UNIFIED 3D PARTITIONING FROM POINT PROMPTS

Anonymous authors
Paper under double-blind review

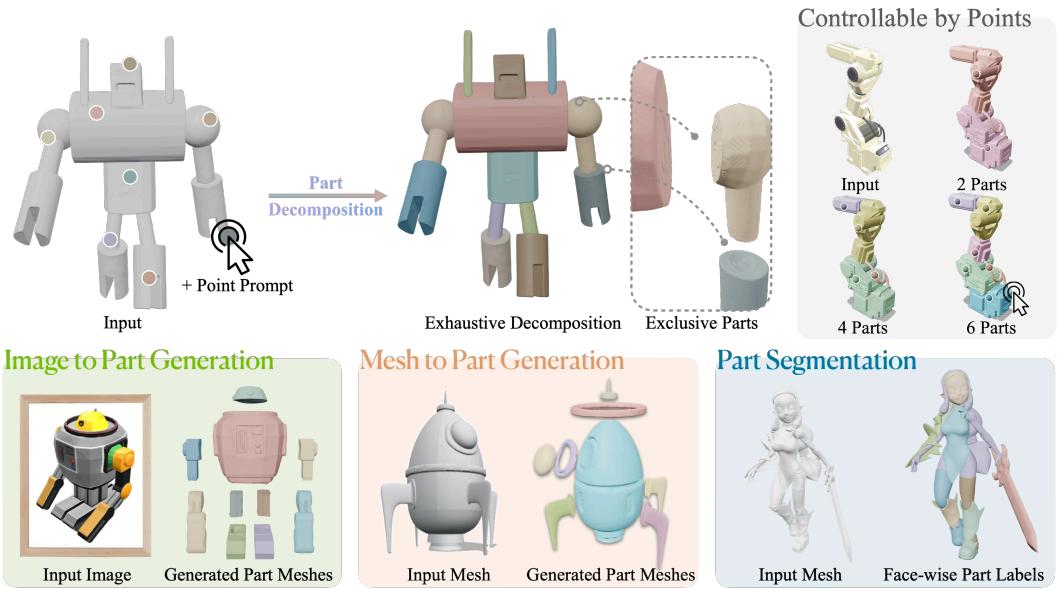

Figure 1: **A unified framework for promptable 3D partitioning.** *(Top)* We support 3D point prompts to specify desired parts and partition the input into closed, exhaustive, and mutually exclusive parts. *(Right)* Users can interactively control the partition through point prompts. *(Bottom)* We support image-to-part generation, mesh-to-part generation, and part segmentation within a single unified model.

ABSTRACT

Existing 3D part decomposition methods do not necessarily partition the original shape into non-overlapping parts that collectively cover the entire shape, allowing overlaps or gaps that hinder downstream part-level applications. We instead formulate part decomposition as a *joint partitioning of the entire shape*, where the predicted parts are non-overlapping and jointly recover the entire shape. Our key insight is that part decomposition should consider all desired parts jointly, rather than modeling each part independently. To this end, we develop a promptable model for 3D part decomposition from images or meshes. Users can specify desired parts through 3D point prompts for controllable decomposition. Given one point prompt per desired part, our model produces the corresponding parts as a complete partition of the entire shape. We build on a pretrained 3D generation model and first obtain a shape latent from either an input image or mesh. We then introduce a prompt encoder that maps each 3D point prompt to a part token while attending to the shape latent. To decode the desired parts, we propose a novel part decoder jointly scoring the entire shape against all part tokens in a coarse-to-fine manner, assigning every position within the shape volume to exactly one part. We perform part decomposition in this shared shape latent space, enabling a unified model for image-to-part generation, mesh-to-part generation, and part segmentation. Our method outperforms existing works on all part-quality metrics across all three tasks, and improves compatibility among parts by an order of magnitude over previous SOTA methods. Code and models will be released.

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.

1 INTRODUCTION

Recent 3D generative models produce high-quality object-level meshes from a text prompt or a single image (Li et al., 2025; Xiang et al., 2026; Hunyuan3D et al., 2025; Chen et al., 2026). However, downstream applications such as part-level editing require decomposing these objects into distinct components (Mo et al., 2019b;a). This has motivated growing interest in part-level 3D generation (Zhu et al., 2026a; Yang et al., 2025). Yet existing part-level methods typically produce a collection of parts that may overlap, without jointly considering whether these parts form a disjoint partition of the entire shape.

Existing part generators (Lin et al., 2025; Tang et al., 2025; Yang et al., 2025; 2026; Yan et al., 2026; Zhu et al., 2026a) may use cross-part attention to model relationships between parts, but still decode each part separately, allowing the same spatial region to be occupied by multiple parts (*i.e.,* penetration) or by no part at all (*i.e.,* gaps). They typically control the desired decomposition through a predefined number of parts, text prompts, or 2D image masks (Zhu et al., 2026a; Yang et al., 2025), offering indirect or limited control. Other methods support prompt-based 3D part segmentation (Ma et al., 2026; Zhou et al., 2025; Su et al., 2026), which segments the mesh surface into parts based on 3D point prompts placed directly on the surface, but they typically predict a separate face mask independently for each prompt. While this provides direct 3D control over the desired parts, the resulting masks often require post-processing to resolve overlaps and are not constrained to jointly form a complete, non-overlapping segmentation of the object. Consequently, both approaches may produce overlapping parts or missing components (see Fig. 3), rather than forming a coherent decomposition of the whole object.

To address this, we formulate part decomposition as *a joint partitioning of the entire shape*, where the resulting parts are *exclusive*, meaning their volumes do not overlap, and *exhaustive*, meaning their union recovers the entire volume of the object (see Eq. (1) for the formal definition). The key to enforcing this structure is to define the decomposition jointly over the whole shape, rather than generating each part independently. Given an input image or mesh, users specify each desired part with a 3D point prompt, and we propose a model that decomposes the whole shape into the prompted parts. Our model builds on a pretrained 3D generation model (Hunyuan3D et al., 2025) and consists of three components: a shape latent backbone, a prompt encoder, and a part decoder. The shape latent backbone represents the input image or mesh as a shape latent capturing the entire geometry, while the prompt encoder maps each point prompt to a corresponding part token while attending to the shape latent. The part decoder then jointly scores the entire shape against all part tokens, assigning every query point within the shape volume to exactly one part. We further refine part boundaries at higher resolutions in a coarse-to-fine manner. This finally yields an exclusive and exhaustive partition by construction.

By representing both image and mesh inputs in a shared shape latent space, our model supports three tasks in a unified way: image-to-part generation, mesh-to-part generation, and part segmentation of mesh surface (see Fig. 1 *bottom*). For generation, the model generates a closed mesh for each prompted part, while for segmentation, it assigns each mesh face to one of the parts. Moreover, our model offers interactive part decomposition, where users can flexibly control the decomposition by adjusting the number and locations of 3D point prompts (see Fig. 1 *right*). We evaluate all three tasks on benchmark datasets (Yang et al., 2024), and our method outperforms state-of-the-art (SOTA) generation and segmentation methods on all part-quality metrics across all three tasks, while improving geometric compatibility between parts by an order of magnitude.

In summary, we make the following contributions:

1. We formulate 3D part decomposition as a joint partition of the entire shape, identifying the overlap and missing components issues of existing part-level methods, yielding an exclusive and exhaustive partition by construction.
2. We introduce a promptable partition design that represents each user-specified part with a part token. Given a 3D point prompt per part, the model jointly assigns every point in the shape to exactly one prompted part, producing a complete partition of the shape.
3. By leveraging a shared shape latent, we support image-to-part generation, mesh-to-part generation, and part segmentation within a unified framework, achieving SOTA performance and substantially improving geometric compatibility between parts.

2 Related Work

Promptable part segmentation. Promptable segmentation was popularized in 2D by SAM (Kirillov et al., 2023), where a click yields a mask, and extended to concept prompts by SAM 3 (Carion et al., 2026). In 3D, Point-SAM (Zhou et al., 2025) brings the click interface to point clouds, P3-SAM (Ma et al., 2026) operates natively on meshes and merges per-click masks by suppression and flood fill, PartSAM (Zhu et al., 2026b) scales the design with a triplane encoder, and S^2AM3D (Su et al., 2026) adds a continuous scale signal. Prompt-free methods cluster a learned feature field at a chosen count (Liu et al., 2025), lift SAM 2 masks from rendered views onto the mesh (Tang et al., 2024; Yang et al., 2024), or repurpose a generative model as a segmenter (Li et al., 2026), and PartObjaverse-Tiny (Yang et al., 2024) is a common benchmark among them. Each prompt is answered independently, so the masks may overlap or leave gaps, and in 3D, a mask is only a patch on the existing surface, not a closed part. In contrast, our model answers all prompts jointly as one partition, and it not only segments the surface but also generates each part as a closed mesh.

Part-level generation. Different from segmentation, part-level generation produces a closed mesh per part from a given image or mesh, building on whole-shape generators (Li et al., 2025; Xiang et al., 2026; Hunyuan3D et al., 2025; Chen et al., 2026; Gu et al., 2026). They differ in what specifies the parts. PartCrafter (Lin et al., 2025) and PartPacker (Tang et al., 2025) take only a part count and generate compositional latents, X-Part (Yan et al., 2026) and FullPart (Ding et al., 2026) take a box layout and generate each part within its box, OmniPart (Yang et al., 2025) lifts 2D masks from the input image, CubePart (Zhu et al., 2026a) takes a list of part names, and HoloPart (Yang et al., 2026) and UniPart (He et al., 2026) work from a segmentation, given to the former and generated jointly with the geometry by the latter. To our knowledge, these methods produce closed geometry, but none constrains one part to stay out of another or reports whether it does, and control is indirect, through counts, boxes, names, or masks. We decode from a single whole-shape field with user points into exclusive and exhaustive parts.

Primitive-based decompositions. An older line represents shapes using cuboids (Tulsiani et al., 2017), superquadrics (Paschalidou et al., 2019), learned convexes (Deng et al., 2020; Chen et al., 2020), or deformed spheres (Paschalidou et al., 2021). These methods approximate shapes with a small set of geometric primitives rather than recovering the part decomposition, so their outputs are not directly comparable under our part-level metrics.

3 Method

3.1 Problem Formulation

Given a 3D object, represented either by a complete mesh or an image, together with user-specified 3D point prompts, we seek to decompose the object into one 3D part per prompt such that the parts form an exclusive and exhaustive partition of the object. We first formulate part decomposition as a partition of the object, and then describe how point prompts specify the desired partition.

