CS543 Final Report:
Generating High-quality 3D Assets by 3D Gaussian Primitives Generation

Yen-Chi Cheng, Hao-Yu Hsu
{yenchic3,haoyuyh3}@illinois.edu

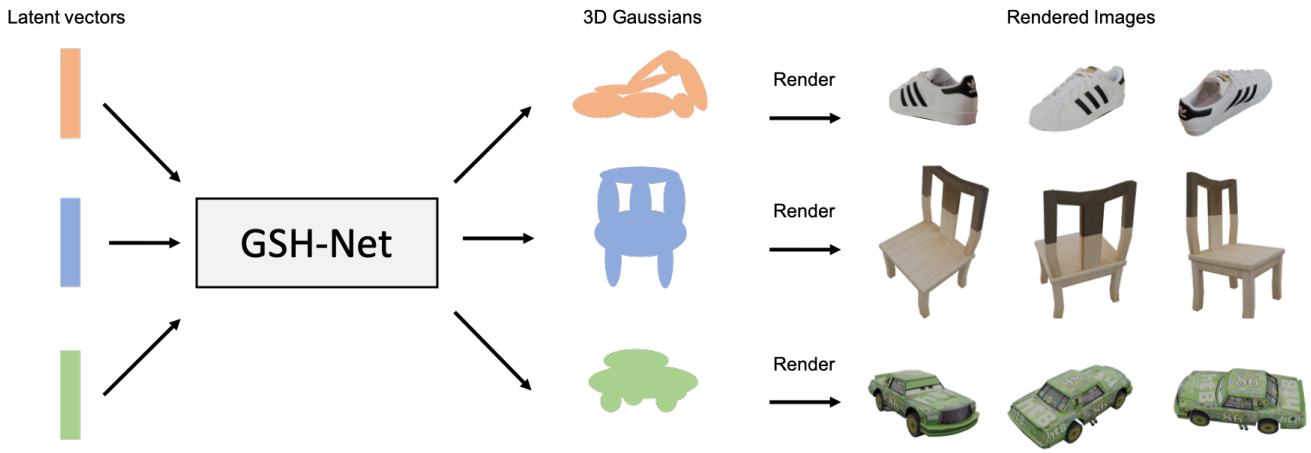

Figure 1. **Gaussian Primitives Generation.**

Abstract

*3D Gaussian Splatting (3DGS) has demonstrated state-of-the-art performance in multi-view synthesis, achieving photo-realistic rendering quality with minimal training time and allowing for real-time rendering (∼ 100 FPS) during the inference stage. However, existing methods predominantly focus on applying 3D Gaussian Splatting to a single scene, resulting in a learned representation that lacks generalizability across different objects. In this project, we explore the possibility of using 3D Gaussians as the representation of 3D object, and try to model the distribution of 3D objects with a generator. Specifically, we collect a custom set of 3D objects from Objaverse, and optimize 3D Gaussians on all of the objects. To reduce the overhead of modeling the distribution of 3D Gaussians, we adopt a Clipped Coarse-to-fine Adaptive Density Control approach to reduce the dimensionality of 3D Gaussians significantly. In the next stage, we introduce the Gaussian Splatting Hypernetwork (GSH-Net) to directly generate the parameters of Gaussian primitives. These generated primitives are then fed into a Gaussian Splatting rasterized renderer for real-time rendering. The implementation of a simple Variational Autoencoder (VAE) learning paradigm facilitates the unconditional generation abilities of GSH-Net. Although the experimental result of GSH-Net still has room for improvement, we show that it is possible to adopt 3D Gaussians as an underlying representation of 3D objects with the hope to push the boundary of 3D generative AI. **The code and the presentation video are uploaded to Canvas**.*

