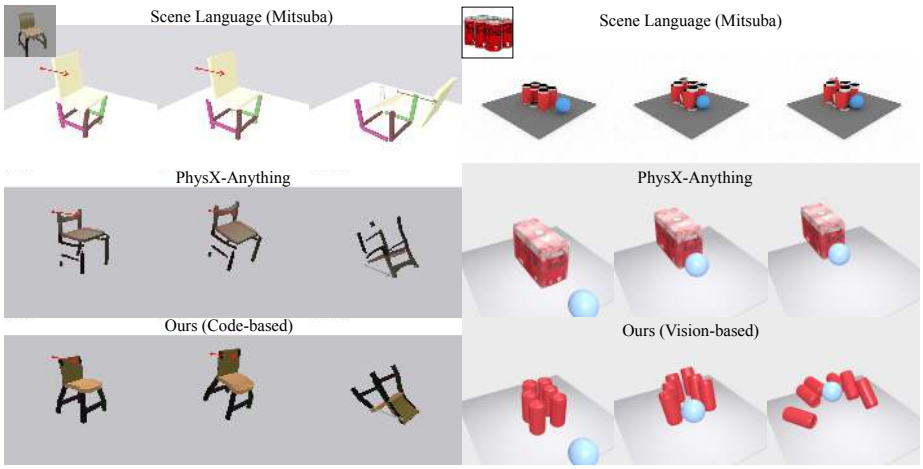

Fig. 5: Object Simulation. As Scene Language does not model part connectivity, the chair falls apart when external force is applied during simulation. In contrast, both our method and PhysX-Anything predict part-level connectivity, allowing the object to move as a coherent whole, although some geometry predicted by PhysX-Anything is not connected visually.

4.2 Multi-object Scenes

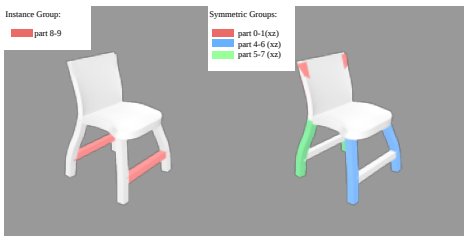

Fig. 4: Relation Estimation. Our method predicts part-level relations, including instance and symmetry correspondences, which can benefit downstream applications such as 3D editing and structured manipulation.

A scene contains multiple objects, each represented by one connected component of the predicted part-tree: the node whose parent chain terminates in `null` is the object root, and every descendant shares that root. Recognising each scene element as an individual object – rather than as parts inadvertently wired into another object – amounts to predicting the same partition of nodes into trees as the ground truth. We compute this on the $n = 192$ Multi-object 3DCoder data test split and report (i) per-scene partition statistics (object-count match, fully-correct partition), and (ii) per-node and per-pair partition agreement (Hungarian-matched cluster accuracy, pair-wise random index, and same-object precision/recall/F_1). Results are shown in Table 2.

We further illustrate the importance of accurate object recognition with two simulation examples in Figure 5, particularly the ability to distinguish independent objects while preserving connectivity within each object. As shown in the figures, Scene Language [32] can reconstruct the geometry of the chair and coke cans, but treats all can components as independent parts without modelling their connectivity, causing the structure to fall apart during physical simulation.

Table 2: Object-recognition accuracy on Multi-object 3DCoder Data. Objects per scene is a count (ground truth 4.44); all other values are percentages. *Same-object* precision, recall and F_1 are computed over node pairs.

	Overall		*Per-scene partition*		*Per-node / per-pair partition*				
	Parse ↑	Obj./ scene	Count match↑	Full part.↑	Cluster acc↑	Rand ↑	Prec. ↑	Rec. ↑	F_1 ↑
Qwen3-VL (zero-shot)	99.5	5.70	29.2	25.5	19.1	78.0	18.4	5.3	8.2
Ours (SFT)	**100.0**	**4.53**	**99.5**	**97.9**	**99.1**	**99.7**	**99.8**	**98.6**	**99.2**

