LEGO-SAM: SEGMENT ANY LEGO® MODEL THROUGH CHAIN OF DISASSEMBLY

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper introduces LEGO-SAM, the first 3D segmentation framework for partitioning LEGO® models into semantic parts. To start, we formulate LEGO® segmentation at atom level with our generalized LEGO-Atomizer to bypass ambiguities in brick-level semantics. We then design a training-free Chain-of-Disassembly (CoD) method to segment the atomized LEGO® assemblies. Our method jointly leverages visual evidence from a 2D MLLM and structural priors from foundational 3D segmentation models, and takes out well-segmented parts iteratively based on multi-view consistency to reduce occlusion and simplify part relations. Also, we take CoD as a data engine to generate LEGO® annotations (1.86M atoms and 67K semantic parts) and take them as supervision to train LEGO-SAM-Flash, a fast native end-to-end LEGO® segmentor, with residual adapters and an atom-level segmentation head. Further, we build LEGO-SAM-Bench, the first LEGO® segmentation benchmark for quantitative evaluation, providing 347K atoms and 6.4K parts for a wide variety of LEGO® models, *e.g.*, indoor rooms, animals, buildings, *etc.* Experiments show that LEGO-SAM achieves leading performance, outperforming the strongest baseline by 31.18%, and also facilitates various assembly-aware downstream applications such as semantic-aware instruction generation, text-guided grounding, and object/part retrieval. Our code and dataset will be publicly available.

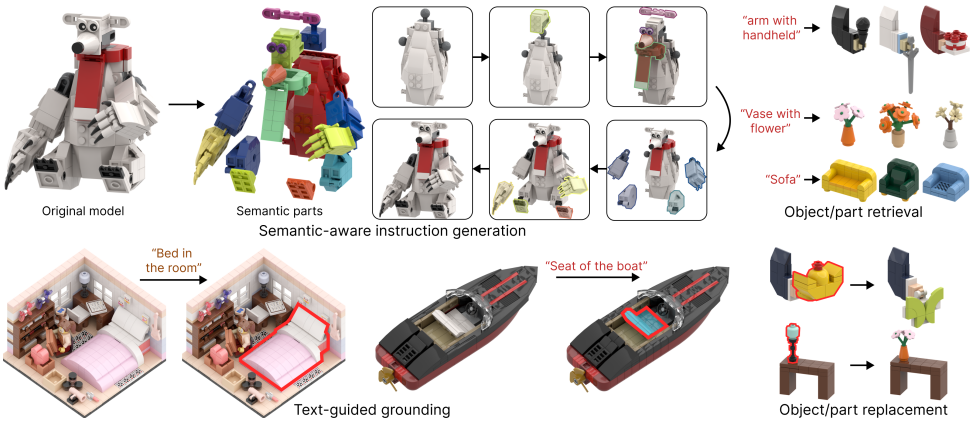

Figure 1: Downstream applications enabled by the semantic information introduced by LEGO-SAM.

1 INTRODUCTION

The LEGO® system (The LEGO® Group, n.d.) provides versatile building blocks for reproducing many different things in the world. As LEGO® models are expressive and physically realizable, they have attracted immense research interest. While early works center on automating LEGO® construction (Gower et al., 1998; Testuz et al., 2013; Luo et al., 2015), learning-based methods are introduced recently for various LEGO®-related tasks, *e.g.*, instruction grounding (Wang et al., 2022), brick classification (Vidal et al., 2023), assembly sequence prediction (Yan & Mu, 2025), manual-guided robotic manipulation (Gu et al., 2026), diffusion-based generation (Ge et al., 2024b), and autoregressive generation (Pun et al., 2025; Xu et al., 2025a; Kulits & Schmid, 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Despite the great progress, current methods mainly study spatial relations and brick connectivity, overlooking the *semantics in LEGO® assemblies*. As LEGO® models are often created to reproduce the real world, local groups of bricks are usually associated with meaningful parts in a LEGO® model. Moreover, compared with existing LEGO® design approaches, which are largely based on a brick-by-brick construction, approaches with semantic understanding of LEGO® models can open up many new downstream applications. As Figure 1 shows, *e.g.*, we can automate the generation of instruction manuals with part semantics, *i.e.*, after building each semantic part, we can then compose the parts into the overall assembly. We can retrieve objects/parts in existing LEGO® designs and ground natural-language descriptions to locate specific objects in a scene. Also, we can edit LEGO® models at object/part level rather than tediously editing brick by brick.

LEGO® models are fundamentally different from conventional 3D shapes, posing challenges when transferring existing 3D segmentation methods to handle them. Multi-view lifting methods (Deng et al., 2026; Tang et al., 2025) infer labels mainly from visible surfaces and miss occluded interior structures, while feedforward 3D segmentors (Ma et al., 2026; Liu et al., 2025) operate on points or mesh faces, thus fragmenting the brick boundaries. Also, there are no dedicated data for LEGO® segmentation. These limitations motivate us to derive a LEGO®-aware approach that jointly exploits semantic cues, 3D structural priors, and the assemblability of LEGO® models.

We therefore propose LEGO-SAM, the first LEGO® segmentation framework. First, we formulate LEGO® segmentation over atoms, the smallest non-decomposable LEGO® units, achieving assembly-aware granularity while resolving brick-level semantic ambiguity without over fragmentation. LEGO-SAM is built on our Chain of Disassembly (CoD), which integrates 2D semantic evidence from a multimodal large language model (MLLM) with structural priors from a pretrained 3D segmentor. By iteratively removing well-segmented parts based on multi-view consistency, we can expose occluded structures and simplify the remaining part relations. Further, we develop LEGO-SAM-Flash, a fast native end-to-end LEGO® segmentor with lightweight residual adapters and an atom-level segmentation head. To support training, we build the first LEGO® segmentation dataset over 4,000 LEGO® models using CoD as an annotation engine. To support evaluation, we build LEGO-SAM-Bench, a manually annotated benchmark of 300 models that span diverse categories. Experimental results show that both LEGO-SAM and LEGO-SAM-Flash substantially outperform existing 3D segmentation baselines and support downstream applications such as semantic-aware instruction generation, text-guided grounding, object/part retrieval, and part replacement.

Main contributions are summarized below. (i) We propose to segment LEGO® assemblies at atom level with a generalized LEGO-Atomizer to handle diverse brick types, enabling fine-grained LEGO® segmentation. (ii) We propose LEGO-SAM with CoD, the first framework for atom-level LEGO® segmentation, integrating semantic reasoning, 3D structural priors, and iterative disassembly. (iii) We distill the semantic understanding of LEGO-SAM into LEGO-SAM-Flash, an efficient end-to-end LEGO® segmentor with residual adapters and an atom-level segmentation head. (iv) We construct the first LEGO® segmentation dataset and benchmark, LEGO-SAM-Bench, and demonstrate multiple semantics-aware downstream applications. All the code and data will be made publicly available.

2 Related Work

Traditional 3D Part Segmentation. Traditionally, 3D part segmentation is cast as a classification task that operates at the primitive level of various 3D representations. Methods for point clouds (Qi et al., 2017a;b; Wang et al., 2019; Zhao et al., 2021; Wu et al., 2024; Graham et al., 2018) operate on points typically sampled or scanned from object surfaces. Methods for 3D meshes (Hanocka et al., 2019; Sharp et al., 2022; Liang et al., 2022; Zhou et al., 2023b) exploit the mesh topologies to define learning operators on mesh elements, *i.e.*, edges, vertices, faces, and local patches, whereas methods for CAD models (Lambourne et al., 2021; Jayaraman et al., 2021) learn boundary representations through message passing among adjacent topological entities.

Zero-shot 3D Part Segmentation. Early 3D segmentation methods are limited to objects of closed-set categories and require costly manual annotations. One approach to address these limitations is to leverage 2D foundation models (Radford et al., 2021; Li et al., 2022; Kirillov et al., 2023; Ravi et al., 2025; Carion et al., 2026; Oquab et al., 2024; Liu et al., 2024; Gemini Team, 2025), *e.g.*, rendering 3D objects into multiple views and lifting 2D predictions to 3D. Others aggregate SAM masks across views for class-agnostic segmentation (Tang et al., 2025; Yang et al., 2023; Xu et al., 2025b; Yang

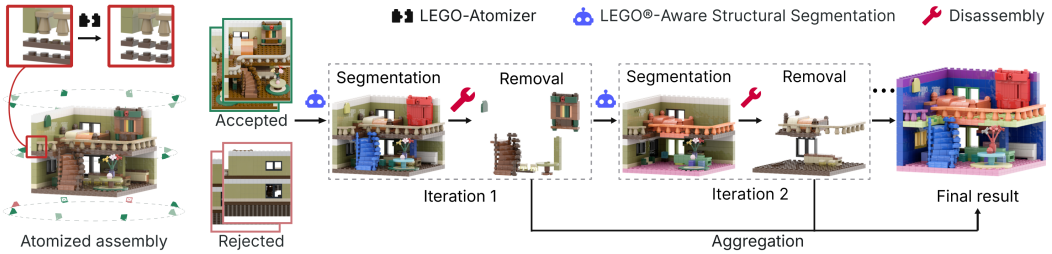

Figure 2: Overview of LEGO-SAM. Given an input LEGO® assembly, our generalized LEGO-Atomizer first decomposes bricks into atomized forms. Then, we query the MLLM to obtain reliable camera poses and segment the atomized assembly through a proposed Chain of Disassembly. At each iteration, we generate 2D masks on the rendered views, perform LEGO®-aware structural segmentation, and disassemble well-segmented parts based on multi-view consistency. Ultimately, we achieve the final LEGO® segmentation by aggregating the segmentation results across all iterations.

et al., 2024; Deng et al., 2026), or incorporate text prompts for open-vocabulary segmentation (Liu et al., 2023; Abdelreheem et al., 2023; Zhou et al., 2025b; Xue et al., 2025; Zhong et al., 2024; Umam et al., 2024; Garosi et al., 2025). Recent feedforward methods learn part-aware 3D representations from native 3D segmentation data and pseudo-labels (Liu et al., 2025; Ma et al., 2026; 2025; Zhou et al., 2025a; Zhu et al., 2026; Hadgi et al., 2026; Su et al., 2026; Jin et al., 2026; Zhou et al., 2026). Despite the great progress, LEGO® models are fundamentally very different from conventional 3D data. They are 3D assemblies built by discrete interconnected bricks. The segmentation should account not only for the exterior surfaces but also the interior bricks that are possibly occluded when viewed from the outside. Hence, we innovate a chain-of-disassembly framework to jointly leverage 2D visual evidence from an MLLM and 3D structural prior from an open-world 3D segmentor.

