

PARTFIELD: Learning 3D Feature Fields for Part Segmentation and Beyond

Supplementary Material

In this supplement, we provide details on our hard negative mining strategy, the baselines and additional qualitative results. We also refer readers to the included supplemental video, which gives additional results and renderings from multiple views.

1. Details of Hard Negative Mining

Positives and negatives for our contrastive learning training strategy are defined on each part mask (a SAM mask for 2D and a part label mask for 3D proposals). We first randomly sample K masks for each batch of shapes. For each mask, we then sample $N = 64$ positive pairs $(\mathbf{p}_a, \mathbf{p}_b)$ using the mask labels. Then we sample $M_1 = 256$ random negatives \mathbf{p}_c^1 using uniform strategy. For 3D-hard strategy, we sample $M_2 = 256$ hard negatives \mathbf{p}_c^2 also using the mask labels. The hard negatives are drawn based from a distribution weighted by the negative Euclidean distance of each negative candidate \mathbf{p}_c to query \mathbf{p}_a . That is $\text{prob}(c_1^{(i)}) = \frac{\text{dist}(c_1^{(i)}, A)}{\sum_c \text{dist}(c, A)}$. For feature-hard, we use a similar strategy as 3D-hard, except the distance metric is computed on the feature space. Then the loss for a given triplet $(\mathbf{p}_a, \mathbf{p}_b, \{\mathbf{p}_c\})$ is as follows:

$$\mathcal{L} = -\frac{1}{2} \left(\log \left(\frac{\text{sim}(f(\mathbf{p}_a), f(\mathbf{p}_b))}{\text{sim}(f(\mathbf{p}_a), f(\mathbf{p}_b)) + \sum_{\mathbf{p}_c} \text{sim}(f(\mathbf{p}_a), f(\mathbf{p}_c))} \right) + \log \left(\frac{\text{sim}(f(\mathbf{p}_b), f(\mathbf{p}_a))}{\text{sim}(f(\mathbf{p}_b), f(\mathbf{p}_a)) + \sum_{\mathbf{p}_c} \text{sim}(f(\mathbf{p}_b), f(\mathbf{p}_c))} \right) \right) \quad (1)$$

2. Details of Baseline Comparison

SAMPart3D [7]. We use the official codebase and the released checkpoint ¹. The original code takes meshes as input and predicts mesh face labels. We directly uses the output to compute the metrics without any output alignment or label transfer. We obtain predictions across scales ranging from 0.0 to 2.0 at intervals of 0.125, compute the metric for each scale, and select the one with the best metric for each shape.

Find3D [3]. We use the official codebase and the released checkpoint ². Following the instructions, we sample 5,000 points per shape, assigning them a dummy white color. We input the ground truth text prompts into the network, which then predicts point-wise labels corresponding to the text prompt. The point-wise labels are transferred to the input

mesh by finding the nearest neighbor for the center of each input mesh face.

SAMesh [4]. We use the official codebase ³. The original code takes meshes as input and predicts mesh face labels. For the PartObjaverseTiny dataset [6], we use the default settings, which produce reasonably good results. However, the default settings fail for the PartNetE dataset [2]. We contacted the authors via email and confirmed that this issue arises due to the low-poly characteristics and poor mesh topology of the PartNet meshes. For the PartNetE dataset, we refined the hyperparameters and made modifications to the code following the authors’ suggestions, achieving better results compared to the default setup.

PartSLIP [2]. We used the official codebase and released checkpoint ⁴. We employed the zero-shot version and utilized the instance segmentation predictions. Since the model takes a dense point cloud as input, we first rendered multi-view RGB images and depth maps of the input 3D meshes, which we then fused to obtain a colored dense point cloud. We fed both the dense point cloud and the ground truth part names into the network, which predicted the corresponding point-wise labels. These labels were transferred to the input mesh by finding the nearest neighbor to the center of each mesh face.

