P^3-SAM: Native 3D Part Segmentation

Changfeng Ma[1,2], Yang Li[2*], Xinhao Yan[2,3], Jiachen Xu[2], Yunhan Yang[2,4], Chunshi Wang[2,5], Zibo Zhao[2], Zhuo Chen[2], Yanwen Guo[1†], and Chunchao Guo[2†]

∗ Project Leader, † Corresponding Author

[1] Nanjing University, Nanjing, China
[2] Tencent Hunyuan, Shenzhen, China
[3] ShanghaiTech University, Shanghai, China
[4] The University of Hong Kong, Hong Kong, China
[5] Zhejiang University, Zhejiang, China

Fig. 1. P^3-SAM produces precise part segmentation results for any object.

Abstract. Segmenting 3D assets into their constituent parts is crucial for enhancing 3D understanding, facilitating model reuse, and supporting various applications such as part generation. However, current methods face limitations such as poor robustness when dealing with complex objects and cannot fully automate the process. In this paper, we propose a native 3D point-promptable part segmentation model termed P^3-SAM, designed to fully automate the segmentation of any 3D objects into components. Inspired by SAM, P^3-SAM consists of a feature extractor, multiple segmentation heads, and an IoU predictor, enabling interactive segmentation for users. We also propose an algorithm to automatically select and merge masks predicted by our model for part instance segmentation. Our model is trained on a newly built dataset containing nearly 3.7 million models with reasonable segmentation labels. Comparisons show that our method achieves precise segmentation results and strong robustness on any complex objects, attaining state-of-the-art performance. Our code will be released soon.

Keywords: Part Segmentation · 3D · SAM

1 Introduction

As a fundamental 3D version task, the segmentation of 3D assets plays a crucial role in shape analysis, editing, and reuse, as well as other downstream tasks such as mesh simplification and animation design. Many works have been proposed to address 3D segmentation and have achieved notable success, but there are still challenges in this task.

Traditional learning-based segmentation methods attempt to segment 3D point clouds using predefined part categories, relying on direct supervision from part labels. They can only segment certain parts within specific categories and struggle to handle objects with arbitrary parts. To this end, recent works leverage the capabilities of the 2D SAM [11] by lifting 2D segmentation results to serve as 3D segmentation ground truth for training 3D segmentation models. Among them, SAMPart3D [31] and PartField [13] train feature learning networks through distillation or contrastive learning and achieve 3D segmentation by feature clustering. Point-SAM [35], on the other hand, directly adapts the 2D SAM framework to 3D point clouds, training a promptable 3D segmentation model. However, these methods still rely on 2D SAM results, which cannot provide 3D-consistent segmentation signals and often produce imprecise part boundaries. As a result, these method suffer from imprecise results and lack strong robustness when dealing with arbitrarily complex objects. Additionally, they are still one step away from full automatic segmentation, as they require users to provide the number of parts or prompt points.

In this paper, we propose a native 3D **P**oint-**P**romptable **P**art segmentation model termed **P**^{3}-SAM, designed to fully automate the segmentation of any complex 3D objects into components with precise mask and strong robustness. The improvements and differences of our method compared to previous methods are summarized in Table1. As a pioneering promptable image segmentation work, SAM provides a feasible implementation approach. However, our method focuses on achieving precise part segmentation automatically, and we simplify the architecture of SAM. Without adopting the complex segmentation decoder and multiple types of prompts from SAM, our model is designed to handle only one positive point prompt. Specifically, **P**^{3}-SAM contains a feature extractor, three segmentation heads, and an IoU prediction head. We employ PointTransformerV3 [27] as our feature extractor and integrate its features from different levels as extracted point-wise features. The input point prompt and feature are fused and passed to the segmentation heads to predict three multi-scale masks and an IoU (Intersection of Union) predictor is utilized to evaluate the quality of the masks. To automatically segment an object, we apply our segmentation model using point prompts sampled by FPS (Farthest Point Sampling) and utilize NMS (Non-Maximum Suppression) to merge redundant masks. The point-level masks are then projected onto mesh faces to obtain the part segmentation results.

Another key aspect of this paper is to eliminate the influence of 2D SAM, and rely exclusively on raw 3D part supervision for training a native 3D segmentation model. While existing 3D part segmentation datasets are either too small (e.g.

Table 1. The comparison of our method with related works across several key aspects.

Methods	Data Num.	Data Type	Param.	Time(Seg.)	Time(Inter.)	Auto.
SAMesh	-	2D Lifting	-	∼7min	-	✔
Find3D	30K	2D Data Engine	46M	∼10s	-	✖
SAMPart3D	200K	2D Data Engine	114M	∼15min	-	✖
PartField	360K	2D Data Engine	106M	∼10s	-	✖
Point-SAM	100K	2D Data Engine	**311M**	-	∼5ms	✖
Ours	**3.7M**	**3D Native**	112M	**∼8s**	**∼3ms**	✔

PartNet [17]) or lack part annotation (e.g. Objaverse [7]), this work addresses the data scarcity by developing an automated part annotation pipeline for artist-created meshes and used it to generate a dataset comprising 3.7 million meshes with high-quality part-level masks. Our model demonstrates excellent scalability with this dataset and achieves robust, precise, and globally coherent part segmentation.

Our extensive experiments demonstrate that our method achieves state-of-the-art performance in part segmentation for any parts of any objects, especially on complex objects with highly detailed geometry. The main contributions of our P^3-SAM are summarized as follows:

- We propose a native 3D point-promptable part segmentation model to segment any parts of any objects.
- We propose a fully automatic part segmentation approach using our model and a mask merging algorithm.
- With high accuracy, generalization, and robustness across various tasks and data types, our method can be applied to interactive, multi-head and hierarchical part segmentation.