LEGO® Systems. Research on LEGO® has a long history. Early works focus on creating LEGO® sculptures using basic bricks (Gower et al., 1998; Testuz et al., 2013; Petrovič, 2001; Peysakhov et al., 2000; Lee et al., 2015; Luo et al., 2015). Subsequent efforts extend to broader domains, *e.g.*, houses (Zhou et al., 2019), Technic (Xu et al., 2019), sketch arts (Zhou et al., 2023a), and figurines (Ge et al., 2024a). Some recent learning-based methods classify bricks (Boiński et al., 2022; Vidal et al., 2023), ground instructions (Wang et al., 2022), and predict assembly sequences (Yan & Mu, 2025; Guo et al., 2024; Chung et al., 2021; Walsman et al., 2022; 2024). Very recently, ManualVLA (Gu et al., 2026) unifies multimodal manual generation with robotic actions. Recent generative models further enable automatic LEGO® creation using diffusion-based (Ge et al., 2024b) and autoregressive approaches (Pun et al., 2025; Ge et al., 2025; Xu et al., 2025a; Kulits & Schmid, 2026).

Overall, existing works focus mainly on the LEGO® model geometry and the brick assemblies, largely overlooking the semantics. In essence, LEGO® models are often created to mimic real objects, so local component parts in a LEGO® model usually represent certain objects or parts. This work introduces the first segmentation framework for LEGO® models with a LEGO® segmentation dataset, showcasing various downstream applications enabled by our new framework.

3 Atom-Level LEGO® Segmentation Formulation

Conventional 3D segmentation decomposes a given object into parts or a given scene into objects by considering structures and semantics. Often, the input is a 3D point cloud or mesh and we are tasked to assign a semantic label to each primitive, *i.e.*, a point or mesh face. To extend this formulation to LEGO® segmentation, straightforward approaches include sampling the LEGO® model surface and segmenting it at point level, or directly assigning a label to each assembly primitive, *i.e.*, a brick.

However, both approaches have limitations. While point-level segmentation allows leveraging existing 3D segmentation methods, its primitives are determined by sampling, thus overly fragmenting the LEGO® bricks. So, the segmentation results cannot support downstream applications at the brick level. On the other hand, brick-level segmentation is inherently ambiguous, since bricks may span adjacent semantic parts. See, *e.g.*, the 2×4 plate in Figure 3(a); it spans across the wall (blue) and bottle rack (red). This ambiguity affects downstream applications such as retrieving or removing an object from a scene, *e.g.*, removing the bottle rack in Figure 3(a) will leave a hole in the wall.

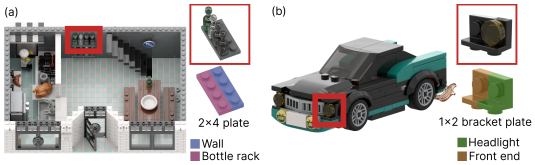

Figure 3: Brick-level segmentation is ambiguous for bricks crossing different parts. (a) The 2×4 plate spans the wall and bottle rack. (b) The 1×2 bracket plate spans the headlight and front end.

Therefore, we propose to formulate 3D LEGO® segmentation at atom level to resolve the brick-level ambiguity while retaining the connectivity and assemblability of parts. An important observation is that larger LEGO® bricks can be divided into smaller ones, *e.g.*, the 2×4 plate in Figure 3(a) can be divided into two 2×2 plates or eight 1×1 plates. Thus, we propose to decompose bricks into the smallest non-decomposable LEGO® bricks, termed *atoms*; see Figure 4. By this means, we can preserve the geometry and connectivity of the original bricks and assign a part label to each atom to delineate each semantic part at finer granularity while respecting the brick boundaries. Further, labeled atoms can be merged back into larger bricks for retrieval, editing, *etc.*

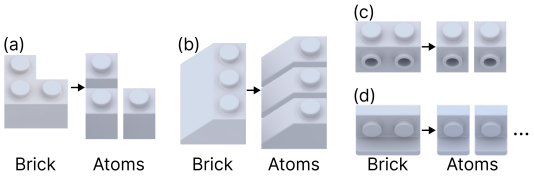

Figure 4: Atoms produced by our generalized LEGO-Atomizer on various brick types, *e.g.*, rectangular (a), slope (b), modified (c), and bracket (d); the atomization preserves the bricks' geometry and also the external connections (studs and tubes).

Generalized LEGO-Atomizer. We cast brick atomization as an integer linear program (ILP) that maps a target (decomposable) brick into a set of smaller bricks. Figure 4 shows examples across several brick families. We describe the target brick by its occupied volume V and locate all its exposed connecting regions *i.e.*, studs and tubes [1]. Then, we enumerate all possible valid placements of smaller bricks inside V and use a binary variable x_i to indicate whether the i-th placement is selected. The ILP aims to find a maximal set of placements that fully covers volume V without overlaps, while covering all the studs and tubes in the target brick. If no feasible decomposition is found, the target brick is retained as a single atom. To the best of our knowledge, this is the first generalized LEGO-Atomizer capable of decomposing diverse brick types in a unified formulation. Also, we precompute atomization only once on each brick in the brick database.

Atom-level Formulation for LEGO® Segmentation. Given a LEGO® model, our atomizer uses a table lookup to decompose each of its bricks into atoms, say $\mathcal{A} = \{a_i\}_{i=1}^{n}$. Then, atom-level segmentation aims to assign a semantic part label p_i to each atom a_i, so partitioning $\mathcal{A}$ into m semantic parts, $\bar{\mathcal{P}} = \{P_j\}_{j=1}^{m}$. The atomized representation serves for semantic label assignment.

4 LEGO-SAM: Segmentation through Chain-of-Disassembly

Existing approaches address open-world 3D segmentation either by lifting 2D masks to 3D or by directly applying a pretrained 3D segmentor. Despite their great recent progress, it remains challenging to handle LEGO® models due to the discrete and occluded assembly structures. Neither approach transfers cleanly to LEGO® models. Lifting-based methods rely on 2D segmentation of visible surfaces and often miss occluded interior structures, whereas direct 3D segmentation tends to produce fragmented or over-smoothed predictions; see Figure 5.

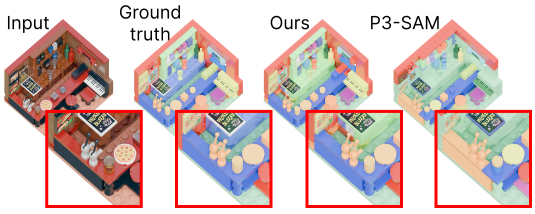

Figure 5: Directly applying the state-of-the-art model P3-SAM (Ma et al., 2026) tends to produce fragmented or over-smoothed results, while LEGO-SAM can produce more coherent semantic parts.

We address these limitations with a training-free Chain-of-Disassembly (CoD) method, LEGO-SAM, built around three components: (i) joint semantic–structural segmentation that combines semantic reasoning of an MLLM with structural prior of an open-world 3D segmentor (Sec. 4.1); (ii) progressive removal of reliably segmented parts, exposing occluded structures (Sec. 4.2); and (iii) an agentic feedback loop to assess intermediate results and adapt the input views (Sec. 4.3).

[1] Studs are cylindrical protrusions on the surface of a brick, whereas tubes are hollow cylindrical structures within a brick that help grip the studs of another brick (Christiansen, 1961).

4.1 LEGO®-AWARE STRUCTURAL SEGMENTATION

We integrate the two existing 3D segmentation paradigms as two complementary priors: (i) the *semantic reasoning* of an MLLM, which helps to understand LEGO® models, filter unrecognizable views, reveal parts in models, and produce view-dependent 2D masks; and (ii) the *structural prior* of a pretrained 3D segmentation model, which provides spatial coherence to complete the interior structures. We integrate the two priors in three steps: (i) filter unreliable views and generate view-wise 2D masks via an MLLM, (ii) lift and merge 2D masks into global 3D proposals, and (iii) prompt a 3D segmentation model with 3D proposals to label invisible atoms.

MLLM-Guided View Selection and 2D Segmentation. Since LEGO® models are abstract, their renderings from certain views may not be easily recognizable; see the rejected views in Figure 2 (left). Using an ambiguous view may introduce noise into the segmentation pipeline. Hence, we render the input LEGO® model from 24 viewpoints and query the MLLM to predict an object category for each view; see Appendix for viewpoint sampling and query prompts. Majority voting over the recognizable predictions determines the dominant category and we reject the views whose prediction(s) do not agree. To obtain a fixed set of twelve views for segmentation, we use farthest-point sampling if there are more than twelve views; otherwise, we interpolate camera poses between the accepted views to ensure adequate visual evidence for the subsequent cross-view merging. By then, we can obtain a set of resulting views that are semantically consistent and spatially diverse, and employ the same MLLM to generate a semantic mask for each of the views.

Lifting and Merging into 3D Proposals. Each view of the LEGO® model is associated with a 2D segmentation map of the segmented objects and parts, each with a unique mask ID. For each view, we assign each visible atom a mask ID through majority voting among its projected pixels. We then construct a graph over the lifted masks across views, connecting two masks if they are from different views and their 3D bounding-box IoU exceeds τ. Leiden community detection is then employed to group the matched masks across views into communities, and the union of the masks in each community forms a global 3D proposal P.

Proposal-prompted 3D Structural Segmentation. A lifted proposal contains only visible atoms, so we use the merged proposals P to prompt a pretrained 3D segmentation model, P3-SAM (Ma et al., 2026), to label invisible atoms. Given an atomized assembly $\mathcal{A}$, we sample a point cloud X from all atom surfaces. For each proposal P, its sampled points serve as prompts, and P3-SAM predicts a confidence score for assigning each point $x \in X$ to P. Each point x is assigned to the highest-scoring proposal among those with confidence above δ; if no proposal satisfies the threshold, x remains unassigned. Finally, atom labels are recovered by majority voting over the sampled points per atom. Setting δ=0 ensures that every atom receives a label.

