Isolated 3D Model Reconstruction Using Gaussian Splatting

Naveen Meena

8-Feb-2025

Abstract

This report outlines the methodology for reconstructing an isolated 3D model of an object from a 360-degree video using Gaussian Splatting. The approach involves video preprocessing, background removal, depth estimation, camera pose estimation, and Gaussian Splatting optimization. The final result is a high-quality, segmented 3D representation of the car, independent of its surroundings.

1 Introduction

1.1 Bidirectional Reflectance Distribution Function (BRDF)

The BRDF defines how light is reflected at a surface. Mathematically, BRDF is expressed as:

$$f_r(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{L_i(\omega_i) \cos\theta_i d\omega_i}, \tag{1}$$

where ω_i and ω_o are the incoming and outgoing directions, respectively. The BRDF is fundamental in physically-based rendering, enabling realistic modeling of materials by accurately capturing specular highlights, diffuse reflections, and other surface interactions.

1.2 Point-Based Rendering

Point-based rendering represents scenes as a cloud of points rather than traditional polygon meshes. Each point contains attributes like position, color, normal, and reflective properties. Rendering involves splatting these points onto the image plane, significantly improving computational efficiency, especially in complex lighting environments. Gaussian Splatting leverages point-based rendering due to its inherent efficiency and scalability.

1.3 Global Illumination

Global illumination (GI) encompasses all light interactions within a scene, including direct illumination from sources and indirect illumination from inter-reflections. Modeling GI accurately requires handling diffuse and specular inter-reflections, shadows, and occlusions, which significantly contribute to realism. Modern Gaussian-based methods strive to capture these intricate interactions through approximations such as path tracing, spherical harmonics, and tensorial models.

1.4 Inverse Rendering

Inverse rendering aims to recover scene parameters such as geometry, materials, and lighting conditions from images. Recent Gaussian-based methods apply inverse rendering frameworks to estimate detailed material and illumination properties per Gaussian primitive, thus enhancing the relighting and editing capabilities of reconstructed scenes.

1.5 COLMAP

COLMAP is an open-source photogrammetry software widely used for Structure-from-Motion (SfM) and Multi-View Stereo (MVS). It extracts keypoints from images, matches these points across multiple views, and computes camera poses along with sparse and dense point clouds. COLMAP serves as a foundational step in many Gaussian-based reconstruction methods by providing accurate geometric initialization and camera parameters, essential for subsequent optimization processes.

1.6 Spherical Harmonics (SH)

Spherical Harmonics are mathematical functions used to represent lighting environments efficiently. SH decomposes complex lighting into frequency bands, allowing compact storage and fast computation. It is extensively used in relighting applications to approximate global illumination and efficiently render scenes under novel lighting conditions.

For a more in-depth explanation of the terms and processes discussed above, refer to the handwritten notes available at: `https://drive.google.com/drive/folders/1Y6tJg5vmYu4PBYchiAiGNW_rPkjoS03O?usp=sharing`.

2 Overview of the Approach

The proposed pipeline consists of the following key stages:

1. **Video Capture and Preprocessing**
2. **Foreground Extraction via Background Removal**
3. **Depth Estimation for Geometric Information**
4. **Structure-from-Motion for Camera Pose Estimation**
5. **Gaussian Splatting-based 3D Reconstruction**
6. **Post-processing and Refinement**

