Object Removal and Re-colorization within 3D Gaussian Splatting Scene

Raushan Joshi

Master of Science in Artificial Intelligence

from the

University of Surrey

School of Computer Science and Electrical and Electronic Engineering

Faculty of Engineering and Physical Sciences

University of Surrey

Guildford, Surrey, GU2 7XH, UK

September 2025

Supervised by: Dr Jean-Yves Guillemaut

DECLARATION OF ORIGINALITY

I confirm that the project dissertation I am submitting is entirely my own work and that any material used from other sources has been clearly identified and properly acknowledged and referenced. In submitting this final version of my report to the JISC anti-plagiarism software resource, I confirm that my work does not contravene the university regulations on plagiarism as described in the Student Handbook. In so doing I also acknowledge that I may be held to account for any particular instances of uncited work detected by the JISC anti-plagiarism software, or as may be found by the project examiner or project organiser. I also understand that if an allegation of plagiarism is upheld via an Academic Misconduct Hearing, then I may forfeit any credit for this module or a more severe penalty may be agreed.

MSc Dissertation Title: Object Removal and Re-colorization within 3D Gaussian Splatting Scene

Author Name : Raushan Joshi

Author Signature: Date: 01/09/2025

Supervisor's name: Dr Jean-Yves Guillemaut

WORD COUNT

Number of Pages: 85

Number of Words: 19488 (excl. ref, app.)

ABSTRACT

The rapid evolution of 3D Gaussian Splatting (3D-GS) has revolutionized real-time 3D scene reconstruction, yet its potential for scene-level editing, such as object removal and recolorization, remains constrained by the lack of robust segmentation and high-quality datasets. Existing methods that lift 2D segmentations often suffer from view inconsistencies, fragmented masks, and visual artifacts. The significance of this work lies in enhancing 3D-GS to support advanced editing tasks, which are critical for applications in virtual reality (VR), augmented reality (AR), robotics, simulation, and digital 3D content creation. Our key contributions are threefold. First, we leverage SAM-HQ (Segment Anything Model - High Quality) to generate significantly improved 2D segmentation masks, addressing limitations of standard SAM, such as coarse boundaries, inconsistent regions, and challenges with thin structures. Second, we propose a novel prior-guided object label reassignment method that assigns multiview-consistent and semantically robust 3D labels to Gaussian primitives by integrating priors learned jointly during 3D-GS training. We simultaneously explored the training of compressed, high-quality semantic features into each Gaussian using a scene-specific autoencoder, enabling richer object-level representation. Third, we construct a high-quality multi-view mask dataset, enhanced by an efficient preprocessing pipeline, and demonstrate interactive object removal and recolorization using user-specified 3D point-based selection, while maintaining real-time inference speed and high visual fidelity. Emphasizing on segmentation accuracy and consistency, our approach matches state-of-the-art performance while embedding object-level semantic capabilities. Qualitative results highlight the superiority of SAM-HQ in preserving boundary fidelity, resolving fine structures, and confirm the practical utility of high-quality 3D-GS editing. This work advances the ongoing research in high-quality interactive 3D scene editing, robotics, and real-time AR/VR applications, while providing a valuable high-quality dataset for future studies in 3D reconstruction and edtiting.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Accurate representation and manipulation of three-dimensional (3D) scenes have become fundamental problems in computer vision and computer graphics, due to their central role in modern applications such as virtual reality (VR), augmented reality (AR) [2], autonomous navigation, robotics, and digital content creation. A well-reconstructed realistic 3D scene not only enables accurate rendering from arbitrary viewpoints but also provides the foundation for downstream tasks such as object detection, scene understanding, and interactive editing. In particular, the ability to selectively remove, recolor, or manipulate objects within 3D environments is of growing importance for practical applications, such as AR-assisted interior design, film production, and immersive gaming [21, 38], as well as research applications, such as controlled simulation environments, robotic interaction modeling, and medical 3D imaging [17].

Earlier, a variety of representations have been proposed for modeling 3D scenes. Point clouds [29], while lightweight, lack surface connectivity and are often too sparse for photorealistic rendering. Meshes [57] provide structural coherence, but require complex mesh pipelines and often break down for fine-grained details or thin structures. Voxel grids [58], on the other hand, offer volumetric consistency but are prohibitively expensive in terms of memory and computational requirements at high resolutions. These limitations have motivated the search for more expressive and efficient representations that balance rendering fidelity with computational feasibility.

Recently, in this context, Neural Radiance Fields (NeRFs) [32] introduced a transformative paradigm for implicit neural scene representation. NeRFs encode volumetric radiance and density into the weights of Multi-Layer Perceptrons (MLPs), enabling novel view synthesis with unprecedented fidelity. However, their implicit representation complicates object-level decomposition and editing, and the high computational cost of training and inference limits their suitability for real-time applications. Moreover, they lack direct semantic interpretability, which limits their usability for interactive editing tasks such as object removal and recolorization.

To overcome these barriers, 3D Gaussian Splatting (3D-GS) [23] has recently emerged as an explicit and computationally efficient alternative. By modeling scenes as collections of anisotropic 3D Gaussians projected through differentiable rasterization, 3D-GS achieves photorealistic ren-

dering at real-time frame rates. The explicit nature of this representation suits more naturally to scene editing tasks, since each Gaussian can be directly manipulated in terms of its position, scale, color, or opacity. This makes 3D-GS a promising framework not only for efficient rendering but also for real-time object-level manipulations in immersive systems.

