1 Implemantation Details

1.1 Stage 1: GaussianVolume Fitting

In this stage, we precisely fit each object using 96 uniformly rendered images, capturing various poses. The initial 72 images are rendered with camera poses uniformly distributed around a camera-to-world-center distance of 2.4 units. The remaining 24 images are rendered from a closer distance of 1.6 units to provide a more detailed view. We set the volume resolution at $N = 32$ and the spherical harmonics (SH) order at 0, resulting in each Gaussian point having a feature channel number $C = 14$.

The fitting volume is assumed to be a cube with a side length of 1 world unit, centered within the global coordinate system. Inside this cube, Gaussian points are uniformly distributed to ensure a comprehensive coverage of the volume. Initially, the offset for each Gaussian point $\Delta\mu$ is set to zero. To optimize each instance, we conduct training against a white background for 20,000 iterations. A densification strategy is employed from iteration 500 to 15,000, with subsequent operations executed every 100 iterations to incrementally enhance the model's density. After this densification phase, we periodically clip features to predefined ranges every 100 iterations to maintain consistency and prevent outliers. We refrain from resetting the opacity of each Gaussian point during the training process. This decision is made to avoid introducing instability into the model's learning and adaptation phases.

For weights selection in $\mathcal{L}_{fitting}$, we set $\lambda_1 = 0.8, \lambda_2 = 0.2, \lambda_3 = 20.0$, and the offsets threshold as 1.5 voxel distances. Mathematically, this is expressed as $\epsilon_{offsets} = \frac{1.5}{32-1}$, signifying the calculated distance between adjacent grid points in our defined volume space. For other parameters, we adhere to the default configurations in 3D Gaussian Splatting [3].

1.2 Stage 2: Text-to-3D Generation

Data Pre-processing In the generation phase, we first preprocess GaussianVolume data to improve the stability and convergence of the learning process.

First, we standardize the decomposition of the covariance matrix by imposing an ordering constraint on the scaling vector $s \in \mathbb{R}^3$ of each Gaussian, ensuring they are arranged in ascending order. This organization, while maintaining the orthogonality of the eigenvectors, does not modify the resultant covariance matrix. The rotation vector $q \in \mathbb{R}^4$ is also adjusted to align with the sequence of transformations introduced by scaling. Secondly, we represent quaternions using spherical coordinates. This conversion reduces the rotation vector q to a three-dimensional representation, enhancing the uniqueness and consistency of each Gaussian's parameters. By normalizing quaternions in this manner, we ensure that identical shapes are represented by identical parameters. Finally, we normalize each feature channel individually based on its mean and variance. This normalization process equilibrates the scales across different channels.

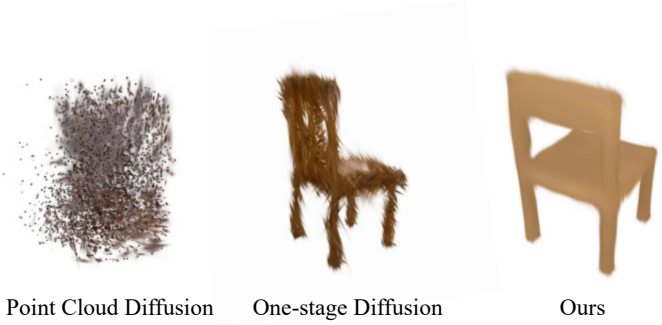

Fig. 1: Visual Results of Generated Assets in Explorational Experiments.

Model Architecture and Training In the generation phase of Gaussian Density Fields (GDF), we utilize a 3D U-Net modified from [2]. The text condition is 77×768 embeddings extracted with CLIP ViT-L/14 [6] text encoder, and it is injected into the 3D U-Net using cross-attention mechanisms. An additional MLP layer is employed to bridge the inter-modal gap. The architecture of the prediction model mirrors the 3D U-Net, with adjustments including a modification in the input channel number and the omission of timestep blocks.

