OpenShape: Scaling Up 3D Shape Representation Towards Open-World Understanding - Supplementary Material

Anonymous Author(s)
Affiliation
Address
email

1 More Examples of Multi-Modal 3D Shape Retrieval

In Figures 1 and 2, we showcase more examples of multi-modal 3D shape retrieval.

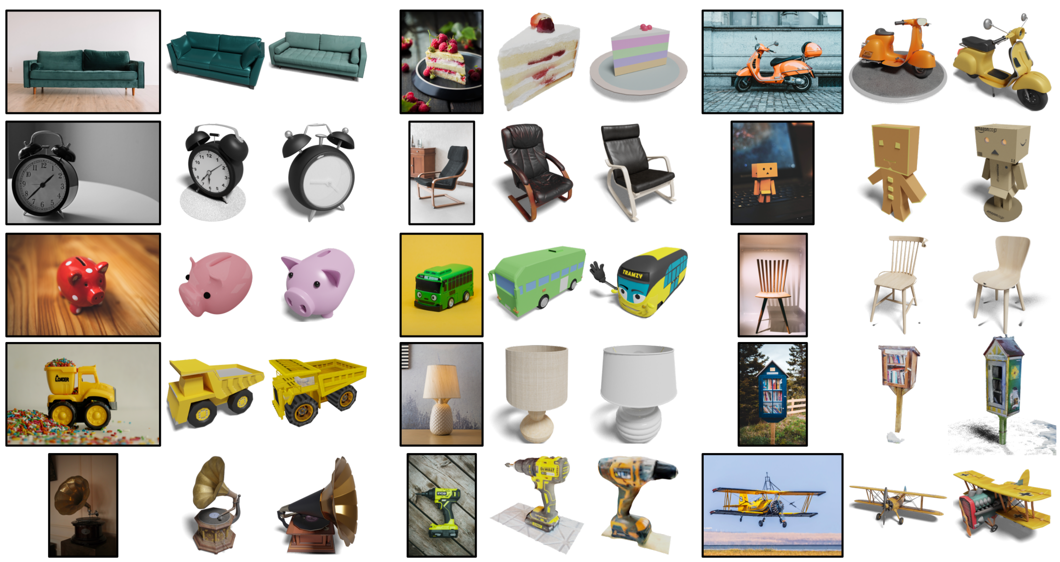

Figure 1: **Image-input 3D shape retrieval.** In each triplet, we present the input image and two 3D shapes retrieved using OpenShape embeddings from the Objaverse [2] dataset. Input images are from `unsplash.com`.

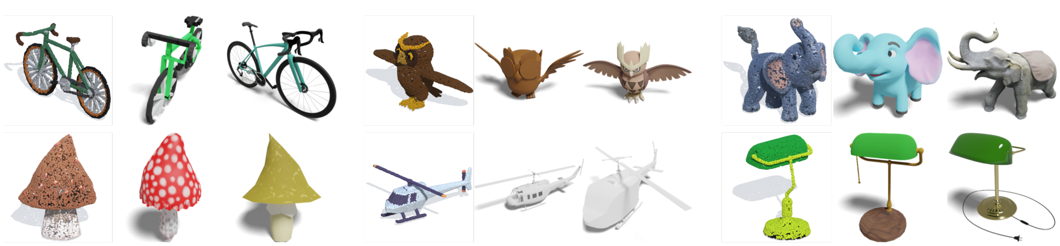

Figure 2: **Point cloud-input 3D shape retrieval.** In each triplet, we present the input point cloud and two 3D shapes retrieved using OpenShape embeddings from the Objaverse [2] dataset.

2 More Examples of Shape-Conditioned Multimodal Generation

In Figure 3 and Figure 4, we showcase more examples of point cloud captioning and point cloud-conditioned image generation.

