COSE474-2024F: Final Project
From Geometry to Semantics: A Deep Learning Approach to 3D Object Descriptions

2020100010 Sehyeon Park

1. Introduction

1.1. Motivation

Converting complex 3D mesh data into simple and meaningful language is important for many purposes, such as helping users interact with virtual environments or making it easier to search for 3D models in libraries. This project aims to provide a new way to understand and interact with 3D content by transforming 3D mesh data into a descriptive sentence.

1.2. Problem definition

Despite being a relatively straightforward task, creating a model that can describe 3D models in natural language has received far less attention compared to other 3D deep learning models. Most research in the 3D domain focuses on tasks like segmentation, or reconstruction, while generating descriptions for 3D objects remains underexplored.
This project aims to bridge this gap by building a pipeline that takes 3D models as input, renders them into 2D images, and uses a foundation models to generate accurate text descriptions.

1.3. Contriubtion

The main contribution of this project is creating a framework that efficiently processes 3D mesh data using image-based representation, extracts a class through an image classification model, and generates coherent and contextual descriptions with a language model. This approach offers a new way of dealing with 3D mesh representation. In addition, By connecting 3D data with natural language, this approach can make 3D content more accessible and easier to understand, helping applications like XR environments.

2. Related Works

CLIP(Radford et al., 2021) bridges the gap between visual and textual domains. By converting 3D mesh data into 2D images, CLIP can be used to classify objects, which can then be transformed into descriptions in a large language model. Although it lacks direct 3D geometry awareness, its proven capabilities as a pre-trained foundation model provide a practical solution for this task. In addittion, ULIP(Xue et al., 2023) proposes a unified pipeline to concatenate language, images, and 3D point clouds into a shared representation space. Similar to ULIP, the project integrates information by rendering 3D meshes into 2D images and learning image-text relationships through a fine-tuned CLIP model. ULIP's approach provides valuable insights for bridging 3D data and natural language, which aligns closely with my project goals.
To complement these visual models, LLaMA(Touvron et al., 2023), a large language model, offers a framework for generating context-aware text. Its pre-training on extensive text makes it a component for generating accurate and natural descriptions from extracted features, bridging the gap between visual and descriptive understanding.
Datasets like ShapeNet(Chang et al., 2015) and Objaverse 1.0(Deitke et al., 2022) serve as a cornerstone for such process. ShapeNet provides structured 3D mesh data across diverse categories, making it a decent dataset for training classification models. Objaverse 1.0, with its extensive collection of over 800,000 annotated models, introduces unprecedented diversity in 3D content, including animated and rigged objects, making it an invaluable resource for advancing neural networks capable of bridging 3D geometry and semantics.

3. Methods

Algorithm 1 Data Preparation Pipeline

```
1: for each $mesh, class\_label$ in Dataset do
2:     Normalize $mesh$ scale to fit within $[-1, 1]^3$
3:     images = []
4:     for each $angle$ in $\{v1, v2, v3, v4\}$ do
5:         images.append(Render($mesh$, $angle$))
6:     end for
7:     Store $(images, class\_label)$
8: end for
```