However, a major challenge remains to enable object-level editing in 3D-GS which requires accurate and consistent 3D segmentation of Gaussian elements. Traditional 3D datasets [35] are costly to annotate, and existing methods that lift 2D segmentations [61, 15, 37] into 3D scene suffer from view inconsistencies, broken masks, and semantic misalignments. This often leads to unreliable editing results, where objects may be partially removed or recolorized, or where structural artifacts appear in the rendered scene.

In this dissertation, we propose a novel framework for object removal and recolorization within 3D Gaussian Splatting that leverages recent advances in foundation models for segmentation and incorporates learned priors into the 3D Gaussian labeling process. Specifically, we adopt SAM-HQ (Segment Anything Model - High Quality) [22] to generate improved 2D segmentation masks, addressing common weaknesses of SAM [27] such as coarse boundaries and difficulty with thin structures. Furthermore, we introduce a prior-guided linear programming formulation that assigns labels to 3D Gaussian primitives in a way that is both multiview consistent and semantically robust. By integrating priors which are learned jointly during 3D-GS training, our method regularizes the segmentation process and significantly improves robustness of object-level scene editing.

1.1 Background and Context

The problem of scene representation in 3D has a long history that spans both computer vision and computer graphics [57]. The early methods in graphics were dominated by polygonal meshes [58], which remain a standard in computer-aided design and gaming pipelines. In computer vision, multiview stereo (MVS) techniques[44] and structure-from-motion (SfM) [36] pipelines allowed the recovery of dense point clouds, which are effective for geometry estimation but offer limited capabilities for photorealistic rendering.

The need for volumetric reasoning led to voxel-based [58] representations, where the scene is discretized into a 3D grid. Although conceptually simple, voxel representations suffer from cubic memory scaling, making them impractical for large or detailed environments. The breakthrough came with Neural Radiance Fields (NeRFs) [32], which encode a continuous volumetric scene

function within the weights of a neural network, specifically Multi-Layer Perceptrons (MLPs). NeRFs demonstrated that neural representations could achieve state-of-the-art novel view synthesis, rivaling photogrammetry. Extensions such as Semantic-NeRF [10], N3F [51], and DFF [28] integrated semantic supervision or feature distillation to enable scene understanding and editing. However, NeRF-based methods remain slow to train (often requiring hours to days per scene) and costly to render, limiting their real-time usability.

The emergence of 3D Gaussian Splatting (3D-GS) [23] represents a significant shift towards explicit, real-time neural rendering. Here, scenes are represented by a set of 3D Gaussian ellipsoids, each parameterized by its position, orientation, scale, opacity, and color. This representation achieves high visual quality and rendering speeds of hundreds of frames per second, making it a promising candidate for interactive applications [12] such as AR/VR and real-time scene editing. However, 3D-GS in its standard form does not provide semantic segmentation or object-level decomposition, and thus cannot directly support tasks such as object removal or recolorization.

To bridge this gap, researchers have explored the integration of 2D foundation models into the 3D pipeline. The Segment Anything Model (SAM) [27] offers general-purpose segmentation capabilities, enabling automatic and prompt-based extraction of 2D object masks across a wide variety of scenes. By projecting these masks into 3D scene, one can assign semantic labels to Gaussian primitives without requiring manual 3D annotations. This approach avoids the painful cost of 3D anontated ground-truth datasets [19], which are very expensive to produce. Few other works [61, 10] have employed DINO [3] and SAM features to enforce multiview semantic consistency. Despite these advances, lifting 2D masks into 3D comes with challenges. SAM-generated masks are often inconsistent across views, leading to fragmented or incomplete 3D segmentations [31, 61, 22]. Further, they also struggle with fine structures, occlusions, and complex boundaries, resulting in artifacts that propagate into downstream editing tasks for example, removing an object may leave residual fragments in the scene, or recolorization may include adjacent structures. For consistent masks across views, few works [56, 9, 47] apply video-tracking across views to enforce that the same object is labeled consistently. Video tracking method (DEVA) [8] requires high-quality segmentation to achieve superior masks and consistent labeling. Thus, another practical issue lies in the 2D mask quality itself. The original SAM [27], despite powerful, often yields coarse boundaries and missing fine structures. For example, parts of thin objects or detailed contours may be omitted or broken in SAM-generated masks. To overcome this, recent work [31] has tried

SAM-HQ [22] mask in NeRF-based representation. SAM-HQ, a minimal adaptation of SAM that delivers significantly sharper and more accurate masks, achieves this by training on a curated set of 44K very high-quality masks and injecting a small high-quality output token into the SAM decoder. In 3D Gaussian Splatting (3D-GS), high-quality segmentation masks are critical for precise Gaussian-to-mask matching and preserving object boundaries. To achieve segmentation in 3D scenes, each Gaussian needs to be aware of the object class or label to which it belongs. One principled way to do this is to frame segmentation as an optimization or assignment problem. Recent work [47] shows that 2D rendering of a given 3D-GS scene can be formulated as a linear function of the labels assigned to each Gaussian supervised by ground-truth segmentation masks. Exploiting this property, we can set up a linear integer program (LP) whose solution gives the optimal assignment of each 3D Gaussian to an object label while exactly matching the multi-view masks. The authors [47] reported that this optimal solver takes only 30 seconds and greatly improves downstream editing tasks since segmented Gaussians can be clearly separated and manipulated. However, matching Gaussians to segmentation masks using pixel overlap counts can fail in situations where an object appears very differently in each view. For example, consider a small object that is partially occluded or seen at very oblique angles, its projected size and location may vary, so simply counting overlapping pixels may not correctly link all its Gaussians. To handle such a scenario, the authors [47] added the heuristic-based background bias term, which, while helpful, lacks robustness in capturing intricate scene details.