2 Related Work

2.1 Traditional 3D Part Segmentation

Traditional 3D part segmentation methods usually train their networks on specific part labels from object or scene datasets, such as PartNet [17], Princeton Mesh Segmentation [4], ScanNet [5], and S3DIS [2]. These methods employ point cloud encoders like PointNet [21] and PointTransformerV3 (PTv3) [27] or mesh encoders like MeshCNN [9] and Mesh Transformer (Met) [34] to extract 3D features for the segmentation head to predict part labels. In addition to segmenting parts with semantic meaning, recent works aim to segment out geometrically significant parts. However, traditional methods suffer from limited categories and part labels and struggle to generalize to arbitrary categories and parts.

2.2 2D Lifting 3D Part Segmentation

With the development of 2D foundation models, significant progress has been made by models such as CLIP [22], GLIP [12], SAM [11], Dinov2 [18] and VLM in image-text alignment and zero-shot detection and segmentation. Rendering 3D models into multi-view images and leveraging these 2D foundation models for lifting 2D capabilities to 3D is an obvious but effective approach. Recent methods, such as SAMesh [24], SAM3D [32] and SAMPro3D [28], directly apply SAM

to rendered 2D images and aggregate multi-view masks to achieve class-agnostic segmentation for any 3D objects or scenes. Additionally, several methods utilize text descriptions of categories as prompts on 2D rendered images to enhance the querying of 3D parts. PartSLIP [14] and SATR [1] employ text prompts and GLIP [12] to detect parts, followed by post-processing to segment parts on point clouds and meshes. Besides GLIP, PointSLIP++ [36] and ZeroPS [29] also leverage SAM, achieving more precise segmentation results. The MeshSegmenter [33] employs Stable Diffusion [23] to generate textures for a mesh, enabling SAM [11] and Grounding DINO [15] to clearly segment and detect parts. PartDistill [25] utilizes 2D Vision-Language Models for forward and backward knowledge distillation to achieve 3D part segmentation. COPS [8] leverages Dinov2 [18] to project visual features onto 3D point clouds and then uses a VLM for part segmentation. Directly lifting 2D knowledge to 3D may encounter limitations such as data gaps, 3D consistency issues, and unstable post-processing, leading to poor robustness and inaccurate segmentation results. Text-query-based methods also require prompt engineering. Rendering multi-view images and using SAM or VLM to process these images can also consume significant resources and time.

2.3 2D Data Engine for 3D Part Segmentation

To alleviate the 3D consistency issues and data gaps brought by directly lifting 2D knowledge, recent works attempt to use 2D foundation models to build a data engine for training feed-forward networks on 3D point clouds and meshes. OpenScene [20] employs a feature extractor to learn the CLIP features projected onto scene point clouds and uses text prompts to query these features for segmentation. Segment3D [10] and SAL [19] use feed-forward networks to learn the projected masks of scene meshes and LiDAR data predicted by SAM, and then use CLIP to assign categories to each part. Find3D [16] trains a network to segment objects given text prompts by building a data engine that allows the VLM to query parts after SAM processes multi-view images. SAMPart3D [31] employs a network to distill the projected Dinov2 features of point clouds. To achieve more accurate segmentation of each object, it then trains a lightweight MLP for each object to predict segmentation masks by conducting contrastive learning on SAM projections. Finally, a MLLM is utilized to annotate each part. PartField [13] directly supervises a network composed of a voxel CNN and a tri-plane transformer with contrastive learning loss on both 2D and 3D masks, where the 2D masks are obtained using SAM. Point-SAM [35] adapts SAM to 3D point clouds and utilizes SAM to design a data engine based on multi-view images. This data engine continuously trains and refines a PointViT model to achieve part segmentation based on prompt points. Although a 2D data engine can reduce 3D inconsistencies and improve the network's generalization ability, segmentation based on 2D data can still suffer from boundary ambiguities and data gaps, leading to inaccurate segmentation results, especially on complex data. Additionally, these methods either require specifying the number of categories or need user-provided prompt points, which means they cannot fully automate object segmentation.

3 Method

Given the mesh $\mathbf{M} = (\mathbf{V}, \mathbf{F} \in \mathbb{N}^{N_f \times 3})$ of an object, our goal is to predict a mask $\mathbf{m}_{part} \in \{1, 2, 3, ..., N_{part}\}^{N_f}$ that segments each face into N_{part} parts, where N_f indicates the number of faces and N_p represents the number of parts for the object. Here, each part is instance-specific but class-agnostic. We first introduce our data curation (Section 3.1), then describe the architecture of our point-promptable part segmentation module termed P^3-SAM (Section 3.2), and finally present the automatic segmentation algorithm (Section 3.3).

3.1 Data Curation

To construct our dataset, we aggregated 3D models from multiple sources, including Objaverse [7], Objaverse-XL [6], ShapeNet [3], PartNet [17], and other internet repositories. The aforementioned 3D assets are mainly created by artists. Artists tend to craft 3D shapes part by part and assemble them together. The assembly process may not forge parts' meshes together; this gives us the chance to reverse the part information from the asset. Specifically, the complete mesh is first decomposed into sub-meshes based on connected components. Then we calculate the surface area of each sub-mesh, and build adjacency graph between parts (we voxelize the space with resolution of 128, two sub-meshes are considered adjacent if they share any voxel). We iteratively merge small parts (with a surface area less than 1% of the total) with their adjacent, larger parts. This bottom-up process continues until all parts exceed the 1% area threshold. After merging, we filter out objects that have too few (less than 2) or too many (more than 50) parts. To prevent the impact of mask area imbalance, we filtered out objects with disproportionately large parts (where a single part occupies more than 0.85%) and objects with a significant number of very small parts (where parts smaller than 1% in area collectively account for more than 10% of the total area). After the aforementioned filtering steps, we obtain nearly 3.7 million objects.