1. Introduction

1.1. Motivation

3D content generation is important in fields such as game design, artworks design, virtual reality, and robotic vision. In recent years, this area has undergone significant evolution, driven by advancements in computational power and algorithmic design. One of critical challenges in this domain is developing a 3D scene representation that not only maintains spatial properties but also accurately captures the appearance and geometry of the scene. Traditional explicit representations, like point clouds, voxel grids, and meshes, struggle with modeling detailed structures of the scenes due to their discretization nature. Similarly, implicit methods such as Signed Distance Functions (SDF) [10] and Neural Radiance Fields (NeRF) [8] often face difficulties in learning precise surface boundaries. In response to these limitations, 3D Gaussian Splatting (3DGS) has emerged as a

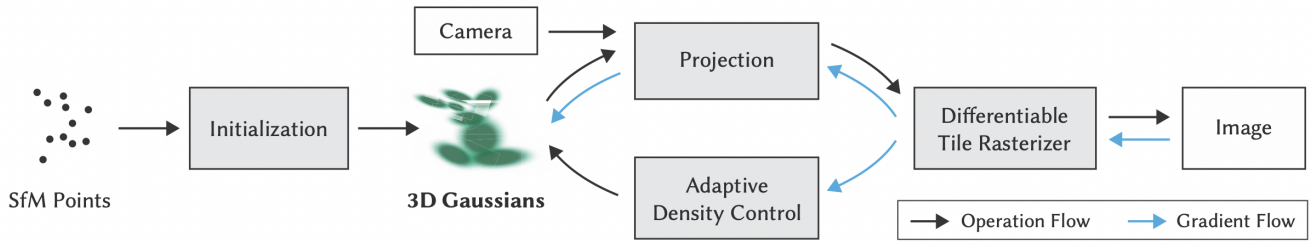

Figure 2. Overview of Gaussian Splatting.

promising solution, blending the robust spatial characteristics of explicit representations with the smooth, continuous nature of implicit methods. This hybrid approach facilitates the creation of detailed, realistic, and geometrically-consistent 3D models, essential for photo-realistic rendering and accurate scene reconstruction.

In this project, we introduce the Gaussian Splatting Hypernetwork (GSH-Net), a novel framework that leverages the strengths of 3DGS for 3D content generation. Our methodology involves two primary components: firstly, training the 3DGS model on multi-view synthesis tasks, and secondly, developing a hypernetwork that reconstructs Gaussian primitives. We test our approach on a customized dataset, collecting 3D assets from the Objaverse dataset [3]. The experimental results of our method are two-fold. Initially, we demonstrate the capability of GSH-Net in reconstructing various known object instances, accurately capturing their geometry and appearance. Subsequently, we explore the potential of GSH-Net in generating a diverse range of novel, unseen 3D objects through latent space sampling. Overall, this project showcases the effectiveness of GSH-Net in 3D content creation.

1.2. Related Works

3D Scene Representation. Various 3D representations have been explored for reasoning about scene geometry. Explicit representations, such as point clouds [12], voxel grids [11], and meshes [1], are easy to acquire and exhibit strong spatial properties. However, they often fail to represent smooth geometry and appearance due to the discretization process. In contrast, implicit representations, like Signed Distance Functions (SDF) [10] and Neural Radiance Fields (NeRF) [8], model the 3D space with a continuous function. These methods successfully capture both scene geometry and appearance by modeling the occupancy or the signed distance field, yet they face challenges in learning precise decision boundaries for surfaces. Recently, 3D Gaussian Splatting (3DGS) [5] has emerged as a popular approach in modeling both scene geometry and appearance. It represents scenes using multiple Gaussian primitives, each including attributes such as position, orientation, size, opacity, and a series of coefficients for spherical harmonics (SH). 3DGS embraces the advantages of both explicit and implicit methods. It employs Gaussians in 3D space to maintain robust spatial properties, while simultaneously learning the size and SH coefficients implicitly to better model the geometry and the appearance.