Our work introduces a new robust method by incorporating a learned label prior to this linear programming framework [47] for label reassignment. This learned label prior, trained jointly during 3D-GS optimization, captures scene-specific information that guides the label assignment process, thereby overcoming the limitations of purely count-based formulations. By combining SAM-HQ for high-quality 2D masks with labels prior guided LP-based label assignment, we establish a framework that ensures robust 3D segmentation, object removal, and recolorization in 3D Gaussian Splatting.

1.2 Objectives

The primary aim of this dissertation is to develop a novel framework for 3D Gaussian Splatting (3D-GS) scene that enables precise object removal and recolorization in 3D scenes through feature integration and interactive user prompts, enhancing scene editing capabilities for real-time applications. The challenges discussed above provide a natural set of objectives that guide the path toward achieving this aim. These objectives are as follows:

- Investigation of existing methods for 3D scene reconstruction and editing to establish a foundation for advancements in 3D Gaussian Splatting based research.
- Design and implement a method to integrate semantic features into Gaussian splats, creating a semantic 3D Gaussian splatting scene representation for enhanced scene understanding and manipulation.
- To generate high-quality, noise-free, and view-consistent segmentation masks for scene datasets, and establish them as a new resource for advancing future research in 3D scene editing tasks.
- Implement a robust prior-guided label assignment method for Gaussians in 3D Gaussian Splatting, enabling accurate object removal and re-colorization within 3D scene.
- To design and assess interactive object removal and recolorization in 3D Gaussian Splatting, enabled through user-specified point-based object selection.

In this dissertation, we propose a novel framework for object removal and recolorization within 3D Gaussian Splatting that leverages recent advances in foundation models for segmentation and incorporates learned priors into the 3D labeling process. Specifically, we adopt SAM-HQ (Segment Anything Model – High Quality) to generate improved 2D segmentation masks, addressing common weaknesses of SAM such as coarse boundaries and difficulty with thin structures. Furthermore, we introduce a prior-guided linear programming formulation that assigns 3D labels to Gaussian primitives in a way that is both multiview consistent and semantically robust. By integrating priors learned jointly during 3D-GS training, our method regularizes the segmentation process and significantly improves reliability for object-level scene editing.

1.3 Achievements

Our contribution includes:

- A high-quality multi-view mask dataset for 3D scene editing, generated using SAM-HQ and robust noise removal with video tracking.
  - The dataset ensures consistent object segmentation across multiple views of a scene, addressing typical noise and inconsistency issues in existing datasets.
  - This benchmark dataset can be used for future research in 3D scene editing, object removal, and recolorization tasks.
- Prior-based label reassignment for robust per-Gaussian label assignment in 3D Gaussian Splatting scenes.
  - A learned prior, trained jointly during 3D-GS optimization, guides the linear programming-based label assignment, improving segmentation accuracy.
  - The approach provides more reliable per-Gaussian labels, enabling higher fidelity object editing in complex 3D scenes.

1.4 Overview of Dissertation

Chapter 1: Introduction

This chapter presents the research objectives and summarizes the primary contributions such as high-quality mask dataset, high-quality feature integration, and prior-guided label reassignment for 3D Gaussian Splatting. Outline of the thesis will be given.

Chapter 2: Background Theory and Literature Review

This chapter surveys prior work in 3D scene reconstruction and 3D scene understanding, contrasting explicit representations with neural implicit methods such as NeRF. We justify the choice of 3D Gaussian Splatting (3D-GS) as the backbone for interactive editing, review feature-distillation, and mask-lifting approaches using 2D foundation models, and identify gaps in mask quality and label assignment methodologies that motivate our contributions.

Chapter 3: Semantic Feature and Object Mask Dataset Preparation

This chapter details the end-to-end pipeline used to construct the semantic features and high-quality object mask dataset. Semantic features extraction involves leveraging foundation model to create high-quality features, followed by a scene-specific autoencoder to compress these features, mitigating memory and computational challenges. Later, we discussed the need to generating consistent, high-quality object masks across multiple views. We describe a post-processing refinement procedure to remove noise, smooth boundaries, and eliminate small artifacts. The resulting new dataset of high-quality multiview masks forms a robust foundation for object-aware 3D-GS training and enables accurate segmentation in downstream 3D scene editing tasks.

Chapter 4: Semantic Feature Based 3D-GS Training

This chapter presents the methodology for embedding compressed 2D semantic features into 3D Gaussians and jointly training these features with standard Gaussian parameters. We describe the scene-specific autoencoder for feature compression, the differentiable rasterization that renders both RGB and feature maps, and the combined loss formulation that enforces photometric fidelity and semantic consistency during optimization.

Chapter 5: Prior Based Object Label Reassignment in 3D-GS

This chapter introduces our label reassignment framework that leverages learned per-Gaussian priors to regularize voting-based mask lifts. We explain how learned label priors are incorporated into a linear programming formulation to produce robust, semantically meaningful per-Gaussian labels. Our approach circumvents the heuristic background-bias adjustments employed in previous methods, therefore achieving enhanced editing robustness without compromising inference speed.