However, these object models are non-watertight at the object-level, often containing internal structures and clear boundaries. Training solely on such data can lead to poor generalization on watertight 3D models, such as scanned mesh or AI-generated ones. We then made the filtered models watertight, resulting in nearly 2.3 million successfully watertight models. These watertight models do not contain internal structures and only include the outer surfaces of the models. To obtain the labels of each face on watertight models, we query their nearest faces on the non-watertight model and assign the label of the nearest face as the label for the corresponding face on the watertight model. During training, if a model has a watertight version, we set an 80% probability of selecting the watertight data for training. This allows our network to handle both watertight and non-watertight data.

3.2 Point-Promptable Part Segmentation Model

Network Architecture To achieve class-agnostic part segmentation for arbitrary objects, providing prompts related to parts is more efficient than direct label supervision or contrastive learning. Previous methods utilize text as prompts to query parts, while methods like SAM [11] and Point-SAM [35] use positive

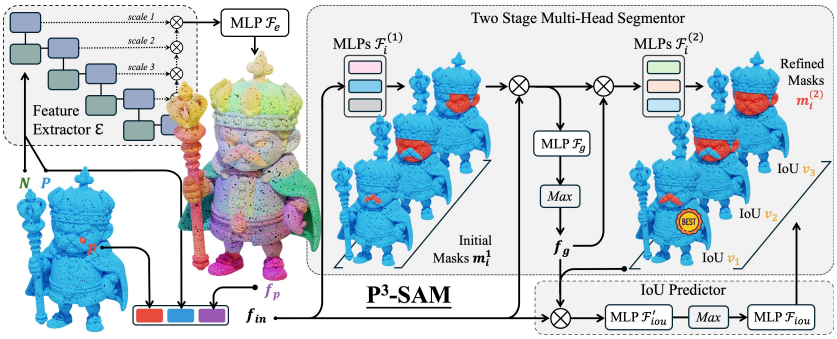

Fig. 2. The Network Architecture of P^3-SAM. Input point clouds are fed to feature extractor to obtain point features. The features, point prompts, and original point clouds are then fed to a two stage multi-mask segmentor to obtain three masks in various scales for the prompts. Finally, the IoU predictor is utilized to evaluate the quality of the masks and select the best one as the final prediction.

and negative prompt points or bounding boxes as prompts. To fully facilitate the automatic part segmentation of objects using point-based prompts, we design our P^3-SAM to segment a part using only a single point prompt. This allows the network to avoid adapting to diverse prompts, simplifying the network and improving its convergence, generalization, and accuracy. The input to our P^3-SAM consists of the point cloud $\mathbf{P} \in \mathbb{R}^{N_p \times 3}$ and its normals $\mathbf{N} \in \mathbb{R}^{N_p \times 3}$ sampled from the input mesh $\mathbf{M}$ and a point $\mathbf{p} \in \mathbb{R}^3$ as a prompt to indicate the part that needs to be segmented. As shown in Figure 2, the architecture of our method consists of a feature extractor, a multi-head segmentor, and an IoU predictor. Since our method only requires a single point prompt, there is no need to design a prompt encoder. We directly input the prompt into the multi-head segmentor.

Feature Extractor. Recent point cloud encoders have achieved excellent results on various point cloud tasks, especially Sonata [26], a self-supervised pre-trained Point Transformer V3 [27]. We then employ Sonata with its pre-trained weights as our feature extractor $\mathcal{E}$ to extract multi-scale features from point clouds. To better handle complex objects, we reduced the voxel size of the input to Sonata. We then aggregate these multi-scale features together and use an weight-shared MLP $\mathcal{F}_e$ to obtain point-wise features f, as shown in Figure 2,

$$\mathbf{f}_p = \mathcal{F}_e(\mathcal{E}(\mathbf{P}, \mathbf{N})_1, \mathcal{E}(\mathbf{P}, \mathbf{N})_2, ..., \mathcal{E}(\mathbf{P}, \mathbf{N})_n),$$

where the subscripts indicate features at different scales. The point features need to be predicted only once and can be used for predicting part masks with different point prompts.

Multi-Head Segmentor. Our part dataset, introduced in Section 3.1, integrates multiple data sources which may involve varying granularity and conflicting criteria for part separation. Additionally, the point prompts might be ambiguous in indicating the specific scale of a part, as mentioned in SAM. Therefore, we use a multi-head segmentor to predict multiple alternative masks at various scales in order to mitigate this conflict and ambiguity. Our multi-head segmentor

contains a two-stage prediction process. In the first stage, three MLPs $\mathcal{F}_i^{(1)}$ take a mixed input $\mathbf{f}_{in}$, including the point-wise features $\mathbf{f}_p$, the input points $\mathbf{P}$ and N_p copies of the point prompt $\mathbf{p}$, to predict three different masks,

$$\mathbf{m}_i^{(1)} = \mathcal{F}_i^{(1)}(\mathbf{f}_{in}) = \mathcal{F}_i^{(1)}(\mathbf{f}_p, \mathbf{P}, \mathbf{p}), i = 1, 2, 3.$$

However, the first stage is a naive implementation that lacks support for global information. Therefore, in the second stage, we introduce a global feature and re-predict the three masks based on the results from the first stage. As shown in the Figure2, we use an MLP $\mathcal{F}_g$ to predict point-wise features and apply max pooling along the point dimension. We utilize an MLP $\mathcal{F}_g$ to predict point-wise features, and apply max pooling along the point dimension to derive a global feature,

$$\mathbf{f}_g = MaxPool(\mathcal{F}_g(\mathbf{f}_{in}, \mathbf{m}_1^{(1)}, \mathbf{m}_2^{(1)}, \mathbf{m}_3^{(1)})).$$

Finally, three new MLPs $\mathcal{F}_i^{(2)}$ are employed to predict more accurate results based on the global features and the outcomes from the first phase,

$$\mathbf{m}_i^{(2)} = \mathcal{F}_i^{(2)}(\mathbf{f}_{in}, \mathbf{f}_g, \mathbf{m}_1^{(1)}, \mathbf{m}_2^{(1)}, \mathbf{m}_3^{(1)}), i = 1, 2, 3.$$

While the second stage optimizes the results from the first stage, the initial masks $\mathbf{m}_i^{(1)}$ also help the second stage to focus the extraction of global features on the parts that need segmentation, making feature extraction more efficient and improving the accuracy of segmentation results.