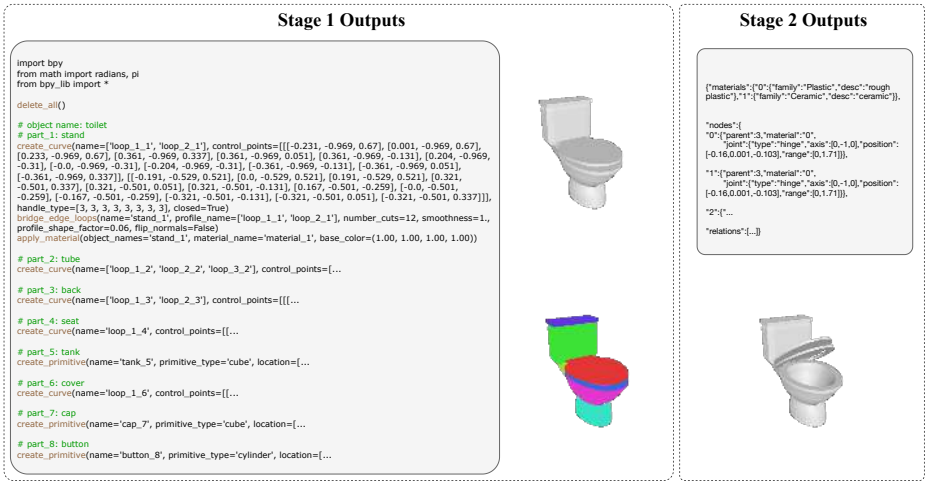

Fig. 6: Example of outputs from each prediction stage. We show the per-stage outputs for the toilet example in Figure 3. For stage 1, the model first predicts the Blender Python code from the input image. The code naturally contains part-level information, which allows us to render a segmentation map. For stage 2, the model estimates a JSON file that contains material, nodes (which contain parent and articulation information) and relations. The articulation prediction can then be added to the 3D model.

PhysX-Anything [3], despite being trained on examples containing detachable components such as battery covers or drip trays, fails to recognise the cans as separate objects. In contrast, our model can predict connectivity for the chair and distinguishes independent objects from connected parts, even when objects are placed in close proximity, producing more coherent and simulation-ready structures.

4.3 Part-level 3D Estimation Analysis

The motivation for supporting different part-level 3D estimation models in the first stage of our pipeline is to accommodate diverse application scenarios. For example, Figure 3 presents a case where we aim to articulate a toilet model. Although the voxel-based representation used in PhysX-Anything can reconstruct high-quality geometry for the closed toilet, applying articulation reveals severe

Table 3: Decomposition augmentation ablation on the PhysX-Mobility test split. **w/o aug** is the predictor trained on PhysX-Mobility without the augmentation; **w/ aug** adds the *split* and *merge* operations. Column groups match Table 1; all values are percentages, except joint position error (in metres).

	Structure		*Joint*					Artic.	*Relations*		
	Parent ↑	Edge F_1 ↑	Pres. F_1 ↑	Type ↑	Axis ↑	Pos. ↓	Range IoU↑	succ. ↑	Sym F_1 ↑	Inst F_1 ↑	Plane ↑
w/o aug	83.0	81.2	88.6	93.8	74.4	0.110	66.8	61.9	31.7	15.8	92.7
w/ aug (**Ours-PhysX**)	**86.2**	**81.9**	**91.5**	**96.5**	**84.8**	**0.060**	**71.7**	**76.6**	**44.6**	**16.6**	**96.4**

artifacts in the inner structure. This is primarily because overlapping regions are difficult to separate in voxel space, and the model lacks priors for reconstructing internal geometry that is occluded in the input image. In contrast, our code-based representation predicts each part independently and leverages semantic part names inferred by the VLM. Trained on a wide range of common object categories, the model learns reusable structural priors that facilitate clean part decomposition and articulation. As illustrated in the intermediate predictions in Figure 6, this representation naturally supports accurate part segmentation and manipulation.

Although the flexibility of the first-stage geometry estimation enables compatibility with different reconstruction methods, we observe that out-of-distribution segmentations remain a common challenge at test time. Since object parts can be decomposed in multiple valid ways, both over-segmentation and under-segmentation frequently occur. To improve robustness against segmentation biases in the training data, we simulate such variations through dedicated data augmentations including *split* and *merge* as discussed in Section 3.3. These augmentations expose the model to diverse segmentation granularities and improve robustness to both over- and under-segmentation at inference time. Results in Table 3 demonstrate the effectiveness of this strategy.