Chapter 6: Experiments And Results

This chapter presents the experimentation details and clearly provide quantitative and qualitative results. We compare reconstruction and segmentation performance across baselines, analyze ablations, and demonstrate practical editing tasks of object removal and recolorization to validate the effectiveness and generalization of our approach. We conduct several ablation studies to evaluate the impact of key parameters, such as the prior weight, on the effectiveness of object removal and recolorization in 3D Gaussian Splatting.

Chapter 7: Conclusions And Future Work

The last chapter concludes with a discussion of the presented approaches and results. Significance of findings, impact of the work, key challenges and future work are covered.

2 BACKGROUND THEORY AND LITERATURE REVIEW

This chapter presents the literature review of key advancements in 3D scene representation and segmentation, focusing on Neural Radiance Fields (NeRF), 3D Gaussian Splatting, and 2D-to-3D segmentation techniques. We discuss NeRF's limitations in rendering speed and editability, which 3D Gaussian Splatting overcomes as an efficient, editable alternative for novel view synthesis. We also survey advancements in 2D segmentation, focusing on state-of-the-art foundation models that excel in semantic and instance segmentation tasks. Subsequently, we analyze prior work on two key approaches, namely feature field distillation and 2D mask-lifting, that leverage 2D segmentation for 3D scene understanding. We highlight their strengths, such as improved feature transfer, and challenges, including issues with cross-view consistency and mask quality. Building on these challenges, we propose a novel methodology that utilizes 3D Gaussian Splatting and high-quality 2D segmentation features to achieve our objectives of object removal and recolorization in 3D scenes, ensuring efficiency and robustness.

2.1 Radiance Fields and 3D Gaussian Splatting

In the past several years, various representations have been employed for the Novel View Synthesis (NVS) applications [52, 36]. Radiance fields, particularly Neural Radiance Fields (NeRF) [32], have revolutionized 3D scene representation by encoding scenes as continuous volumetric functions learned from multiple 2D images. NeRF [42] enables high-fidelity novel view synthesis, capturing complex geometries and appearances through a neural network that predicts color and opacity for any point in 3D space. However, its reliance on neural network evaluations for rendering results in slow training and rendering times, often requiring seconds per frame even on high-end hardware [42]. This limitation stems from the need for per ray neural evaluations, making NeRF unsuitable for real-time applications like virtual reality or robotics. Few approaches like Instant-NGP [33] reduces the cost of neural primitives with a versatile new input encoding that permits the use of a smaller network without sacrificing quality, thus significantly reducing the number of floating point and memory access operations. Moreover, NeRF's implicit representation, lacking explicit geometric primitives, complicates scene editing. Editing tasks such as object removal or re-coloring, necessitate re-optimizing the entire neural network, which is computationally expensive and impractical for dynamic scenes.

These limitations prompted research into more computationally efficient and editable 3D representations, leading to the development of 3D Gaussian Splatting, often termed as significant breakthrough. 3D Gaussian Splatting, introduced by Kerbl et al [23], addresses the shortcomings of NeRF and traditional MVS methods by representing scenes explicitly as a collection of 3D Gaussians ellipsoids. Each Gaussian is defined by position, covariance (determining shape and orientation), opacity, and color, offering a point-based representation. It preserves the desirable properties of continuous volumetric radiance fields while avoiding unnecessary computation in empty space [12]. This explicit representation facilitates real-time rendering and editing, leveraging techniques like frustum culling and tile-based parallel rendering for efficiency, often accelerated by CUDA. The explicit nature of 3D GS allows for straightforward scene editing, such as object removal or re-coloring, by manipulating individual Gaussians without retraining the entire model [56, 37, 35, 61, 9]. A survey by Chen et al [12] provides a systematic overview, highlighting its transformative impact on 3D reconstruction and representation. Given the advantages, our work follows 3D Gaussian Splatting as the backbone scene representation in our proposed framework for creating consistent 3D segmentation tasks.

2.2 Segmentation in 2D Scene

Image segmentation has long been a core task in computer vision. With recent breakthroughs in 2D segmentation driven by the availability of large-scale datasets and advances in model architectures. Classical 2D segmentation paradigms, such as semantic segmentation [16, 45] and instance segmentation [19, 34, 50] have been extensively studied, benefiting from powerful backbone networks and rich training corpora. The advent of transformer-based models has revolutionized 2D image segmentation. The journey of transformer-based models in computer vision began with the introduction of vision transformers (ViTs) [11], which adapted the transformer architecture [25] from natural language processing to handle image data. This shift laid the groundwork for foundation models such as DINO (self-DIstillation with NO labels) [3], which emerged as a breakthrough in self-supervised learning for vision transformers. One of DINO's most striking properties is its ability to produce features that inherently contain information about semantic segmentation. In recent research, the Segment Anything Model (SAM) [27] has emerged which is designed as a foundation model that can segment any object in an image based on user prompts, such as points, boxes, or text descriptions. This prompt nature allows SAM to be highly versatile, enabling users to interactively refine segmentation results without retraining the model. Due to SAM's versatile architecture, which consists of an image encoder, a prompt encoder, and a mask decoder, sev-

eral recent works [41, 8] have successfully integrated SAM into downstream applications, further enhancing segmentation capabilities.