IoU Predictor. To achieve automatic identification of the best mask, we introduce an IOU predictor to assess the quality of $\mathbf{m}_1^{(2)}, \mathbf{m}_2^{(2)}, \mathbf{m}_3^{(2)}$ and select the best mask as the network's final prediction. The assessment is achieved by directly predicting the IoU values of the predicted masks and the ground truth masks. The IOU predictor first uses an MLP $\mathcal{F}'_{iou}$ and max pooling to obtain a global feature from the global feature and three masks of second stage, and then employs another MLP $\mathcal{F}_{iou}$ to predict three IoU values,

$$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 = \mathcal{F}_{iou}(MaxPool(\mathcal{F}'_{iou}(\mathbf{f}_{in}, \mathbf{f}_g, \mathbf{m}_1^{(2)}, \mathbf{m}_2^{(2)}, \mathbf{m}_3^{(2)}))).$$

The multi-head segmentor and IoU predictor are lightweight models capable of real-time computation. Consequently, once the global feature of a given 3D model is extracted, our P^3-SAM can be utilized for real-time interactive segmentation.

Training To enhance the robustness of our network, we introduce random noise to the input points $\mathbf{P}$, normals $\mathbf{N}$, and point prompts $\mathbf{p}$ during training. Furthermore, we randomly remove normals with a probability of 0.3. We also mix watertight and non-watertight data, as mentioned in Section 3.1.

During training, for a given model, we randomly select K part masks and then randomly choose one point from the part points corresponding to each mask, resulting in K prompts $\mathbf{p}_j \in \mathbb{R}^3$ and K ground truth part masks $\mathbf{m}_j^{(gt)} \in \{0, 1\}^{N_p}$, where $j = 1, 2, ..., K$. For the three masks generated by the network in

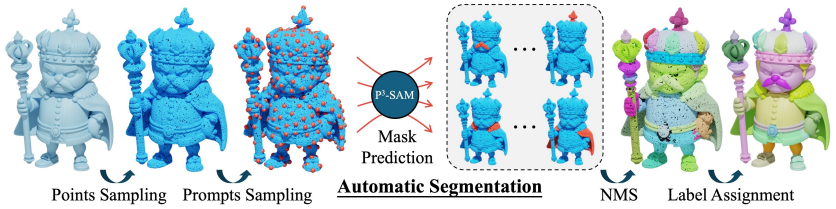

Fig. 3. Automatic Segmentation Pipeline. Point prompts are sampled by FPS and go through the P^3-SAM to obtain multiple masks. NMS is then adopted to merge redundant masks. The point-level masks are then projected onto mesh faces to obtain the part segmentation results.

Algorithm 1 Automatic Segmentation

Input: Mesh $\mathcal{M}$ with N_f faces
Output: Mask $\mathbf{m}_{part} \in \{1, 2, ..., N_{part}\}^{N_f}$ with N_{part} parts
1: Sample N_p points $\mathbf{P}$ with normals $\mathbf{N}$ from $\mathcal{M}$
2: Sample N_{pp} prompt points $\mathbf{p}_j$ from $\mathbf{P}$ using FPS
3: Extract point-wise feature $\mathbf{f}_p$ from $\mathbf{P}$ and $\mathbf{N}$ using P^3-SAM
4: Predict N_{pp} masks $\mathbf{m}_j$ and IoU value $\mathbf{v}_j$ based on $\mathbf{p}_j$ using P^3-SAM
5: Filter masks $\mathbf{m}_j$ using NMS and retain N_{part} masks
6: Assign the N_{part} point masks $\mathbf{m}_j$ to $\mathcal{M}$ with part labels
7: Fill the faces without labels using flood fill algorithm and produce the mask $\mathbf{m}_{part}$

the first and second stages, we apply both Dice loss $\mathcal{L}_{dice}$ and Focal loss $\mathcal{L}_{focal}$ for supervision. Backpropagation is applied only to the output with the lowest loss, which encourages each segmentation head to predict masks at different scales. So, the mask loss $\mathcal{L}_{mask}$ can be calculated as:

$$\mathcal{L}_{mask}^{(t)} = \frac{1}{K}\sum_{j=1}^{K}\min_{i=1}^{3}\left(\alpha_{dice}\mathcal{L}_{dice}(\mathbf{m}_{ij}^{(t)}, \mathbf{m}_{j}^{(gt)}) + \mathcal{L}_{focal}(\mathbf{m}_{ij}^{(t)}, \mathbf{m}_{j}^{(gt)})\right),$$

where α_{dice} is a weighting parameter, and $t = 1, 2$ indicates whether the loss is computed for the first or second stage of the network. To supervise the IoU, we first calculate the IoU between $\mathbf{m}_{ij}^{(2)}$ and $\mathbf{m}_{j}^{(gt)}$. We then use MSE to compute the loss based on these IoU values,

$$\mathcal{L}_{IoU} = \frac{1}{3K}\sum_{j=1}^{K}\sum_{i=1}^{3}\mathcal{L}_{MSE}\left(\mathbf{v}_{ij}, IoU(\mathcal{I}(\mathbf{m}_{ij}^{(2)}), \mathbf{m}_{j}^{(gt)})\right),$$

where $\mathcal{I}$ is an indicator function that indicates whether the mask value is greater than 0.5. The overall loss is the sum of the mask losses from both the first and second stages and the IOU loss, that is $\mathcal{L} = \mathcal{L}_{mask}^{(1)} + \mathcal{L}_{mask}^{(2)} + \mathcal{L}_{IoU}$.