5 Conclusion

We presented 3DCoder, a framework that reconstructs structured, editable 3D objects and scenes from one or more images by decoupling dense part-level geometry from the inference of a hierarchical structure that unifies articulation, part–object connectivity, semantic relations, and materials in a single frame. Trained on our new Infinigen-based dataset of articulated and rigid, single- and multi-object instances, the approach demonstrates good accuracy in terms of articulation and structure relation estimation, and shows various downstream applications from 3D editing to simulation.

References

1. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S.,

Li, Z., Li, M., Li, M., Li, K., Lin, Z., Lin, J., Liu, X., Liu, J., Liu, C., Liu, Y., Liu, D., Liu, S., Lu, D., Luo, R., Lv, C., Men, R., Meng, L., Ren, X., Ren, X., Song, S., Sun, Y., Tang, J., Tu, J., Wan, J., Wang, P., Wang, P., Wang, Q., Wang, Y., Xie, T., Xu, Y., Xu, H., Xu, J., Yang, Z., Yang, M., Yang, J., Yang, A., Yu, B., Zhang, F., Zhang, H., Zhang, X., Zheng, B., Zhong, H., Zhou, J., Zhou, F., Zhou, J., Zhu, Y., Zhu, K.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)

2. Cao, Z., Chen, Z., Pan, L., Liu, Z.: Physx-3d: Physical-grounded 3d asset generation. arXiv preprint arXiv:2507.12465 (2025)
3. Cao, Z., Hong, F., Chen, Z., Pan, L., Liu, Z.: Physx-anything: Simulation-ready physical 3d assets from single image. arXiv preprint arXiv:2511.13648 (2025)
4. Chen, B., Jiang, H., Liu, S., Gupta, S., Li, Y., Zhao, H., Wang, S.: Physgen3d: Crafting a miniature interactive world from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6178–6189 (2025)
5. Chen, Z., Walsman, A., Memmel, M., Mo, K., Fang, A., Vemuri, K., Wu, A., Fox, D., Gupta, A.: Urdformer: A pipeline for constructing articulated simulation environments from real-world images. arXiv preprint arXiv:2405.11656 (2024)
6. Dai, B., Li, L., Tang, Q., Wang, J., Lian, X., Xu, H., Qin, M., XU, X., Dai, B., Wang, H., et al.: Meshcoder: Llm-powered structured mesh code generation from point clouds. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems
7. Her, L.C., Yan, M., Bai, Y., Li, R., Zhang, H.: Zero-shot 3d map generation with llm agents: A dual-agent architecture for procedural content generation. arXiv preprint arXiv:2512.10501 (2025)
8. Hu, R., Li, W., Van Kaick, O., Shamir, A., Zhang, H., Huang, H.: Learning to predict part mobility from a single static snapshot. ACM Transactions On Graphics (TOG) **36**(6), 1–13 (2017)
9. Huang, I., Yang, G., Guibas, L.: Blenderalchemy: Editing 3d graphics with vision-language models. In: European Conference on Computer Vision. pp. 297–314. Springer (2024)
10. Jiang, Z., Hsu, C.C., Zhu, Y.: Ditto: Building digital twins of articulated objects from interaction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5616–5626 (2022)
11. Le, L., Lucas, R., Wang, C., Chen, C., Jayaraman, D., Eaton, E., Liu, L.: Pixie: Fast and generalizable supervised learning of 3d physics from pixels. arXiv preprint arXiv:2508.17437 (2025)
12. Le, L., Xie, J., Liang, W., Wang, H.J., Yang, Y., Ma, Y.J., Vedder, K., Krishna, A., Jayaraman, D., Eaton, E.: Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. arXiv preprint arXiv:2410.13882 (2024)
13. Li, R., Yao, Y., Zheng, C., Rupprecht, C., Lasenby, J., Wu, S., Vedaldi, A.: Particulate: Feed-forward 3d object articulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 27708–27718 (2026)
14. Li, Z., Bai, X., Zhang, J., Wu, Z., Xu, C., Li, Y., Hou, C., Zhang, S.: Urdf-anything: Constructing articulated objects with 3d multimodal language model. arXiv preprint arXiv:2511.00940 (2025)
15. Liu, J., Iliash, D., Chang, A., Savva, M., Mahdavi Amiri, A.: Singapo: Single image controlled generation of articulated parts in objects. In: International Conference on Learning Representations. vol. 2025, pp. 97511–97532 (2025)