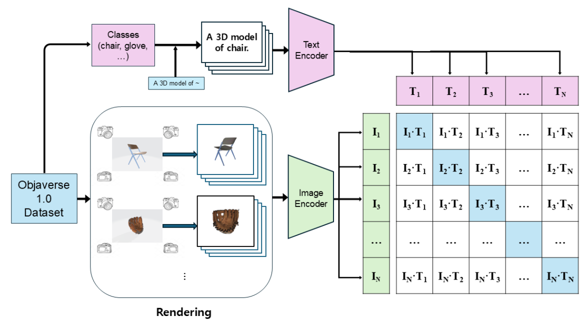

Figure 1. The training pipeline of the CLIP with 3D meshes

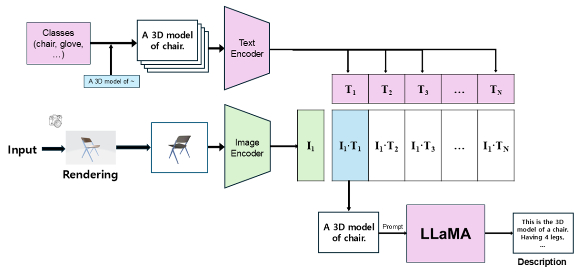

Figure 2. The pipeline concatenating the CLIP and the LLaMA

The overall training pipeline is shown in Figure 1. First, I extract 3D models and their annotations from the dataset. To prevent randomness of results, I choose the models by using the sequential indices, instead of using random number of indices. From the annotations, I focus on extracting the class information.
A potential challenge is the risk of overfitting during training due to the limited size and diverse shapes of the 3D models. To address this, As you can see Algorithm 1, data augmentation is achieved by rendering each 3D model from four different camera poses, capturing multiple viewpoints. During rendering, all models are normalized to fit within a 2×2×2 cube centered at the origin (0,0,0) to ensure consistent scaling and reproducibility.
To enhance performance, the class information is converted into a descriptive text format, such as ”A 3D model of class.” The rendered images and their corresponding class texts are then processed using the CLIP. The images are input into the CLIP's Image Encoder, while the class texts are input into the Text Encoder. For each image-text pair, the similarity of their representations is calculated using a dot product. The cross entropy loss in the training process is used as follows,

$$L = -\frac{1}{2N}\sum_{i=1}^{N}\left(\log\frac{\exp(s_{ii})}{\sum_{j=1}^{N}\exp(s_{ij})} + \log\frac{\exp(s_{ii})}{\sum_{j=1}^{N}\exp(s_{ji})}\right)$$

$$s_{ij} = \frac{\mathbf{i}_i \cdot \mathbf{t}_j}{\|\mathbf{i}_i\|\|\mathbf{t}_j\|}$$

the image-to-text loss and the text-to-image loss are calculated in average, so that I take advantage of the averaged loss function in the training process. Finally, I fine-tune the CLIP model through contrastive learning, aiming to increase the similarity between matching image-text pairs.
After the fine-tuning, the model processes a 3D mesh input by rendering it into a single image from a specific camera viewpoint. As illustrated in Figure 2, the rendered image is passed into the fine-tuned Image Encoder. This embedding is then compared with the text embeddings of pre-prepared text queries in the format ”A 3D model of class.” The model predicts the text query with the highest similarity score and outputs it.
The predicted text query is fed into the LLaMA with a pre-engineered prompt, as shown in Figure 2. Without further fine-tuning, the language model generates the final descriptive output, which is reviewed as the final prediction.

4. Experiments

4.1. Dataset

In this project, I decided to choose the Objaverse 1.0 dataset due to its scale and diversity. The Objaverse 1.0 contains about 800K 3D models, which are significantly larger than the ShapeNet's approximately 50K models. This vast scale provides not only more data for training but also greater diversity within individual classes, allowing for more robust learning and generalization.
Another key advantage of Objaverse 1.0 is its inclusion of LVIS annotations, which represent classes and model UIDs in a Python dictionary format. This makes accessing and organizing the data much more convenient compared to ShapeNet.

4.2. Computing Resources

I utilize a local server equipped with an AMD Ryzen 9 3900X 12-Core 3900X CPU, a NVIDIA RTX 3090 GPU with 24GB of VRAM, Windows 10, and CUDA 12.4. The decision to use a local server rather than the Google Colaboratory was driven by the need for better control over the environment and greater computational power during network training. Using a local server ensures that training can proceed without interruptions from time or resource constraints commonly encountered in free cloud environments.

4.3. Experimental Design

For this experiment, I selected classes from the Objaverse 1.0 dataset's LVIS annotations that had at least 10, 50, and 100 models. The number of classes in each case was 1,061, 319, and 76, respectively. For each case, I picked 10, 50, or 100 models per class sequentially and rendered images from four different camera angles. Some models were fail to be rendered due to their attributes, such as having animation,

type mismatch, etc.
The rendered images were input to the Image Encoder, while the class names were modified into the format ”A 3D model of class.” and input into the Text Encoder. All training were done with the same hyperparameters: batch size = 256, epochs = 5, learning rate = 5×10^{-6}, and AdamW as the optimizer. After training, I evaluated the model using a test dataset and measured the test accuracy. In addition, I compared the accuracy of the zero-shot CLIP model and that of my training methods.
There were the LLaMA model without any additional fine-tuning. Three versions of the model were tested: LLaMA 3.2-1B, LLaMA 3.2-3B, and LLaMA 3.1-8B. These versions vary in parameters size, allowing me to evaluate their performance across different scales.
I used two main evaluation criteria in the language model: the quality of the generated answers and the response speed. The quality was assessed by how accurate, compliable to a given prompt, while response speed was measured as the time taken to generate each output. This combination allowed me to balance the trade-offs between computational efficiency and descriptive accuracy.
The hyperparameters of LLaMA were kept consistent across all models. Specifically, the parameters were set as follows: max length = 70, the number of return sequences = 1, and temperature = 0.2.

