GaussFusion: Seamless 3D Object Integration using Gaussian Splatting

Weng Haoyang
wenghy22@mails.tsinghua.edu.cn

Huang Yingyi
hyy24@mails.tsinghua.edu.cn

Abstract

This paper presents a streamlined pipeline to merge 3D objects with background scenes using 3D Gaussian Splatting (3DGS). Our method creates realistic 3D models by integrating objects into new environments. We describe our data preprocessing techniques, 3DGS training process, and an interactive visualization tool for merging and viewing the results. Our approach offers advantages over traditional mesh-based methods, requiring simpler data collection setup and faster training. Our demonstration uses only 5 minutes in preprocessing and 10 minutes for training the gaussian splatting model on 8 GPU in parallel. We demonstrate the effectiveness of our method through qualitative visual inspections. The demonstration videos can be accessed at the following URL: https://cloud.tsinghua.edu.cn/d/f0e541f8eeca4aa1bd5f/, and the code is published on Github at https://github.com/jasmine27-alt/weng-wong-project-cv.

1. Introduction

The integration of 3D objects into existing/novel scenes is a fundamental challenge in computer vision and graphics, with applications spanning from indoor furniture placement to novel scene synthesis. Conventional approaches typically rely on mesh-based object representations, which often demand intensive manual engineering to define geometry and texture maps. In contrast, our project explores the use of 3D Gaussian Splatting (3DGS) [1] to rasterize images of an object onto a background, generating a realistic 3D model where the object is seamlessly merged into the environment.

3D Gaussian Splatting offers several key advantages over traditional mesh-based methods:

- Simpler training setup
- Less demanding data collection process
- Real-time rendering and interactive control

Our project uniquely supports two distinct data sources for 3D reconstruction: synthetic 3D models and real-world video scans. However, training a merged scene comprising two 3DGS models presents several challenges:

- For synthetic data, visually continuous camera positions and comprehensive coverage of all view angles are required to adequately represent the object from multiple perspectives while ensuring successful keypoint matching in COLMAP.
- When working with video scans of real environments, the lack of ground truth camera positions necessitates non-trivial handling of the rendering camera's coordinate system.
- Merging two independently reconstructed objects or scenes introduces an additional layer of complexity, as we must align their respective coordinate systems to create a cohesive final scene.

To address these challenges, we have developed a comprehensive pipeline encompassing data preprocessing, 3DGS training, and scene merging. Our preprocessing stage is tailored to handle both synthetic and real-world data sources, ensuring consistent input for the 3DGS training process. The training phase utilizes distributed computing to efficiently create high-quality 3D Gaussian representations of objects and scenes. Finally, our merging algorithm tackles the intricate task of combining multiple 3DGS models into a single, visually coherent scene by concatenating the Gaussian model parameters.

Complementing this pipeline, we have also created an interactive visualization tool that enables real-time manipulation of the integrated objects. This tool not only showcases the results of our method but also provides an intuitive interface for users to fine-tune object placement and explore the merged scenes.

From an application-oriented perspective, our work is motivated by the growing need (and interest) in integrating novel objects into familiar scenes, or placing familiar objects in synthetic backgrounds. This capability has numerous potential applications, from virtual reality and augmented reality to architectural visualization and film production. Our pipeline simplifies this process by addressing the complex camera coordinate transformations and 3D

model view rendering, making such applications more accessible to a wider audience.

Moreover, our work extends beyond mere object integration. The viser-based visualizer we've developed enables researchers to compare different training checkpoints jointly. This feature provides valuable insights into the training process, allowing for more effective fine-tuning of 3DGS models.

In this paper, we present our methodology, implementation details, and results, offering comprehensive guidelines for each stage of the process. We demonstrate the effectiveness of our approach through qualitative visual assessments.

2. Related Work

Our work builds upon recent advancements in 3D scene representation and rendering, with a particular focus on the integration of objects into existing scenes. The foundation of our approach is the 3D Gaussian Splatting (3DGS) technique introduced by [1]. This method represents 3D scenes as a set of Gaussian primitives, offering an effective balance between rendering quality and computational efficiency.