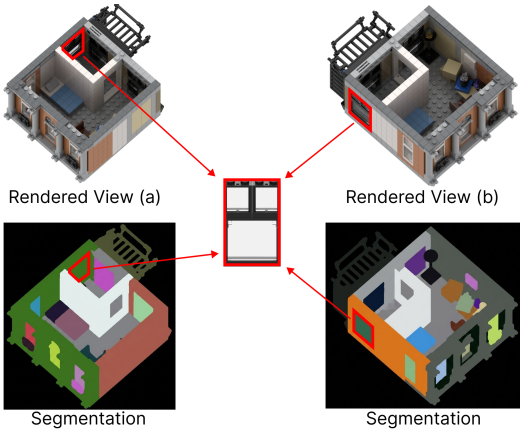

Figure 6: The multi-view consistency mechanism. We measure the reliability of a proposal by checking the consistency of segmentation maps produced by the MLLM, filtering out unreliable ones before the disassembly, *e.g.*, the same window marked in red is assigned to different semantic parts. In view (a), it is merged with the wall, but in view (b), it is segmented as an individual part. This inconsistency indicates that the segmentation is unreliable, *i.e.*, not ready for disassembly.

4.2 MULTI-VIEW CONSISTENCY-GUIDED DISASSEMBLY

For assemblies with heavily occluded or entirely invisible parts, a single pass may not segment the assembly reliably. We exploit the assemblability of LEGO® models by iteratively detaching only reliable proposals and passing the residual assembly to the next iteration. Each detachment exposes previously hidden structure and reduces the complexity of the remaining segmentation problem, while retaining uncertain proposals prevents early errors from propagating down the chain.

As Figure 6 illustrates, we estimate proposal reliability by examining the agreement across views. For an atom a, let $\mathcal{V}(a)$ denote the set of views in which a is visible. Each visible view $v_k \in \mathcal{V}(a)$ associates a with a global proposal, which we denote by $\bar{P}_k(a)$. The consistency score for assigning a to proposal P_i is defined as the fraction of visible views that agree with this assignment:

$$s_i(a) = \frac{1}{|\mathcal{V}(a)|} \sum_{v_k \in \mathcal{V}(a)} \mathbf{I}[\bar{P}_k(a) = P_i],$$

where $\mathbf{I}$ denotes the indicator function. Each atom is then assigned to the proposal with the highest $s_i(a)$. We then compute the multi-view consistency of each proposal by averaging the consistency scores of its constituent atoms: $S_i = \frac{1}{|P_i|} \sum_{a \in P_i} s_i(a)$. Proposals with $S_i > \theta$ are disassembled, where θ is the reliability threshold. The residual assembly is then passed to the next iteration.

Aggregation and Termination. Let $\mathcal{P}^t$ denote the set of reliable parts detached at iteration t. After T iterations, we aggregate them to obtain the final segmentation $\mathcal{P} = \bigcup_{t=1}^{T} \mathcal{P}^t$. To keep the chain compact, after removing the reliably segmented proposals at each iteration, we terminate if fewer than five proposals remain or if the remaining proposals account for less than 20% of the predicted proposals before removal. In the final iteration, we set the confidence threshold to $\delta = 0$, ensuring that every atom is assigned before aggregation.

4.3 Agentic LEGO® Segmentation Loop

Complex local structures may require denser view coverage to achieve reliable multi-view-consistent segmentation, whereas uniformly increasing the number of views substantially increases inference time. To balance segmentation quality and computational efficiency, we introduce an optional agentic feedback loop that adds targeted viewpoints only where additional visual evidence is needed. After each iteration, the MLLM jointly examines the rendered views and their predicted masks to assess whether the current segmentation is semantically and structurally plausible. If the assessment identifies a poorly segmented region, the agent adds viewpoints targeted at that region and reruns the iteration; for example, an unreliable vehicle underside triggers additional views from below.

5 LEGO-SAM-Flash

LEGO-SAM produces high-quality segmentation results, but its iterative pipeline requires relatively long inference time. To achieve a better trade-off between segmentation quality and inference efficiency, we propose LEGO-SAM-Flash, distilling the predictions of LEGO-SAM into an end-to-end LEGO®-native segmentation model. Specifically, we use LEGO-SAM as an offline annotation engine to generate atom-level pseudo-labels, and design a lightweight residual adapter and an atom-level segmentation head that are tailored to the discrete nature of LEGO® assemblies. LEGO-SAM-Flash substantially reduces inference time while achieving comparable segmentation quality.

5.1 Annotation Engine through LEGO-SAM

We collect LEGO® models from BrickNet (Kulits & Schmid, 2026), the LDraw Official Model Repository (OMR) (LDraw.org, n.d.), and our in-house designs. To ensure the data quality, we preprocess the data with four operations: (i) filtering out models with inter-brick collisions; (ii) canonicalizing model orientations to ensure upright poses for view rendering; (iii) splitting models with multiple disconnected components into separate models; and (iv) removing duplicate models. After preprocessing, we obtain 4,000 models and apply LEGO-SAM to generate atom-level part pseudo-labels. The obtained training set contains 737,349 bricks, 1,857,904 atoms, and 67,466 parts.

5.2 LEGO-SAM-Flash: End-to-End LEGO® Segmentation

As LEGO® models are intrinsically composed of discrete atoms with explicit structural boundaries instead of continuous point or mesh representations, based on the general 3D segmentation model (Ma et al., 2026) architecture, we present LEGO-SAM-Flash by introducing two lightweight adaptations tailored to LEGO® assemblies. First, a zero-initialized residual adapter learns LEGO®-specific

representations while preserving general geometric priors. Second, an atom-level segmentation head aggregates point features within each atom and directly predicts atom-level masks.

LEGO® Adapter. LEGO-SAM-Flash adopts the architecture of P3-SAM (Ma et al., 2026), consisting of a pretrained point-cloud encoder, Sonata (Wu et al., 2025), and a mask decoder. We sample points from the surfaces of all atoms, including both exterior and interior atoms, and feed the resulting point cloud to the encoder. To preserve general 3D geometric priors while learning LEGO®-specific representations, we insert trainable residual adapters into both the point-cloud encoder and mask decoder. For an intermediate feature $\mathbf{z}$, an adapter takes the form

$$\mathrm{Ada}(\mathbf{z}) = \mathbf{z} + sW_{\mathrm{up}}\left(\sigma\left(W_{\mathrm{down}}(\mathbf{z})\right)\right),$$

where Ada denotes the residual adapter, W_{down} and W_{up} are two linear layers, σ denotes GELU, and s is a learnable residual scale. We initialize W_{up} to zero, making the adapter an identity mapping. Thus, the model starts from the pretrained representation while gradually adapting to LEGO® space.

Atom-level Segmentation Head. A conventional point-wise decoder may assign inconsistent predictions to points belonging to the same atom, producing fragmented results. To impose atom boundary priors, we build the atom-level segmentation head by introducing an atom pooling module between point feature extraction and mask decoding. Concretely, let $\mathcal{I}_a$ denote the index set of sampled points belonging to atom a. For each $i \in \mathcal{I}_a$, let $\mathbf{f}_i$ and $\mathbf{x}_i$ denote the feature vector and 3D coordinates of the corresponding point, respectively. We compute the pooled feature $\bar{\mathbf{f}}_a$ and mean coordinates $\bar{\mathbf{x}}_a$ of atom a as

$$\bar{\mathbf{f}}_a = \frac{1}{|\mathcal{I}_a|}\sum_{i\in\mathcal{I}_a}\mathbf{f}_i, \qquad \bar{\mathbf{x}}_a = \frac{1}{|\mathcal{I}_a|}\sum_{i\in\mathcal{I}_a}\mathbf{x}_i.$$

The pooled atom token $(\bar{\mathbf{f}}_a, \bar{\mathbf{x}}_a)$ is fed into the mask decoder, which predicts one mask logit for each atom and converts the logits into the final atom-level mask. Aggregating the decoder input from points to atoms not only ensures atom consistency but also substantially improves the efficiency.

Training Objective. We train the model with Dice and Focal losses for mask prediction and an MSE loss for confidence estimation:

$$\mathcal{L} = \mathcal{L}_{\mathrm{Dice}}(\hat{\mathbf{m}}, \mathbf{m}) + \mathcal{L}_{\mathrm{Focal}}(\hat{\mathbf{m}}, \mathbf{m}) + \mathrm{MSE}(\hat{c}, \mathrm{IoU}(\hat{\mathbf{m}}, \mathbf{m})),$$

where $\hat{\mathbf{m}}$ and $\mathbf{m}$ denote the predicted and ground-truth masks, respectively, and $\hat{c}$ is the predicted confidence score trained to estimate their IoU.

6 Experiments

6.1 LEGO-SAM-Bench

Given the scarcity of semantic LEGO® data, no high-quality public benchmark for LEGO® segmentation is currently available. We therefore built LEGO-SAM-Bench, the first dataset for LEGO® segmentation, covering a broad range of common LEGO® categories with instance annotations at atom level. We identified ten major categories commonly appearing in LEGO® designs: indoor scenes, human-shaped figurines, animals, cars, ships, planes, spaceships, botanicals, houses, and micro buildings. In total, the dataset contains 300 models: 100 indoor scenes, 100 human-shaped figurines, and 100 models from eight other categories. For the third group, we collected 12-13 models per category to ensure category diversity. We recruited two professional LEGO® designers to segment the LEGO® models into semantically-coherent parts and annotate each part with a text label. Since our dataset spans a wide diversity of categories, we do not restrict the label annotations to a closed label set. For example, indoor scenes typically contain parts such as windows, doors, walls, floors, beds, and tables; human-shaped figurines contain heads, torsos, legs, arms, handhelds, and accessories; and cars contain wheels, lights, fenders, seats, and windshields. After annotation, LEGO-SAM-Bench contains 133,211 bricks (347,831 atoms after atomization) and 6,420 part annotations.