Although SAM revolutionized segmentation with its promptable design, it sometimes struggles with producing high-quality masks for objects with intricate structures. To address this limitation, SAM-HQ (High-Quality Segment Anything Model) [22], trained on 44K fine-grained masks, enhances SAM by introducing a learnable High-Quality Output Token into the mask decoder and combining features from early and final ViT layers. SAM-HQ builds on this foundation by ensuring that segmentation masks are of high quality, which is essential for tasks where precision is paramount, making it ideal for multiview consistent 3D segmentation.

2.3 From 2D to 3D Segmentation in Neural Scene Representations

Despite the maturity of 2D segmentation, 3D segmentation remains relatively underexplored. Traditional approaches rely on RGB-D images [19] or point clouds [34, 59], which require explicit depth or 3D representations as input. However, these methods often suffer from limited generalization due to the scarcity of large-scale 3D datasets and the high computational cost of processing 3D data directly [35]. This has led to growing interest in leveraging 2D segmentation to 3D for scene understanding, particularly from multi-view images. By exploiting the vast availability of 2D image datasets, foundation models like DINO and SAM have significantly helped in 3D segmentation with methods majorly categorized into feature field distillations and 2D mask-lifting.

2.3.1 Feature Field Distillations for 3D Segmentation

Feature field distillation transfers rich semantic features from 2D vision foundation models [3, 27] to 3D representations. Earlier works [10, 51, 28, 31] learn additional feature fields that are aligned with implicit representations using NeRFs, and used 2D visual features from pre-trained models or language embeddings to query the 3D features. These methods usually require modifying or retraining the original NeRF models or training other specific parameters to obtain the 3D segmentation. LERF [24] splits images into patches of different sizes to obtain multiscale CLIP [40] feature maps that can supervise neural field training. ISRF [13] uses DINO [3] features and K-Means clustering to separate user-selected regions from the background. However, lacking a powerful decoder, the segmentation based on these features is not accurate and sharp along the boundaries. Inspired by real-time 3D Gaussian Splatting, recent works [61, 37, 9, 30] have been proposed to achieve feature-based segmentations in Gaussian Splatting. SAGA [4] and

Feature-3DGS [61] distill the knowledge embedded in the SAM decoder into the feature field of 3D-GS, and Gaussian-Grouping [56] supervises the Identity Encodings during the differentiable rendering leveraging the 2D mask predictions by SAM. Few methods [37] utilize CLIP model to enable novel view semantic segmentation using language-guided editing.

However, distilling low-resolution DINO or SAM feature maps results in aliasing in the output masks, in the form of jagged mask edges. Therefore, 3D segmentation approaches that rely on feature distillation face inherent limitations, primarily stemming from the quality of the extracted 2D features and the constraints imposed by their dimensionality when embedding them into individual Gaussians. In this work, we employ high-quality 2D features extracted from SAM-HQ [22], which currently represents the state-of-the-art in segmentation feature quality. However, the exploration of feature dimensionality remains limited, primarily due to the significant computational and memory overhead associated with higher-dimensional embeddings.

2.3.2 2D Mask-lifting for 3D Segmentation

The integration of 2D segmentation masks into 3D scene representations has emerged as a promising strategy to enable instance-level scene understanding. This process, often referred to as mask lifting, leverages precomputed or model-generated segmentation masks to supervise 3D object separation. Recent research has explored lifting 2D segmentation masks from multiview images into 3D scene representations using both implicit neural fields and explicit volumetric models. In NeRF-based implicit representation, SA3D [5] enables interactive 3D segmentation by leveraging the Segment Anything Model (SAM) [27] on a single view. Nevertheless, this pipeline relies on the first view mask and is susceptible to the ambiguity inherent in SAM in delineating intricate structures during its self-prompting. While SA3D claims to be faster, it is still reliant on per-object prompting, SAM mask quality, and slow rendering. With similar work in 3D-Gaussian splatting, Gaussian Grouping [56] augments Gaussians with Identity Encodings, supervised by SAM's 2D masks and 3D spatial consistency, enabling fine-grained 3D segmentation and editing like object removal, but incurs significant memory and computational costs. Unified-Lift [62] lifts multiview SAM's 2D instance segmentation, using a codebook for accurate 3D segmentation. FlashSplat [47] redefines 3D-GS segmentation as a globally optimal linear assignment problem. Flashsplat states the rendering process as a linear function of Gaussian labels. The authors proposed a method that solves segmentation in a single forward pass using linear programming with the supervision of SAM's 2D masks. They also introduced a heuristic-based background bias term that, although

useful, exhibits limited robustness in accurately capturing detailed scene elements. While effective, reliance on 2D masks introduces challenges such as cross-view inconsistency, noise amplification, and limited adaptability to unseen viewpoints. In this work, we address the challenge of segmentation mask quality employing high-precision masks from the SAM-HQ [22] model, augmented with a noise filtering module to suppress artifacts or incomplete boundaries. Our framework extends previous work [47] pipeline by integrating learned per-Gaussian segmented labels, lifted from the filtered high-quality 2D masks, directly into the rendering process. This formulation enables segmentation to be solved in a single forward pass via linear programming, thereby improving both efficiency and robustness to noisy 2D supervision, without the need of any heuristic-based bias term.