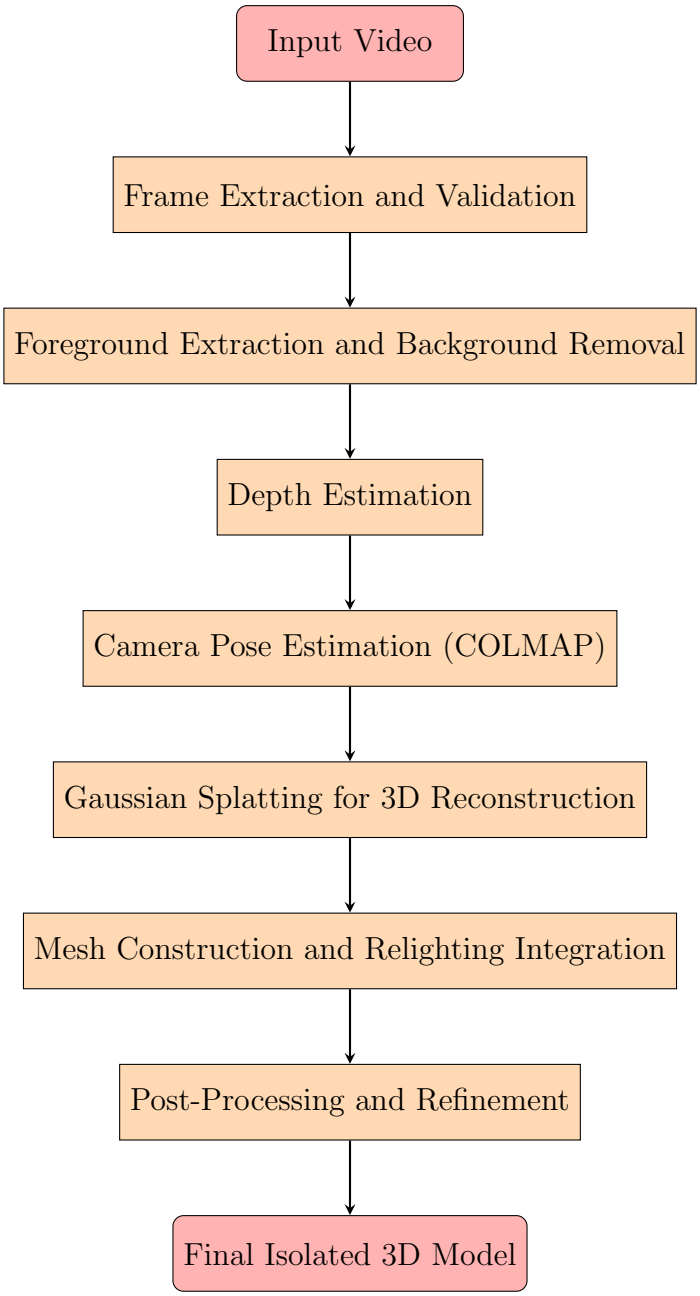


Figure 1: Flowchart of the Isolated 3D Model Reconstruction Pipeline

3 Methodology

3.1 Frame Extraction and Validation

- Video Loading and Frame Selection: The video is loaded and frames are selected at equidistant intervals to cover all views uniformly.

```
video_cap = cv2.VideoCapture(video_path)
frame_idxs = np.linspace(0, total_frames, num_frames_to_extract)
```

- Frame Quality Assessment: Frames are evaluated for blur, brightness, and contrast to ensure optimal quality.

```
blur_score, brightness_score, contrast_score = predict(image)[1:4]
```

- Non-Blur Frame Replacement: Frames failing quality checks are replaced by searching nearby frames for improved quality.

```
least_blur_frame = min(alternative_frames, key=lambda
x: predict(x.image)[1])
```