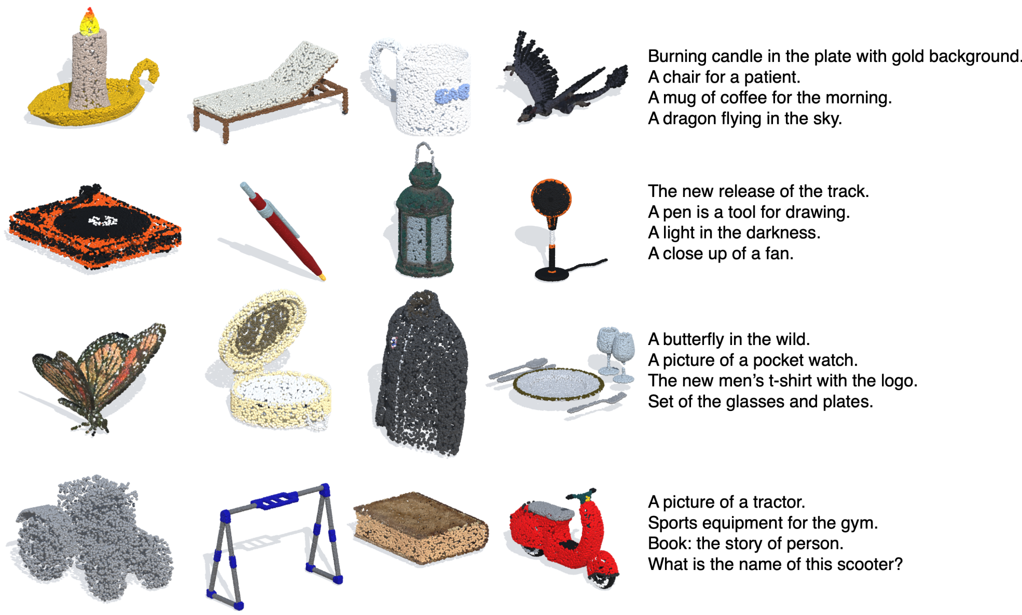

Figure 3: **Point cloud captioning**. In each row, we show the input point clouds on the left and the generated captions on the right.

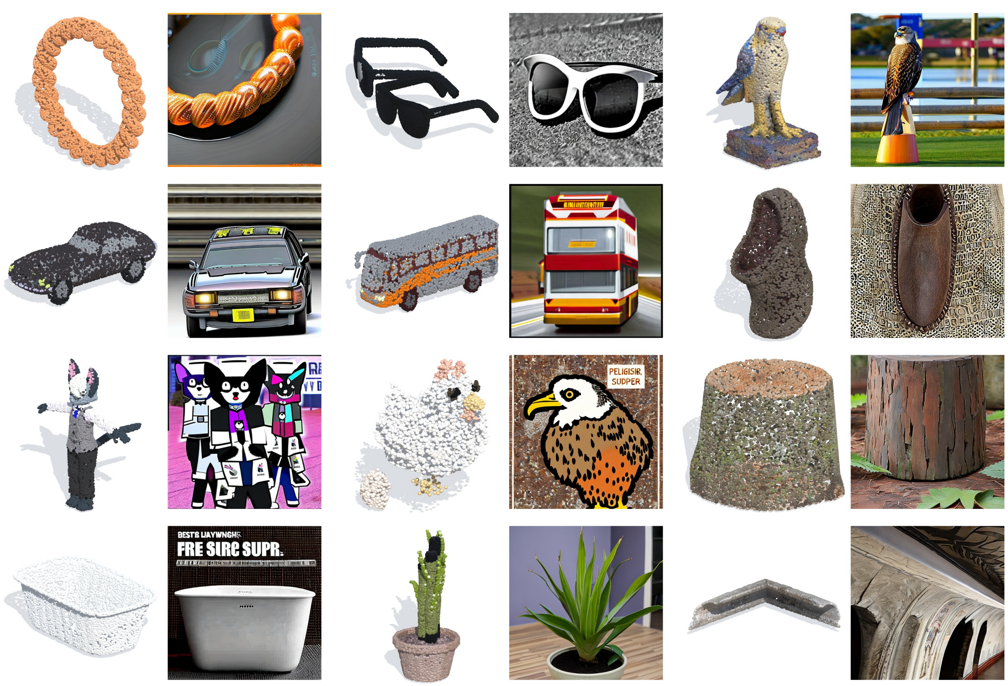

Figure 4: **Point cloud-conditioned image generation**. Each row shows three examples (input point clouds and generated images).

3 Details on Raw Text Generation and Filtering

3.1 Raw Text Generation

We leverage the metadata from the four datasets to generate the raw texts. Although the original datasets may contain numerous attributes for each shape, we carefully choose the most informative ones to compose the text, ensuring its quality and relevance.