Since its introduction, the 3DGS method has seen several significant improvements. [2] proposed a hierarchical approach to 3D Gaussian Splatting, enhancing both the scalability and efficiency of the original technique. In a similar vein, [5] developed a grouping strategy for 3D Gaussians, which led to notable improvements in rendering performance. These advancements have collectively pushed the boundaries of what's possible with 3DGS, making it an increasingly attractive option for various 3D rendering applications.

A particularly noteworthy recent development is the work by [6] on distributed training for 3DGS. Their approach achieves linear scaling of 3DGS training across multiple GPUs, significantly reducing the computational barriers to creating custom 3DGS models. This advancement makes the technology more accessible to a broader range of researchers and practitioners, potentially accelerating innovation in the field.

While these works have focused on improving the core 3DGS technique, our project takes a different direction. We address the specific challenge of integrating objects into existing scenes, a problem that has traditionally been tackled using mesh-based methods or image-based rendering techniques. Our approach leverages the strengths of 3DGS while focusing on the unique challenges posed by object integration.

3. Methodology

Our methodology comprises three main components: data preprocessing, 3D Gaussian Splatting (3DGS) training, and scene merging with visualization.

3.1. Data Preprocessing

Our pipeline supports two types of input data:

1. 3D models sourced from online repositories
2. Video scans of real environments

For 3D models, we employ a custom rendering script to generate multiple views of the object, along with corresponding camera parameters. For video scans, we extract frames and utilize COLMAP [3, 4] to perform structure-from-motion and generate a sparse 3D reconstruction.

The preprocessing pipeline consists of the following steps:

1. Image generation or extraction
2. Feature extraction and sparse 3D reconstruction using COLMAP
3. Coordinate system alignment (specifically for video scans)

3.2. 3DGS Training

For training our 3D Gaussian Splatting models, we leverage the Grendel-GS implementation [6]. This process involves optimizing a set of 3D Gaussians to best represent the input scene or object. The training is conducted separately for the background scene and the object to be integrated, ensuring independent, high-quality representations for each component.

3.3. Scene Merging and Visualization

To merge the pre-trained object and background models, we implement a straightforward strategy of concatenating the Gaussian parameters of the two models. We then apply scaling and translation transformations to the position and scaling parameters of the object's Gaussians to integrate it into the background scene.

It's worth noting that our framework does not support rotation transformations for two primary reasons:

1. Rotation transformations would require non-trivial modifications to the spherical harmonics parameters of the Gaussians, which would be computationally intensive to calculate in real-time.
2. Controlling rotation intuitively can be challenging for users. While methods like roll, pitch, and yaw could be employed, we opted to limit the degrees of freedom to prevent user confusion.

Given that we cannot automatically determine the optimal position and scaling for object integration, we have developed an interactive visualization tool. This tool allows for real-time manipulation of the merged scene, enabling users to fine-tune the object's position and scale within the background environment.

4. Implementation and Results

4.1. Data Processing Pipeline

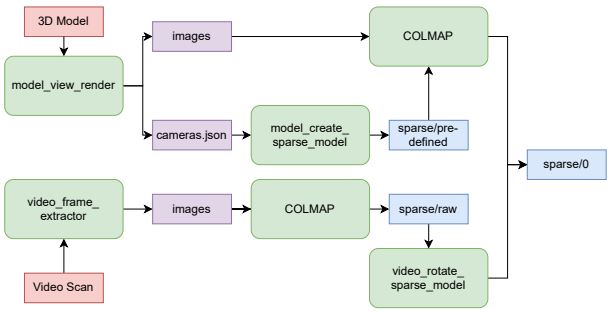


Figure 1. Data Preprocessing Pipeline

Our data processing pipeline is implemented in Python and leverages existing computer vision libraries. Key components include:

- `model_view_renderer.py`: Generates multiple views of 3D models
- `video_frame_extractor.py`: Extracts frames from video scans
- `create_sparse_model_from_json.py`: Creates COLMAP-compatible sparse models from ground truth camera extrinsics for synthetic 3d models
- `rotate_sparse_model.py`: Aligns coordinate systems for video scans

4.1.1 Image Generation

Regarding the processing of 3D models, creating a custom script to render models and capture multiple views with a virtual camera posed a number of challenges. Choosing a compatible file format and finding a library that could accurately render textures were primary obstacles. Initially, we tried Pyglet for its windowing capabilities, but it had issues with texture rendering and macOS compatibility. Trimesh managed 3D model data well but struggled with texture accuracy and camera transformations. Open3D, despite its advanced features, faced GLFW errors and segmentation faults on macOS.