3.3 Automatic Segmentation

Methods such as interactive segmentation, text prompt extraction, and clustering often require human intervention during the segmentation process. To

achieve fully automatic segmentation, we propose an automated approach based on our P^3-SAM, as shown in Algorithm 1. Our method begins by sampling points $\mathbf{P}$ and normals $\mathbf{N}$ from given mesh $\mathcal{M}$. Subsequently, we use Farthest Point Sampling (FPS) to select N_{pp} point prompts $\mathbf{p}_j$ from $\mathbf{P}$. After extracting features $\mathbf{f}_p$ from $\mathbf{P}$, we predict a mask $\mathbf{m}_j$ and an IoU value $\mathbf{v}_j$ based on each point prompt $\mathbf{p}_j$ utilizing our P^3-SAM. To ensure that each part can be segmented out, point prompts are always over-sampled, with their number being significantly greater than the actual number of parts in the object. To obtain the true number of parts, we use Non-Maximum Suppression (NMS) to filter out the numerous duplicate masks. We first sort the masks in descending order according to their IoU values to form a candidate queue. We take out the first mask and use it to filter out other masks in the queue that have an IoU greater than T_{NMS} with this mask. We repeat this process of selecting and filtering until no masks remain in the queue. The set of all selected masks constitutes the final result of NMS. The part number N_{part} is the number of the selected masks, and each mask has its own part label. According to which face and mask each point belongs to, we assign the corresponding part labels to each face and determine a final part label for each patch through voting. We use the flood fill algorithm to assign labels to faces that do not have a label. Specifically, for each unlabeled face, we assign it the most frequent label among its neighboring faces (or its nearest several faces if there is no connectivity in the mesh). We repeat this process until all faces have been assigned a label and obtain the final mask $\mathbf{m}_{part}$ of each faces.

4 Experiments

4.1 Implementation Details

The channel of the point-wise feature $\mathbf{f}_p$ is 512. The point number N_p is set to $100,000$ during training, evaluation, and inference. We randomly select $K = 8$ parts in the training process and set α_{dice} to 0.5. For automatic segmentation, we sample $N_{pp} = 400$ prompts from points, and the threshold T_{NMS} is set to 0.9. Our network is trained on our dataset using 64 H20 for 9 epochs. We set the batch size to 2 per GPU, and the training took approximately 4 days. We employ the Adam optimizer with a learning rate of 10^{-5}.

4.2 Comparison

Evaluation Datasets. We evaluate each method on three datasets: PartObj-Tiny [31], PartObj-Tiny-WT, and PartNetE [14]. PartObj-Tiny is a subset of Objarvse [7], containing 200 data samples across 8 categories, with manually annotated part segmentation information. PartObj-Tiny-WT is the watertight version of PartObj-Tiny. To evaluate the performance of various networks on watertight data, we converted the meshes from PartObj-Tiny to watertight versions and successfully obtained 189 watertight meshes. We then acquired the ground truth segmentation labels following the method described in Section 3.1. And there is no color on the watertight mesh, which could pose a challenge for methods based on rendering multi-view images. PartNetE, derived from PartNet-Mobility, contains 1,906 shapes covering 45 object categories in the form of point clouds. We also evaluate various networks on it to verify their generalization performance on point cloud.

Table 2. The comparison of our method with previous methods on PartObjaverse-Tiny. The first two blocks represent class-agnostic part segmentation without and with connectivity, respectively, and the last block represents interactive segmentation.

Task	Method	Human	Animals	Daily	Build.	Trans.	Plants	Food	Elec.	AVG.
Seg. w/o Connect.	Find3D	23.99	23.99	22.67	16.03	14.11	21.77	25.71	19.83	21.28
	SAMPart3D	55.03	57.98	49.17	40.36	47.38	62.14	**64.59**	51.15	53.47
	PartField	54.52	58.07	56.46	42.47	49.09	59.16	55.4	56.29	53.93
	Ours	**60.77**	**59.43**	**62.98**	**50.82**	**57.72**	**70.53**	54.04	**61.96**	**59.88**
Seg. w/ Connect.	SAMesh	66.03	60.89	56.53	41.03	46.89	65.12	60.56	57.81	56.86
	PartField	80.85	83.43	77.83	**69.66**	**73.85**	80.21	85.27	**82.30**	79.18
	Ours	**80.77**	**86.46**	**80.97**	67.77	68.44	**90.30**	**92.90**	81.52	**81.14**
Interact.	Point-SAM	26.13	29.25	28.85	23.58	22.91	31.44	33.04	28.05	27.91
	Ours	**49.01**	**53.45**	**52.36**	**38.50**	**51.52**	**62.57**	**50.80**	**51.86**	**51.23**

Table 3. The comparison of our method with previous methods on the watertight version of PartObjaverse-Tiny.

Task	Fully Segmentation w/o Connectivity					Interactive Seg.	
Method	Find3D	SAMPart3D	SAMesh	PartField	Ours	Point-SAM	Ours
PartObj-Tiny-WT	20.76	48.79	-	51.54	**58.10**	24.16	**49.11**
PartNetE	21.69	56.17	26.66	59.1	**65.39**	45.85	**63.48**

Tasks. There are three tasks: full segmentation without connectivity, full segmentation with connectivity, and interactive segmentation. The meshes in PartObj-Tiny are non-watertight, with each face exhibiting strong connectivity, forming distinct connected components that are strongly correlated with the segmentation results. Introducing such connectivity may improve segmentation performance. However, in real-world applications, the connectivity relationships of meshes are often chaotic or may not even exist. We then divide the full segmentation task into the first two tasks. For watertight data and point clouds, there is no connectivity, so the second task does not apply.

Baseline Methods. We compare our P^3-SAM with recent related works including SAMesh [24], Find3D [16], SAMPart3D [31], PartField [13] and Point-SAM [35]. Among these, SAMesh is a method based on 2D lifting that enables fully automatic segmentation. The other methods based on 2D data engines require human intervention in the overall segmentation process: Find3D requires text prompts, SAMPart3D and PartField need part categories for clustering, and Point-SAM necessitates manual selection of prompt points. Since Point-SAM cannot segment the entire model, we only compare its performance on interactive segmentation. More comparisons on different aspects, including time cost, number of parameters, amount of training data, etc., are shown in Table 1.