Part decomposition as a partition. We represent the object as a solid $\Omega \subset \mathbb{R}^3$ with surface $\partial\Omega$, and its decomposition as N closed solids $\{\Omega_1, \ldots, \Omega_N\}$. For a mesh input, Ω is the solid enclosed by the input mesh; for an image input, it is the solid enclosed by the mesh generated from the image. Each part is represented as a closed solid, and the complete object is recovered by assembling all parts. Let $\mathrm{vol}(\cdot)$ denote the volume of a set. We formally define the decomposition such that distinct parts have no volume overlap (*exclusive*) and their union recovers the entire object (*exhaustive*):

$$\underbrace{\mathrm{vol}(\Omega_i \cap \Omega_j) = 0, \quad \text{for } i \neq j,}_{\textit{exclusive}} \qquad \underbrace{\bigcup_{j=1}^{N} \Omega_j = \Omega\,.}_{\textit{exhaustive}} \tag{1}$$

Point-prompted decomposition. A 3D mesh admits valid decompositions at different granularities. To provide fine-grained and flexible control, we enable users to interactively control the decomposition by directly placing *3D point prompts* on the object Ω, with one prompt for each desired

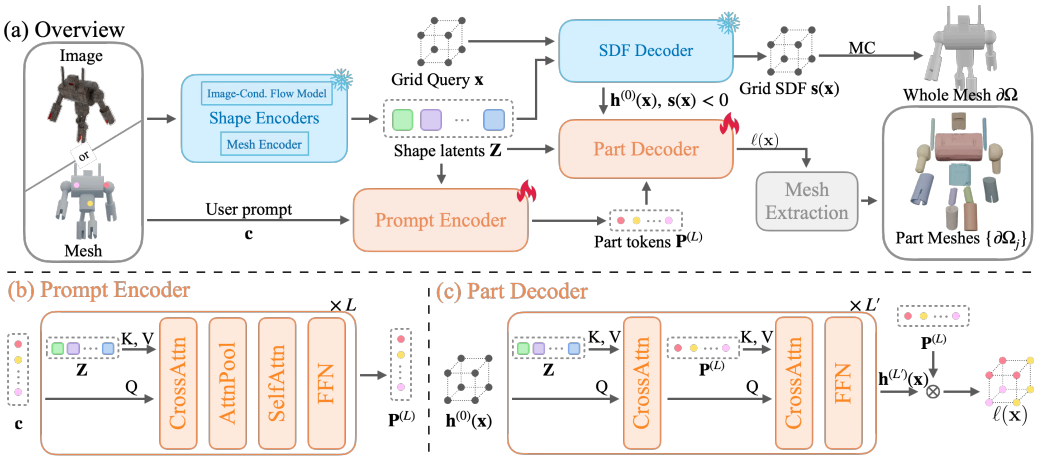

Figure 2: **Method overview.** (a) Given a mesh or an image, a shape encoder first encodes it to shape latents $\mathbf{Z}$; the prompt encoder processes user prompts by attending to shape latents $\mathbf{Z}$ to get part tokens $\mathbf{P}^{(L)}$; the part decoder scores query points $\mathbf{x}$ with the part tokens to predict their part assignments; mesh extraction converts these predictions into exclusive and exhaustive closed part meshes. (b, c) One layer of the prompt encoder and the part decoder, respectively.

part. In contrast to indirect controls such as text (Zhu et al., 2026a), 2D image masks (Yang et al., 2025), or part counts (Lin et al., 2025), 3D point prompts provide finer control over the entire object, including regions that may be occluded in a 2D view. Our model aims to decompose Ω to the prompted parts to produce a partition satisfying Eq. (1). For *generation*, each prompt yields a closed part mesh, while for *segmentation*, each face is assigned a part label corresponding to a prompt.

3.2 Model

Overview. Figure 2 presents our unified model for promptable part generation and segmentation. Given a mesh or an image together with N user prompts, our model generates either N closed part meshes or assigns one part label to each input face for the *segmentation* task. Our model builds on a pretrained latent shape model (Hunyuan3D et al., 2025) and extends it with a prompt encoder and a part decoder. For either input modality (*i.e.*, image or mesh), the backbone provides shape latents $\mathbf{Z}$ in a shared latent space. For image input, the decoded whole mesh is used to place 3D prompts, while the same shape latents $\mathbf{Z}$ are used for decomposition. The prompt encoder processes user prompts by attending to $\mathbf{Z}$, producing one token $\mathbf{P}^{(L)}$ per part, and the part decoder jointly scores each query point $\mathbf{x}$ against all part tokens, producing part assignments $\ell(\mathbf{x})$. These part assignments $\ell(\mathbf{x})$ derive the exclusive and exhaustive output parts Ω_j by mesh extraction.

Shape Backbone. The shape backbone consists of shape encoders and an SDF decoder, as shown in Fig. 2. The shape encoder maps the given image or mesh to shape latents, and the SDF decoder reconstructs a full mesh from shape latents. For either a mesh or an image input, our shape encoders produce M shape latents $\mathbf{Z} \in \mathbb{R}^{M \times C}$ in the same latent space, where C is the channel dimension. For mesh input, $\mathbf{Z}$ is obtained from a mesh encoder (Hunyuan3D et al., 2025); for image input, it is predicted by the image-conditioned flow model (Hunyuan3D et al., 2025). We leverage pretrained alignment between the shape latents of the two inputs and perform the decomposition on the shared shape latent $\mathbf{Z}$, allowing the same prompt encoder and part decoder to operate on either input.

With the encoded shape latent $\mathbf{Z}$, SDF decoder provides this whole mesh $\partial\Omega$ through its signed distance field, *i.e.*, for a spatial location $\mathbf{x} \in \mathbb{R}^3$, its SDF decoder predicts

$$\mathbf{s}(\mathbf{x}) = f_0(\mathrm{CrossAttn}(\phi(\mathbf{x}), \mathbf{Z}))\,, \tag{2}$$

where ϕ is a positional embedding, $\phi(\mathbf{x})$ serves as the query, $\mathbf{Z}$ provides the keys and values, and f_0 is a linear readout. The signed distance is negative inside the shape, defining $\Omega = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{s}(\mathbf{x}) < 0\}$. For the subsequent decomposition, we initialize the query feature at each interior location $\mathbf{x} \in \Omega$ as $\mathbf{h}^{(0)}(\mathbf{x}) = \phi(\mathbf{x}) \in \mathbb{R}^C$. A mesh is recovered by evaluating $\mathbf{s}$ on a regular grid and extracting the zero level set $\partial\Omega$ with marching cubes (Lorensen & Cline, 1987).

Prompt Encoder. The prompt encoder takes point prompts $\mathbf{c}$ for N parts and predicts one part token $\mathbf{P}^{(L)} \in \mathbb{R}^{N\times C}$, corresponding to each specified part by attending to shape latent and inter-part relationships. Figure 2(b) illustrates the prompt encoder. A point prompt for each part consists of K points for each j-th part for a total of N parts. Each point is embedded independently as

$$\mathbf{P}^{(0)} = \big[\phi(\mathbf{c}_{j,k})\big]_{j\le N,\ k\le K} \in \mathbb{R}^{N\times K\times C}, \tag{3}$$

where ϕ is the backbone positional embedding. A user prompt built from positional embeddings alone carries no information about the shape or about the other prompts. The prompt encoder supplies both, letting each point prompt reference the asset through $\mathbf{Z}$ and relate to the other prompts. Its first layer lets every point attend to $\mathbf{Z}$, aggregate the K tokens of each part by attention pooling, and then lets the parts attend to one another. The remaining layers repeat both steps,

$$\begin{aligned}\mathbf{P}^{(1)} &= \mathrm{FFN}\Big(\mathrm{SelfAttn}\big(\mathrm{Pool}_k\,\mathrm{CrossAttn}(\mathbf{P}^{(0)},\mathbf{Z})\big)\Big),\\ \mathbf{P}^{(l+1)} &= \mathrm{FFN}\Big(\mathrm{SelfAttn}\big(\mathrm{CrossAttn}(\mathbf{P}^{(l)},\mathbf{Z})\big)\Big), \qquad l=1,\dots,L-1,\end{aligned} \tag{4}$$

where Pool_k aggregates the K point tokens of each part, and $\mathbf{P}^{(l)} \in \mathbb{R}^{N\times C}$ for $l \ge 1$. Pooling after cross-attention lets each point gather local geometry before forming one token per part.

Part Decoder. Given the part tokens $\mathbf{P}^{(L)}$ from the prompt encoder, shape latents $\mathbf{Z}$, and an interior query point $\mathbf{x} \in \Omega$, the part decoder produces one part assignment per part by relating $\mathbf{x}$ to the local geometry in $\mathbf{Z}$ and to all prompted parts. These are collected as $\ell(\mathbf{x}) \in \mathbb{R}^N$, indicating how strongly $\mathbf{x}$ belongs to each prompted part, as illustrated in Figure 2(c). Starting from $\mathbf{h}^{(0)}(\mathbf{x}) = \phi(\mathbf{x})$, the decoder refines the query feature through L' layers. Each layer first attends to $\mathbf{Z}$ to gather local geometry and then to $\mathbf{P}^{(L)}$ to incorporate part-specific information,

$$\mathbf{h}^{(l+1)}(\mathbf{x}) = \mathrm{FFN}\Big(\mathrm{CrossAttn}\big(\mathrm{CrossAttn}(\mathbf{h}^{(l)}(\mathbf{x}),\mathbf{Z}),\mathbf{P}^{(L)}\big)\Big), \quad l=0,\dots,L'-1, \tag{5}$$

where $\mathbf{x} \in \Omega$. The part assignments are dot products between $\mathbf{h}^{(L')}$ and part tokens $\mathbf{P}^{(L)}$,

$$\ell_j(\mathbf{x}) = \frac{1}{\sqrt{C}}\big\langle f_h(\mathbf{h}^{(L')}(\mathbf{x})), f_s(\mathbf{P}_j^{(L)})\big\rangle, \quad j=1,\dots,N, \quad \mathbf{x}\in\Omega, \tag{6}$$

where f_h and f_s are learned projections, and each part token acts as a dynamic linear classifier for its part, as in Mask2Former (Cheng et al., 2022) but on interior points. One pass over the queries decodes all N parts, so the cost is set by the number of queries and grows only slightly with N.