Ultimately, PyVista emerged as the most effective solution. By directly importing the GLTF file, PyVista handled texture rendering accurately and offered robust camera setup, lighting adjustments, and off-screen rendering capabilities, allowing us to capture high-quality images from various angles. To meet our unique needs, we developed a custom script for generating a spiral camera path using cv2, capturing multiple views from calculated positions and intrinsics. This approach enabled precise control over camera movements and ensured comprehensive coverage of the 3D models.

For video scans, we extract 600 frames as input images at identical time intervals using opencv-python.

4.2. Camera Coordinate System Alignment

Our pipeline addresses two critical challenges in aligning camera coordinate systems:

1. **Pyvista to COLMAP Transformation**: To use COLMAP with the 3D model views generated by Pyvista, we must provide camera intrinsics and extrinsics in COLMAP's specific format (as detailed at https://colmap.github.io/format.html). However, Pyvista and COLMAP use different camera coordinate systems. This necessitates a transformation of the camera extrinsics matrix to ensure compatibility with COLMAP.

2. **Video Scan Frame Alignment**: For frames extracted from video scans, we lack ground truth camera extrinsics and rely on COLMAP to estimate them. However, COLMAP's output is subject to arbitrary rotation, making these extrinsics unsuitable for training and integration with other 3DGS scenes. Post-training rotation of parameters is complex, particularly for spherical harmonic coefficients.

To address these challenges, we implement the following solutions:

1. **Pyvista to COLMAP Transformation**: We developed a custom transformation process to convert Pyvista's camera extrinsics to COLMAP's format. This ensures correct camera positioning and orientation, as illustrated in Figure 2. Without this transformation, camera positions appear erratic or incorrectly oriented.

2. **Video Scan Frame Alignment**: For video scans, we enforce a consistent world coordinate system by applying the following constraints:

   - We assume the first frame is captured with an upright camera pose.
   - For the first frame, we align the camera's y-axis with the world's negative z-axis and the camera's x-axis with the world's x-axis.

This approach ensures a consistent up direction in the world frame and enables seamless integration of the constructed scene with other 3DGS environments.

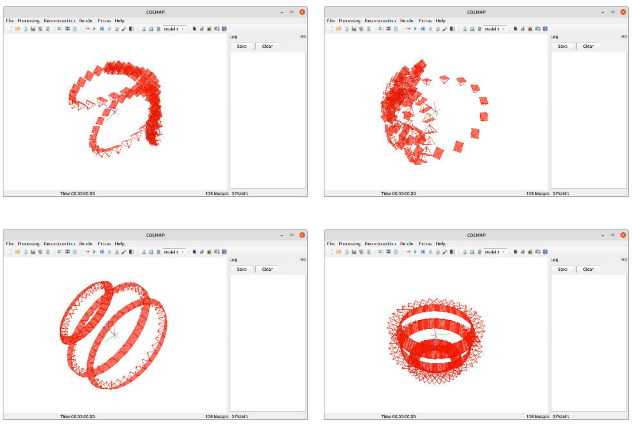


Figure 2. Effect of Correct Camera Extrinsics Transformation. Without correct transformation, the camera positions appear erratic and messy (top row), or facing in the opposite direction (bottom left). With correct transformation, the cameras are in the correct location and point inwards (bottom right).

4.3. Gaussian Splatting Training

For training our 3D Gaussian Splatting (3DGS) models, we utilize the Grendel-GS framework [6]. This framework supports distributed training, allowing for efficient handling of large scenes and complex objects. Leveraging GPU acceleration, Grendel-GS efficiently manages the computational demands of 3DGS using PyTorch for high-performance tensor computations.

The training process involves several key components working together to achieve high-quality 3D representations. Initially, the algorithm constructs rotation matrices and calculates covariance from scaling and rotation parameters using methods implemented in PyTorch. These operations are crucial for accurately modeling the geometric transformations required in 3D space. Additionally, spherical harmonics are employed for color representation, which enhances the realism of the rendered images.