When training the generative model predicting GDF, we only use MSE loss $\mathcal{L}_{3D}$ as the supervision. For training the prediction model, we set $\lambda = 0.2$ for the rendering loss $\mathcal{L}_{2D}$ through the whole process. Then, for the total loss $\mathcal{L} = \lambda_{3D}\mathcal{L}_{3D} + \lambda_{2D}\mathcal{L}_{2D}$, we set $\lambda_{3D} = 1.0, \lambda_{2D} = 0$ for the first 100 epochs, providing a robust initialization that focuses solely on 3D consistency. Subsequently, the weights are adjusted to $\lambda_{3D} = 0.2$ and $\lambda_{2D} = 0.8$, shifting the focus towards optimizing 2D rendering and further refining.

2 Explorational Experiments

In our early exploration, we investigated the efficacy of various generative models, initially training three unconditional models: a point cloud diffusion model, and a primal 3D U-Net-based diffusion model, and compared them with our full model. We specifically utilized 8 fitted GaussianVolumes belonging to the chair LVIS-category as our training data and trained each model for 4,000 epochs. The qualitative results are shown in Fig. 1.

Our initial foray involved adapting the point cloud diffusion model, inspired by prior work [4], to facilitate the generation of 3D Gaussians. Unfortunately, this approach struggled with convergence issues, yielding unsatisfactory results. The inherent limitations of point cloud diffusion models in capturing and generating the nuanced structures of 3D Gaussians became apparent, prompting us

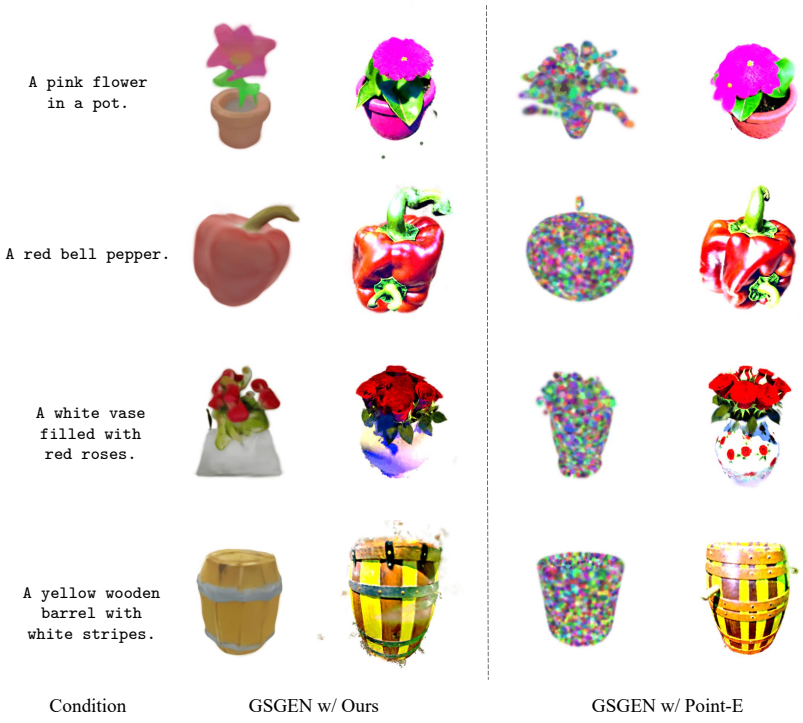

Fig. 2: GSGEN [1] Results Initialized with Our Method and Point-E [5]. The left column represents rendering results initialized with different methods, and the right column stands for rendering results after optimization with GSGEN.

to explore alternative strategies. Employing a primal one-stage diffusion model offered a glimpse into generating basic 3D structures. However, this model predominantly produced coarse outputs characterized by spiny shapes, highlighting the need for a more nuanced generation technique to achieve plausible 3D geometry. Our full model, incorporating a coarse-to-fine generation pipeline, demonstrated significant improvements in generating 3D Gaussians. This approach not only simplified the generation process but also empowered the model to produce instances with more accurate and realistic 3D geometry. By sequentially refining the generated outputs, the coarse-to-fine pipeline effectively addresses the limitations observed in the earlier models, showcasing its superiority in generating complex 3D structures.