Metric. The evaluation metric for fully segmentation is the same as in previous work [13], using IOU to measure the accuracy of mask predictions. To evaluate the interactive segmentation, we sample 10 prompt points for each part, then measure the average IOU between the predicted masks for all prompts of all parts and their corresponding ground truth masks.

Results. Table 2 shows the evaluation results of various methods on PartObj-Tiny across three tasks. In the second task, the results for PartField [13] are based on its original version, which incorporates connected components as the basis for hierarchical clustering and selects the optimal results from multiple levels.

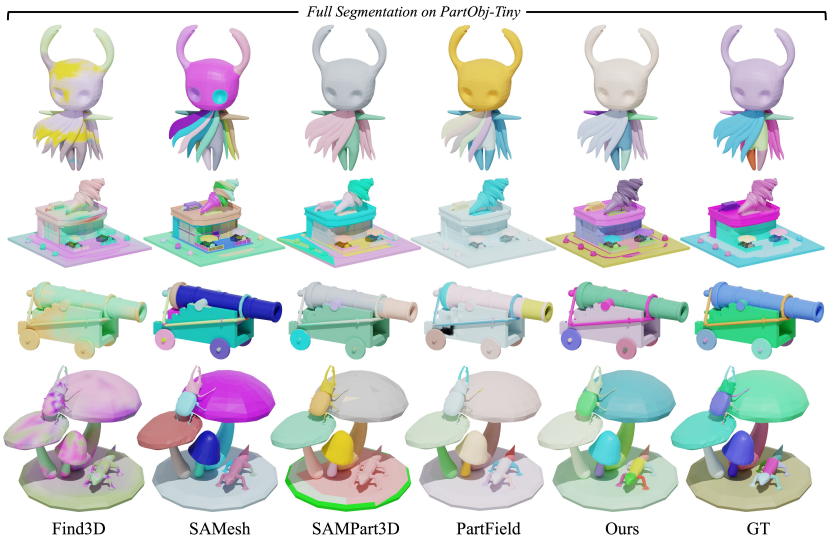

Fig. 4. The comparison of our automatic segmentation approach with previous methods on PartObj-Tiny.

To ensure a fair comparison, we also introduced connected components and used random prompts for each part. The detailed methodology can be found in Section 4.3. In the third task, since Point-SAM can only segment point clouds, we sample the point clouds from the meshes for comparison. Note that our method is also capable of segmenting point clouds because we do not require the connectivity of the mesh. Although PartField [13] performs well in the second task, once connectivity information is removed, its performance drops significantly, indicating that the PartField method is not robust to meshes without connected components. This is further evidenced by the comparison on watertight data, where the absence of connectivity also affects its performance. In contrast, our method consistently achieves the best performance regardless of whether connectivity information is present or not. This indicates that our approach effectively learns the geometric features of objects, enabling accurate part segmentation. In interactive segmentation, our method also performs the best, thanks to our unique prompt point segmentation head and IOU prediction module. These components enable precise multi-scale segmentation predictions and automatically select the optimal results.

The comparisons on PartObj-Tiny-WT and PartNetE are shown in Table 3. Since watertight data and point clouds lack connectivity information, PartField's performance is not as good as on non-watertight data. This again validates that our method effectively learns the geometric features of objects. Here, SAMesh will get stuck when processing watertight meshes due to the high number of faces. Another observation is that the metrics for interactive segmentation are lower than those for full segmentation. This is because, in the evaluation of interactive segmentation, the method can only rely on a single prompt point, focusing more on the accuracy of individual mask segmentation. This further

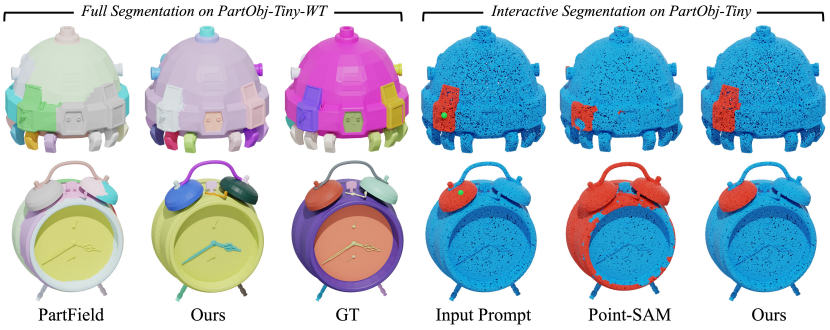

Fig. 5. The comparison of our method with previous methods on PartObj-Tiny and PartObj-Tiny-WT across different tasks including full segmentation and interactive segmentation. Here, the green points indicate the input prompt pints.

validates the precision of our approach. Comparisons across various datasets and tasks, involving different data forms such as non-watertight meshes, watertight meshes, and point clouds, demonstrate the remarkable effectiveness, robustness and generalization ability of our methods, confirming its superior performance under diverse conditions.

We also conduct a qualitative comparison of our method and previous methods on PartObj-Tiny for full segmentation with connectivity, as shown in Figure 4. SAMesh tends to over-segment objects, while other methods struggle with handling complex objects, resulting in inaccurate masks, failure to separate multiple part masks, and other segmentation errors. However, our method can accurately segment complex objects, even performing part segmentation on the lizard and beetle in the scene of the last row. Figure 5 shows a comparison between our method and PartField on the PartObj-Tiny-WT dataset for full segmentation without connectivity. PartField struggles to segment watertight meshes, while our method can maintain the segmentation quality. Figure 5 also illustrates the interactive segmentation results of our method and Point-SAM given the green point prompts, where our segmentation results exhibit accurate boundaries and scales. The results show that our method can predict masks with a reasonable number of categories, clear and accurate boundaries, and can handle complex models as well as different types of data and tasks. They also demonstrate the great generalization and robustness of our approach.