Ultrametric Feature Field [1]. We used the official codebase ⁵. Since the method takes multi-view images as input and performs a NeRF optimization, we first rendered multi-view images following the camera poses provided for the example asset in the code repository. We then followed the code instructions to generate multi-view SAM predictions and their corresponding hierarchy, and ran the NeRF optimization using the provided default configuration. After the NeRF optimization, we used ground truth multi-view depth to extract a fused point cloud and query its corresponding predicted features. Finally, we utilized the provided clustering algorithm, converted the clustering labels back to the 3D input mesh by finding the nearest neighbor for the center of each input mesh face, and evaluated 20 scales—computing the metric for each scale and selecting the one with the best performance for each shape.

³<https://github.com/gtang12/samesh>

⁴<https://drive.google.com/drive/u/0/folders/19j6PZfW8TDQ1lfHZwHIhn6X4BHjYRFCL>

⁵https://github.com/hardyho/ultrametric_feature_fields

¹<https://github.com/Pointcept/SAMPart3D>

²<https://github.com/ziqui-ma/Find3D>

3. Multiclass Regression Interactive Cosegmentation

In addition to the clustering-based cosegmentation described in Section 4.3, the supplemental video includes an additional demonstration where we interactively cosegment all of the guitar shapes from the COSEG dataset [5]. The user clicks a small number of points on any shape, which are then extended in real-time to segmentations of all shapes, thanks to our precomputed feature field on each shape.

This also demonstrates an alternate learning-based setup using our feature field. Rather than clustering, we use user-annotated points as a (very small) training set to fit a logistic regression model for one-vs-rest multiclass classification, mapping from our features to the segmentation label. These models are easily fit and evaluated on all shapes in real time on the GPU, providing the user with helpful interactive feedback.

4. PartNetE Class Grouping

When reporting PartNetE results, we grouped the original 45 classes into five clusters to save space. Here is the mapping:

- Electronics & Computing Devices: Keyboard, Mouse, Laptop, Phone, Camera, USB, Display (monitor), Remote, Printer, Switch (if treated as a network or power switch)
- Large Home Appliances: WashingMachine, Dishwasher, Refrigerator, Oven, Microwave
- Kitchen & Food-Related Items: KitchenPot, Kettle, Toaster, CoffeeMachine, Faucet, Dispenser, Knife, Bottle, Bucket (often used in kitchen/cleaning contexts)
- Furniture & Household Infrastructure: Table, Chair, FoldingChair, StorageFurniture, Door, Window, Lamp, TrashCan, Safe (often a household or office fixture)
- Tools, Office Supplies, & Miscellaneous: Stapler, Scissors, Pen, Pliers, Lighter, Box, Cart (e.g., utility cart), Globe (decorative/educational), Suitcase (travel/personal), Eyeglasses (personal), Clock

References

- [1] Haodi He, Colton Stearns, Adam W. Harley, and Leonidas J. Guibas. View-consistent hierarchical 3d segmentation using ultrametric feature fields, 2024. [1](#)
- [2] Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and Hao Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21736–21746, 2023. [1](#)
- [3] Ziqi Ma, Yisong Yue, and Georgia Gkioxari. Find any part in 3d, 2024. [1](#)
- [4] 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, 2024. [1](#)
- [5] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6): 1–10, 2012. [2](#)
- [6] Mutian Xu, Xingyilang Yin, Lingteng Qiu, Yang Liu, Xin Tong, and Xiaoguang Han. Sampro3d: Locating sam prompts in 3d for zero-shot scene segmentation. *arXiv preprint arXiv:2311.17707*, 2023. [1](#)
- [7] 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. [1](#)



Figure 1. Additional qualitative examples segmentation results.