4.4. Results

Methods	# of Classes	# of Models per Class	Test Accuracy
Zero-shot(Baseline)	1061	10	22.81%
1	1061	10	61.47%
2	319	50	80.37%
3	73	100	95.76%

Table 1. Summary of The Methods Statistics and Test Accuracy (Quantitative Result)

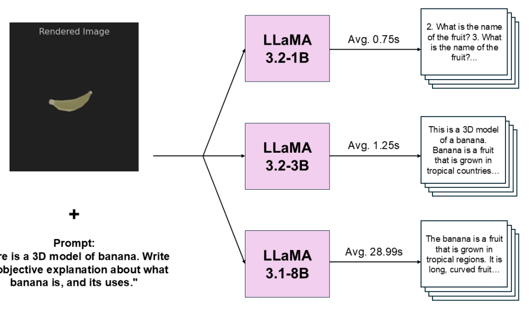
Rendered Image
+
Prompt:
"Here is a 3D model of banana. Write an objective explanation about what banana is, and its uses."
LLaMA 3.2-1B
Avg. 0.75s
2. What is the name of the fruit? 3. What is the name of the fruit?...
LLaMA 3.2-3B
Avg. 1.25s
This is a 3D model of a banana. Banana is a fruit that is grown in tropical countries...
LLaMA 3.1-8B
Avg. 28.99s
The banana is a fruit that is grown in tropical regions. It is long, curved fruit...

Figure 3. Comparison of the LLaMA models on generated descriptions with response time. (Qualitative Result)

The results of the experiments are shown in Table 1. Zero-shot CLIP, the baseline of this project, recorded a tremendously low test accuracy, 22.81%. As the number of classes in the dataset increases and the number of rendered images per class decreases, I observed a clear drop in test accuracy. When there were 1,061 classes and 10 images per class, the test accuracy was 61.47%. For 319 classes with 50 images per class, the accuracy improved significantly to 80.37%. Finally, with 73 classes and 100 images per class, the test accuracy reached the highest value of 95.76%.
Figure 3 shows the qualitative results for three different LLaMA models. The small model (LLaMA 3.2-1B) generated shorter and much less detailed, occasionally irrelevant, even weird outputs with faster response times (Avg. 0.75s). The medium model (LLaMA 3.2-3B) produced more balanced outputs, providing accurate and descriptive texts with moderate response times (Avg. 1.25s). The largest model (LLaMA 3.1-8B) created the analogous detailed descriptions but required significantly longer processing time (Avg. 28.99s).

4.5. Discussion

One reason the proposed method was successful is that it used fine-tuned CLIP. The existing zero-shot CLIP had bad performance to this 3D model classification. By introducing my method, this helped achieve good results, especially in linking images and text more effective than the zero-shot CLIP. However, a limitation of the method lies in rendering 3D models as 2D images. While this simplifies the processing, it also restricts the full representation of 3D properties, such as depth and geometry, which could limit the model’s ability to fully understand complex 3D objects.
For the LLaMA usage, the experiments revealed both the advantages and challenges of using different model sizes. The medium model, LLaMA 3.2-3B, showed the best balance between speed and description quality, making it the most practical choice for deployment on the local server setup. However, since the LLaMA model was used without fine-tuning, it may not fully capture the context-specific nuances of the generated descriptions, making this aspect of the method partially unsuccessful in achieving optimal performance.

5. Future Direction

For future work, I am going to build a model using MeshCNN(Hanocka et al., 2019), which can better capture the features of 3D meshes, and compare its results with the current approach using CLIP.
Fine-tuning the LLaMA model is another goal to generate more precise results. By tailoring the language model specifically to the context of this project, it can generate more accurate and detailed descriptions while maintaining consistency.

References

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., and Yu, F. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., and Farhadi, A. Objaverse: A universe of annotated 3d objects, 2022. URL https://arxiv.org/abs/2212.08051.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4):1–12, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Xue, L., Gao, M., Xing, C., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J. C., and Savarese, S. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding, 2023. URL https://arxiv.org/abs/2212.05171.