Mesh Extraction. Before decoding $\ell(\mathbf{x})$ into part meshes, note that although the backbone of Eq. (2) regresses a signed distance, and mesh extraction uses only its zero level set, and hence only the sign of $\mathbf{s}$, a binary decision at every point, inside or outside. Extracting parts therefore needs one further decision on the interior Ω, and Eq. (1) fixes its form: exclusive and exhaustive together say that every point of Ω is assigned to exactly one part, that is, a single-valued map $\pi : \Omega \to \{1,\dots,N\}$ defined on all of Ω. Any such map satisfies Eq. (1), since

$$\sum_{j=1}^{N} \mathbf{1}[\pi(\mathbf{x}) = j] = 1 \quad \forall\,\mathbf{x}\in\Omega, \tag{7}$$

which implies the exclusivity and exhaustiveness conditions in Eq. (1). We take π to be the argmax of the part assignments of Eq. (6), the part with the largest score at each point,

$$\pi(\mathbf{x}) = \operatorname*{arg\,max}_{j\in\{1,\dots,N\}} \ell_j(\mathbf{x}), \qquad \Omega_j = \{\mathbf{x}\in\Omega : \pi(\mathbf{x}) = j\}. \tag{8}$$

Since Eq. (1) follows from the form of π and not from its values, it holds whatever the model predicts, without using suppression, flood filling, or an overlap penalty. The part meshes follow from the same field. Marching cubes (Lorensen & Cline, 1987) extracts the zero level set of a field, the surface between its negative and non-negative regions, so a closed mesh of part j needs a field that is negative exactly on Ω_j. We obtain it from the whole field by keeping $\mathbf{s}$ where $\pi = j$ and discarding its sign elsewhere,

$$\mathbf{s}_j(\mathbf{x}) = \begin{cases}\mathbf{s}(\mathbf{x}) & \pi(\mathbf{x}) = j,\\ |\mathbf{s}(\mathbf{x})| & \text{otherwise},\end{cases} \tag{9}$$

so that $\mathbf{s}_j < 0$ exactly on Ω_j, since outside the object $\mathbf{s}$ is already non-negative and inside the other parts the sign is dropped. Evaluating $\mathbf{s}_j$ on the grid that covers Ω and running marching cubes returns one closed mesh per part, whose faces on $\partial\Omega$ coincide with the whole and whose faces where Ω_j meets another part are the cut surfaces that close it.

Supervision and sampling. We train the model to assign each location inside the whole to a single GT part. To support a flexible number of points per prompt, during training we sample $K \sim \mathcal{U}\{1, \ldots, 4\}$ surface points $\{\mathbf{c}_{j,k}\}_{k=1}^{K}$ for each part prompt from the j-th GT part. Unless otherwise specified, we use $K = 1$ at inference. Given the signed distance $d_j(\mathbf{x})$ to each annotated part, we define the target assignment as

$$y(\mathbf{x}) = \underset{j \in \{1,\ldots,N\}}{\arg\min}\ d_j(\mathbf{x}), \qquad \mathbf{x} \in \Omega. \tag{10}$$

This assigns a unique label to interior locations, resolving overlaps in the annotations. We concentrate training queries near part boundaries, where the assignment is most ambiguous, while also sampling the shape interior and regions on and near its surface.

Training objective. We train the model to predict the prompted part assignment for each interior query, while encouraging part tokens from the same part to be consistent and query features to be discriminative. We optimize

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{dice}}\,\mathcal{L}_{\text{dice}} + \lambda_{\text{cons}}\,\mathcal{L}_{\text{cons}}, \tag{11}$$

where $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{dice}}$ supervise the predicted partition, and $\mathcal{L}_{\text{cons}}$ encourages consistency across prompts for the same part. Since each interior location is assigned to the part with the highest score, we formulate $\mathcal{L}_{\text{cls}}$ as an N-way classification loss. Let $p_j(\mathbf{x}) = \mathrm{softmax}_j\, \ell(\mathbf{x})$ denote the predicted probability of assigning $\mathbf{x}$ to part j. Given the target assignments $y(\mathbf{x})$, we apply a focal cross-entropy (Lin et al., 2017) over the labeled training queries $\mathcal{X} \subset \Omega$,

$$\mathcal{L}_{\text{cls}} = -\frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} w_{y(\mathbf{x})} \left(1 - p_{y(\mathbf{x})}(\mathbf{x})\right)^{\gamma} \log p_{y(\mathbf{x})}(\mathbf{x}), \tag{12}$$

where w_j balances parts of different sizes and γ reduces the contribution of easy queries, emphasizing locations whose assignment remains uncertain. To complement pointwise classification with part-level region supervision, we add a soft Dice loss (Milletari et al., 2016),

$$\mathcal{L}_{\text{dice}} = \frac{1}{N} \sum_{j=1}^{N} \left(1 - \frac{2 \sum_{\mathbf{x} \in \mathcal{X}} p_j(\mathbf{x})\, \mathbf{1}[y(\mathbf{x}) = j] + 1}{\sum_{\mathbf{x} \in \mathcal{X}} p_j(\mathbf{x}) + \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{1}[y(\mathbf{x}) = j] + 1}\right). \tag{13}$$

A part token should represent the intended part rather than the particular surface locations used to specify it. We therefore encode two independently sampled prompt sets for the same ground-truth parts, obtaining tokens $\mathbf{P}^{(L)}$ and $\mathbf{P}'^{(L)}$, and encourage their agreement,

$$\mathcal{L}_{\text{cons}} = \frac{1}{N} \sum_{j=1}^{N} \left\| \mathbf{P}_j^{(L)} - \mathbf{P}_j'^{(L)} \right\|_2^2. \tag{14}$$

3.3 Inference

Generation task. We support both mesh and image inputs for part generation. For either input, the predicted partition is converted into N closed part meshes using Eq. (9) followed by per-part marching cubes. For image input, prompts are placed on the generated mesh rather than on the image, so that parts on the back of the object, which the image does not show, can also be assigned.

Following octree-based isosurface extraction (Meagher, 1982; Wilhelms & Van Gelder, 1992) and its adaptation to neural shape decoding in FlashVDM (Lai et al., 2025), we perform coarse-to-fine refinement of both the whole-shape and part SDFs. The whole-shape SDF is refined within a narrow surface band, defined by $|s(\mathbf{x})| < \eta$ for a chosen threshold η. The part SDFs are refined only where $s(\mathbf{x}) < \eta$ and the two highest part-assignment scores, $\ell_{(1)}(\mathbf{x})$ and $\ell_{(2)}(\mathbf{x})$, differ by less than a margin δ:

$$\mathcal{R}^{\text{whole}} = \{\mathbf{x} : |s(\mathbf{x})| < \eta\}, \qquad \mathcal{R}^{\text{part}} = \{\mathbf{x} : s(\mathbf{x}) < \eta \text{ and } \ell_{(1)}(\mathbf{x}) - \ell_{(2)}(\mathbf{x}) < \delta\}. \tag{15}$$

Elsewhere, the fine grid inherits the coarse value. The N part meshes are cut from the one field, and their interiors $\Omega_1, \ldots, \Omega_N$ reassemble Ω, a whole from which none of its parts is absent.

Segmentation task. We support promptable part segmentation for mesh input. Given a mesh and one point prompt per part, we query the decoder at each face centroid $\mathbf{x}_f$ instead of $\mathbf{x} \in \Omega$ and assign the face the predicted part label $\pi(\mathbf{x}_f)$, leaving the input mesh geometry unchanged.

4 EXPERIMENTS

4.1 SETUP

Baselines. We compare against SOTA methods for part generation (Zhu et al., 2026a; Yang et al., 2026; Yan et al., 2026; Lin et al., 2025; Tang et al., 2025; Yang et al., 2025) and part segmentation (Li et al., 2026; Zhou et al., 2025; Zhu et al., 2026b; Su et al., 2026; Ma et al., 2026; Liu et al., 2025). To keep the comparison as consistent as possible, we run all baselines using their official checkpoints and inference settings. As these methods differ in inputs and training data, with some relying on private training data (Liu et al., 2025), our quantitative results serve as a proof-of-concept comparison under their respective released settings. Generation methods (Lin et al., 2025; Tang et al., 2025) requiring a part count receive the GT part count; CubePart (Zhu et al., 2026a) and OmniPart (Yang et al., 2025) receive part names and 2D segmentation masks, respectively. Unless stated otherwise, promptable methods (Su et al., 2026) receive one point per GT part (*i.e.*, $K{=}1$), sampled uniformly from the corresponding part surface where it does not overlap with other parts. For image input, the point is sampled on the generated mesh surface after aligning the generated mesh to the GT mesh. Please refer to Appendix B for detailed baseline settings.

Datasets. We train on the public dataset HY3D-Bench (Hunyuan3D et al., 2026), which has 240K assets with part-level labels. The baselines were trained on their own datasets of comparable or larger size, almost all of which are not released. We evaluate on PartObjaverse-Tiny (Yang et al., 2024), a widely used benchmark for 3D part decomposition, with 200 meshes and face-level part-instance labels, and we remove any assets from the HY3D-Bench training set that overlap with PartObjaverse-Tiny. We use the same assets to evaluate all three tasks: a mesh for segmentation and mesh-conditioned generation, and a single rendered view for image-conditioned generation.

Metrics. We evaluate three aspects of the decomposition: part quality, whole-shape geometry, and part compatibility. For part quality, *pCD* and *pF1@*τ measure matched-part Chamfer distance and surface F1; for segmentation, we additionally report face-level *mIoU*, where face correspondence is available. For whole-shape geometry, *CD* and *F1@.05* compare the union of generated parts with the ground-truth whole. For part compatibility, we report inter-part penetration (*pen%*) and the fraction of closed parts (*wt%*); *pen%* measures violations of exclusivity, while *pCD* and *pF1* also reflect failures of exhaustiveness (see Appendix B). Distances are reported at 10^{-2} scale in the normalized unit cube and averaged over assets. *Inference time* is measured per asset on one A100, and image predictions are ICP-aligned before evaluation.

Implementation details. We train the HY3D-2.1 model (Hunyuan3D et al., 2025), and also provide an ablation on using TripoSG (Li et al., 2025) as backbone, which has similar architecture to HY3D-2.1; Please refer to Appendix A for detailed architectures, sampling budgets, loss, *etc*.

4.2 MAIN RESULTS

Part generation. Table 1 compares our method with prior methods for 3D part generation from mesh (top) and image (bottom) inputs. Our method achieves the best part quality for both inputs and reduces inter-part penetration by over an order of magnitude, demonstrating that our decomposition results are exclusive. The small residual penetration comes from marching-cubes interpolation at part seams (Appendix B). Whole-shape fidelity is bounded by the backbone's reconstruction of the input. Our method is the fastest from a mesh and second fastest from an image, since all parts are decoded jointly in one pass. Figure 3 visualizes the same behavior, further comparisons in Appendix D.

Table 1: **Comparison of our method with SOTA methods for 3D part generation from a mesh (top) and an image (bottom).** We evaluate part decomposition quality, part compatibility, whole-shape geometry, and inference time.