Hypernetworks for 3D Representation. Most existing methods employ a single hypernetwork to facilitate generalizable 3D scene representations. For instance, ATT3D [7] learns to map text prompts into spatial features within 3D voxel grids and then utilizes a single NeRF model to render the results from these generated feature grids. HyperDiffusion [4] addresses the unconditional generative modeling of implicit neural fields by operating directly on MLP weights, generating new neural implicit fields encoded by synthesized MLP parameters. Similarly, Chiang et al. [2] approach the 3D scene stylization problem by employing a hypernetwork to predict the parameters of MLPs in the appearance branch. In this project, we use a hypernetwork to directly generate a 3DGS representation. The point cloud-like properties of 3DGS allow for effective implementation by adapting from existing point cloud-based networks.

2. Method

Our GSH-Net is based on two key components: (1) the training of the 3D Gaussian Splatting (3DGS) model on multi-view synthesis task, which enables us to obtain specific 3DGS model parameters for each object instance. (2) the training of a hypernetwork by reconstructing Gaussian primitives, which facilitates the generation of 3D objects through random sampling within the latent space.

In Section 2.1 of our paper, we elaborate on the training of the 3D Gaussian Splatting. A detailed explanation of how we employ a VAE learning approach to attempt to achieve unconditional 3D object generation in Section 2.2.

2.1. Training Gaussian Splatting on Objects

3D Gaussian Splatting. Gaussian Splatting [5] represents an explicit 3D scene using point clouds. Each point clouds

have several features to describe the structure of the scene. Each Gaussian $\mathbf{P} \in \mathbb{R}^D$ with a center $x \in \mathbb{R}^3$, and a covariance matrix Σ defined in world space:

$$G(x) = e^{-\frac{1}{2}x^T\Sigma^{-1}x}. \quad (1)$$

Each 3D Gaussian has additional attributes – color represented by spherical harmonics $c \in \mathbb{R}^k$, where k indicates the degrees of freedom, and opacity $\alpha \in \mathbb{R}$, to represent the radiance field. To project the 3D Gaussians to 2D for rendering, they follow [15] to perform this projection. Given a viewing transformation W the covariance matrix Σ' in camera coordinates is given as follows:

$$\Sigma' = JW\Sigma W^T J^T. \quad (2)$$

where J is the Jacobian of the affine approximation of the projective transformation. To optimize the 3D Gaussians, Kerbl *et al*. proposes to use a rotation matrix $\mathbf{R}$ and scale matrix $\mathbf{S}$ to decompose the covariance matrix:

$$\Sigma = \mathbf{RSS}^T\mathbf{R}^T, \quad (3)$$

which is differentiable and the calculation of gradient for each parameters is detailed in [5]. Specifically, the trainable parameter for each 3D Gaussian point is: 3 for center $x \in \mathbb{R}^3$, 48 for spherical harmonics $c \in \mathbb{R}^{45}$ (in our project), 1 for opacity $\alpha \in \mathbb{R}$, 4 for rotation $q \in \mathbb{R}^4$ in quaternion, and 3 for scale $s \in \mathbb{R}^3$, a total of 59 parameters for each Gaussian.

Clipped Coarse-to-Fine Adaptive Density Control. However, it is difficult to model the 3D Gaussians if we do not pose regularization during the Adaptive Density Control step. The number of Gaussians can grow arbitrarily even if we only need to model a single object instead of a complex scene. To avoid overparameterizing when optimizing the 3D Gaussians of a single object, we adopt a coarse-to-fine approach to regularize number of points used during optimization. We start from 4096 Gaussians instead 100000 points, and densify or prune the Gaussians if the number of points does not exceed 10000. We found that this strategy yields the similar reconstruction results but substantially reduce the number of parameters we need when training the 3D Gaussians.