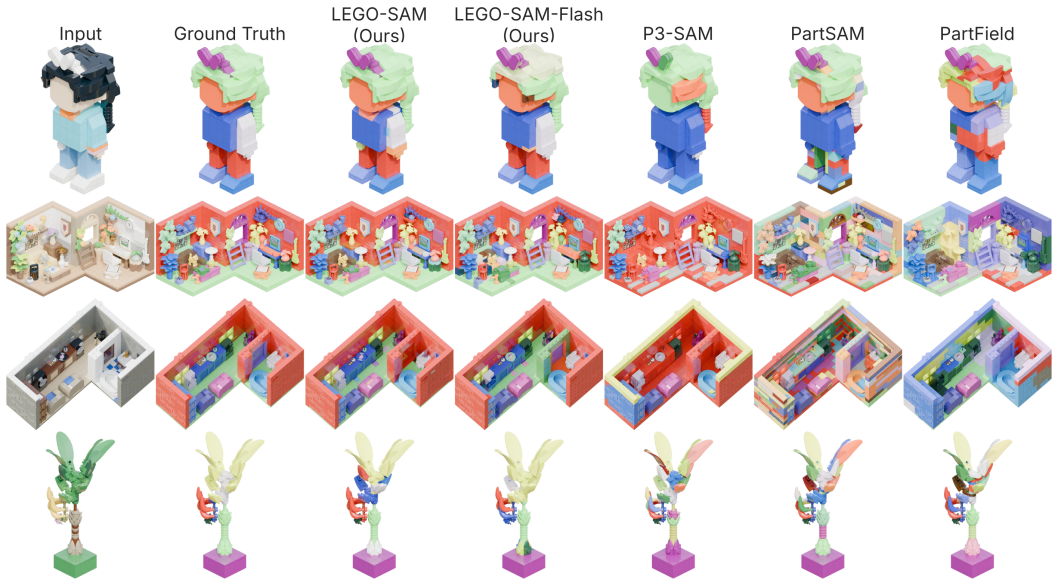

Figure 7: Visual comparison between our methods and baselines. Both LEGO-SAM and LEGO-SAM-Flash can more accurately segment objects and parts in these LEGO® models. Please refer to the webpage submitted in the supplemental materials for interactive visualization of our results.

Table 1: Quantitative comparison with baseline 3D shape segmentation methods. We report mean bounding-box IoU (mIoU), F1 score, precision (Prec.), and recall (Rec.), all expressed as percentage (%). **Dark blue** indicates the best result, whereas **light blue** indicates the second-best result.

Method	Runtime↓	Overall				Figurines		Indoor Scenes		Others	
		mIoU↑	F1↑	Prec.↑	Rec.↑	mIoU↑	F1↑	mIoU↑	F1↑	mIoU↑	F1↑
SAMesh	∼7 min	20.24	12.46	8.31	24.92	23.53	15.46	18.30	10.84	18.90	11.08
GeoSAM2	∼30 s	10.02	12.94	21.04	18.08	12.51	14.89	10.54	12.49	7.01	11.43
PartField	∼10 s	27.12	20.08	13.43	39.88	35.13	28.90	22.59	14.65	23.64	16.68
P3-SAM	**∼8 s**	32.24	23.19	15.87	44.63	41.57	28.47	29.31	22.71	25.84	18.38
PartSAM	∼12 s	36.22	24.49	16.32	48.97	48.10	33.28	31.15	21.20	29.40	18.98
LEGO-SAM-Flash (ours)	**∼6 s**	**66.69**	**44.41**	**42.26**	**53.66**	**85.92**	**66.60**	**58.51**	**37.41**	**55.64**	**29.23**
LEGO-SAM (ours)	∼17 min	**67.40**	**42.11**	**43.62**	**65.54**	**71.45**	**45.22**	**69.57**	**40.04**	**61.18**	**41.06**

6.2 Implementation Details

For LEGO-Atomizer, we solve the ILP for atomization using the HiGHS (Huangfu & Hall, 2018) solver. For LEGO-SAM, we use the open-source MLLM HunyuanImage-3.0-Instruct (Tencent Hunyuan Foundation Model Team, 2026) for rendered-view selection and view-wise 2D mask generation. For LEGO-SAM-Flash, we initialize from P3-SAM and Sonata and train lightweight residual adapters while keeping the pretrained backbone frozen. Training takes approximately 40 hours on one NVIDIA H200 GPU. Additional implementation details and hyperparameters are provided in Appendix.

6.3 Evaluation

Experimental Setting. We compare our method with state-of-the-art 3D segmentation models as baselines, including SAMesh (Tang et al., 2025), P3-SAM (Ma et al., 2026), PartField (Liu et al., 2025), PartSAM (Zhu et al., 2026), and GeoSAM2 (Deng et al., 2026). Each baseline is applied with its recommended best practice.

We evaluate segmentation quality with the following metrics: mean bounding-box IoU (mIoU), F1 score, precision, and recall. We first establish one-to-one correspondences between predicted and ground-truth parts using Hungarian matching, with the same assignment used for all metrics. mIoU measures the average overlap between the axis-aligned bounding boxes of matched parts. Precision and Recall measure the proportions of correctly matched parts among all predicted and ground-truth parts, respectively, while F1 score is their harmonic mean. Here, a matched pair is considered a true positive only if its bounding-box IoU is at least 0.3.

Quantitative Evaluation. Table 1 reports mIoU, F1, precision, and recall of our methods and baselines. Both our methods consistently outperform the baselines and generalize well across diverse LEGO® categories. Notably, LEGO-SAM performs particularly well on structurally complex categories, such as indoor scenes, benefiting from its iterative segmentation pipeline, while LEGO-SAM-Flash substantially reduces inference time through an efficient end-to-end design.

Qualitative Evaluation. Figure 7 visually compares our methods and baselines. Both LEGO-SAM and LEGO-SAM-Flash consistently segment coherent semantic parts across diverse LEGO® models and closely match the ground-truth decompositions made by human designers. Particularly, through Chain of Disassembly, LEGO-SAM can robustly segment heavily occluded objects like floor, wall and furniture in indoor scenes, while LEGO-SAM-Flash retains similar visual quality with substantially reduced inference time. See the Appendix and our webpage for more visual results.

Ablation study. We conduct an ablation study on the proposal-prompted 3D structural segmentation and Chain of Disassembly (CoD) in LEGO-SAM to assess their effectiveness. Table 2 shows that removing either component substantially degrades the segmentation quality. The prompted segmentation predicts complete structures, while CoD progressively exposes the hidden structures.

Table 2: Ablation of proposal-prompted 3D structural segmentation (Seg.) and Chain of Disassembly (CoD) in LEGO-SAM.

Method	mIoU↑	F1↑	Prec.↑	Rec.↑
w/o Seg.	29.61	29.02	35.83	28.20
w/o CoD	57.04	37.88	39.85	54.27
LEGO-SAM	**67.40**	**42.11**	**43.62**	**65.54**

6.4 Applications

We showcase three downstream applications in Figure 1. Additional details are provided in Appendix.

(i) Semantic-aware Instruction Generation. Given a segmented LEGO® model, we generate semantic-aware assembly instructions by assembling local semantic groups separately and attaching them to the main body, following the practice in human-designed instructions in official LEGO® products.

(ii) Text-guided Grounding. Given a text query and a LEGO® model, we use an MLLM to identify the target part in a reference view and associate it with the segmentation produced by LEGO-SAM.

(iii) Object/Part Retrieval and Replacement. We build a reusable semantic part library from segmented LEGO® models, enabling text-based retrieval of objects and parts. Retrieved parts can further be used for replacement when the insertion is compatible and does not introduce collisions.

7 Conclusion

We presented LEGO-SAM, the first framework for segmenting 3D LEGO® models into semantic parts. Through our atomization formulation and Chain of Disassembly method, LEGO-SAM iteratively segments objects and parts, despite occlusion and complex part relations. We also introduce LEGO-SAM-Flash for fast end-to-end segmentation. Further, we build LEGO-SAM-Bench, the first LEGO® segmentation benchmark with manual annotations. Beyond improving segmentation accuracy over state-of-the-art 3D segmentation models, LEGO-SAM introduces semantic information to LEGO® modeling and opens new possibilities for a variety of assembly-aware downstream applications.

Discussions. Please refer to the Appendix for comprehensive method analysis and additional discussions, including implementation details, additional results, and applications.

AI USE STATEMENT

In this work, we used our proposed LEGO-SAM as an annotation engine to generate part pseudo-labels for the LEGO® models, which were used as training data for LEGO-SAM-Flash. We have not used generative AI tools for developing theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, implementing methods, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting results. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for summarizing or analyzing existing literature, editing the research paper to improve readability, and drafting MLLM prompts. We have reviewed all AI-assisted work. All authors carefully read the original literature, and all AI-assisted edits were also carefully reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. SATR: Zero-shot semantic segmentation of 3D shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15166–15179, October 2023.

Tomasz Boiński, Konrad Zawora, and Julian Szymański. How to sort them? a network for LEGO bricks classification. In *Proceedings of the International Conference on Computational Science (ICCS)*, pp. 627–640, 2022.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, François Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 138846–138923, April 2026.

Godtfred Kirk Christiansen. Toy building brick. U.S. Patent 3,005,282, October 1961.

Hyunsoo Chung, Jungtaek Kim, Boris Knyazev, Jinhwi Lee, Graham W. Taylor, Jaesik Park, and Minsu Cho. Brick-by-brick: Combinatorial construction with deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 5745–5757, 2021.

Ken Deng, Yunhan Yang, Jingxiang Sun, Xihui Liu, Yebin Liu, Ding Liang, and Yan-Pei Cao. GeoSAM2: Unleashing the power of SAM2 for 3D part segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6367–6376, June 2026.

Marco Garosi, Riccardo Tedoldi, Davide Boscaini, Massimiliano Mancini, Nicu Sebe, and Fabio Poiesi. 3D part segmentation via geometric aggregation of 2D visual features. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pp. 3257–3267, February 2025.

Jiahao Ge, Mingjun Zhou, Wenrui Bao, Hao Xu, and Chi-Wing Fu. Creating LEGO figurines from single images. *ACM Transactions on Graphics (SIGGRAPH 2024)*, 43(4):1–16, 2024a.