3.2 Foreground Extraction and Background Removal

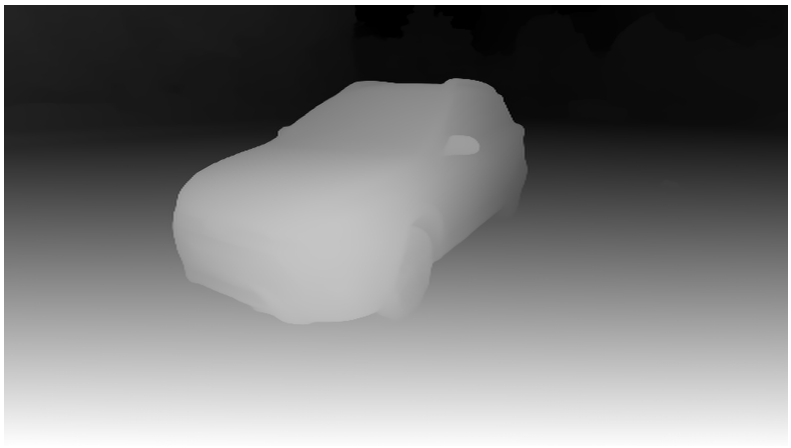
To improve the quality of our Gaussian model, I explored using segmentation masks to isolate objects from their backgrounds. For this, I trained a segmentation model using the Swin Transformer (SwinB) architecture, inspired by the **Inspyrenet** framework [1] `https://github.com/plemeri/InSPyReNet`. It operates by leveraging deep learning architectures, such as convolutional neural networks (CNNs) or vision transformers (e.g., Swin Transformers), to extract hierarchical features from the input image. These features capture both local and global contextual information, enabling the model to distinguish between different regions. During training, the model minimizes a loss function (e.g., cross-entropy) by comparing predicted pixel labels against ground truth annotations. At inference, the model outputs a dense pixel-wise mask, isolating objects or regions of interest from the background. The segmentation masks were generated using proprietary APIs from the company I work with, which cannot be shared due to confidentiality. These masks allowed us to train the model on background-removed images, focusing solely on the object of interest. While the current Gaussian model is trained on whole images, this approach demonstrates the potential for future improvements by isolating objects during training. For reference, I will share the isolated object mesh (isolated.ply) in the GitHub repository, showcasing the results of this method.

(a) w background (b) w/o background (c) mask

Figure 2: Inspyrenet Segmentation

3.3 Depth Estimation

Initially, depth estimation was performed using external methods from prior work, such as monocular depth estimation networks or multi-view stereo algorithms. However, the original Gaussian Splatting repository introduced its own robust depth handling mechanism, which was subsequently adopted in this work.



(a) depth

3.4 Camera Pose Estimation and Gaussian Initialization

To initialize the 3D Gaussian model, I used **COLMAP** [2] for camera pose estimation and sparse point cloud generation. COLMAP reconstructs the scene structure by extracting and matching features from multi-view images, followed by bundle adjustment to optimize camera poses and 3D points. This sparse reconstruction serves as the initial input for Gaussian Splatting. I made adjustments to COLMAP's configuration to handle the video-specific challenges, such as ensuring consistent frame sampling and optimizing feature matching for sequential frames. Alternatively, the built-in Gaussian Splatting pipeline provides a `convert.py` script, which can directly use COLMAP's output to initialize the Gaussians.

3.5 Gaussian Splatting for 3D Reconstruction

The implementation of the 3D Gaussian Splatting pipeline is based on the original repository provided by Inria's GRAPHDECO research group [3], available at `https://github.com/graphdeco-inria/gaussian-splatting`. This repository served as the foundation for the core components of the pipeline, including the Gaussian rendering, depth handling, and anti-aliasing mechanisms. Gaussian Splatting reconstructs the 3D model by optimizing a set of 3D Gaussians. The main steps include:

- Initialization of Gaussian splats based on SfM output.
- Optimization of Gaussian parameters (position, scale, rotation, opacity, and color) through differentiable volume rendering.
- Minimization of photometric loss between the rendered and input images using gradient descent.

We introduce several key modifications to the original 3D Gaussian Splatting training pipeline to better handle object-centric reconstruction from video inputs. These changes address three main challenges: (1) Visibility-Conscious Pruning, (2) Depth-Guided Normal Alignment, and (3) Progressive Spherical Harmonics.