2.2. VAE on Gaussian Primitives

Given each 3D object represented as 3D Gaussians $\mathbf{P} \in \mathbb{R}^{N\times 59}$, we leverage a Variation Autoencoder [6] to generate the 3D Gaussians. The encoder first maps the input 3D Gaussians $\mathbf{P}$ to obtain the latent variable $\mathbf{z_P}$, which is drawn from a normal distribution. The encoder predicts the mean and standard deviation of a normal distribution

$$\mu_\mathbf{P}, \sigma_\mathbf{P} = \text{Enc}(\mathbf{P}) \quad (4)$$

$$\mathbf{z_P} \sim N(\mu_\mathbf{P}, \sigma_\mathbf{P}). \quad (5)$$

Figure 3. Visualization of 3D assets from Objaverse.

where $\mathbf{z_P} \in \mathbb{R}^{1024}$ and Enc is the encoder of the VAE implemented with the encoder of PointNet++ [14]. The reconstruction of the 3D Gaussian is obtained with a decoder:

$$\hat{\mathbf{P}} = \text{Dec}(\mathbf{z_P}). \quad (6)$$

The model is trained with reconstruction loss and KL-divergence loss:

$$\mathcal{L}_\text{VAE} = \mathbb{E}_q \log p(\mathbf{P}|\mathbf{z}) - \lambda_\text{KL} D_\text{KL}(q(\mathbf{z}|\mathbf{P})||p(\mathbf{z})). \quad (7)$$

where λ_KL is the weight parameter for KL-divergence loss. During test time, we randomly sample the latent variable from a standard Gaussian $\mathbf{z_P} \sim N(0, \mathbf{I})$ and use the decoder to generate the 3D Gaussian.

2.3. Implementation Detail

For training 3D Gaussian Splatting, we use the source code from the original author [5]: https://github.com/graphdeco-inria/gaussian-splatting. We initialize the 3D Gaussian with 4096 points randomly. We use degree of 3 for spherical harmonics. For the clipped coarse-to-fine adaptive density control, we densify and prune the 3D Gaussians as along as the number of 3D Gaussians does not exceed 10000. We use 30000 iterations to optimize the 3D Gaussians for each object. For training the VAE, we use a encoder similar to PointNet++ [14] which has 5 Conv1D layers followed by two layers of MLP. For the decoder, we have 5 layers of Conv1D layers. The dimensionality of the latent variable $\mathbf{z_P}$ is 1024. We use $\lambda_\text{KL} = 1e-6$ when training the VAE. The VAE is trained with 100000 iterations. We explore as many implementation choices we can to obtain better results. For instance, we tried different learning rate ($1e-2 \sim 1e-5$), KL-divergence weight ($1e-7 \sim 100$), encoders (plain ConvNet, PointNet [13], PointNet++ [14]), opacity weight ($1e-2 \sim 10$), and scale weight ($1e-3 \sim 1e-1$). For the details of training the 3D Gaussian Splatting and the VAEs, please refer to the code.

(a) Car viewpoint #1 (b) Car viewpoint #2 (c) Car viewpoint #3

(d) Chair viewpoint #1 (e) Chair viewpoint #2 (f) Chair viewpoint #3

(g) Shoe viewpoint #1 (h) Shoe viewpoint #2 (i) Shoe viewpoint #3

Figure 4. Demonstration of dataset images sampled from different viewpoints.

3. Experimental Results

3.1. Dataset.

3D Assets Collection. For this project, we have curated custom object assets from Objaverse 1.0 [3]. Objaverse 1.0 is a massive dataset with 800K+ annotated 3D objects retrieved from the Internet. To ensure class-specific collection, we utilized LVIS annotations from Objaverse 1.0 to select objects from those classes appears more frequently in the dataset. Specifically, we chose seven distinct classes for our study: chairs, swords, armor, mugs, shoes, motorbikes, and cars. These classes were selected based on their fre-