4.3 Applications

Multi-Prompts Auto-Segmentation. Our method can also segment the object given several point prompts that indicate specific parts. In this setting, we have a strong condition where each prompt corresponds to a specific part of the object. Therefore, instead of using the predicted IoU to evaluate mask quality, we only need to select one mask for each part such that all masks collectively cover the entire object as much as possible while minimizing overlap between the masks. As shown in Figure 6, the multi-prompts segmentation can follow the user's instructions. Compared to automatic segmentation, it can both extend unsegmented regions, such as the horse's body, and merge over-segmented regions, such as flower petals.

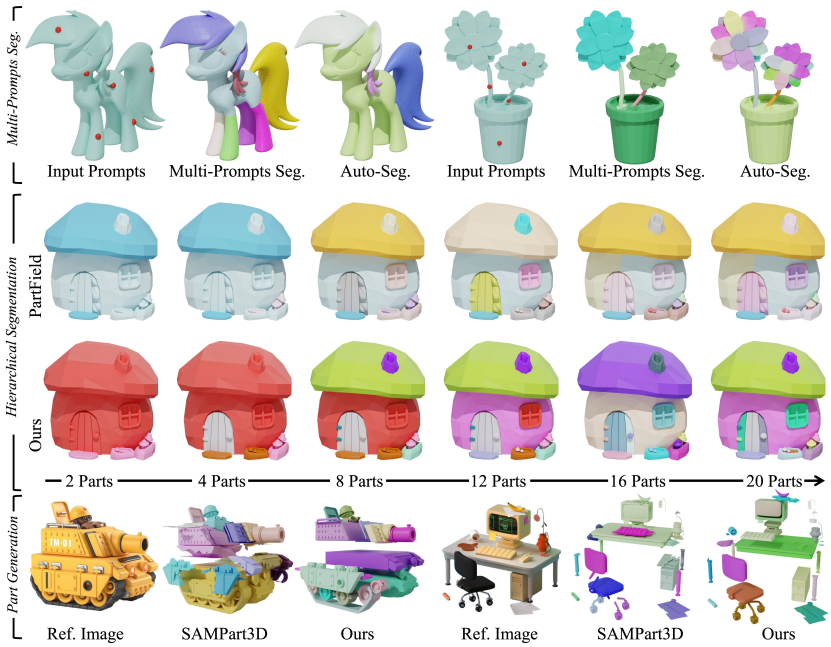

Fig. 6. The applications of our method including multi-prompts and hierarchical part segmentation and part generation.

Hierarchical Part Segmentation. After segmentation, we can obtain the average feature of each part by finding the corresponding point-wise feature $\mathbf{f}_p$. We then directly employ hierarchical clustering on these average features to obtain hierarchical segmentation results. As shown in Figure 6, our hierarchical segmentation results effectively aggregate different parts at various levels, validating the effectiveness of our feature extraction method in accurately representing the information of each part. Compared to the results from PartField, our method's aggregation better adheres to the relationships between parts.

Part Generation. Our results can also be applied to part generation as an instruction for splitting objects. Figure 6 demonstrates the exploded part generation results of HoloPart[30] when given the segmentation masks from SAMPart3D and ours. Our more accurate segmentation masks can significantly improve HoloPart's performance.

4.4 Ablation Study

We first conduct ablation studies on our network architecture. The feature extractor can be any point cloud encoder, and we select the current state-of-the-art one that is Sonata. The IoU predictor is critical for selecting the best mask; without it, our method cannot function properly. We then focus on conducting ablation studies of our two-stage multi-head segmentor. As shown in Table 4, we evaluate four ablated models and our full method on the test set of our dataset. The first model contains only one segmentation head of the first stage. The second and third models respectively include only the first and second stages.

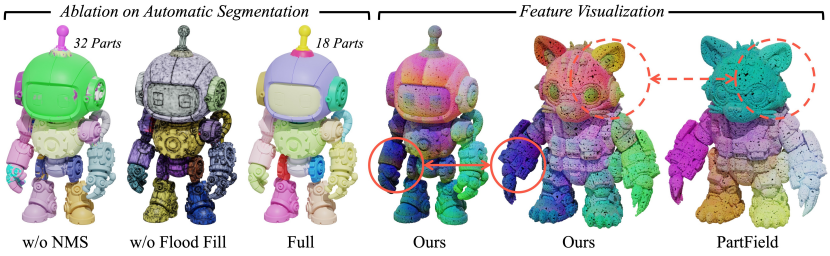

Fig. 7. The ablation study of our automatic segmentation and the visualization of our point-wise feature.

The fourth model includes both stages but is trained without data augmentation. The last model is our full version. This progressive ablation study clearly demonstrates the importance of each component in our method. Notably, the difference between the second and third models is that the latter extracts a global feature during segmentation. The better performance metrics of the third model highlight the importance of this global feature. As shown in Figure 7, we also present the full segmentation results without using the NMS or flood fill algorithm in our automatic segmentation approach. The results highlight the necessity of these two steps, as without them, the segmentation masks are not complete, have unclear boundaries, and do not yield a reasonable number of parts. Figure 7 also visualizes the point-wise features of objects. The results show that for the same type of data, our method produces similar features for corresponding parts, while our features can capture more detailed geometry compared to PartField, such as the eyes and ears of the person on the right. This fully demonstrates the advantages of our method in extracting accurate features.