Objaverse:We utilize the `name` associated with each shape to serve as the text.

ShapeNetCore: For each shape, we generate three types of texts: (a) the `name`, (b) the `category name` (with a total of 55 categories), and (c) the concatenation of the `sub-category names` (with a total of 336 sub-categories), separated by commas.

3DFuture: For each shape, we generate two types of texts: (a) the `category`, and (b) the concatenation of `category`, `style`, `theme`, and `material`, separated by commas.

ABO: For each shape, we generate two types of texts: (a) the `item_name`, and (b) the `product_type`.

In this way, we generate one or more raw texts for each shape.

3.2 Raw Text Filtering

We employ GPT-4 [4] to filter out uninformative raw texts. To accomplish this, we divide all the raw texts into batches, each containing 256 entries, and process each batch independently using GPT-4. Here is an example illustrating the prompt we used and the corresponding response generated by GPT-4.

I am analyzing a 3D dataset with various text descriptions for the 3D models. However, many of these texts are inaccurate or uninformative, and therefore, not suitable as descriptions for 3D models. I need your help to identify such incorrect texts. Specifically, if a text primarily consists of irrelevant or uninformative content, such as timestamps, model numbers, incomprehensible descriptions, random filenames (e.g., "my project"), random characters, etc., please respond with "N". If a text contains a clear noun (or noun phrase) that could potentially describe a 3D object, please respond with "Y". You will find a list of texts below, and each line contains a three-digit ID and associated text. For each text, please respond with "Y" or "N", following the ID number (e.g., "001 Y" or "002 N"). Please evaluate all 256 texts.
000 New project (19)
001 3December - Chemestry
002 Fake Brand Soda Can
003 Spartan Shild
004 Apple3d
005 Landmine
006 FaunveinB-S
007 FIGURA 5
008 Sphero Blue
009 Sofa
010 Maddox
011 A3 Complete
012 Suspension Bridge
013 Maung
014 Captain-americas-shield
015 sphorb4
......

000 N
001 Y
002 Y
003 Y
004 Y
005 Y
006 N
007 N
008 Y
009 Y
010 N
011 N
012 Y
013 N
014 Y
015 N
......

Afterward, we combine all the responses to create the final filtering results, effectively removing approximately 30% of the raw texts.

4 Details on the Backbone Scaling Experiment

In Figure 4 of the main paper, we investigate the performance and scalability of various backbones when scaling up their model sizes. For this experiment, we employ a default resolution of 10,000 points for input point clouds, a batch size of 200, and conduct the experiment on a single A100 GPU. In general, if instructions are given in the original paper of a backbone, we scale up the model as instructed. Otherwise, we scale up the model by expanding width or depth (i.e., stacking blocks or layers). Specifically, we scale up each backbone as follow:

PointBERT [13] The scaling parameters are shown in Table 1. We scaled PointBERT to 72.1M parameters beyond the 32.3M version reported in Figure 4 of the main paper. However, at this scale, the model dramatically overfits on the training data and performs worse on all benchmarks than the 32.3M version.

Table 1: Hyperparameters for scaling up PointBERT [13].

# Parameters	# Layers	Width	# Heads	MLP Dim	# Patches	Patch Embed Dim
5.1M	6	256	4	1024	64	96
13.3M	6	512	8	1024	64	128
32.3M	12	512	8	1536	384	256
72.1M	12	768	12	2304	512	256

SparseConv [1] The smallest version (5.3M parameters) of the model is adapted from the MinkowskiFCNN model by adjusting the width of the final convolution and linear layers. The remaining three models are adaptations of MinkowskiResNet, each varying in the number of basic ResNet blocks used. See Table 2 for the specific scaling parameters.

Table 2: Hyperparameters for scaling up SparseConv [1].

# Parameters	# Convolution Layers	# Linear Layers
5.3M	7	4
29.0M	18	3
33.7M	26	3
41.3M	42	3

PointNeXt [7] PointNeXt is proposed as a scalable version of PointNet++ [6], and includes S/B/L/XL variants in the original paper. We simply adopt these official configurations.