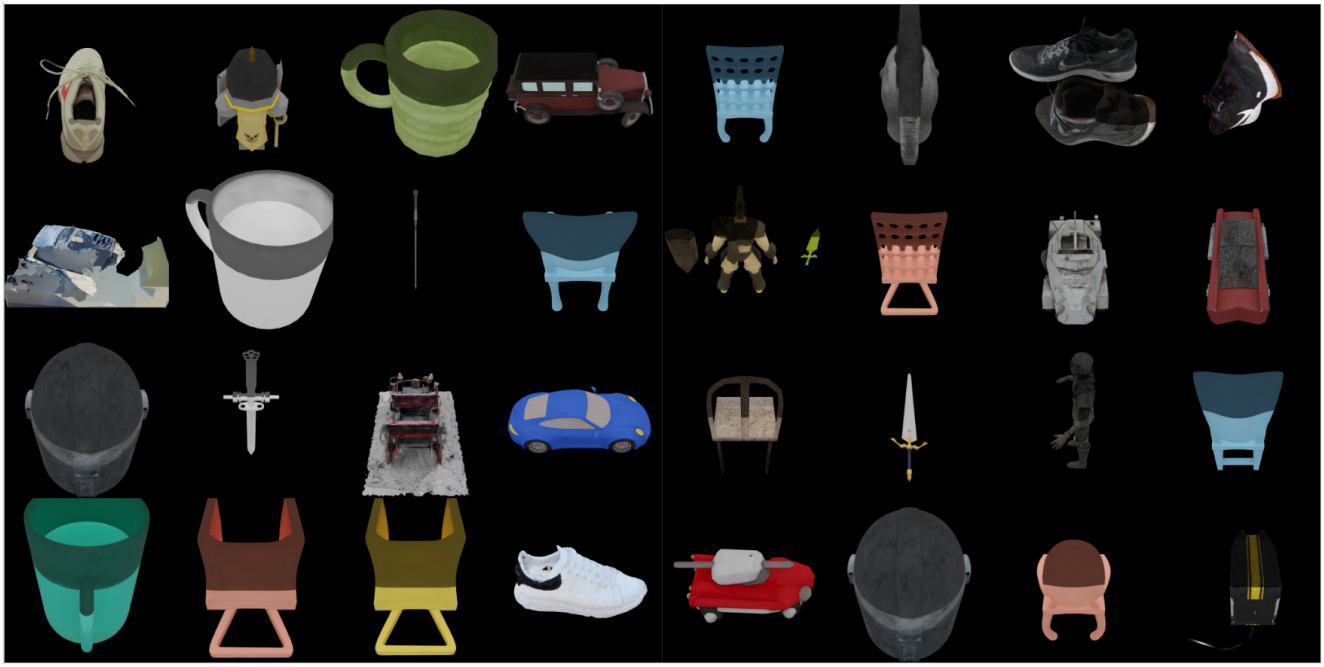

Figure 5. **Rendering of the optimized 3D Gaussians.** We show the rendering of 3D Gaussian for different objects after the optimization.