2.4 Summary

Segmentation quality in 3D Gaussian Splatting is jointly constrained by the fidelity of extracted 2D features and the reliability of corresponding masks. Prior feature distillation methods often suffer from degraded 2D feature quality due to dimensionality reduction, while mask lifting approaches propagate noise from inconsistent or fragmented masks. In this work, we address both challenges employing high-quality SAM-HQ features to preserve fine-grained semantic cues, alongside a noise filtering module with SAM-HQ generated masks to enhance mask reliability. Building upon these refined inputs, we present our framework by embedding learned per-Gaussian semantic labels into 3D Gaussian Splatting representations, lifted from the filtered 2D masks. Later, these learned labels act as a regularizer in the rendering formulation, enabling globally optimal segmentation via a single-pass linear programming solution that is both computationally efficient and robust to imperfect annotations.

3 SEMANTIC FEATURE EXTRACTION AND OBJECT MASK GENERATION

3.1 Introduction

In this chapter, we focus on building the foundation for integrating feature information into the 3D Gaussian Splatting (3D-GS) framework. Since 3D-GS by itself only encodes geometric and appearance cues, we introduce 2D semantic feature extraction for semantic feature-based 3D-GS training method from Section 4.3, and object mask generation pipeline for our object mask-based 3D-GS training method from Section 5.2. We also present the end-to-end procedure developed to construct our new high-quality object mask dataset for 3D segmentation, designed to support and facilitate future research.

Section 3.2 outlines the procedure for preparing a dataset to train 2D semantic features from multiview images. The focus is on leveraging SAM-HQ [22] for high-quality semantic feature extraction, followed by a feature compression technique using a scene-specific autoencoder to address memory and computational constraints.

Section 3.3 focuses on generating consistent high-quality object masks in multiview images using the SAM-HQ [22] and DEVA [8] pipeline. It further describes a post-processing refinement step to reduce noise, smooth object boundaries, and remove small artifacts. The new dataset of high-quality multiview masks is created that forms the basis for object-level 3D-GS training.

3.2 2D Semantic Feature Extraction using SAM-HQ

Given set of input image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ for the scene consisting of mutliview images, we utilize a 2D foundation model, specifically SAM-HQ [22] to extract the feature vector. These D dimension features encode rich semantic information per pixel area or region in an image, making them suitable for downstream tasks such as segmentation. DINOv2 [3], while effective for self-supervised feature learning, lacks the pixel-wise granularity needed for precise segmentation. On the other hand, SAM [27] presents a new paradigm for segmentation tasks. It can accept a wide variety of input prompts and produce segmentation masks of different semantic levels as output. The versatility and generalizability of SAM suggest a new way to perform promptable object segmentation. And SAM-HQ, built on SAM, trained on a curated dataset of 44K fine-grained masks, enhances

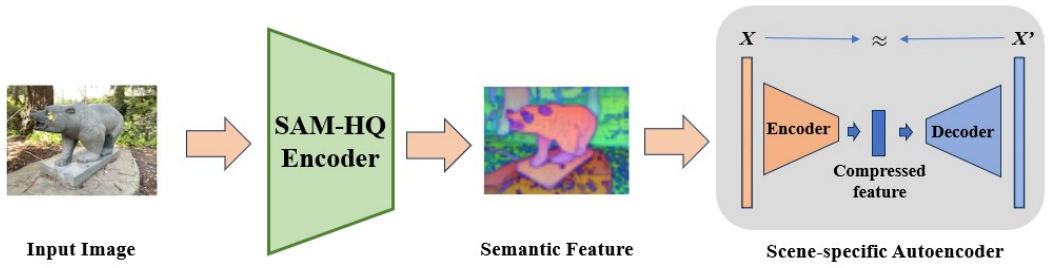

Figure 3.1: Semantic feature extraction via SAM-HQ encoder, compressed to a lower dimension using a scene-specific autoencoder for 3D Gaussian Splatting training.

feature quality by capturing detailed local and global contextual information. The pipeline for the extraction of the semantic feature is shown in Figure 3.1.

3.2.1 Feature Compression Using Scene-Specific Autoencoder

Integrating SAM-HQ D-dimensional feature vectors directly into 3D GS training poses significant memory and computational challenges [61], as each Gaussian would need to store high-dimensional features, increasing memory usage and slowing down the differentiable rasterization process [23]. To address this, a scene-specific autoencoder is employed to compress the D-dimensional SAM-HQ features $\mathbf{f} \in \mathbb{R}^D$ $(D = 256)$ into a lower-dimensional latent feature, $\hat{\mathbf{f}} \in \mathbb{R}^d$ $(d \ll D)$. The autoencoder shown in Figure 3.1, is trained specifically for the images used in a given 3D scene reconstruction, ensuring that it captures fine-grained, scene-specific details. The autoencoder consists of an encoder that maps the D-dimensional features to a latent space and a decoder that reconstructs the original features, optimized to preserve semantic integrity. Training is guided by a combination of L2 loss and cosine similarity loss, which together ensure numerical accuracy and semantic consistency within the compressed features.

$$\mathcal{L}_{\text{L2}}(\mathbf{f}) = \frac{1}{D} \left\| \mathbf{f} - \phi_d(\hat{\mathbf{f}}) \right\|_2^2, \quad \text{where } \hat{\mathbf{f}} = \phi_e(\mathbf{f}), \tag{3.1}$$

$$\phi_e : \mathbb{R}^D \to \mathbb{R}^d, \quad \text{the encoder that compresses } \mathbf{f} \text{ into } \hat{\mathbf{f}}.$$