Jiahao Ge, Mingjun Zhou, and Chi-Wing Fu. Learn to create simple LEGO micro buildings. *ACM Transactions on Graphics (SIGGRAPH Asia 2024)*, 43(6):1–13, 2024b.

Jiahao Ge, Mingjun Zhou, Hanyou Zheng, Hao Xu, and Chi-Wing Fu. LEGO-maker: Autoregressive image-conditioned LEGO model creation. *ACM Transactions on Graphics (SIGGRAPH Asia 2025)*, 44(6):1–15, 2025.

Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805v5*, 2025. Version 5, May 9, 2025; first released in 2023.

Rebecca A. H. Gower, Agnes E. Heydtmann, and Henrik G. Petersen. LEGO: Automated model construction. In *32nd European Study Group with Industry - Final Report*, pp. 81–94, 1998.

Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9224–9232, June 2018.

Chenyang Gu, Jiaming Liu, Hao Chen, Runzhong Huang, Qingpo Wuwu, Xiaoqi Li, Zhuoyang Liu, Ying Li, Renrui Zhang, Peng Jia, Pheng-Ann Heng, and Shanghang Zhang. From manuals to actions: A unified VLA model for chain-of-thought manual generation and robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13540–13552, June 2026.

Mengqi Guo, Chen Li, Yuyang Zhao, and Gim Hee Lee. TreeSBA: Tree-transformer for self-supervised sequential brick assembly. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 35–51, 2024.

Souhail Hadgi, Bingchen Gong, Ramana Sundararaman, Emery Pierson, Lei Li, Peter Wonka, and Maks Ovsjanikov. PatchAlign3D: Local feature alignment for dense 3D shape understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3014–3023, June 2026.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A network with an edge. *ACM Transactions on Graphics (SIGGRAPH 2019)*, 38(4): 1–12, 2019.

Qi Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D. D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. UV-Net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11703–11712, June 2021.

Li Jin, Weikai Chen, Yujie Wang, Yingda Yin, Zeyu Hu, Runze Zhang, Keyang Luo, Shengju Qian, Xin Wang, and Xueying Qin. CoSMo3D: Open-world promptable 3D semantic segmentation through LLM-guided canonical spatial modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14325–14334, June 2026.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, October 2023.

Peter Kulits and Cordelia Schmid. BrickNet: Graph-backed generative brick assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39252–39261, June 2026.

Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. BRepNet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12773–12782, June 2021.

LDraw.org. Official model repository (OMR). `https://library.ldraw.org/omr`, n.d. Accessed September 3, 2026.

Sangyeop Lee, Jinhyun Kim, Jae Woo Kim, and Byung-Ro Moon. Finding an optimal LEGO® brick layout of voxelized 3D object using a genetic algorithm. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 1215–1222, July 2015.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10965–10975, June 2022.

Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. MeshMAE: Masked autoencoders for 3D mesh data analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 13663, pp. 37–54, 2022.

Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and Hao Su. PartSLIP: Low-shot part segmentation for 3D point clouds via pretrained image-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21736–21746, June 2023.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. PartField: Learning 3D feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9704–9715, October 2025.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 38–55, 2024.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: Optimizing LEGO designs. *ACM Transactions on Graphics (SIGGRAPH Asia 2015)*, 34(6):1–12, 2015.

Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P^3-SAM: Native 3D part segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2026.

Ziqi Ma, Yisong Yue, and Georgia Gkioxari. Find any part in 3D. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7818–7827, October 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Pavel Petrovič. Solving LEGO brick layout problem using evolutionary algorithms. In *Proceedings of the Norsk Informatikkonferanse (NIK)*, pp. 87–97, 2001.

Maxim Peysakhov, Vlada Galinskaya, and William C. Regli. Representation and evolution of LEGO-based assemblies. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI)*, pp. 1089, 2000.

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14798–14809, October 2025.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, July 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pp. 5099–5108, 2017b.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pp. 8748–8763, 2021.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 28085–28128, May 2025.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. DiffusionNet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics*, 41(3):1–16, 2022.

Han Su, Tianyu Huang, Zichen Wan, Xiaohe Wu, and Wangmeng Zuo. S^2AM3D: Scale-controllable part segmentation of 3D point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14357–14366, June 2026.

George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh. *arXiv preprint arXiv:2408.13679v2*, 2025. Version 2, March 9, 2025; first released in 2024.

Tencent Hunyuan Foundation Model Team. HunyuanImage 3.0 technical report. *arXiv preprint arXiv:2509.23951v3*, 2026. Version 3, June 26, 2026; first released in 2025.

Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. Automatic generation of constructable brick sculptures. In *Eurographics 2013 - Short Papers*, pp. 81–84, 2013.

The LEGO® Group. LEGO® system in play. `https://www.lego.com/en-us/history/articles/lego-system-in-play`, n.d. Accessed September 26, 2026.

Ardian Umam, Cheng-Kun Yang, Min-Hung Chen, Jen-Hui Chuang, and Yen-Yu Lin. PartDistill: 3D shape part segmentation by vision-language model distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3470–3479, June 2024.

Joel Vidal, Guillem Vallicrosa, Robert Martí, and Marc Barnada. Brickognize: Applying photo-realistic image synthesis for LEGO bricks recognition with limited data. *Sensors*, 23(4):1898, 2023.

Aaron Walsman, Muru Zhang, Klemen Kotar, Karthik Desingh, Ali Farhadi, and Dieter Fox. Break and make: Interactive structural understanding using LEGO bricks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 90–107, 2022.

Aaron Walsman, Muru Zhang, Adam Fishman, Ali Farhadi, and Dieter Fox. Learning to build by building your own instructions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 261–278, 2024.

Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Chin-Yi Cheng, and Jiajun Wu. Translating a visual LEGO manual to a machine-executable plan. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 677–694, 2022.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4840–4851, June 2024.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22193–22204, June 2025.

Hao Xu, Ka-Hei Hui, Chi-Wing Fu, and Hao Zhang. Computational LEGO® technic design. *ACM Transactions on Graphics (SIGGRAPH Asia 2019)*, 38(6):1–15, 2019.

Hao Xu, Yuqing Zhang, Yiqian Wu, Xinyang Zheng, Yutao Liu, Xiangjun Tang, Yunhan Yang, Ding Liang, Yingtian Liu, Yuanchen Guo, Yanpei Cao, and Xiaogang Jin. LegoACE: Autoregressive construction engine for expressive LEGO assemblies. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–11, December 2025a.

Mutian Xu, Xingyilang Yin, Lingteng Qiu, Yang Liu, Xin Tong, and Xiaoguang Han. SAMPro3D: Locating SAM prompts in 3D for zero-shot instance segmentation. In *Proceedings of the International Conference on 3D Vision (3DV)*, pp. 1222–1232, 2025b.

Yuheng Xue, Nenglun Chen, Jun Liu, and Wenyun Sun. ZeroPS: High-quality cross-modal knowledge transfer for zero-shot 3D part segmentation. In *Proceedings of the International Conference on 3D Vision (3DV)*, pp. 1328–1339, 2025.

Hongyu Yan and Yadong Mu. Neural assembler: Learning to generate fine-grained robotic assembly instructions from multi-view images. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, pp. 14717–14725, 2025.

Yunhan Yang, Xiaoyang Wu, Tong He, Hengshuang Zhao, and Xihui Liu. SAM3D: Segment anything in 3D scenes. In *ICCV Workshop on Open-Vocabulary 3D Scene Understanding (OpenSUN3D)*, October 2023.

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y. Lam, Yan-Pei Cao, and Xihui Liu. SAMPart3D: Segment any part in 3D objects. *arXiv preprint arXiv:2411.07184*, 2024.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16259–16268, October 2021.

Ziming Zhong, Yanyu Xu, Jing Li, Jiale Xu, Zhengxin Li, Chaohui Yu, and Shenghua Gao. MeshSegmenter: Zero-shot mesh semantic segmentation via texture synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 182–199, 2024.

Jie Zhou, Xuejin Chen, and Ying-Qing Xu. Automatic generation of vivid LEGO architectural sculptures. *Computer Graphics Forum*, 38(6):31–42, 2019.

Jinfan Zhou, Richard Liu, Itai Lang, and Rana Hanocka. MeshFM: 2D features are all you need for 3D shape understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2026.

Mingjun Zhou, Jiahao Ge, Hao Xu, and Chi-Wing Fu. Computational design of LEGO® sketch art. *ACM Transactions on Graphics (SIGGRAPH Asia 2023)*, 42(6):1–15, 2023a.

Pengwei Zhou, Xiao Dong, Juan Cao, and Zhonggui Chen. MeT: Mesh transformer with an edge. *The Visual Computer*, 39(8):3235–3246, 2023b.

Yuchen Zhou, Jiayuan Gu, Tung Yen Chiang, Fanbo Xiang, and Hao Su. Point-SAM: Promptable 3D segmentation model for point clouds. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 56850–56869, May 2025a.

Yuchen Zhou, Jiayuan Gu, Xuanlin Li, Minghua Liu, and Hao Su. PartSLIP++: Enhancing low-shot 3D part segmentation via multi-view instance segmentation and maximum likelihood estimation. In *ICCV Workshop on Wild 3D: 3D Modeling, Reconstruction, and Generation in the Wild (Wild3D)*, October 2025b.

Zhe Zhu, Le Wan, Rui Xu, Yiheng Zhang, Honghua Chen, Zhiyang Dou, Cheng Lin, Yuan Liu, and Mingqiang Wei. PartSAM: A scalable promptable part segmentation model trained on native 3D data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 132469–132493, April 2026.

APPENDIX: LEGO-SAM: SEGMENT ANY LEGO® MODEL THROUGH CHAIN OF DISASSEMBLY

Anonymous authors
Paper under double-blind review

A LEGO-ATOMIZER

In this section, we describe the construction of our LEGO-Atomizer. We first introduce the discrete representation of LEGO® bricks, and then formulate atomization as a generalized exact-cover problem, which is solved using integer linear programming.

A.1 TASK DEFINITION

Given a LEGO® model $\mathcal{M} = \{b_i\}_{i=1}^{n_\mathcal{M}}$ consisting of $n_\mathcal{M}$ bricks, the goal of the atomizer is to decompose each brick into an assembly of finer atom-level units. After atomization, the model is represented as $\mathcal{A} = \{a_i\}_{i=1}^{n_\mathcal{A}}$, where each a_i is an atom that cannot be further decomposed.