	Part quality			Part compatibility		Whole geometry		Inference
Method	pCD↓	pF1@.01↑	pF1@.05↑	pen%↓	wt%↑	CD↓	F1@.05↑	time (s)↓
Mesh input								
CubePart (Zhu et al., 2026a)	4.71	51.5	72.5	2.09	**100.0**	1.21	**97.5**	29.1
HoloPart (Yang et al., 2026)	5.29	46.6	73.9	1.01	32.6	1.54	95.0	99.1
X-Part (Yan et al., 2026)	4.53	52.5	73.4	3.05	90.8	**1.19**	97.3	147.2
Ours	**2.73**	**57.0**	**84.4**	**0.06**	**100.0**	1.56	94.1	**24.4**
Image input								
PartCrafter (Lin et al., 2025)	12.75	6.3	29.6	2.45	70.5	3.76	76.9	84.8
PartPacker (Tang et al., 2025)	8.11	17.6	54.2	1.25	86.0	2.22	91.7	**26.7**
OmniPart (Yang et al., 2025)	6.43	23.9	60.8	0.96	94.7	2.20	91.4	73.3
Ours	**5.34**	**27.2**	**68.6**	**0.01**	**100.0**	**2.00**	**93.3**	37.8

Table 2: **Comparison of our method with SOTA methods for 3D part segmentation.** We evaluate part decomposition quality on face mIoU and on part-level CD and F1, and inference time.

	Part quality				Inference
Method	mIoU↑	pCD↓	pF1@.01↑	pF1@.05↑	time (s)↓
SegviGen (Li et al., 2026)	20.50	10.05	22.8	40.3	317.9
Point-SAM (Zhou et al., 2025)	39.12	5.37	47.7	68.7	1.2
PartSAM (Zhu et al., 2026b)	50.31	3.24	63.4	79.2	36.6
S^2AM3D (Su et al., 2026)	50.43	3.16	62.3	80.5	**0.3**
P3-SAM (Ma et al., 2026)	54.66	4.33	61.0	74.8	19.1
PartField (Liu et al., 2025)	69.10	5.31	62.3	72.5	1.1
Ours	**69.80**	**2.10**	**78.3**	**87.0**	**0.3**

Table 3: **Model ablations.** We evaluate part generation from a mesh with the part-quality metrics.

Ablation	pCD↓	pF1@.01↑	pF1@.05↑	pen%↓
(a) w/o prompt enc.	4.24	50.4	75.2	0.07
(b) w/o $\mathcal{L}_{\text{cons}}$	3.11	53.1	80.8	0.06
(c) w/o refine.	2.86	54.6	83.9	**0.05**
(d) Indep. assign.	2.90	54.9	83.2	1.23
Ours	**2.73**	**57.0**	**84.4**	0.06

Table 4: **Ablation on the number of prompt points K.** We evaluate part segmentation with the metrics of Table 2.

Backbone	K	mIoU↑	pCD↓	pF1@.01↑	pF1@.05↑
TripoSG	1	67.40	2.41	75.6	85.3
	4	72.68	1.53	81.6	90.3
HY3D-2.1	1	69.80	2.10	78.3	87.0
	4	**74.84**	**1.35**	**84.0**	**91.8**

Part segmentation. Table 2 shows that our model leads on every metric of the segmentation task, with the largest gain in pF1 at the tight threshold, meaning our boundaries fall close to the ground-truth boundaries rather than merely in the right region. It is also the fastest, tied with S^2AM3D. Figure 3 shows the same on individual assets, and Appendix D provides further comparisons.

4.3 Ablations

Effect of Modules. Table 3 evaluates each component of our model. (a) Removing the prompt encoder causes the largest drop in part quality, showing the importance of letting prompts attend to the shape and other parts. (b) Removing $\mathcal{L}_{\text{cons}}$ produces a smaller but consistent drop, as prompts on the same part may yield different boundaries. (c) Removing coarse-to-fine refinement mainly affects tight F1, indicating that refinement corrects small boundary and seam errors. (d) Independent assignment replaces the joint argmax in Eq. (8) with independent binary assignments from the per-part logits in Eq. (6), causing substantial inter-part penetration despite similar part quality.

Number of prompt points. Table 4 shows the effect of the number of points per part across two backbones. Across both backbones, every metric improves as more points are added. A single point indicates where a part is, while additional points reveal its extent. Appendix C reports the full sweep.

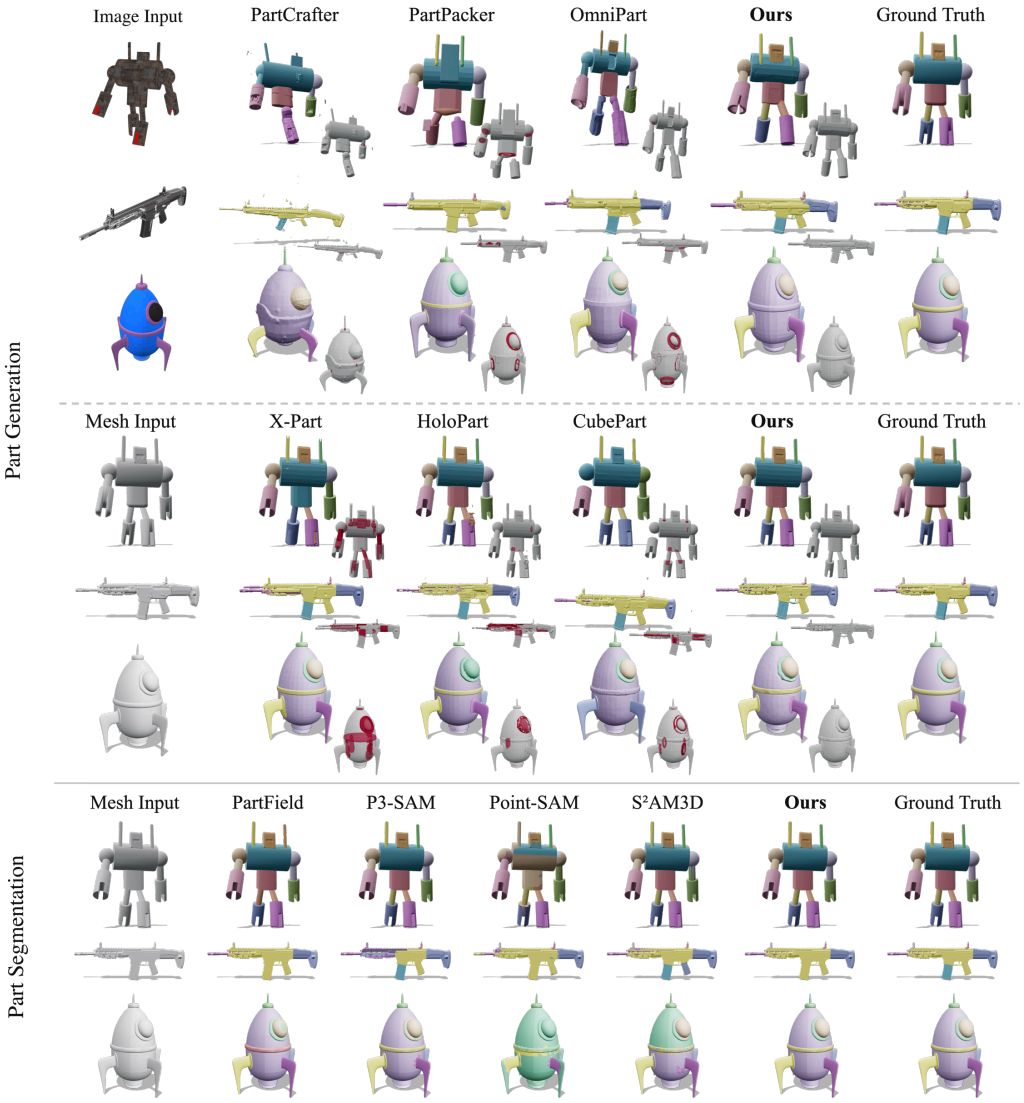

Figure 3: **Qualitative comparison of our method with SOTA methods on the three tasks.** Part generation from an image (top) and from a mesh (middle), and part segmentation (bottom). The grey mesh beside each generated result marks in red the volume claimed by more than one part. Prior methods overlap, drop, split, or merge parts, while ours matches the ground truth.

Limitations. Our method faces challenges when decomposing extremely thin structures or large open surfaces (Appendix Figure 11), which may not enclose a well-defined volume and are therefore difficult to represent as solid parts. This limitation is partly due to the pretrained backbone, which struggles to represent and reconstruct such thin and open geometry. Improving the backbone's representation of these geometries could extend our method to a broader range of shapes.

5 Conclusion

We introduce a unified promptable model for 3D part generation and segmentation that formulates part decomposition as a partition of the whole rather than a collection of independently generated parts. By jointly assigning the whole among prompted parts, our model produces an exclusive and exhaustive partition by construction. A shape can admit multiple valid decompositions; point prompts let users directly specify and control the desired decomposition. A single model supports generation from meshes or images, as well as mesh segmentation, achieving state-of-the-art part quality while reducing part geometric incompatibility by over an order of magnitude.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, to polish the wording of text drafted by the authors. We have not used generative AI tools for research ideation or for writing code and running experiments, and data collection and annotation are not applicable to this work. Additionally, we used generative AI tools for checking grammar and consistency of notation across sections. We have reviewed all AI-assisted work. Every AI-edited sentence was compared against the authors' draft for fidelity of meaning, and all claims, numbers, and citations were written and verified 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.

ETHICS STATEMENT

This work involves no human subjects and no personal data. Training uses HY3D-Bench (Hunyuan3D et al., 2026) (CC BY 4.0) and evaluation uses PartObjaverse-Tiny (Yang et al., 2024) (CC BY-NC 4.0) and PartNeXt (Wang et al., 2025) (CC BY 4.0), all public 3D datasets, which we use under those licenses without redistribution.

REPRODUCIBILITY STATEMENT

The model, losses, and inference procedure are specified in Sec. 3, with every architecture size, sampling budget, loss weight, and schedule listed in Appendix A. The evaluation protocol, including the interpenetration and watertightness tests, the alignment for image input, and the inputs given to each baseline, is in Appendix B. We also evaluate the reproducibility of Table 2 in Appendix C. We will release the code, trained checkpoints, the filtered asset list, and the evaluation scripts.

REFERENCES

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics*, 2018.

P.J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

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, Francois 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 *International Conference on Learning Representations*, 2026.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. SAM 3D: 3Dfy anything in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Lihe Ding, Shaocong Dong, Yaokun Li, Chenjian Gao, Xiao Chen, Rui Han, Yihao Kuang, Hong Zhang, Bo Huang, Zhanpeng Huang, Zibin Wang, Dan Xu, and Tianfan Xue. FullPart: Generating each 3D part at full resolution. In *International Conference on Learning Representations*, 2026.