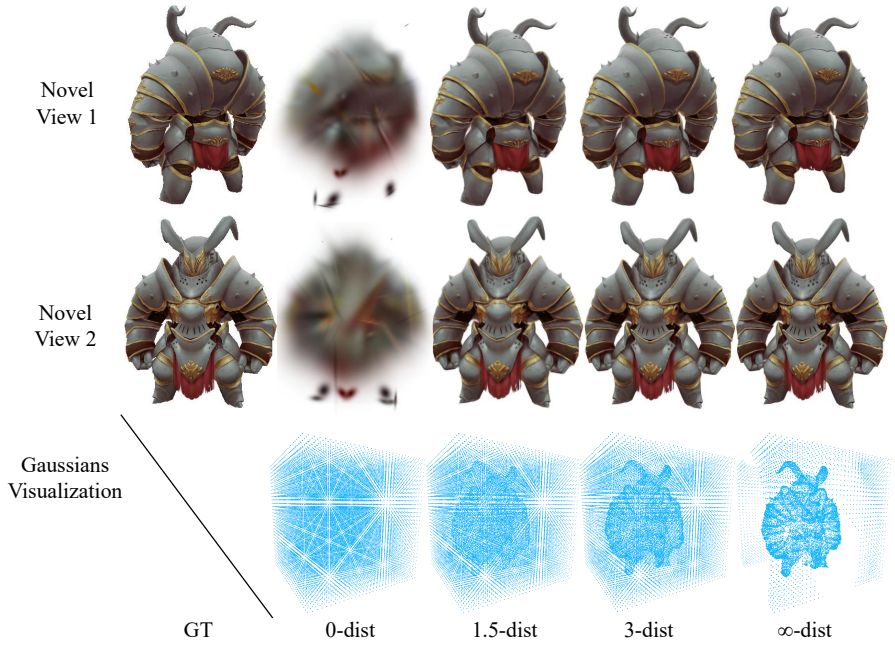

Fig. 3: Visual Results of Rendered Images and Positions of The Gaussian Center. The fitted assets are optimized with different offsets threshold $\epsilon_{offsets}$.

3 Application

To demonstrate the superiority of our method, we show GVGEN's capability to integrate with optimization-based methods, like GSGEN [1], for further refinement (see Fig. 2). Using the generated GaussianVolume[1] as initialization to replace point clouds from Point-E with random colors. This shift enables GSGEN to further optimize 3D Gaussians, achieving better alignment with the text descriptions in both texture and geometry. This enhancement stems from avoiding the adverse impact of color attributes from Point-E on GSGEN's utilization, as the features produced by GVGEN are more compatible and beneficial for the optimization process.

[1] We filter low-opacity Gaussian points for GSGEN initialization.

Table 1: Quantitative results for different offsets threshold selection.

Metrics	PSNR ↑	SSIM ↑	LPIPS ↓
0-dist	18.189	0.859	0.176
1.5-dist	**30.437**	0.966	0.046
3-dist	30.395	0.969	0.038
∞-dist	30.007	**0.969**	**0.034**

Table 2: Quantitative Results for Different Settings of GaussianVolume Resolution.

Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	Num of Gaussians	Time(s)
16-res	23.079	0.887	0.122	4,096	169
32-res	**30.437**	0.966	0.046	32,768	179
64-res	30.395	**0.972**	0.032	262,144	195
3DGS	29.789	0.969	**0.030**	175,532	191

4 Additional Ablation Studies on GaussianVolume Fitting Stage

4.1 Effects of offsets threshold $\epsilon_{offsets}$

In Tab. 1 and Fig. 3, we present the effects of varying the offset threshold, $\epsilon_{offsets}$ during GaussianVolume fitting. The term "0-dist" indicates the absence of an offsets term, whereas "∞-dist" denotes the omission of offsets regularization. Our observations reveal that minimal regularization leads to improved rendering performance. However, this comes at the cost of Gaussian points within the volume becoming more unstructured and deviating from grid points. Without this regularization, the offset term becomes overly flexible, making it challenging for the network to learn effectively. To strike a balance between flexibility and maintaining a well-defined structure, we have selected $\epsilon_{offsets} = \frac{1.5}{32-1}$, equivalent to a 1.5 voxel distance, as our optimal threshold.

4.2 Effects of GaussianVolume Resolution

We present a comparison of rendering results and quantitative metrics in Tab. 2 and Fig. 4, utilizing the "Chunky knight" asset from the Objaverse dataset. An increase in resolution N-res of the GaussianVolume correlates to a higher number of Gaussian points, N^3. To accommodate the varying volume resolutions N, adjustments are made to the initialization and the offsets threshold, specifically $\epsilon_{offsets} = \frac{1.5}{N-1}$. The term "3DGS" denotes the process of fitting the object using the original 3D Gaussian Splatting [3], with iteration settings matched to ours

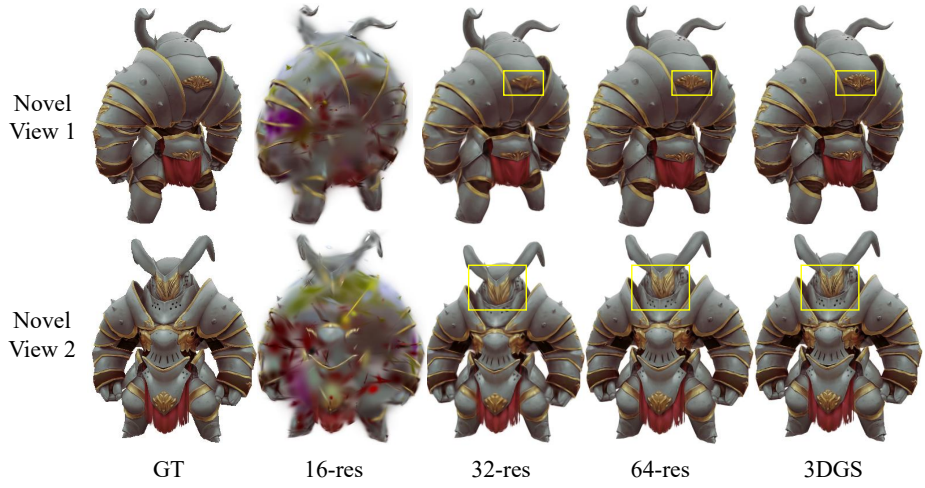

Fig. 4: Qualitative Comparisons among different GaussianVolume resolution settings and original 3DGS.

and other parameters set to default. The analysis reveals that a greater number of Gaussian points leads to improved fitting quality under identical settings, albeit at the expense of increased memory usage. Notably, even when utilizing fewer Gaussian points, our method delivers comparable visual quality to the original 3DGS approach and achieves higher PSNR values in a structured representation. This underscores the efficiency and effectiveness of our method in reconstructing high-quality, memory-efficient representations.

5 Failure Cases

As illustrated in Fig. 5, some of the objects generated by our model suffer from blurred textures and imprecise geometry. This phenomenon largely stems from the relatively small size of our training dataset, which comprises around 46,000 instances. Such a dataset size limits the model's capacity to produce varied outputs in response to a broad spectrum of text inputs. In the future, our work will focus on two main avenues: improving the model architecture and enhancing the data quality. By addressing these aspects, we aim to scale up the model for application in large-scale scenarios, which is expected to improve the generation diversity and lead to better-rendered results.