16. Liu, J., Tam, H.I.I., Mahdavi-Amiri, A., Savva, M.: Cage: Controllable articulation generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17880–17889 (2024)
17. Lu, R., Liu, Y., Tang, J., Ni, J., Wang, Y., Wan, D., Zeng, G., Chen, Y., Huang, S.: Dreamart: Generating interactable articulated objects from a single image. arXiv preprint arXiv:2507.05763 (2025)
18. Lu, S., Chen, G., Dinh, N.A., Lang, I., Holtzman, A., Hanocka, R.: Ll3m: Large language 3d modelers. arXiv preprint arXiv:2508.08228 (2025)
19. Mandi, Z., Weng, Y., Bauer, D., Song, S.: Real2code: Reconstruct articulated objects via code generation. arXiv preprint arXiv:2406.08474 (2024)
20. Raistrick, A., Lipson, L., Ma, Z., Mei, L., Wang, M., Zuo, Y., Kayan, K., Wen, H., Han, B., Wang, Y., Newell, A., Law, H., Goyal, A., Yang, K., Deng, J.: Infinite photorealistic worlds using procedural generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12630–12641 (2023)
21. Raji, F., Petrangeli, S., Gadelha, M., Shen, Y., Bhattacharya, U., Wu, G.: Proc3d: Procedural 3d generation and parametric editing of 3d shapes with large language models. arXiv preprint arXiv:2601.12234 (2026)
22. Sun, C., Han, J., Deng, W., Wang, X., Qin, Z., Gould, S.: 3d-gpt: Procedural 3d modeling with large language models. In: 2025 International Conference on 3D Vision (3DV). pp. 1253–1263. IEEE (2025)
23. Team, T.H.: Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation (2025)
24. Weng, Y., Wen, B., Tremblay, J., Blukis, V., Fox, D., Guibas, L., Birchfield, S.: Neural implicit representation for building digital twins of unknown articulated objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3141–3150 (2024)
25. Xu, J., Wang, C., Zhao, Z., Liu, W., Ma, Y., Gao, S.: Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. arXiv preprint arXiv:2411.04954 (2024)
26. Yamada, Y., Chandu, K., Lin, B.Y., Hessel, J., Yildirim, I., Choi, Y.: L3go: Language agents with chain-of-3d-thoughts for generating unconventional objects. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations). pp. 456–469 (2025)
27. Yan, X., Xu, J., Li, Y., Ma, C., Yang, Y., Wang, C., Zhao, Z., Lai, Z., Zhao, Y., Chen, Z., et al.: X-part: high fidelity and structure coherent shape decomposition. arXiv preprint arXiv:2509.08643 (2025)
28. Yin, S., Ge, J., Wang, Z.Z., Wang, C., Li, X., Black, M.J., Darrell, T., Kanazawa, A., Feng, H.: Vision-as-inverse-graphics agent via interleaved multimodal reasoning. arXiv preprint arXiv:2601.11109 (2026)
29. Yuan, Z., Lan, H., Zou, Q., Zhao, J.: 3d-premise: Can large language models generate 3d shapes with sharp features and parametric control? arXiv preprint arXiv:2401.06437 (2024)
30. Zeng, V., Lee, T.E., Liang, J., Kroemer, O.: Visual identification of articulated object parts. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2443–2450. Ieee (2021)
31. Zhang, S., Jiang, C., Li, Z., Deng, J.: Shapecraft: Llm agents for structured, textured and interactive 3d modeling. arXiv preprint arXiv:2510.17603 (2025)
32. Zhang, Y., Li, Z., Zhou, M., Wu, S., Wu, J.: The scene language: Representing scenes with programs, words, and embeddings. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 24625–24634 (2025)