The cosine similarity loss is defined as:

$$\mathcal{L}_{\cos}(\mathbf{f}) = 1 - \frac{\mathbf{f}^\top \phi_d(\hat{\mathbf{f}})}{\|\mathbf{f}\|_2 \, \|\phi_d(\hat{\mathbf{f}})\|_2}, \quad \text{where } \hat{\mathbf{f}} = \phi_e(\mathbf{f}). \tag{3.2}$$

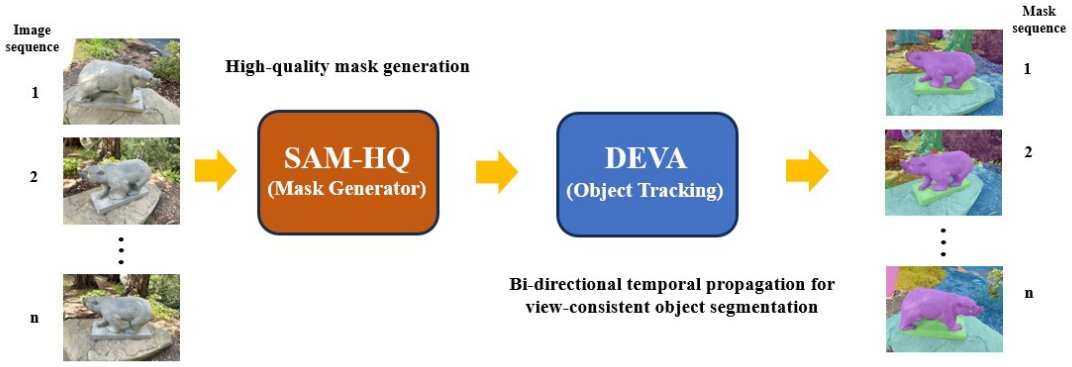

Figure 3.2: Object mask generation pipeline using SAM-HQ for mask generation and DEVA for object tracking to ensure consistent segmentation across views via bi-directional temporal propagation of mask IDs.

$$\phi_d : \mathbb{R}^d \to \mathbb{R}^D, \quad \text{the decoder that reconstructs } \hat{\mathbf{f}} \text{ into } \mathbf{f}.$$

where $\mathbf{f}^\top \phi_d(\hat{\mathbf{f}})$ is the dot product and $\|\mathbf{f}\|_2$ and $\|\phi_d(\hat{\mathbf{f}})\|_2$ are the Euclidean norms of the vectors. The combined loss function for scene-specific autoencoder is:

$$\mathcal{L}(\mathbf{f}) = \lambda_{\text{L2}}\mathcal{L}_{\text{L2}}(\mathbf{f}) + \lambda_{\cos}\mathcal{L}_{\cos}(\mathbf{f}), \tag{3.3}$$

where λ_{L2} and $\lambda_{\cos}$ are weighting factors, in practice, both are set to 1.

This scene-specific autoencoder approach also shown in methodology architecture Figure 4.3, ensures that the compressed features of any dimensions are readily available to associate with 3D Gaussians for learning semantic information within the reconstructed scene from Section 4.3.

3.3 2D Mask generation using SAM-HQ

The scene dataset [43, 56] used in this study consists of segmentation maps generated by the Segment Anything Model (SAM) [27]. Generating masks for images independently will introduce non-consistent labels for same objects in different views. Therefore, we utilize the Decoupled Video Segmentation Approach (DEVA), introduced by Cheng et al [8] at ICCV 2023, which is a framework designed to ”track anything” by leveraging task-specific image-level segmentation models combined with a universal, class-agnostic temporal propagation model. As shown in Figure 3.2, considering multiview images as sequential frames of the video of the scene, DEVA can be perfectly applied for consistent labeling of object segmentation class across multiple views. DEVA functions as a decoupled architecture, which separates the segmentation process into two

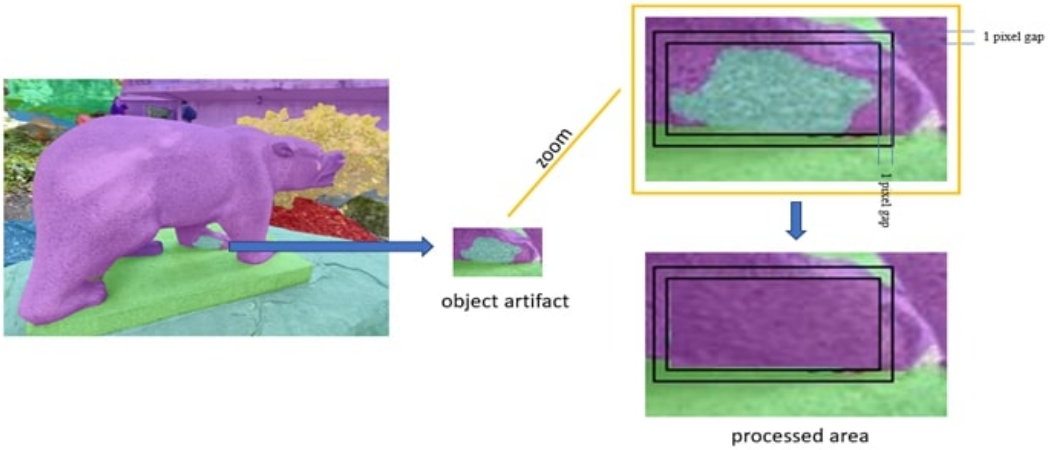

Figure 3.3: Processing of artifacts within 2D masks where for each identified artifacts, a bounding box expanded by 1 pixel is used to determine the most frequent surrounding label, and all object pixels are reassigned to this label, removing artifacts and ensuring segmentation consistency