Since no existing atomization schemes are available for the entire LEGO® library, we compute these schemes automatically for all brick types. For each brick b in the library, we represent its geometry using three discrete sets: $V(b), S(b), T(b)$, where $V(b)$ denotes the occupied volume of the brick, and $S(b)$ and $T(b)$ denote the locations of its studs and tubes, respectively. All sets are defined on the same discrete LEGO® grid. The atomization aims to reconstruct each brick using the smallest non-decomposable atoms while preserving its geometry and external connectivity.

A.2 ATOMIZATION SCHEME SEARCH

We formulate atomization as a generalized exact-cover problem. For a target brick b, we enumerate all feasible placements of smaller candidate atoms a within b. Each placement $p(a)$ of atom a has occupied volume V_p, studs S_p, and tubes T_p, and is feasible only if $V_p \subseteq V(b)$. The selected placements must exactly cover $V(b)$ without gaps or overlaps, while preserving the exposed connectors of b. When a stud from one selected atom coincides with a tube from another, the pair forms an internal connection and is removed from the exposed connector sets. Formally, $\mathcal{Q}$ is required to satisfy

$$\bigcup_{p\in\mathcal{Q}} V_p = V(b),$$
$$\bigcup_{p\in\mathcal{Q}} S_p \setminus \bigcup_{p\in\mathcal{Q}} T_p = S(b),$$
$$\text{and } \bigcup_{p\in\mathcal{Q}} T_p \setminus \bigcup_{p\in\mathcal{Q}} S_p = T(b),$$

where $\bigcup$ denotes a disjoint union and $\setminus$ denotes set difference.

We solve this discrete optimization as an integer linear program. If multiple solutions exist, we choose the solution that produces the largest number of atoms, yielding the finest-grained decomposition. Empirically, we find that every brick in the LEGO® library admits a unique feasible atomization scheme, so no ambiguity arises in practice. If no feasible decomposition exists, the brick itself is treated as an atom.

B IMPLEMENTATION DETAILS

For LEGO-SAM, we initially render the atomized assembly from 24 candidate camera poses, combining eight azimuth angles, $0°$, $45°$, $90°$, $135°$, $180°$, $225°$, $270°$, and $315°$, with three elevation

angles, $-10°$, $30°$, and $60°$. Then, we use the open-source MLLM HunyuanImage-3.0-Instruct (Tencent Hunyuan Foundation Model Team, 2026) for rendered-view selection and view-wise 2D mask generation. The prompts are provided in Appendix H. When constructing the match graph, we set the bounding-box IoU threshold τ to 0.5 for figurines, 0.5 for indoor scenes, and category-specific values ranging from 0.4 to 0.7 for the remaining categories. For proposal-prompted 3D structural segmentation, we sample $100{,}000$ points from the atom surfaces and set the confidence threshold δ to 0.5. During the Chain of Disassembly, the reliability threshold θ is set to 0.5.

For LEGO-SAM-Flash, we initialize from P3-SAM and Sonata and train lightweight residual adapters while keeping the pretrained backbone frozen. Residual adapters are inserted with a bottleneck dimension of 128, totaling 265,603 trainable parameters. We sample 100,000 points from the atom surfaces, and train with a batch size of 32 and a learning rate of 1×10^{-4} for 250 epochs. Training takes approximately 40 hours on one NVIDIA H200 GPU.

C Evaluation Metrics

We evaluate part segmentation using mean bounding-box IoU (mIoU), precision, recall, and F1 score. We establish one-to-one correspondences between predicted and ground-truth parts using Hungarian matching and use the same correspondence across all metrics.

Bounding-Box IoU. For a LEGO® model, let $\{G_i\}_{i=1}^{M}$ and $\{P_j\}_{j=1}^{N}$ denote the ground-truth and predicted parts, respectively. Let $B(X)$ denote the 3D axis-aligned bounding box enclosing all geometry assigned to part X. We compute the pairwise bounding-box IoU as

$$s_{ij} = \frac{\mathrm{Vol}(B(G_i) \cap B(P_j))}{\mathrm{Vol}(B(G_i) \cup B(P_j))}, \tag{1}$$

where $\mathrm{Vol}(\cdot)$ denotes volume.

One-to-One Assignment. We apply the Hungarian algorithm to the pairwise IoU matrix to obtain an assignment $\mathcal{C}^\star$ that maximizes the total bounding-box IoU:

$$\mathcal{C}^\star = \arg\max_{\mathcal{C}} \sum_{(i,j)\in\mathcal{C}} s_{ij}, \tag{2}$$

where $\mathcal{C}$ contains $\min(M, N)$ pairs and each part appears in at most one pair.

Metric Definitions. We compute mIoU by normalizing the total IoU from the matched pairs by the number of ground-truth parts:

$$\mathrm{mIoU} = \frac{1}{M} \sum_{(i,j)\in\mathcal{C}^\star} s_{ij}. \tag{3}$$

Here, unmatched ground-truth parts contribute zero.

For precision, recall, and F1, a matched pair is counted as a true positive if its bounding-box IoU is at least τ_{eval}:

$$\mathrm{TP} = \sum_{(i,j)\in\mathcal{C}^\star} \mathbf{1}[s_{ij} \geq \tau_{\mathrm{eval}}]\,. \tag{4}$$

We use $\tau_{\mathrm{eval}} = 0.3$ for all evaluated methods. Precision and recall are defined as

$$\mathrm{Precision} = \frac{\mathrm{TP}}{N}, \qquad \mathrm{Recall} = \frac{\mathrm{TP}}{M}. \tag{5}$$

F1 is their harmonic mean, equivalently expressed as

$$\mathrm{F1} = \frac{2\,\mathrm{TP}}{M + N}. \tag{6}$$

For baseline methods only, we cap the number of predicted parts at three times the number of ground-truth parts to mitigate the effect of severe over-fragmentation on precision. No such cap is applied to our methods. We compute F1, precision, and recall model by model, and compute the dataset-level metrics by averaging over all models in the dataset.

D METHOD ANALYSIS

D.1 RUNTIME ANALYSIS

Table 1 reports per-model runtime and the length of Chain of Disassembly (CoD). In addition, for LEGO-SAM, the initial view selection takes 114.4 seconds on average. Each CoD iteration takes 403.8 seconds in total, including 12.3 seconds for multi-view rendering, 376.0 seconds for view-wise segmentation, and 15.5 seconds for lifting, merging, and disassembly.

Table 1: Mean per-model runtime and LEGO-SAM chain length on LEGO®-SAM-Bench, reported over the full benchmark and each category group. LEGO-SAM uses four H200 GPUs, while LEGO-SAM-Flash uses one H200 GPU.

Category group	Runtime (s/model)		Chain length (iterations)
	LEGO-SAM	LEGO-SAM-Flash	LEGO-SAM
Overall	1036.4	6.1	2.283
Figurines	970.5	1.5	2.120
Indoor scenes	1119.9	9.1	2.490
Others	1018.9	7.8	2.240

D.2 CHAIN OF DISASSEMBLY ANALYSIS

Table 2 reports the mIoU, F1, precision, and recall from each iteration in our Chain of Disassembly. The consistently decreasing performance across iterations suggests that CoD successfully prioritizes reliable parts in the early stages, thereby preventing error propagation along the chain.

Table 2: Quantitative evaluation of each iteration during Chain of Disassembly for LEGO-SAM.

Iteration	mIoU	F1	Precision	Recall
1	76.56	50.23	55.56	74.79
2	66.67	42.96	43.47	61.02
3	56.49	29.78	38.59	55.76
Final segmentation	67.40	42.11	43.62	65.54

E APPLICATION

We showcase three downstream applications in the main paper Figure 1. The implementation details are provided as follows:

(i) Semantic-aware Instruction Generation. Given a segmented LEGO® model, we can generate semantic-aware assembly instructions by assembling local semantic groups separately and attaching them to the main body, following the practice in human-designed instructions in official LEGO® products. In detail, we build a connectivity graph between the segmented parts according to their connectivity relations. We then grow an assembly tree based on the graph, where the lowest and largest part is chosen as the root of the tree. At each step, we select the candidate part that satisfies two criteria: it can connect to the current partial assembly and can be inserted into it without collision. In addition to checking collisions with the already assembled parts, we perform a look-ahead collision check to ensure adding the candidate will not block insertions of the remaining parts.

(ii) Text-guided grounding. LEGO® segmentations produced by LEGO-SAM can facilitate text-guided grounding. To do so, we employ an MLLM to locate the target part in a reference view; we can then select the segmentation ID corresponding to the text query.

(iii) Object/part retrieval and replacement. We first define a common part vocabulary, including arm with handheld object, sofa, table, vase with flower, window, and so on. By repeatedly applying the

text-guided grounding algorithm introduced above with this vocabulary, we can extract semantic parts from existing LEGO® models and build a reusable LEGO® part library. The resulting library supports text-based part and object retrieval. Given a text prompt, we can retrieve semantically-matched parts from the library; see Figure 1. Further, we can replace a target part in a LEGO® model with a retrieved part. A retrieved part is considered compatible if it shares the same connector(s) as the target part and can be inserted into the LEGO® model without causing collisions.

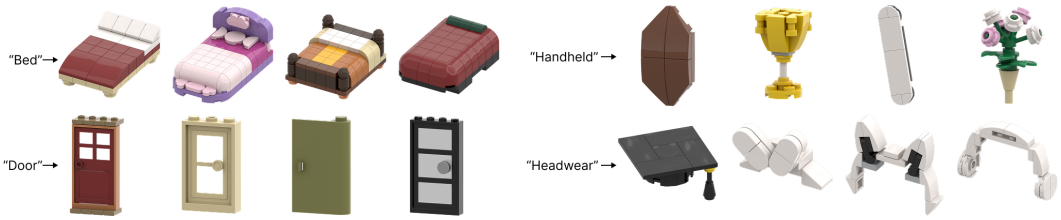

Figure 1: Different objects/parts retrieved using LEGO® SAM from our benchmark dataset.