Diandian Gu, Jing Lin, Gaohong Liu, Jiahang Liu, Su Ma, Guang Shi, Jun Wang, Qinlong Wang, Qianyi Wu, Zhongcong Xu, Xuanyu Yi, Zihao Yu, Jianfeng Zhang, Zhuolin Zheng, Yifan Zhu, Rui Chen, Hengkai Guo, Xiaoyang Guo, Mingcong Han, Xu Han, Xiu Li, Yixun Liang, Weiqiang Lou, Junzhe Lu, Guan Luo, Minghan Qin, Shuguang Wang, and Yuang Wang. Seed3D 2.0: Advancing high-fidelity simulation-ready 3D content generation. *arXiv preprint arXiv:2605.13862*, 2026.

Xufan He, Yushuang Wu, Xiaoyang Guo, Chongjie Ye, Jiaqing Zhou, Tianlei Hu, Xiaoguang Han, and Dong Du. UniPart: Part-level 3D generation with unified 3D geom-seg latents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, Qingxiang Lin, Zeqiang Lai, Xianghui Yang, Huiwen Shi, Zibo Zhao, Bowen Zhang, Hongyu Yan, Lifu Wang, Sicong Liu, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Dongyuan Guo, Junlin Yu, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Shida Wei, Chao Zhang, Yonghao Tan, Yifu Sun, Lin Niu, Shirui Huang, Bojian Zheng, Shu Liu, Shilin Chen, Xiang Yuan, Xiaofeng Yang, Kai Liu, Jianchen Zhu, Peng Chen, Tian Liu, Di Wang, Yuhong Liu, Linus, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3D 2.1: From images to high-fidelity 3D assets with production-ready PBR material. *arXiv preprint arXiv:2506.15442*, 2025.

Team Hunyuan3D, Bowen Zhang, Chunchao Guo, Dongyuan Guo, Haolin Liu, Hongyu Yan, Huiwen Shi, Jiaao Yu, Jiachen Xu, Jingwei Huang, Kunhong Li, Lifu Wang, Linus, Penghao Wang, Qingxiang Lin, Ruining Tang, Xianghui Yang, Yang Li, Yirui Guan, Yunfei Zhao, Yunhan Yang, Zeqiang Lai, Zhihao Liang, and Zibo Zhao. HY3D-Bench: Generation of 3D assets. *arXiv preprint arXiv:2602.03907*, 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*, 2023.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Fuyun Wang, Huiwen Shi, Xianghui Yang, Qingxiang Lin, Jingwei Huang, Yuhong Liu, Jie Jiang, Chunchao Guo, and Xiangyu Yue. Unleashing vecset diffusion model for fast shape generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Lin Li, Haoran Feng, Zehuan Huang, Haohua Chen, Wenbo Nie, Shaohua Hou, Keqing Fan, Pan Hu, Sheng Wang, Buyu Li, and Lu Sheng. SegviGen: Repurposing 3D generative model for part segmentation. *ACM Transactions on Graphics*, 2026.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, and Yan-Pei Cao. TripoSG: High-fidelity 3D shape synthesis using large-scale rectified flow models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017.

Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Yiqiang Feng, Yadong Mu, and Katerina Fragkiadaki. PartCrafter: Structured 3D mesh generation via compositional latent diffusion transformers. In *Advances in Neural Information Processing Systems*, 2025.

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*, 2025.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 1987.

Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-SAM: Native 3D part segmentation. In *Proceedings of the European Conference on Computer Vision*, 2026.

Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 1982.

Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *International Conference on 3D Vision (3DV)*, 2016.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Transactions on Graphics*, 2019a.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019b.

Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3D shape abstractions with invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

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*, 2026.

George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh: Zero-shot mesh part segmentation via lifting segment anything 2 to 3D. *arXiv preprint arXiv:2408.13679*, 2024.

Jiaxiang Tang, Ruijie Lu, Max Li, Zekun Hao, Xuan Li, Fangyin Wei, Shuran Song, Gang Zeng, Ming-Yu Liu, and Tsung-Yi Lin. Efficient part-level 3D object generation via dual volume packing. In *Advances in Neural Information Processing Systems*, 2025.

Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. PartNeXt: A next-generation dataset for fine-grained and hierarchical 3D part understanding. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 1992.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3D generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Xinhao Yan, Jiachen Xu, Yang Li, Changfeng Ma, Yunhan Yang, Chunshi Wang, Zibo Zhao, Zeqiang Lai, Yunfei Zhao, Zhuo Chen, and Chunchao Guo. X-Part: High fidelity and structure coherent shape decomposition and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

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.

Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, Zi-Xin Zou, Yukun Huang, Ying-Tian Liu, Hao Xu, Ding Liang, Yan-Pei Cao, and Xihui Liu. OmniPart: Part-aware 3D generation with semantic decoupling and structural cohesion. In *ACM SIGGRAPH Asia Conference Papers*, 2025.

Yunhan Yang, Yuan-Chen Guo, Yukun Huang, Zi-Xin Zou, Zhipeng Yu, Yangguang Li, Yan-Pei Cao, and Xihui Liu. HoloPart: Generative 3D part amodal segmentation. In *International Conference on Learning Representations*, 2026.

Yuchen Zhou, Jiayuan Gu, Tung Yen Chiang, Fanbo Xiang, and Hao Su. Point-SAM: Promptable 3D segmentation model for point clouds. In *International Conference on Learning Representations*, 2025.

Yiheng Zhu, Kangle Deng, Jean-Philippe Fauconnier, Inaki Navarro, Daiqing Li, Ava Pun, Yinan Zhang, Peiye Zhuang, Xiaoxia Sun, Maneesh Agrawala, Kiran Bhat, and Tinghui Zhou. CubePart: An open-vocabulary part-controllable 3D generator. In *ACM SIGGRAPH Conference Papers*, 2026a.

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*, 2026b.

In this appendix, we provide the architecture, sampling, and optimization details in Appendix A, the evaluation protocol including the interpenetration and watertightness tests in Appendix B, additional results on the number of parts, the number of prompt points, inference time, and a second benchmark in Appendix C, and additional qualitative comparisons, exploded views, control of the decomposition, and failure cases in Appendix D.

A Implementation and Training Details

Architecture. Table 5 lists every hyperparameter of the model reported in the main text. The cross-attention to $\mathbf{Z}$ in each decoder layer starts from the backbone's readout; every other added weight is trained from scratch. Three of them start at zero so that training begins from the backbone's own behavior: the output projection of each added residual branch, so the whole-shape field is unchanged at initialization; the projection f_s, so all N logits are equal; and the attention-pooling query in the first encoder layer, so pooling begins as the mean of the K tokens.

Table 5: **Hyperparameters of our model.** Symbols follow the notation of Sec. 3, and – marks a value with no symbol in the text.

	Hyperparameter	Symbol	Value
Architecture	Shape Latents (TripoSG / Hunyuan3D)	M	2048 / 4096
	Channels	C	1024
	Surface samples per shape	–	20,480
	Prompt encoder layers	L	2
	Part decoder layers	L'	3
	Attention heads	–	8
Training	Points per prompt	K	$\mathcal{U}\{1,\dots,4\}$
	Queries per asset	–	5376
	surface / near surface / bounding box	–	1024 / 1024 / 512
	part surface / inside / part boundary	–	768 / 1024 / 1024
	Focal exponent	γ	2
	Loss weights	$\lambda_{\text{dice}}, \lambda_{\text{cons}}$	1, 0.5
	Batch size, epochs	–	256, 10
	Learning rate (base / readout)	–	2.4×10^{-4} / 4×10^{-4}
	Schedule (warm-up steps, decay fraction)	–	150, 0.2
	EMA decay	–	0.999
Inference	Coarse / fine grid	–	128^3 / 512^3
	narrow-band	η	0.05
	margin	δ	0.2
	Points per prompt	K	1

Query sampling. Training samples its queries instead of evaluating the field on a grid, since most grid cells lie far from any surface or seam, where the loss is already zero. We draw the queries of each asset from the pools of Sec. 3.2, which allocate most of the budget to the surface and the seams between parts, while keeping a sparse, uniform pool for the rest of the bounding box.

Figure 4 shows the results for four assets; the leftmost panel shows the queries crowding the seams. The prompt points, one to four per part, are drawn on each part's surface where they do not overlap any other part. Two prompt sets are drawn per asset, the balls and the cubes in the figure, and the consistency loss $\mathcal{L}_{\text{cons}}$ links their partitions.

Training implementation details. Training runs on four A100 80GB GPUs for 3 days. The learning rate follows a warm-up, hold, and decay schedule, rising over the first steps, staying constant for most of training, and decaying over the final fifth. We keep an exponential moving average of the trained weights and use the averaged weights at inference, which removes the step-to-step variation of the raw weights without changing the model. The base rate and the averaging constant are listed in Table 5.

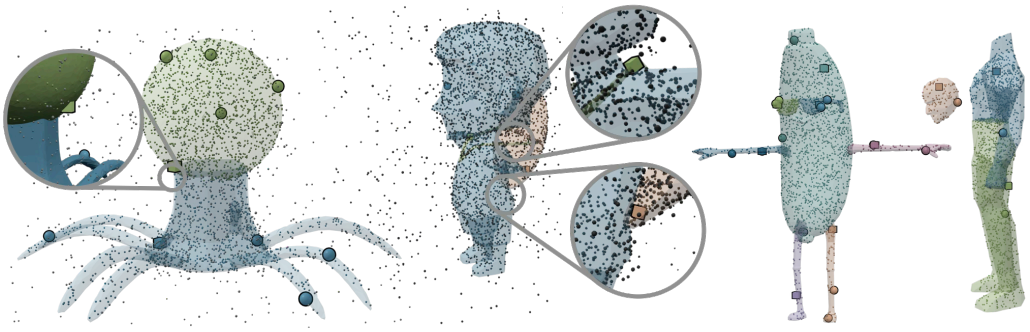

Figure 4: **Training query sampling and prompt points.** Small points are the sampled queries, colored by part; the larger markers are the prompt points, with balls and cubes marking two prompt sets for the same parts.

B METRIC DETAILS

Baselines. Every baseline runs its released checkpoint with its default settings. The one exception is PartField: it evaluates 20 cluster counts per asset and selects the one with the highest mIoU against the ground truth. We instead use the annotated number of parts as a fixed count. FullPart (Ding et al., 2026), and UniPart (He et al., 2026) are discussed in Sec. 2 but not compared, since their inference code has not been fully released.