Table 4. The comparison of four ablated methods and our full method.

	w/o Augmentation				w/ Augmentation
Ablation	Single-Head	Stage 1 Only	Stage 2 Only	Stage 1 + Stage 2	Full
mIoU	0.2801	0.4265	0.6647	0.7464	0.7906

5 Limitations and Conclusions

In this paper, we propose P^3-SAM, a native 3D part segmentation method. Our approach employs Sonata to extract point-wise features and uses a two-stage multi-head segmentor to predict multi-scale masks given a point prompt indicating a part. An IoU predictor is employed to evaluate and select the best mask. We also propose an automatic segmentation approach using our P^3-SAM. We train our model on 3.7 million models, resulting in a part segmentation method with high accuracy, generalization, and robustness across various tasks and data types. Our method is also flexible and can be applied to multiple applications such as real-time interactive, hierarchical, or multi-prompt part segmentation. We observe that our method may rely too heavily on the geometric information of the object's surface and lacks an understanding of the spatial volume of the object's parts. This is because our training data consists solely of surface

point clouds. Therefore, future work may focus on developing models with spatial segmentation capabilities to broaden their applicability to a wider range of tasks.

References

1. Abdelreheem, A., Skorokhodov, I., Ovsjanikov, M., Wonka, P.: Satr: Zero-shot semantic segmentation of 3d shapes. In: ICCV (2023)
2. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: CVPR (2016)
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
4. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3d mesh segmentation. Acm transactions on graphics (tog) (2009)
5. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017)
6. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., VanderBilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., Farhadi, A.: Objaverse-xl: A universe of 10m+ 3d objects. arXiv preprint arXiv:2307.05663 (2023)
7. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. arXiv preprint arXiv:2212.08051 (2022)
8. Garosi, M., Tedoldi, R., Boscaini, D., Mancini, M., Sebe, N., Poiesi, F.: 3d part segmentation via geometric aggregation of 2d visual features. In: WACV (2025)
9. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: a network with an edge. ACM Transactions on Graphics (ToG) **38**(4), 1–12 (2019)
10. Huang, R., Peng, S., Takmaz, A., Tombari, F., Pollefeys, M., Song, S., Huang, G., Engelmann, F.: Segment3d: Learning fine-grained class-agnostic 3d segmentation without manual labels. In: ECCV (2024)
11. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4015–4026 (2023)
12. Li, L.H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.N., et al.: Grounded language-image pre-training. In: CVPR (2022)
13. Liu, M., Uy, M.A., Xiang, D., Su, H., Fidler, S., Sharp, N., Gao, J.: Partfield: Learning 3d feature fields for part segmentation and beyond (2025)
14. Liu, M., Zhu, Y., Cai, H., Han, S., Ling, Z., Porikli, F., Su, H.: Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In: CVPR (2023)
15. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In: ECCV (2024)
16. Ma, Z., Yue, Y., Gkioxari, G.: Find any part in 3d. arXiv preprint arXiv:2411.13550 (2024)

17. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: 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. pp. 909–918 (2019)
18. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
19. Ošep, A., Meinhardt, T., Ferroni, F., Peri, N., Ramanan, D., Leal-Taixé, L.: Better call sal: Towards learning to segment anything in lidar. In: ECCV (2024)
20. Peng, S., Genova, K., Jiang, C., Tagliasacchi, A., Pollefeys, M., Funkhouser, T., et al.: Openscene: 3d scene understanding with open vocabularies. In: CVPR (2023)
21. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
22. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
23. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
24. Tang, G., Zhao, W., Ford, L., Benhaim, D., Zhang, P.: Segment any mesh: Zero-shot mesh part segmentation via lifting segment anything 2 to 3d. arXiv:2408.13679 (2024)
25. Umam, A., Yang, C.K., Chen, M.H., Chuang, J.H., Lin, Y.Y.: Partdistill: 3d shape part segmentation by vision-language model distillation. In: CVPR (2024)
26. Wu, X., DeTone, D., Frost, D., Shen, T., Xie, C., Yang, N., Engel, J., Newcombe, R., Zhao, H., Straub, J.: Sonata: Self-supervised learning of reliable point representations. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 22193–22204 (2025)
27. Wu, X., Jiang, L., Wang, P.S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., Zhao, H.: Point transformer v3: Simpler faster stronger. In: CVPR (2024)
28. Xu, M., Yin, X., Qiu, L., Liu, Y., Tong, X., Han, X.: Sampro3d: Locating sam prompts in 3d for zero-shot scene segmentation. arXiv preprint arXiv:2311.17707 (2023)
29. Xue, Y., Chen, N., Liu, J., Sun, W.: Zerops: High-quality cross-modal knowledge transfer for zero-shot 3d part segmentation. arXiv:2311.14262 (2023)
30. Yang, Y., Guo, Y.C., Huang, Y., Zou, Z.X., Yu, Z., Li, Y., Cao, Y.P., Liu, X.: Holopart: Generative 3d part amodal segmentation. arXiv preprint arXiv:2504.07943 (2025)
31. Yang, Y., Huang, Y., Guo, Y.C., Lu, L., Wu, X., Lam, E.Y., Cao, Y.P., Liu, X.: Sampart3d: Segment any part in 3d objects. arXiv preprint arXiv:2411.07184 (2024)
32. Yang, Y., Wu, X., He, T., Zhao, H., Liu, X.: Sam3d: Segment anything in 3d scenes. arXiv preprint arXiv:2306.03908 (2023)
33. Zhong, Z., Xu, Y., Li, J., Xu, J., Li, Z., Yu, C., Gao, S.: Meshsegmenter: Zero-shot mesh semantic segmentation via texture synthesis. In: ECCV (2024)
34. Zhou, P., Dong, X., Cao, J., Chen, Z.: Met: mesh transformer with an edge. The Visual Computer **39**(8), 3235–3246 (2023)
35. Zhou, Y., Gu, J., Chiang, T.Y., Xiang, F., Su, H.: Point-sam: Promptable 3d segmentation model for point clouds (2024), `https://arxiv.org/abs/2406.17741`

36. Zhou, Y., Gu, J., Li, X., Liu, M., Fang, Y., Su, H.: Partslip++: Enhancing low-shot 3d part segmentation via multi-view instance segmentation and maximum likelihood estimation. arXiv:2312.03015 (2023)