The rendering pipeline is another important aspect of the training process. This involves projecting 3D points onto a 2D image plane, which is achieved by setting up precise camera parameters and configuring rasterization settings. The camera setup defines the field of view, image dimensions, and transformation matrices, ensuring accurate projections and realistic shading and lighting effects.

4.4. Visualization Tool

We have developed a real-time interactive visualization tool using viser (https://github.com/nerfstudio-project/viser), which provides a browser-based interface for interacting with merged 3D Gaussian Splatting (3DGS) scenes. Our tool offers several key features:

1. **Real-time Rendering**: The tool supports real-time rendering of integrated objects and scenes, allowing for smooth interaction and exploration.

2. **Plug-and-Play Integration**: Users can easily load and merge two PLY files, combining their 3DGS parameters into a joint model. This feature enables seamless integration of different objects and scenes.

3. **Interactive Manipulation**: We provide an intuitive GUI with sliders, allowing users to interactively adjust the offset and scaling of objects relative to the scene.

4. **Multi-Resolution Rendering**: Users can dynamically adjust the rendering resolution using sliders. This adaptive approach enables higher frame rates for smooth animation in high-capacity scenes and facilitates faster interactive manipulation for finding optimal offset and scale. Users can then finalize the rendering at a higher resolution for maximum quality.

4.4.1 Scene Merging and Object Manipulation

Figure 4 illustrates the plug-and-play merging capabilities of our visualizer. It demonstrates the versatility of our approach, showing how different objects (a bonsai tree and a Lamborghini car) can be seamlessly integrated into various scenes (a desktop and a garage).

4.4.2 Interactive Manipulation

The user can use sliders provided by viser GUI to control the translation and scaling as illustrated in 3.

4.4.3 Multi-Resolution Rendering

Our tool's multi-resolution rendering capability allows users to balance between performance and visual quality. Figure 5 showcases the visual output at different resolutions, while Table 1 provides a quantitative comparison of performance and file size across resolutions.

Table 1. Performance metrics for multi-resolution rendering

Resolution	Frame Rate (FPS)	File Size (MB)
384p	60	3.68
512p	50	3.87
1024p	35	4.32
2048p	15	4.69
2560p	10	4.77
3072p	8	4.79

This adaptive resolution feature enables users to maintain high frame rates during scene navigation and object

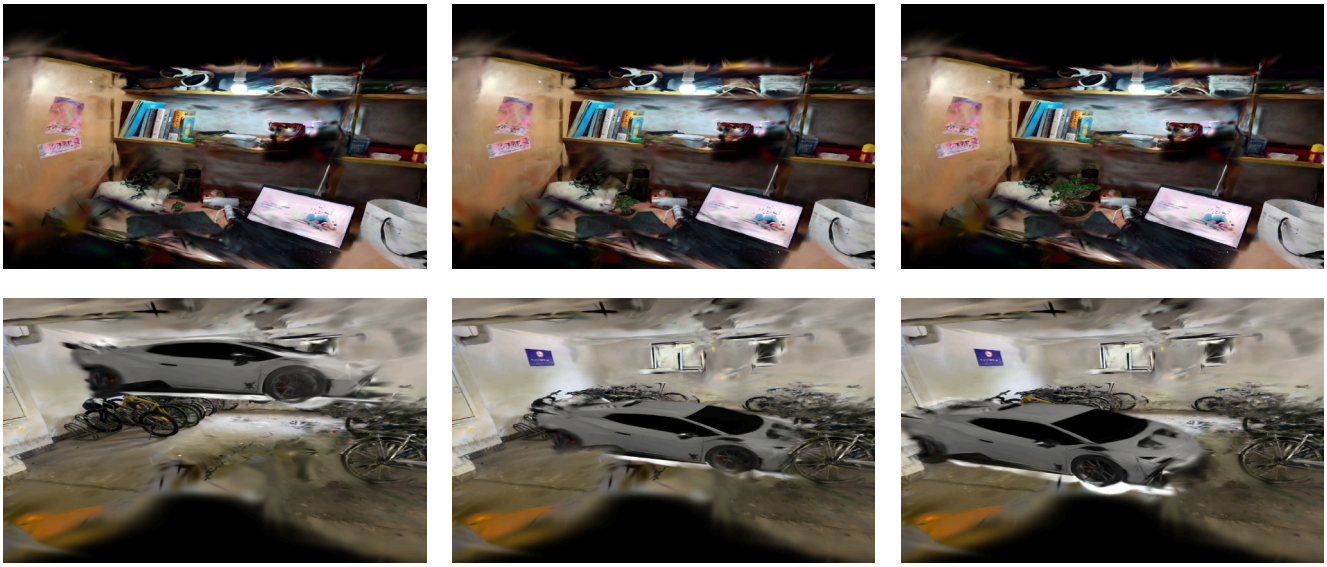
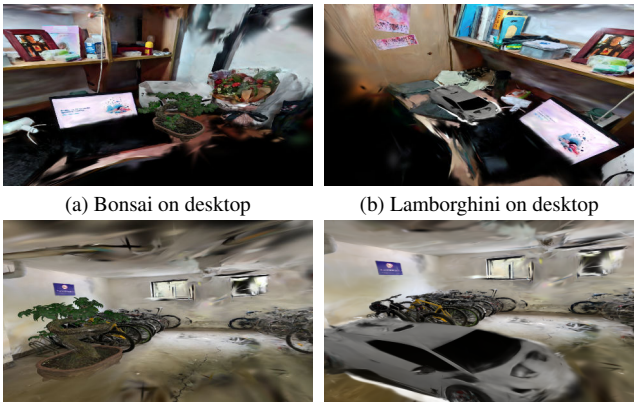