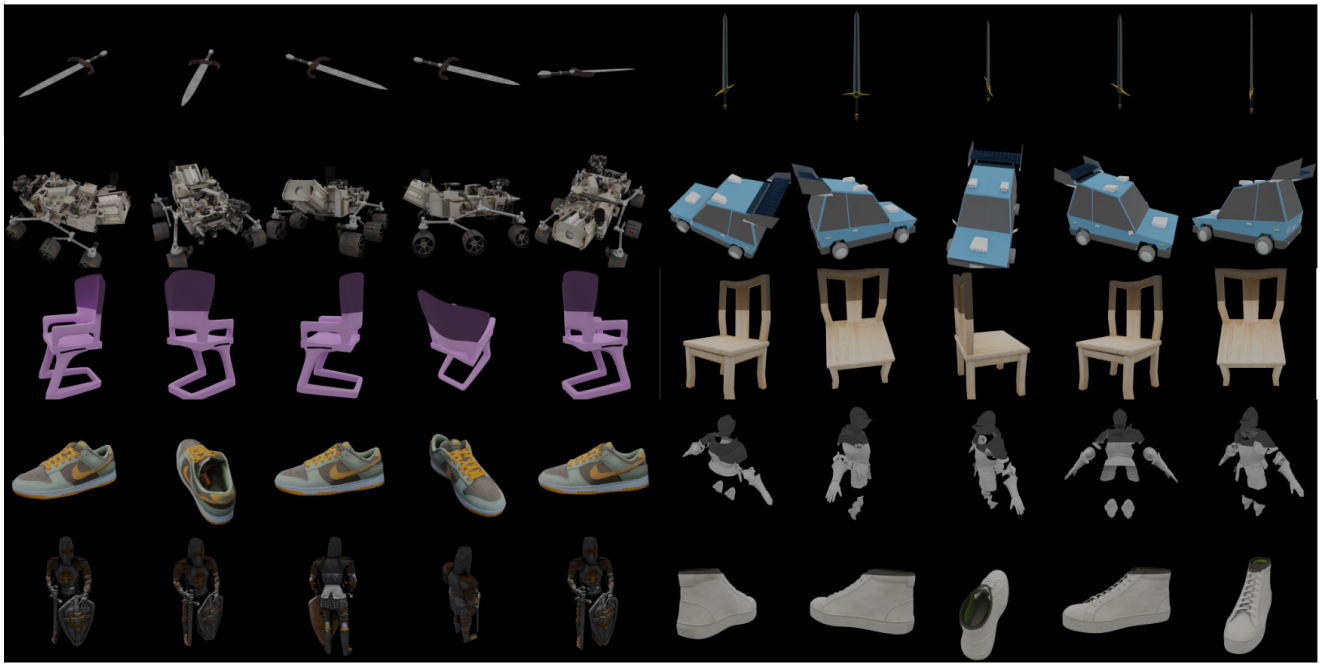

Figure 6. **Rendering of the optimized 3D Gaussians in multiview.** We show the rendering of 3D Gaussians from multiviews.

quency in the dataset. The overall dataset consists of 1186 object instances, offering a diverse range of shapes and appearances for our experimentation.

Blender Rendering. To create images used for training multi-view synthesis task, we use Blender to render the 3D object from multiple viewpoints. Initially, we resize the object to a standard unit size. Then, we position the same 100 cameras around the object to render the 3D asset. For lighting up objects, we place an area light source above the object. We save the camera transformation matrix for train-

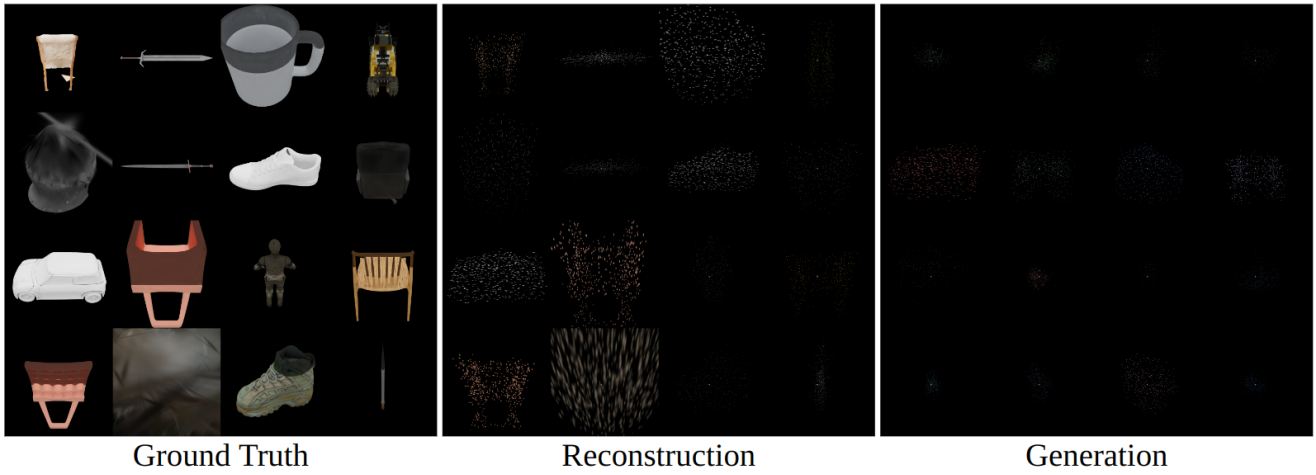

Figure 7. **Results of the VAE.** We show the reconstruction and generation results with our VAE.

ing the 3D Gaussians Splatting. The demonstration of our sampled dataset is shown in Fig. 4.