6 Additional Qualitative Results

We provide more visual results in Fig. 6 and Fig. 7.

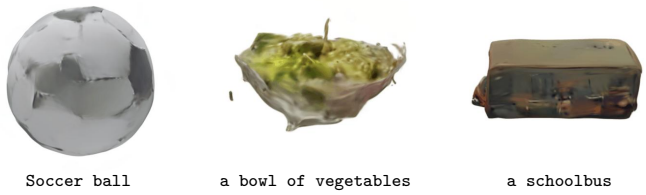

Fig. 5: Failure Cases.

References

1. Chen, Z., Wang, F., Liu, H.: Text-to-3d using gaussian splatting. arXiv preprint arXiv:2309.16585 (2023)
2. Cheng, Y.C., Lee, H.Y., Tulyakov, S., Schwing, A.G., Gui, L.Y.: Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4456–4465 (2023)
3. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
4. Melas-Kyriazi, L., Rupprecht, C., Vedaldi, A.: Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12923–12932 (2023)
5. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)
6. 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: International conference on machine learning. pp. 8748–8763. PMLR (2021)

Fig. 6: More Qualitative Results.

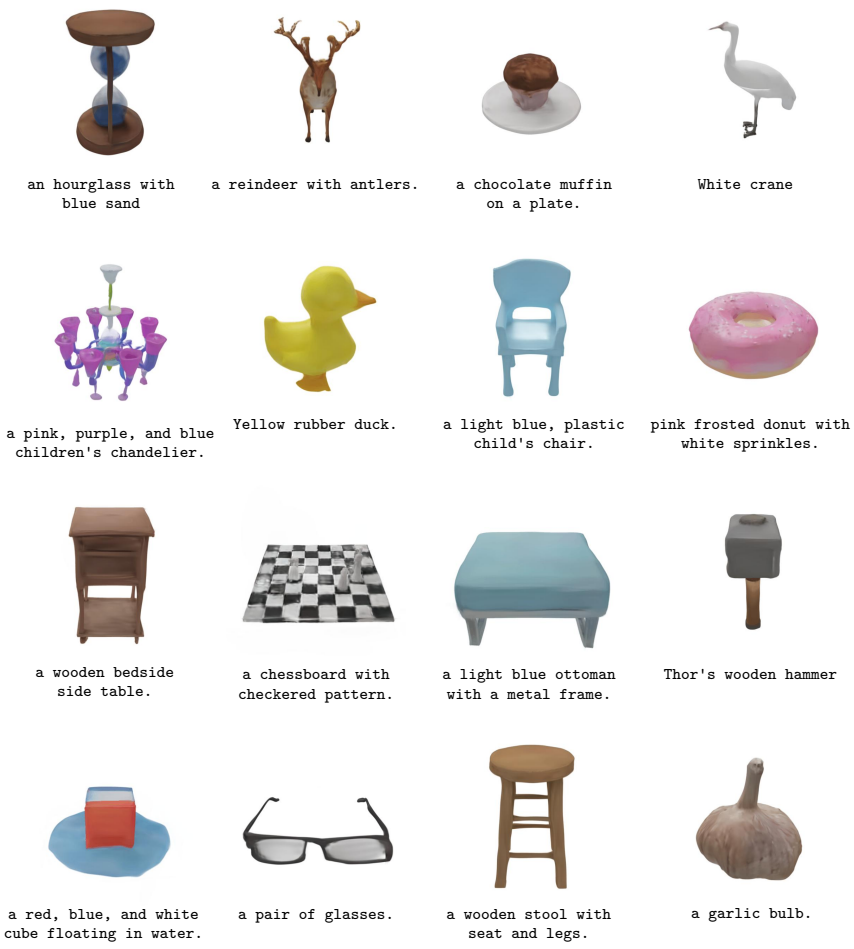

Fig. 7: More Qualitative Results.