F QUALITATIVE EVALUATION

Here, we provide additional visual comparison between our methods and baseline methods.

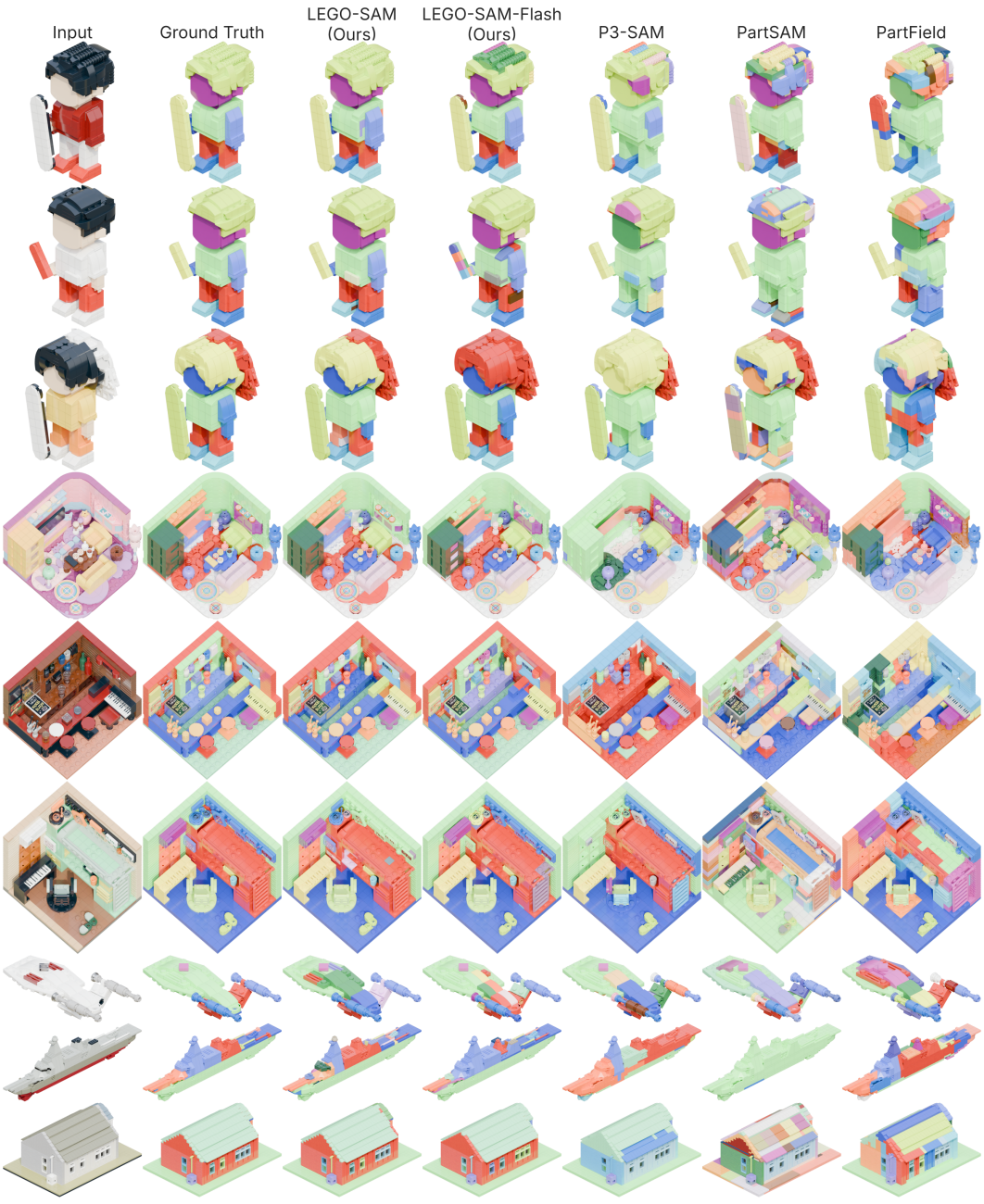

Figure 2: Visual comparison of LEGO-SAM and LEGO-SAM-Flash with P3-SAM (Ma et al., 2026), PartSAM (Zhu et al., 2026), and PartField (Liu et al., 2025).

G MORE RESULTS

In this section, we provide more results from LEGO-SAM and LEGO-SAM-Flash.



Figure 3: Segmentation results for LEGO® figurine models.

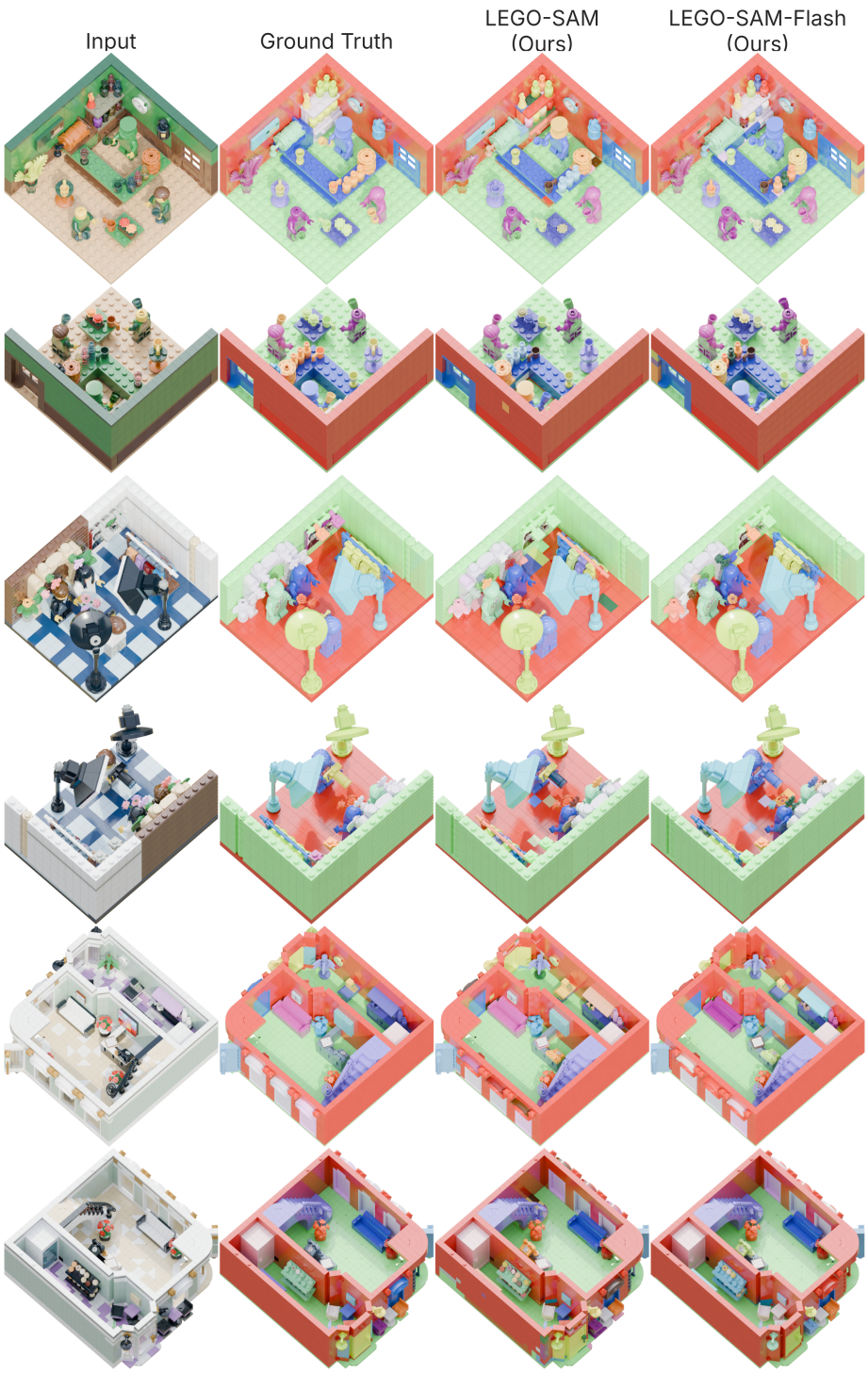

Figure 4: Segmentation results for LEGO® indoor scenes.