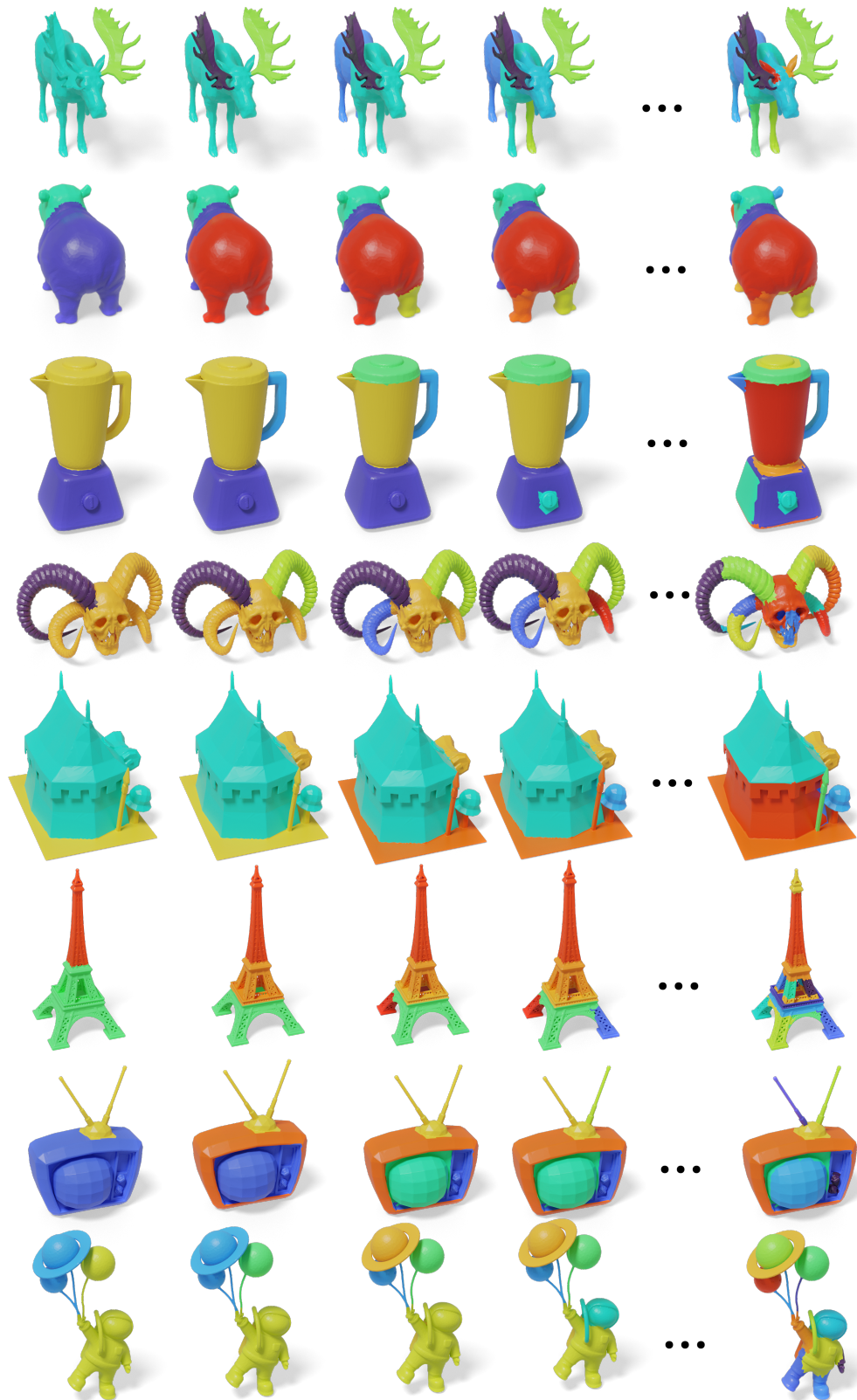


Figure 2. Additional examples of hierarchical segmentation using our method.