DGCNN [10] and PointNet [5] For these two backbones without a hierarchical structure, we increase the width of each layer proportionally to scale up to 4xPointNet and 2xDGCNN before we hit the GPU memory limit. As the models operate completely on dense points, it is impractical to use the default 10k-point resolution. We thus reduce the input resolution for the two backbones, resulting in 1k points for DGCNN and 4k points for PointNet.

5 Details on Training and Evaluation

Training Details We freeze the CLIP text and image encoders and train the 3D encoder and two projection heads on our ensembled dataset using the cross-modal contrastive loss. We train the model on a single A100 GPU with a batch size of 200. Since we precache the text and image CLIP embeddings of all shapes, the training is greatly accelerated and takes about 300 A100 hours for convergence. We utilize an exponential learning rate schedule, and employ an range test to find the initial learning rate. For 32.3M version of PointBERT, we utilize a learning rate of $5e-4$; for 72.1M version of PointBERT, we utilize a learning rate of $4e-4$; and for other models, we utilize a learning rate of $1e-3$. For hard-negative mining, the number of seed shapes s is set to 40, and the number of neighbors m is set to 5 per shape, and the threshold δ is set to 0.1.

Fine-tuning CLIP Text and Image Encoders? After training OpenShape-PointBERT, we conducted experiments to unfreeze and finetune the CLIP text encoder for a single epoch. However, the results obtained did not demonstrate any noticeable improvement on the benchmarks. Moreover, we observed that finetuning the CLIP text encoder could potentially undermine the generalization capabilities of CLIP and hinder the integration of OpenShape embeddings into existing CLIP-based models. As a result, we choose to freeze the CLIP encoders throughout the entire training process.

Evaluation Details We evaluated all baselines using their publicly released pretrained checkpoints. Additionally, we retrained ULIP [12] on our ensembled training shapes using their official code base and backbone networks. Note that the retrained ULIP model utilized the original raw texts from the four datasets during training (prompt engineering is also applied), rather than our filtered and enriched texts. For ModelNet40 [11], the evaluation is conducted on the test split with 2,468 shapes. Regarding ScanObjectNN [9], we follow ULIP [12] to evaluate on the OBJ_ONLY version, which contains 581 test shapes. For Objaverse-LVIS [2], the input is 10,000 sampled points with point colors. For ModelNet40 [11], the input is 10,000 sampled points without color. For ScanObjectNN [9], we utilize the official 2,048 points without color as input. All methods use the same input during evaluation. The forward inference time on an A100 GPU for a 10,000-point point cloud is approximately 0.9ms for OpenShape-SparseConv and 3.8ms for OpenShape-PointBERT.

6 Details on Shape-Conditioned Multimodal Generation

Point Cloud Captioning CLIPCap [3] utilizes a 10-token prefix generated from CLIP image embeddings to enable GPT-2 for captioning. In order to align with the off-the-shelf CLIPCap model, we trained a variant of OpenShape-PointBERT that employs CLIP ViT-B/32 embeddings instead of OpenCLIP ViT-G/14 used in other experiments. Consequently, we directly input the point cloud encoding, *without normalization*, into CLIPCap for captioning.

Point Cloud Conditioned Image Generation We take the Stable Diffusion v2.1 unCLIP model [8] for image generation and replace the CLIP image condition encoder with our OpenShape encoder to perform image generation conditioned on point clouds (and optionally text prompts). The unCLIP model takes CLIP ViT-L/14 embeddings without normalization as input. To match the embedding space, we trained a variant of OpenShape-PointBERT with CLIP ViT-L/14 embeddings. Additionally, we noticed a significant mismatching of scales (L_2-norm of embedding vectors) between ViT-L/14 image embeddings and OpenShape embeddings. To mitigate this issue, we perform a re-normalization on OpenShape embeddings to a L_2-norm of $\frac{1}{2}\sqrt{768}$, which is our observed mean L_2-norm of ViT-L/14 image embeddings. We use 50 diffusion steps. The guidance scale can be tuned freely.

References

[1] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[3] Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[4] OpenAI. Gpt-4 technical report, 2023.

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[6] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[7] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *arXiv:2206.04670*, 2022.

[8] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[9] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

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

[11] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[12] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning unified representation of language, image and point cloud for 3d understanding. *arXiv preprint arXiv:2212.05171*, 2022.

[13] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.