3.2. Qualitative Results

3D Gaussian Splatting on Objaverse. We show the rendering of 3D Gaussian Splatting results on multiple objects in Fig 5. The results show that the optimized 3D Gaussians can faithfully model the geometry and textures of the 3D object. We also show the multiview rendering of different objects in Fig 6. The rendering shows that the optimized Gaussians perferms well on novel view synthesis.

3D Gaussians Reconstruction and Generation We show the reconstruction and generation of 3D Gaussians with our VAE in Fig 7. The results shows that the VAE can reconstruct the location of the Gaussians. For the color, we can only reconstruct the right color but cannot predict the finer details of the color. This suggests that we need a more powerful architecture to reconstruct the spherical harmonics. We can also see that the reconstructed Gaussians do not have the right scales, which results that we cannot reconstruct the finer details of the shape.

4. Discussion and Conclusions

In this project, we propose a potential solution for high quality 3D objects generation with 3D Gaussian Splatting. Given its great abiliy and efficiency to model the 3D scene or object, our goal is to explore the if the 3D Gaussians are good representation for 3D generative AI. We use Objaverse as our test bed, and use Blender to render the 3D objects for training, resulting a total of 1186 objects represented as 3D Gaussians. We use a VAE to model the distribution of 3D Gaussians. The experimental results show that we can learn the center and the color of the Gaussians to some extent, but failed to model the finer details regarding the scales, rotations, and the spherical harmonics. We explore several choices of the encoder (plain Conv1D, PointNet [13], PointNet++[14]), different hyper-parameters (learning rate, KL-divergence weight, opacity weight, scale weight) and different optimizer (SGD, Adam), but none of them can reconstruct the finer details of the 3D objects. We think the main possible solution to further improve the results can be (1) reduce the max degree of spherical harmonics, similar to our coarse-to-fine strategy on the number of Gaussians, (2) add Fourier embedding for center, scale, rotations to better encode the attributes, (3) replace the VAE with Point-E [9], diffusion model implemented with transformer to generate colored point clouds.

5. Individual Contribution

Yen-Chi Cheng and Hao-Yu Hsu developed an algorithm for generating 3D Gaussians and implementing the framework with PyTorch. Hao-Yu Hsu has completed the preparation of a customized dataset using Blender, while Yen-Chi Cheng has successfully trained the Gaussian splatting model on this data, yielding excellent rendering results. We have been developing the code collaboratively via Git and discussing both progress and challenges on a weekly basis. Together, we successfully optimize the 3D Gaussians on a custom set of 3D objects from Objaverse. There are 1186 objects in thie custom set. We finish the full pipeline of training VAE on a dataset of 3D Gaussians: a dataloader for loading 3D Gaussians, a model of VAE to preprocess input, perform forward and backward pass, inference, and rendering of 3D Gaussians, a visualization with TensorBoard to modify training, and a main file for handling the optimization iterations. We write the final report together.

References

[1] A.Kanazawa, S.Tulsiani, A.A.Efros, and J.Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 2

[2] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork, 2022. 2

[3] 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. 2, 4

[4] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *ICCV*, 2023. 2

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 3

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[7] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. Att3d: Amortized text-to-3d object synthesis. In *ICCV*, 2023. 2

[8] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

[9] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 6

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 1, 2

[11] Charles R. Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016. 2

[12] Charles R. Qi*, Hao Su*, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[13] 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. 3, 6

[14] 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. 3, 6

[15] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3