Interpenetration. We sample $2{,}000$ points on the surface of every generated part with a fixed seed. A probe is kept only if it is still inside its own part under the fast-winding-number sign (Barill et al., 2018); a probe that passed through a thin part tells nothing about that part, and without this guard 99.98 percent of the flagged probes on a test set were of this kind. Writing Q_j for the surviving probes of part j and Ω_k for the solid bounded by part k, a probe counts as penetrating if it lies inside any other part by the same test, which involves no grid or resolution, and

$$\mathrm{pen} = \frac{\sum_j \#\{\mathbf{q} \in Q_j : \exists\, k \neq j,\ \mathbf{q} \in \Omega_k\}}{\sum_j |Q_j|}, \tag{16}$$

in percent, per object and averaged over the benchmark. Our parts partition one volume, so the residual we report is marching-cubes interpolation across a seam within one cell, whereas methods that decode each part as an independent field report interpenetration proper; for methods that output open shells the winding number is fractional and the value is approximate.

Watertightness. *wt%* is the share of generated parts whose mesh is closed, every edge shared by exactly two faces, per object and averaged over the benchmark. A part that is not closed has no interior, so it cannot be moved or replaced as a solid, and a decomposition whose parts are not all closed does not partition the whole.

Alignment for image input. A shape generated from an image lives in the generator's own frame, so image-input methods are aligned to the ground truth before scoring, while mesh-input methods use the identity. Both meshes are normalized by their bounding box, centered and scaled so that the longest side is one, and rigid ICP (Besl & McKay, 1992) on 10K surface points per side is run from 24 axis-aligned initial poses, without scale or reflection, keeping the transform of lowest cost. The same transform is applied to the whole, to every part, and to the part-mesh vertices; the ground truth is only normalized.

Matching and face labels. Generated parts are matched to ground-truth parts one to one by the Hungarian algorithm on the part Chamfer distance, and pCD and pF1 are computed on the matched pairs. For mIoU, each face of the ground-truth mesh takes the label of the generated part that projects onto it, which gives a face labelling for generated parts and for predicted labels alike.

C Additional Results

Number of parts. Figure 5 bins the benchmark assets by their number of ground-truth parts and plots part quality per bin. From a mesh, every generator loses accuracy as the part count grows, whereas ours stays close to its level on the simplest objects, so the margin is smallest on objects with a handful of parts and largest on those with more than twenty. From an image, the ordering is the same, and OmniPart, the closest baseline on few parts, falls below PartPacker on the objects with the most parts. A partition does not become harder to keep exclusive and exhaustive as N grows, since every point is still assigned once, and the figure shows that accuracy follows.

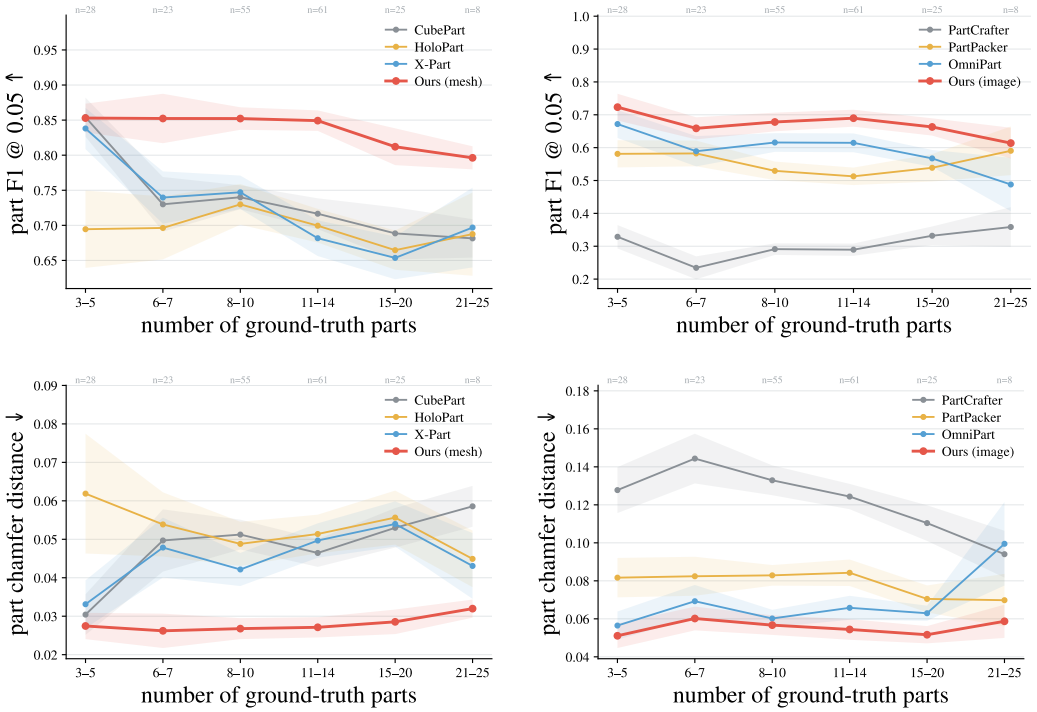

Figure 5: Part quality against the number of ground-truth parts, from a mesh (left) and from an image (right), with part quality metrics.

Number of prompt points. Figure 6 extends Table 4 to K from one to eight points per part. Both metrics improve steeply from one to three points and then level off, and the gap between the backbones stays constant across K, so extra points mainly remove the ambiguity of a single click rather than add information the backbone lacks. Since the decoder cost is independent of K, the measured end-to-end runtime changes negligibly.

Inference time. Figure 7 plots segmentation quality against inference time per asset. Our model is the most accurate and under a second per asset. The three points for $K \in \{1, 2, 4\}$ lie on a vertical line, since the points of a prompt are pooled into one token in the first encoder layer and the decoder cost does not depend on K.

Other benchmarks. Table 6 repeats the segmentation comparison on the PartNeXt evaluation set (Wang et al., 2025), a benchmark with finer and hierarchical part labels from a different source than PartObjaverse-Tiny. Ours still leads every column among the methods, indicating that the gains transfer to a dataset from a different source.

Reproducibility. We randomly sampled point prompts and repeated the experiment over five independent runs. The resulting standard deviations are small: 0.08 for pCD, 0.4 for both pF1@.01 and pF1@.05, and 0.3 for mIoU in the segmentation task. This indicates that the reported results are stable with respect to prompt sampling.

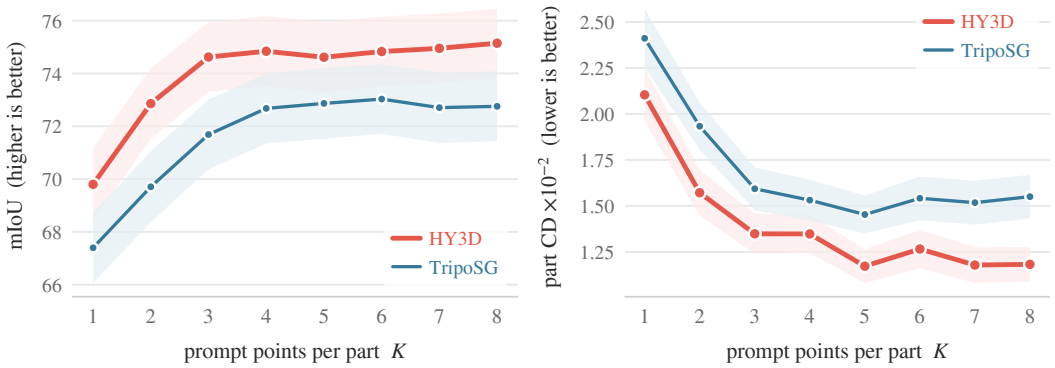

Figure 6: Segmentation quality against the number of prompt points per part K, for both backbones, with mIoU (left) and pCD (right). Both metrics improve steeply up to three points and change little beyond.

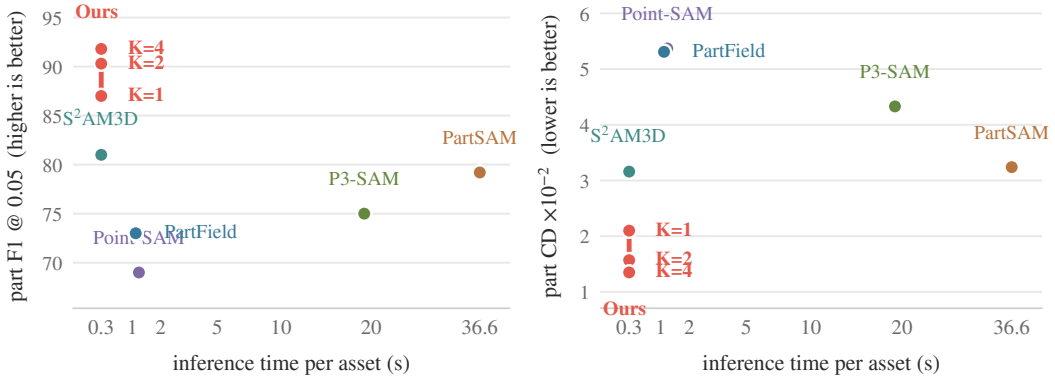

Figure 7: Segmentation quality against inference time per asset, pF1@.05 (left) and pCD (right), with our model at $K \in \{1, 2, 4\}$. Ours is the most accurate at close to the lowest time, and raising K improves quality at no cost in time.

Table 6: **Comparison of our method with SOTA methods for 3D part segmentation on PartNeXt.** We evaluate part decomposition quality on face mIoU and on part-level pCD and pF1.

Method	mIoU↑	pCD↓	pF1@.01↑	pF1@.05↑
Point-SAM (Zhou et al., 2025)	44.41	4.24	60.8	78.8
PartField (Liu et al., 2025)	52.55	7.25	53.0	65.7
S^2AM3D (Su et al., 2026)	52.99	<u>2.66</u>	69.9	<u>85.1</u>
P3-SAM (Ma et al., 2026)	<u>54.89</u>	3.24	<u>72.6</u>	83.1
Ours	**57.76**	**2.39**	**75.7**	**86.1**

D ADDITIONAL VISUALIZATION

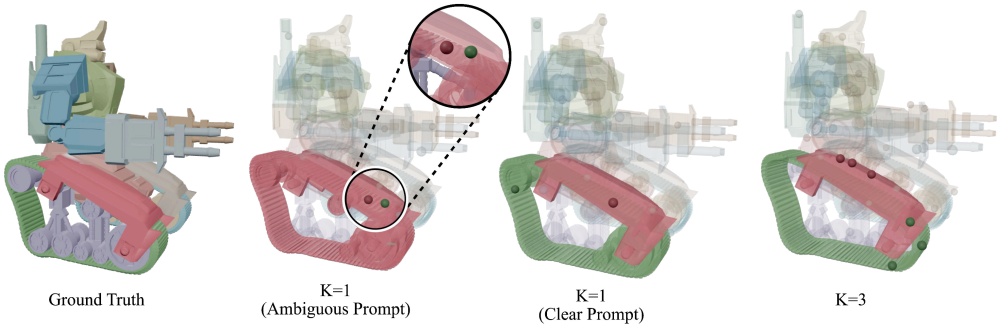

Figure 8: **Failure case and the effect of prompt placement.** The same mesh is decomposed from one point per part placed near a contact between two parts (second column), from one point per part placed on the body of each part (third column), and from three points per part (fourth column).