3.5.1 Visibility-Conscious Pruning

```
observe_cnt[out_observe > 0] += 1
prune_mask = (observe_cnt < 2).squeeze()
```

- Removes Gaussians seen in less than 2 views
- Prevents transient view-dependent artifacts
- Reduces memory footprint

3.5.2 Depth-Guided Normal Alignment

```
normal_loss = weight * (depth_normal - rendered_normal).abs().sum(0).mean()
loss += normal_loss  # Added geometric consistency term
```

- Enforces multi-view normal consistency
- Regularizes geometry in textureless regions

3.5.3 Progressive Spherical Harmonics

```
if iteration % 1000 == 0:
    gaussians.oneupSHdegree()
```

- Starts with low-frequency base colors (SH degree 0)
- Gradually introduces higher-frequency details
- Prevents early overfitting to specular highlights

3.6 Mesh Construction and Relighting Integration

In my work, I initially used traditional methods like COLMAP for mesh construction, but these often failed to capture fine geometric details and realistic surface properties. Incorporating the **Reflective Gaussian Splatting (Ref-Gaussian)** [4] framework `https://fudan-zvg.github.io/ref-gaussian/` significantly improved results by leveraging its advanced relighting capabilities. Relighting methods, in general, enhance mesh quality by modeling light-surface interactions through diffuse, specular, and indirect lighting components. This lighting information refines the mesh, resulting in more accurate and detailed 3D models. Currently, the relighting framework is trained separately, and its outputs are used in a post-processing step to refine the mesh. While this approach has shown promising results, a direct integration of mesh construction into the relighting pipeline remains a **future goal**. Such integration would improve efficiency and accuracy, enabling real-time applications and more robust 3D reconstructions.