Figure 3. Demonstration of the interactive translation and scaling functionality. Top row: Users can easily adjust the size of the integrated object (a bonsai tree in this case), allowing quick and precise size adjustments, enabling users to find the perfect scale for their scene composition. Bottom row: Users can effortlessly move the integrated object (a Lamborghini model in this example) within the scene. The intuitive controls allow for smooth and precise positioning, enabling users to easily experiment with different object placements in real-time.



(a) Bonsai on desktop (b) Lamborghini on desktop

(c) Bonsai in garage (d) Lamborghini in garage

Figure 4. Demonstration of the plug-and-play merging visualizer, showing versatile object integration across different scenes.

manipulation, switching to higher resolutions for final high-quality renders. It significantly enhances the tool's usability, particularly when working with complex, high-capacity scenes.

4.4.4 Training Process Visualization

During development, we discovered the utility of visualizing different stages of the training process. By comparing checkpoints, users can gain valuable insights into the effects of various hyperparameters, such as the densification threshold.

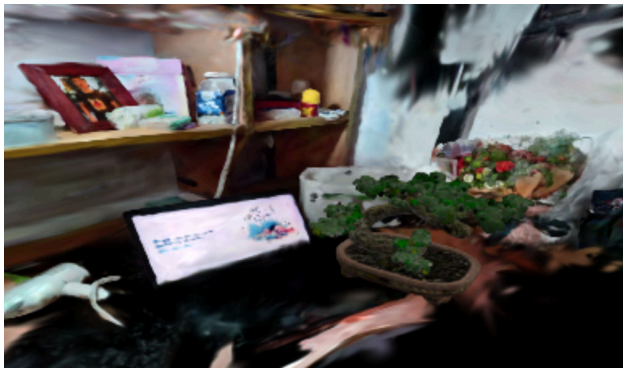
Figure 6 showcases this feature, comparing checkpoints at 30,000 iterations (end of densification) and 50,000 iterations (end of training). This visualization reveals how the training process initially focuses on reconstructing geometry during the densification phase, then transitions to refining color accuracy in the later stages.

This visualization capability proves invaluable for fine-tuning the training process, allowing researchers and practitioners to optimize parameters for improved 3DGS model performance.