Exploded views. Figure 9 moves the parts of our results apart. For generation, from an image or a mesh, every part is a closed mesh with its own cut faces, so the parts separate without any repair and reassemble into the whole, which is the use the introduction asks for. For segmentation, the parts are the labeled faces of the input mesh, so they are open surfaces, and the exploded view shows where the label boundaries fall.

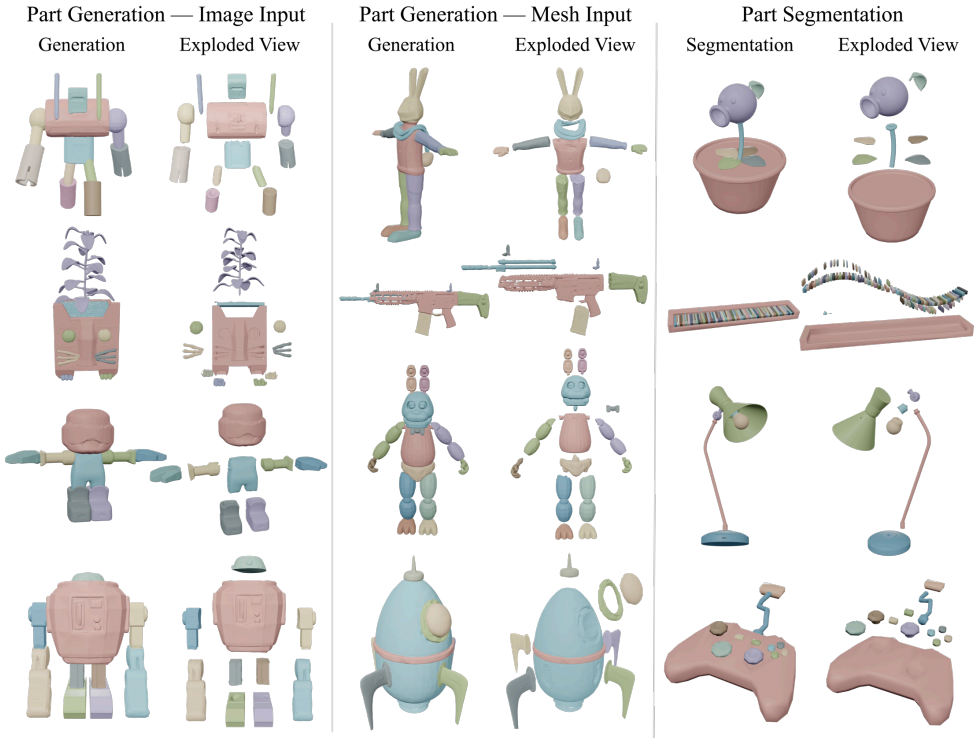

Figure 9: **Exploded views of our results.** Three assets per task, shown assembled and with the parts moved apart: part generation from an image (left), from a mesh (middle), and part segmentation of a mesh (right).

Segmentation. Figure 10 compares per-face labels on assets from both segmentation benchmarks. The baselines err in two ways: merging small repeated parts into their neighbor, as with the cactus spikes and the keyboard keys, or fragmenting a single part into several, as with the roof and the piano body, while a partition with one prompt per part does neither.

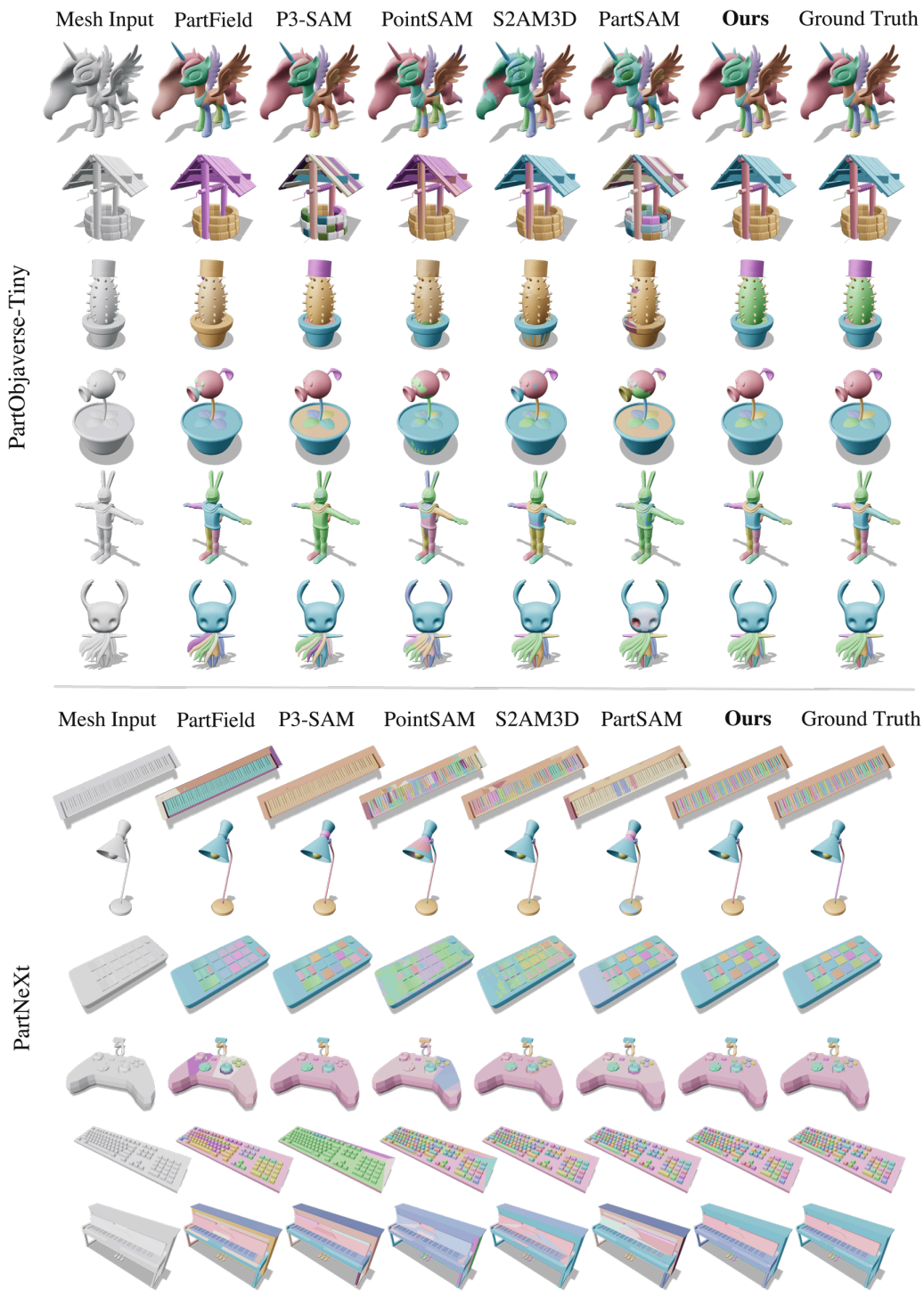

Figure 10: **Part segmentation on PartObjaverse-Tiny (top) and PartNeXt (bottom)**. One point prompt per part, against the segmenters in Table 2 and the ground truth. Ours keeps small repeated parts separate, such as the spikes, keys, and buttons, where the baselines merge or fragment them.

Generation. Figure 12 extends Fig. 3 to twelve further assets from a mesh, and Fig. 13 to ten assets from an image. The red regions, the volume claimed by two parts, appear at the joints of every baseline and are absent from ours, and where a baseline drops a part or returns a fragmentary object, ours keeps the decomposition of the ground truth.



Figure 11: **Limitation on thin and open geometry.** Large open surfaces and thin shells are poorly reconstructed by the pretrained VAE, and our decomposition consequently inherits these geometric errors. From left to right: input, VAE reconstruction, our decomposition, and ground truth.

Controlling the decomposition. Figure 14 decomposes objects at three granularities, from fewer to more parts, through the specification each method takes. The top block shows one object, a robot arm, under every method. A part count fixes how many parts PartField returns but not which, so the boundaries at each count are chosen by its clustering rather than by the user. Part names pass through CubePart's reading of them, and the number of parts it returns does not follow the number of names. Masks for OmniPart are drawn on the image, so only parts visible in the view can be specified. With one point per part, each added point splits off the part it sits on and leaves the other parts in place, so the user moves from a coarse to a fine decomposition by adding points. The bottom block shows the same on two further objects, from the image alone.

Failure cases and prompt placement. Figure 8 shows the failure mode we observe most often. An ambiguous prompt, a point placed where two parts touch, gives both prompts the same neighborhood, and the decoder merges the parts (second column). A clear prompt, a point on the body of each part, avoids this (third column), and more points per prompt recover from an ambiguous one, since the pooled prompt then covers the part rather than one location (fourth column, as in Table 4).