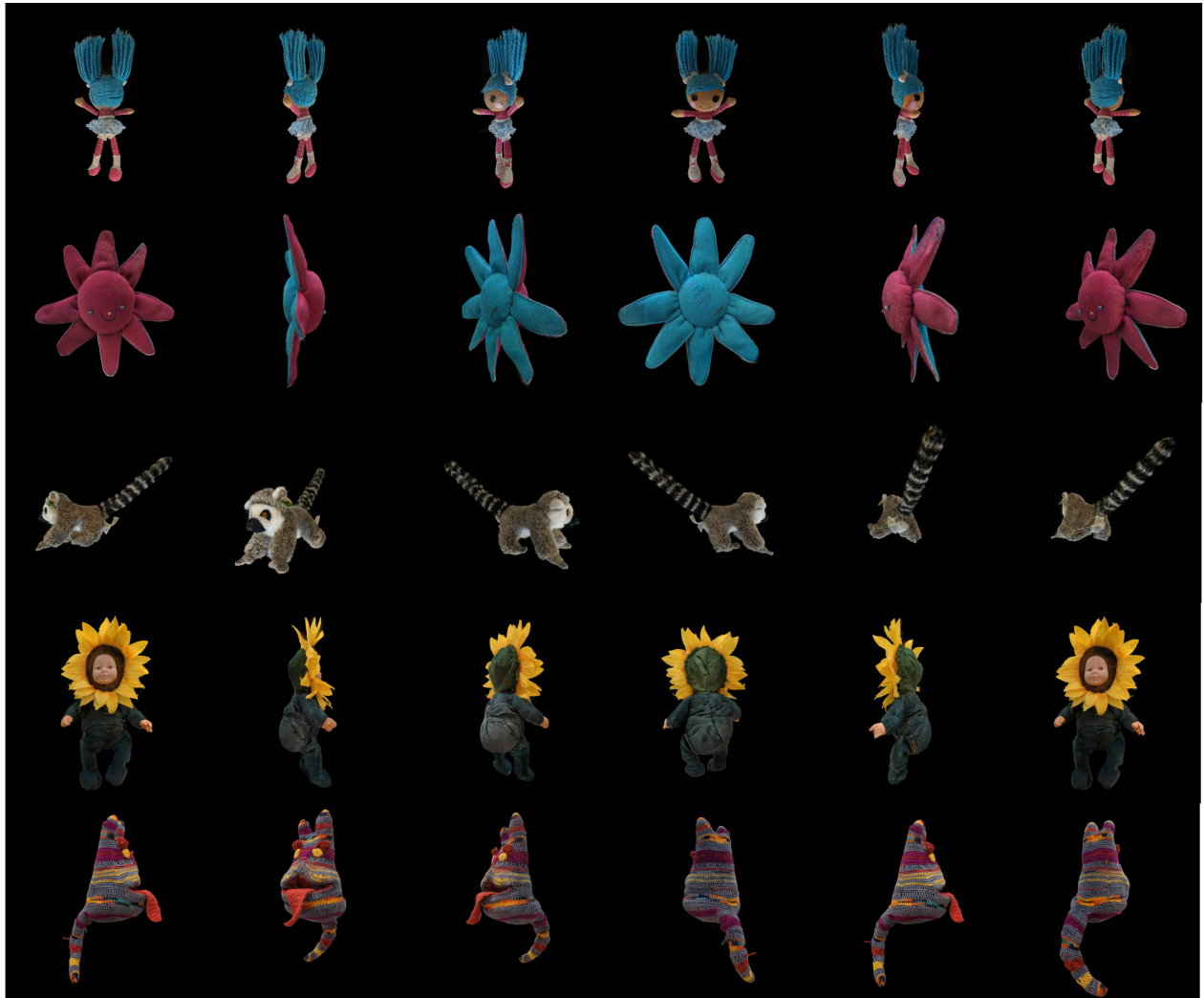


Figure 3. Sample images used for 3D Gaussian splatting reconstruction.