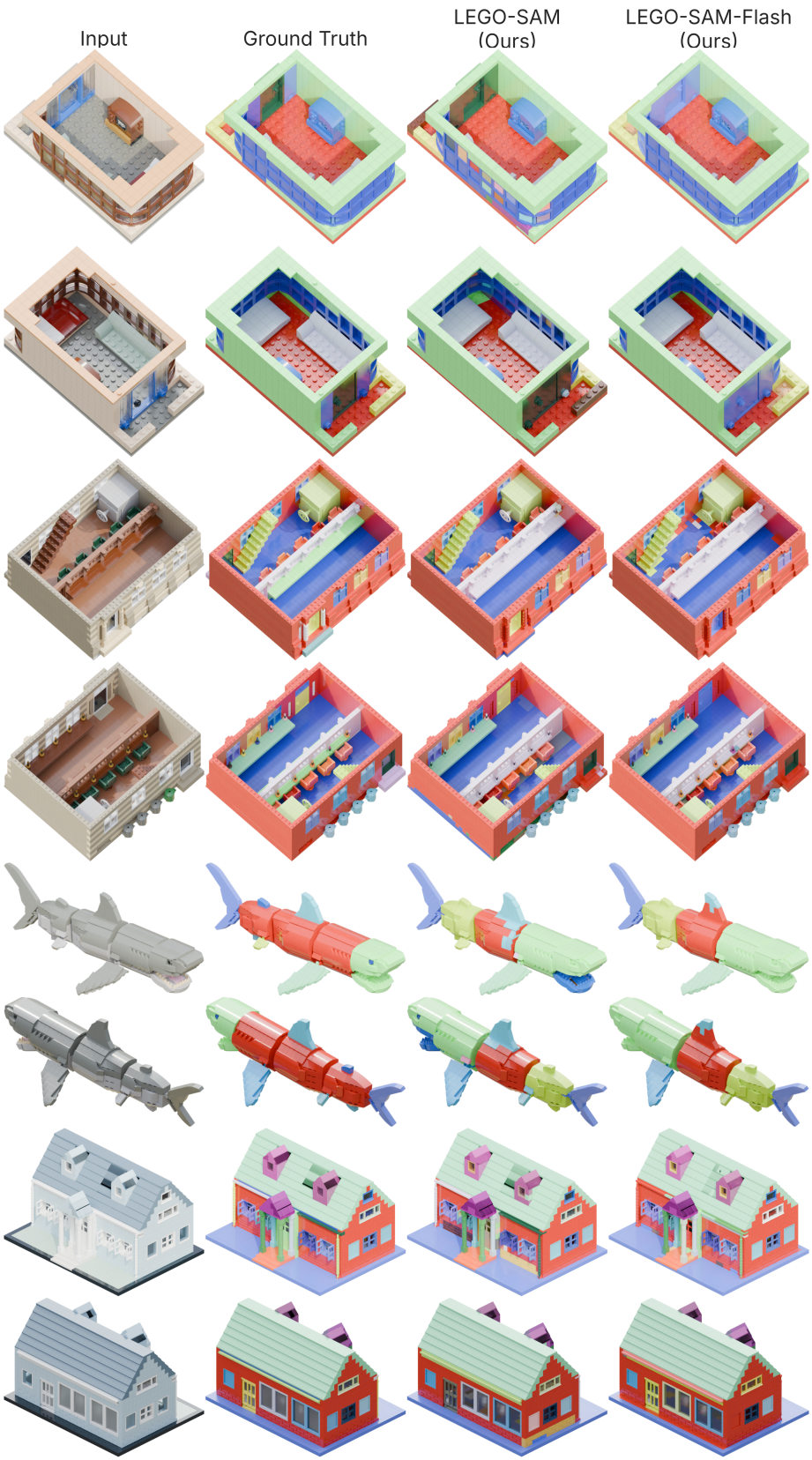

Figure 5: Additional segmentation results for LEGO® indoor scenes and other categories.



Figure 6: Segmentation results for other LEGO® categories.

H PROMPTS

We provide the system and user prompts used with HunyuanImage-3.0-Instruct (Tencent Hunyuan Foundation Model Team, 2026) to categorize rendered models, together with a retry prompt for responses that fail to satisfy the required JSON format or schema. We also provide the overlay generation prompt used for view-wise 2D segmentation.

Listing 1: System prompt for single-view categorization.

```
ROLE

You are a conservative visual analyst for rendered LEGO models.

INPUT

You are given exactly one image showing one view of one LEGO model.

TASK

Determine whether the visible model can be recognized as a specific,
semantically meaningful subject, type, or coherent built environment. Exact
LEGO set identification is neither required nor desired.

EVIDENCE RULES

1. Analyze only what is visibly supported by the image.
2. Do not assume hidden geometry, unseen parts, filenames, set numbers, source
   metadata, or franchise identities.
3. Do not use the plain background or rendering style as semantic evidence.
4. Do not force an abstract, incomplete, disconnected, heavily occluded, or
   semantically indeterminate construction into a category.
5. If the visible evidence admits competing interpretations without supporting
   a stable subject, set recognizability to unrecognizable and category to null.

EVIDENCE-FIRST PROCESS

1. Describe the visible model without assigning a category.
2. Identify concrete visual evidence that supports or weakens recognition.
3. Propose a specific subject and a broad theme when visibly supported.
4. Briefly explain whether the evidence is sufficient.
5. Decide the recognizability and category only after completing the preceding
   steps.

OUTPUT CONSTRAINTS

Return exactly one valid JSON object using the requested key order. Do not
return markdown, commentary, or any text outside the JSON object.
```

Listing 2: User prompt and structured output specification for single-view categorization.

```
Analyze the rendered LEGO model in this single image. Return exactly one JSON
object with the fields below, in the specified order.

FIELD DEFINITIONS

1. description
   Give a concise, neutral description of the complete visible model. Describe
   its overall shape and major visible components without assigning a category.

2. visible_evidence
   State the concrete visual features that support or weaken recognition, such
   as wheels, wings, hull, rooms, windows, limbs, leaves, tracks, cockpit,
   furniture surfaces, tools, or articulated mechanical parts. Do not use the
   background or rendering style as evidence.

3. subject
   Give the most specific visually supported subject or type, such as passenger
   train, helicopter, dragon, tree, robot, house, or chair. Use null if no
   sufficiently specific subject can be proposed. Generic phrases such as LEGO
   model, LEGO construction, structure, object, vehicle-like build, or
   mechanical shape are not valid recognizable subjects.

4. theme
   Give a short broad theme, such as domestic, industrial, medieval, nature,
   science fiction, or transportation. Use null if no theme is visibly
   supported. Do not guess an exact franchise or LEGO product line.
```

```
5. reason
   Briefly explain why the visible evidence is or is not sufficient to recognize
   a specific subject. Mention important ambiguity when multiple interpretations
   remain plausible. Keep the explanation concise and evidence-based.

6. recognizability
   Choose either recognizable or unrecognizable. Follow the definitions in
   RECOGNIZABILITY LABELS below.

7. category
   If recognizability is recognizable, choose exactly one category from the
   normalized vocabulary in CATEGORY LABELS below. Use null when recognizability
   is unrecognizable.

8. confidence
   Choose high, medium, or low to indicate confidence in the complete assessment
   returned for this view, including the recognizability decision.

RECOGNIZABILITY LABELS

recognizable
The visible evidence supports one specific and semantically meaningful subject
or coherent built environment. Exact model identity is not required; a stable
type such as car, aircraft, tree, robot, or house is sufficient.

unrecognizable
The visible evidence does not support a stable semantic interpretation. This
includes views with competing plausible subjects, as well as abstract, generic,
incomplete, disconnected, or semantically indeterminate constructions.
Recognizing only a visual style or vague theme is not sufficient.

CATEGORY LABELS

Building
A building, house, room, interior layout, bridge, station, or other coherent
architectural or spatial environment.

Vehicle
A vehicle that primarily travels on land, including cars, trucks, motorcycles,
trains, tracked vehicles, construction vehicles, and land rovers. Vehicle is
reserved for land transportation.

Aircraft
A vehicle primarily intended to fly in an atmosphere, including airplanes,
helicopters, and clearly aircraft-like drones.

Watercraft
A vehicle primarily intended to travel on or under water, including boats,
ships, and submarines.

Spacecraft
A vehicle or machine clearly intended for space travel or operation, including
spacecraft, rockets, satellites, and space probes.

Animal
A recognizable non-human animal or creature. For this version, fictional
creatures such as dragons may also use Animal.

Plant
A recognizable plant, tree, flower, or other botanical subject.

Character
A human, humanoid, minifigure-like person, statue, or visibly character-like
fictional subject.

Robot
An embodied mechanical agent, robot, droid, or mecha with a recognizable body
or agent-like structure.

Machine
A non-agentive mechanical apparatus, device, tool, mechanism, or industrial
machine whose primary identity is not transportation.

Furniture
A standalone piece of furniture or furnishing, such as a chair, table, bed,
sofa, shelf, cabinet, or desk.

Other
A concrete and recognizable subject that does not fit any category above. A
specific, non-generic subject must be provided.
```

```
CATEGORY SELECTION PRIORITIES

1. Transportation purpose takes priority over generic mechanical appearance.
2. A subject that primarily flies in an atmosphere, travels on or under water,
   or operates in space must use Aircraft, Watercraft, or Spacecraft rather than
   Vehicle.
3. An embodied mechanical agent is Robot rather than Machine.
4. A complete architectural or spatial environment is Building, while a
   standalone furnishing is Furniture.
5. Use Other only when the subject is concrete but genuinely outside the listed
   categories.

OUTPUT TEMPLATE

Return exactly this JSON structure and do not add any keys:

{
  "description": "...",
  "visible_evidence": "...",
  "subject": null,
  "theme": null,
  "reason": "...",
  "recognizability": "recognizable|unrecognizable",
  "category": null,
  "confidence": "high|medium|low"
}

Replace null values for subject, theme, and category with valid strings only
when the rules above require them.
```

To mitigate occasional instruction-following failures, we validate each MLLM response and use the retry prompt below to correct outputs that violate the required JSON format or schema.

Listing 3: Retry prompt for correcting an invalid MLLM response.

```
{original_question}
Your previous response did not satisfy the required output format or schema.
Validation error: {validation_error}
{optional_correction_hint}
Previous response:
{previous_answer}
Correct the response. Return exactly one JSON object and nothing else. The first
non-whitespace character must be '{' and the last must be '}'. Do not include
markdown or explain your reasoning.
```

Finally, we provide the prompt used to generate segmentation maps from rendered views.

Listing 4: Prompt for segmentation overlay generation.

```
Convert the provided rendered LEGO image into a clean panoptic segmentation map.

Preserve the original image size, camera pose, viewpoint, framing, object layout,
     silhouette, geometry, proportions, and occlusion relationships. Keep the segmentation
     spatially aligned with the input image.

Partition all visible objects and scene regions into coherent semantic regions. Within each
     object, distinguish meaningful visible parts and components. Keep distinct physical
     instances separate. Use visible structure and semantic identity to determine regions,
     and keep the elements of each coherent part together. Segment only visible content; do
     not invent hidden parts or unsupported boundaries.

Assign one distinct, flat, uniform, opaque color to each region. Use a separate uniform
     color for the image background. Keep region boundaries crisp and aligned with the
     corresponding visible boundaries in the input.

Do not add, remove, move, reshape, or complete any object or part. Do not include
     transparency, gradients, shading, shadows, highlights, reflections, textures, outlines,
     anti-aliasing, text, labels, or legends.

Return only the segmentation image.
```

REFERENCES

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9704–9715, October 2025.

Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P^3-SAM: Native 3D part segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2026.

Tencent Hunyuan Foundation Model Team. HunyuanImage 3.0 technical report. *arXiv preprint arXiv:2509.23951v3*, 2026. Version 3, June 26, 2026; first released in 2025.

Zhe Zhu, Le Wan, Rui Xu, Yiheng Zhang, Honghua Chen, Zhiyang Dou, Cheng Lin, Yuan Liu, and Mingqiang Wei. Partsam: A scalable promptable part segmentation model trained on native 3d data. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=y8sZUQPYXC`. Poster.