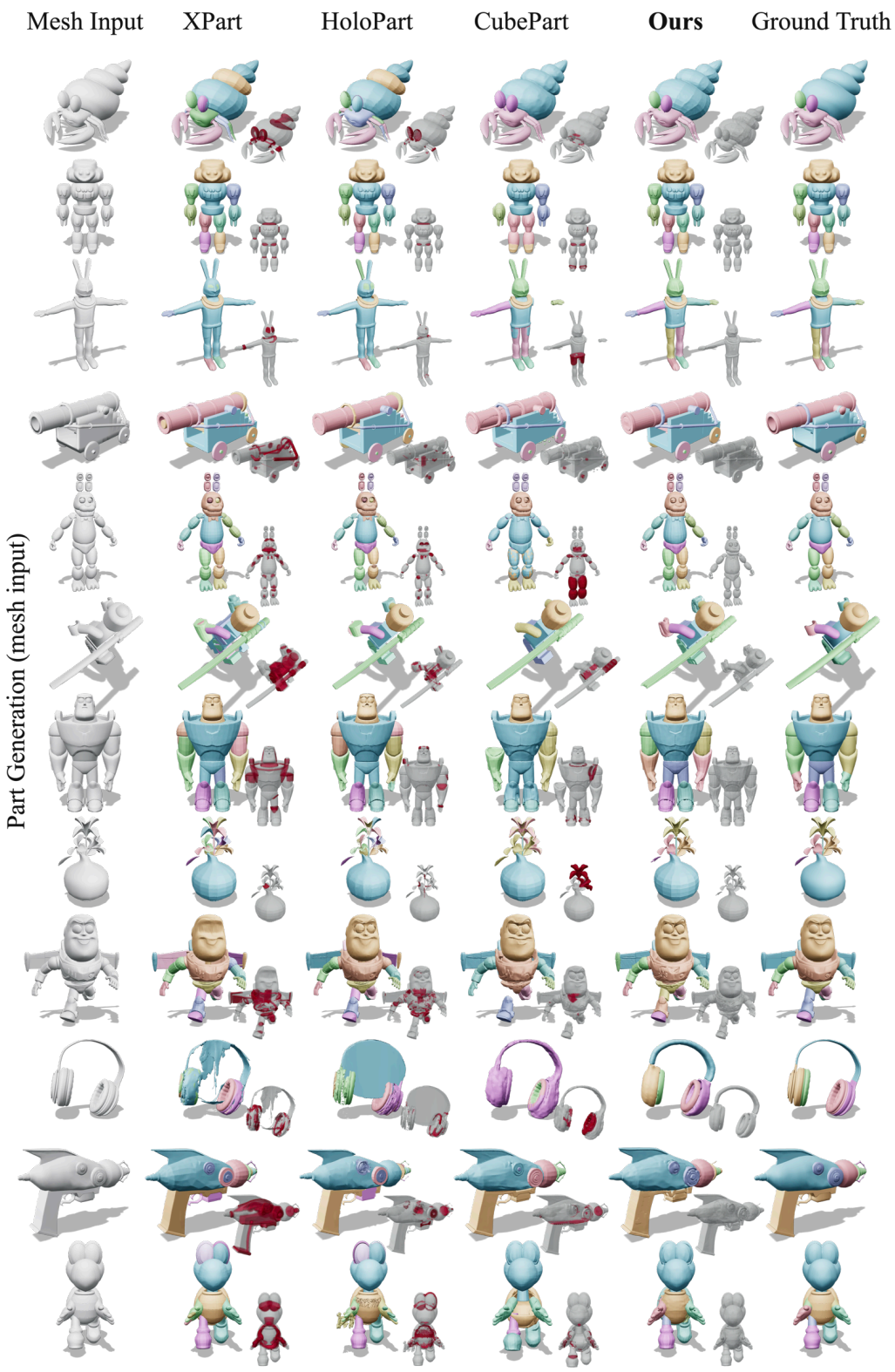

Figure 12: **Part generation from a mesh on further assets**. Compared against the generators in Table 1, with the volume claimed by more than one part marked in red beside each result. Every baseline interpenetrates at the joints, whereas our parts meet with negligible overlap and follow the ground-truth decomposition.

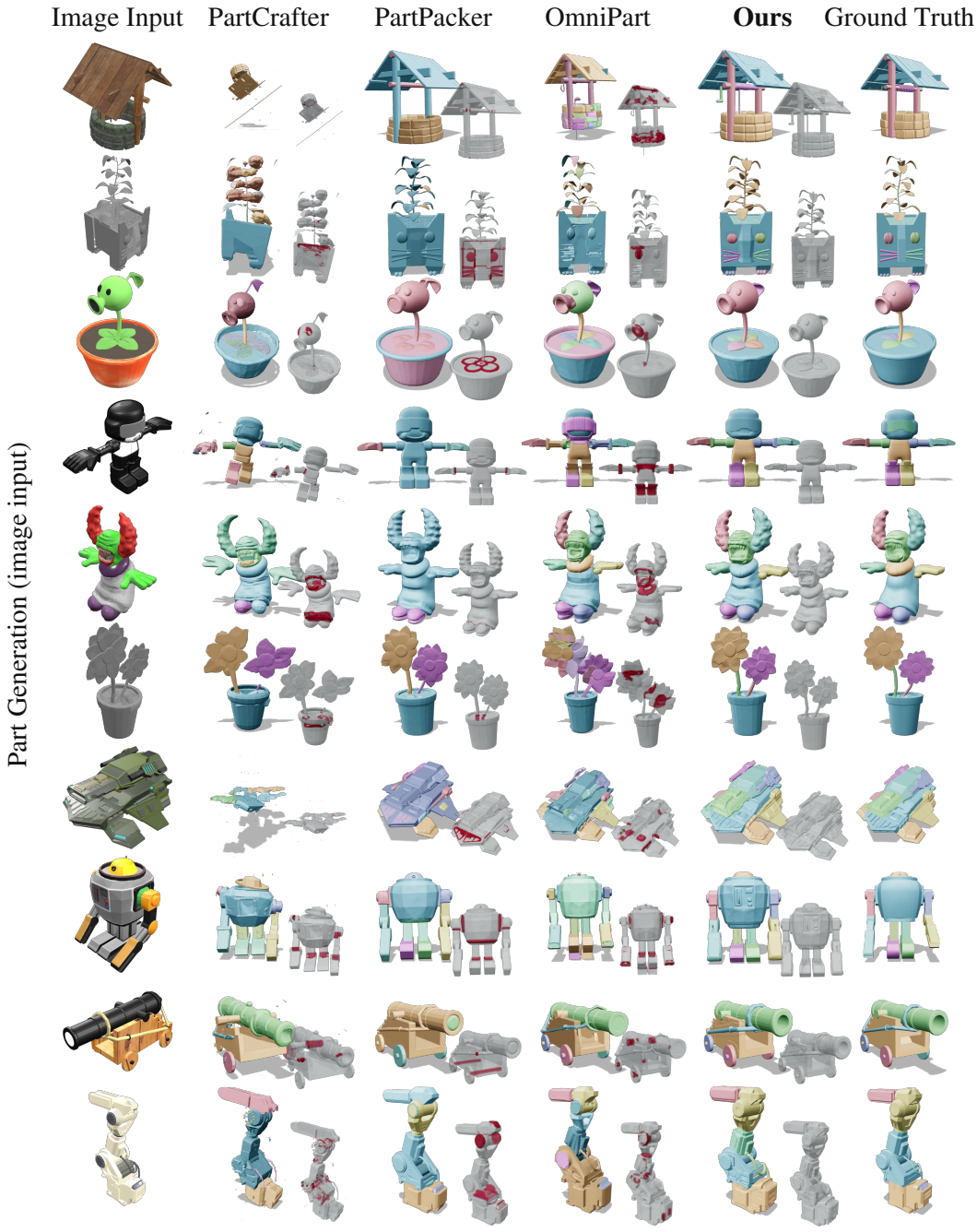

Figure 13: **Part generation from a single image on further assets**. Compared against the generators in Table 1, with the volume claimed by more than one part marked in red beside each result. The baselines interpenetrate at the joints or return a fragmentary object, whereas our parts meet with negligible overlap and follow the ground-truth decomposition.

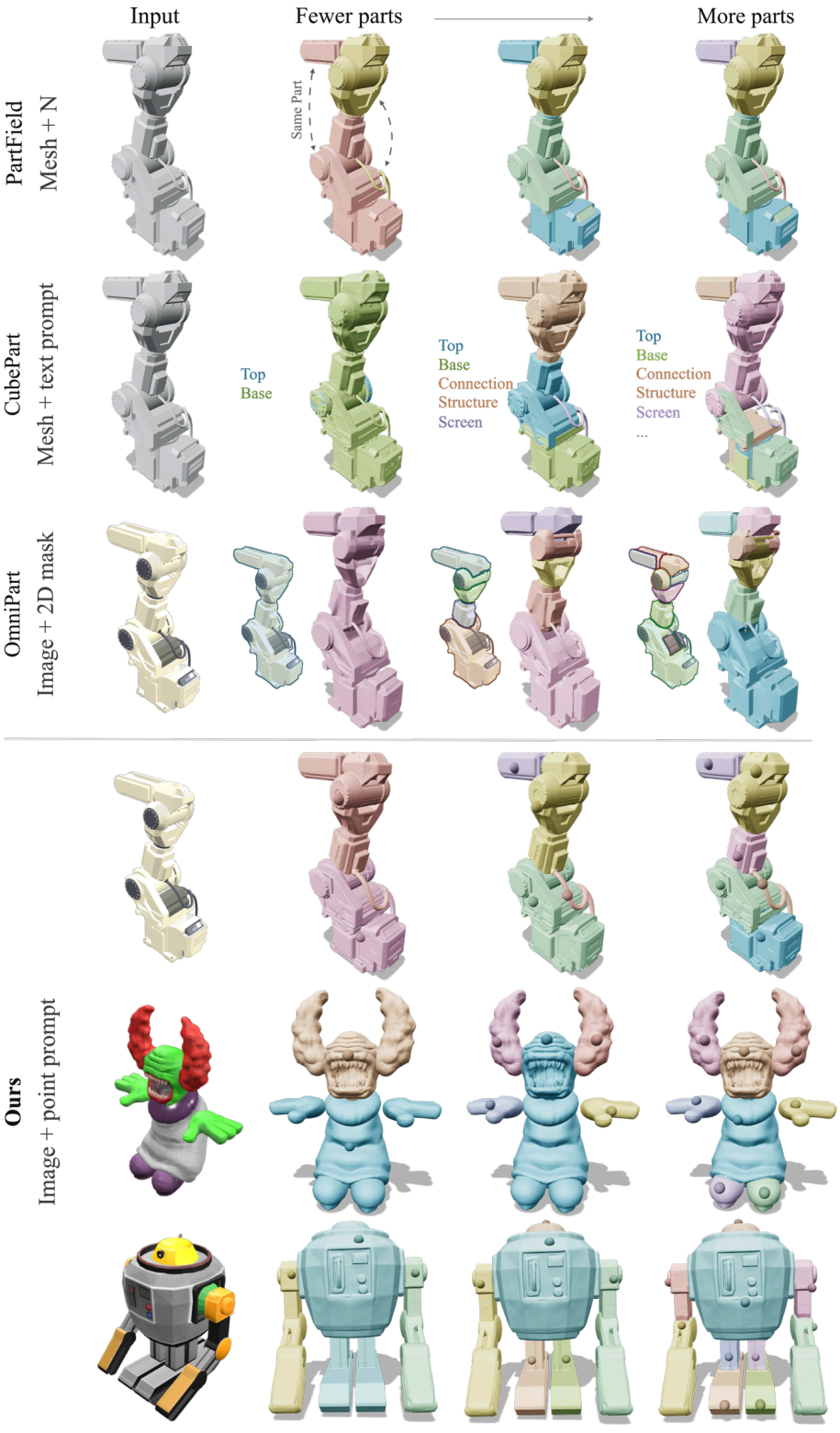

Figure 14: **Controlling the decomposition.** Objects are decomposed at three granularities, from fewer parts to more parts, through the specification each method takes: a part count for PartField, part names for CubePart, 2D masks on the image for OmniPart, and one point per part for ours. The top shows the same sample under every method; the bottom shows ours on two further samples.