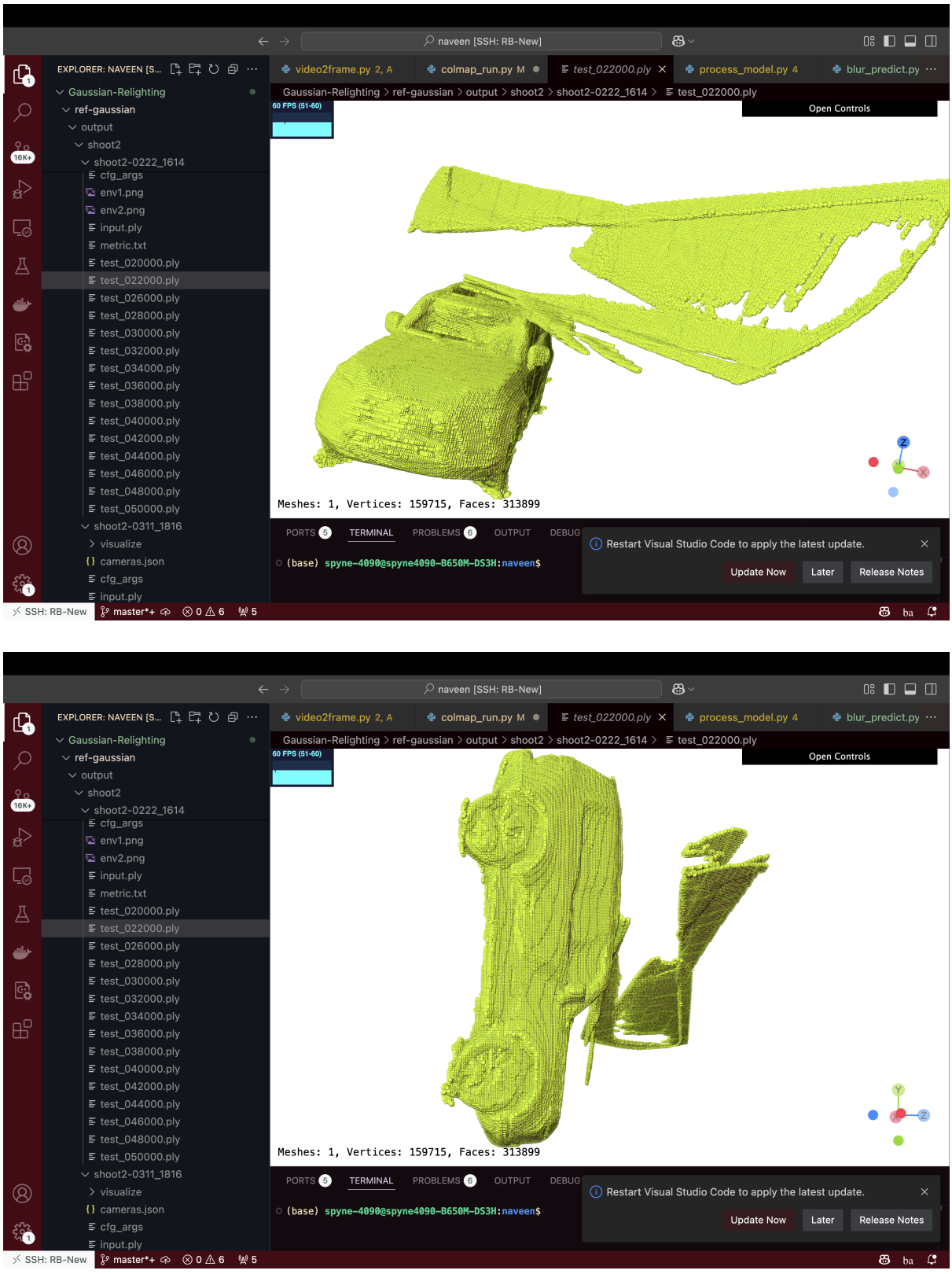


Figure 4: Ref-gaussian Mesh Construction at 22000 iterations

4 Viewing Gaussian Splatting Output

4.1 Why Gaussian Models Require Specialized Viewers

Gaussian Splatting models cannot be directly visualized in traditional viewers designed for polygonal meshes (e.g., STL, GLTF) due to fundamental differences in representation and rendering. Traditional formats like STL and GLTF use polygonal meshes made of vertices, edges, and faces, which cannot store these detailed per-point properties. Gaussian Splatting uses a point-based rendering technique, where each point is "splatted" onto the screen, requiring specialized tools for real-time performance. Traditional formats rely on simpler rendering methods like rasterization or ray tracing

4.2 Specialized Viewers for Gaussian Splatting

To address these challenges, specialized viewers like **SuperSplat** are required. While SuperSplat is the primary choice, other tools can visualize Gaussian Splatting outputs with some limitations:

- **CloudCompare**
- **MeshLab**
- **Blender (with Plugins)**
- **Open3D**

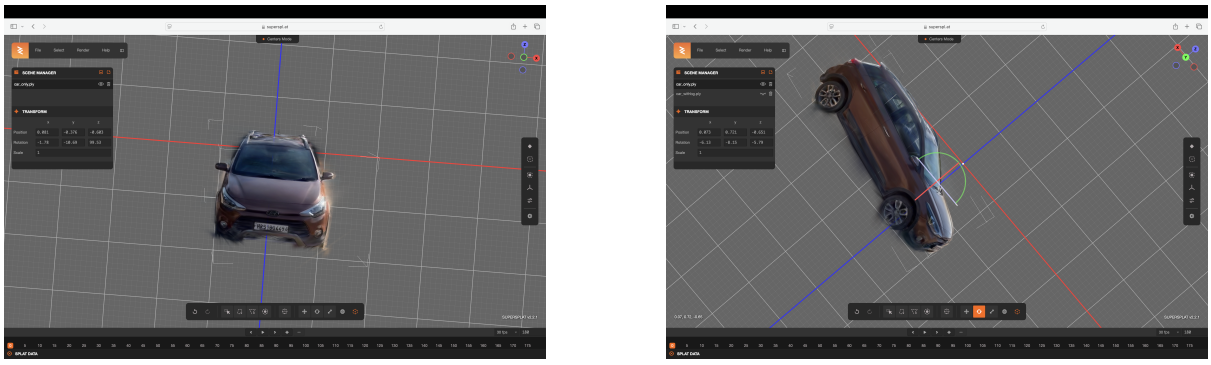


Figure 5: Isolated Object Model

s

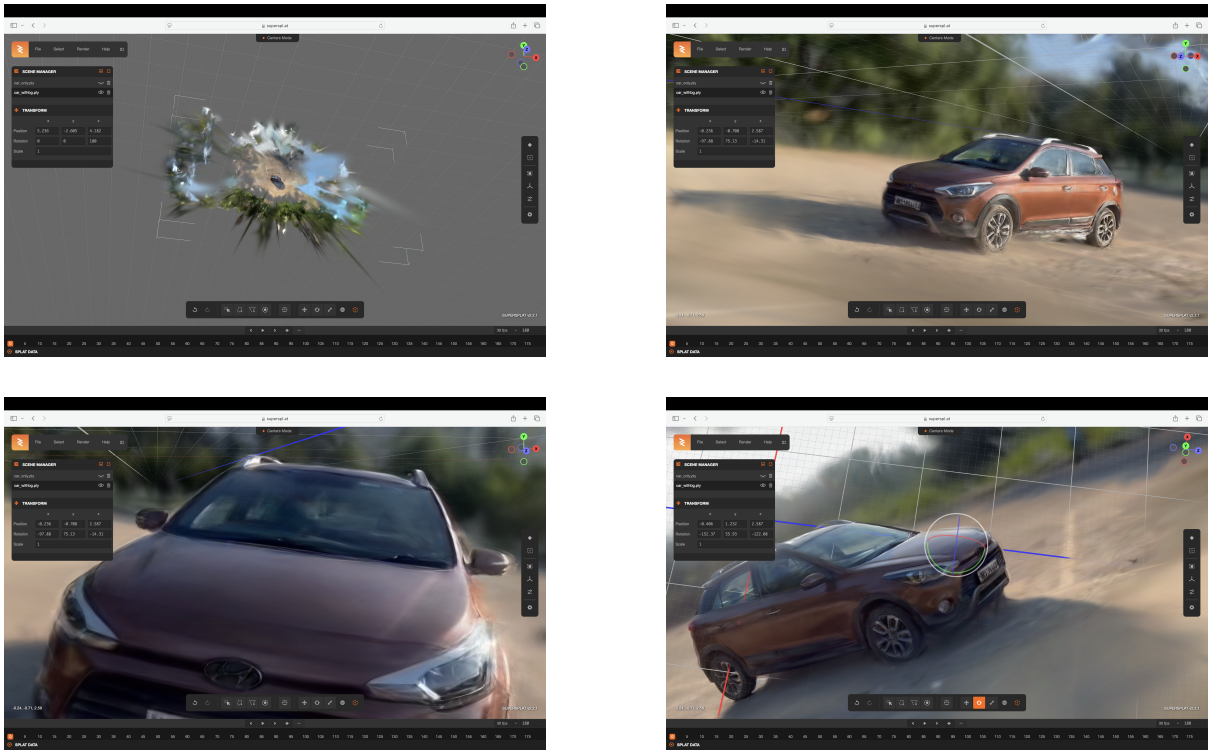


Figure 6: Object Model with Surroundings

References

[1] T. Kim, K. Kim, J. Lee, D. Cha, J. Lee, and D. Kim, "Revisiting image pyramid structure for high resolution salient object detection," in *Proceedings of the Asian Conference on Computer Vision*, pp. 108–124, 2022.

[2] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, July 2023.

[4] Y. Yao, Z. Zeng, C. Gu, X. Zhu, and L. Zhang, "Reflective gaussian splatting," *arXiv preprint*, 2024.