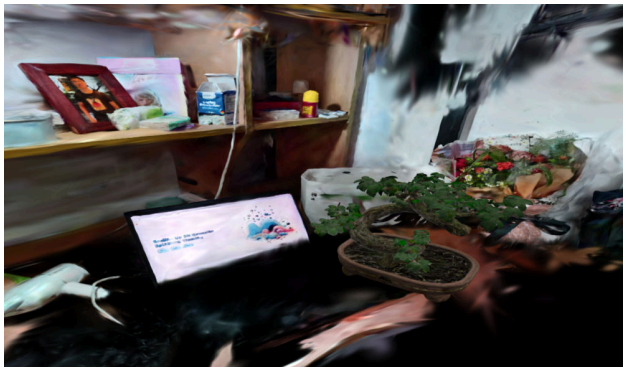
5. Discussion and Future Work

Our current implementation demonstrates the potential of using 3D Gaussian Splatting for object integration in 3D scenes. However, there are several areas for future improvement and exploration:

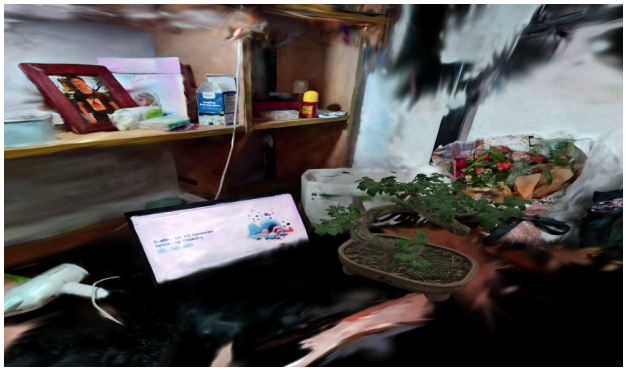
- Reduce floating Gaussians: There are many floating Gaussians present in the current rendered result. We may modify the training algorithm to reduce these floaters or use post-processing techniques to remove them for improved rendering quality.
- Development of a more sophisticated merging algorithm: Current merge algoritm naively concatenates the gaussian parameters and apply translation and scaling on the part of object. We may devise more sophis-



(a) 384p (60 FPS)



(b) 1024p (35 FPS)

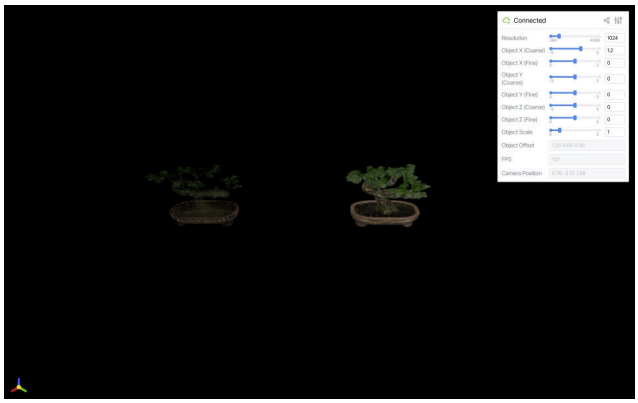


(c) 3072p (8 FPS)

Figure 5. Multi-resolution rendering examples, demonstrating the trade-off between visual quality and performance.

ticated algotithm that considers lighting and material properties, to make the merged scene more visually consistent

- Exploration of multi-resolution training: Current pipelines only support inference time adaptive resolution by setting the rendering window size effectively. We may also modify the training pipeline to generate multiple versions of gaussians for improved rendering efficiency, e.g. support interactive rendering with fewer gaussians then finalize with a high capacity one.



(a) Bonsai model comparison

(b) Lamborghini model comparison

Figure 6. Checkpoint comparison visualizing the training progression. The images compare checkpoints at 30,000 iterations (end of densification) and 50,000 iterations (end of training), illustrating the transition from geometry reconstruction to color refinement.

6. Conclusion

We have presented a novel approach to merging 3D objects with background scenes using 3D Gaussian Splatting. Our method offers a streamlined pipeline from data preprocessing to interactive visualization, providing a powerful tool for realistic 3D scene synthesis. While there is room for further improvement, our results demonstrate the potential of this technique for applications in computer graphics, virtual reality, and augmented reality.

References

[1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. 1, 2

[2] Muhammed Kocabas, Jen-Hao Rick Chang, James Gabriel, Oncel Tuzel, and Anurag Ranjan. HUGS: Human gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[3] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[4] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[5] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes, 2023. 2

[6] Hexu Zhao, Haoyang Weng, Daohan Lu, Ang Li, Jinyang Li, Aurojit Panda, and Saining Xie. On scaling up 3d gaussian splatting training, 2024. 2, 4