independent modules: an image-level segmentation model and a bidirectional temporal propagation model. In our context, SAM-HQ [22] is applied for an image-level segmentation model to generate high-quality segmentation masks for each frame, producing labels ranging from 0 to 255, where each label corresponds to a distinct object or region. For multiview images, this module processes each view independently, generating per-frame masks that serve as the starting point for DEVA's pipeline. Later, DEVA employs a bidirectional propagation strategy, using techniques inspired by models like XMem [7], to track objects across frames. It maintains a memory bank of segmentation hypotheses, which stores object masks and their features over time. For each new frame, the propagation model matches the segmentation of current frame to previous ones, using feature similarity and spatial coherence to propagate labels.

3.3.1 Preprocessing of 2D Segmented Mask

To enhance the quality of multiview segmentation masks generated by SAM-HQ [22] with the DEVA [8] pipeline shown in Figure 3.2, a post-processing step refines the masks by addressing boundary noise and small object artifacts. The pipeline applies morphological closing [14] with a 3×3 square kernel to smooth jagged edges in each view's segmentation map, reducing model-induced noise that causes inconsistent labels. Additionally, small objects with areas below $area\_threshold$ pixels are removed using connected component analysis [18], as described below:

1. **Identify small objects**: Connected component analysis is applied to the binary segmentation mask to identify objects with an area less than $area\_threshold$ pixels.

2. **Determine dominant label**: For each identified object, a bounding box expanded by 1 pixel is created around it. The most frequent label within this bounding box is computed, excluding the object's own label.

3. **Relabel object pixels**: All pixels of the identified object are reassigned to the most frequent label from the expanded bounding box, eliminating spurious regions, and aligning the object with the dominant surrounding class.

This preprocessing pipeline, also shown in Figure 3.3, significantly improves the quality of the mask while the processed mask achieves an almost similar mean intersection over union (mIoU) with the original mask. This step is essential to prevent inconsistencies in 3D segmentation, as noisy masks can lead to fragmented or incorrect object assignments in the 3D scene. The cleaned masks are stored as per-image segmentation maps, with each pixel labeled by its corresponding object ID. These masks will be used for the supervision of the object mask-based 3D Gaussian Splatting training method from Section 5.2 and 5.3.

The pseudocode for the proposed connected component analysis for preprocessing is as follows:

Algorithm 1 Connected Component Analysis for Mask Refinement

```
for each object in segmentation mask do
    if object_area < area_threshold then
        Create 1-pixel-expanded bounding box
        Compute most frequent label (excluding object's label)
        Relabel object pixels with dominant label
    end if
end for
```

4 SEMANTIC FEATURE-GUIDED 3D-GS TRAINING

4.1 Introduction

In this chapter, we describe how the 3D Gaussian Splatting(3D-GS) framework is enhanced to support SAM-HQ [22] semantic feature learning for improved segmentation. The section details the formulation of a joint training approach that augments each Gaussian with a semantic feature, where **semantic features are extracted from the images of the given scene as described in previous Section** 3.2. This enables the creation of a semantic 3D-GS scene and drives capabilities for downstream tasks such as object removal and recolorization, as it allows specific objects to be precisely identified and manipulated in the 3D scene. A high-level overview diagram for the semantic feature-based training piprline is provided in Figure 4.1.

Section 4.2 presents a detailed overview of the 3D Gaussian Splatting framework for scene reconstruction through training and rendering. 3D Gaussian Splatting serves as the backbone for our work on object removal and recolorization in 3D scenes. This section further outlines the formulations for projection, rendering, and optimization, which constitute the most critical components of 3D-GS training.

Section 4.3 outlines the methodology for training semantic features from Section 3.2 as additional parameters within the original 3D Gaussian Splatting. The methodology describes how each Gaussian is augmented with a compressed feature vector and optimized through a differentiable rasterization pipeline. The section further introduces a novel loss formulation to effectively supervise both appearance and semantic consistency during training, and concludes with the steps for inference.

4.2 Preliminaries: 3D Gaussian Splatting

3D Gaussian splatting [23] is a rendering and reconstruction technique that models a 3D scene as a set of 3D Gaussian primitives, each defined by position, covariance, opacity, and color. Gaussian splatting leverages the continuous and differentiable properties of Gaussian functions to achieve smooth rendering and efficient optimization. This method is particularly effective for novel view synthesis and scene reconstruction, as it balances computational efficiency with high-quality rendering, as demonstrated in recent works [23, 12].

Figure 4.1: Overview diagram of the Semantic Feature-Based Training Pipeline with Semantic Features input from Section 3.2 and Promptable Inference Using User-Selected 3D Point.

The primary components of 3D Gaussian splatting include:

- **3D Gaussian Primitives**: Each primitive is a 3D Gaussian function centered at a 3D position $\boldsymbol{\mu} \in \mathbb{R}^3$, with an anisotropic covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, an opacity $\alpha \in [0, 1]$, and appearance attributes such as spherical harmonics(SH) (for view-dependent color) or RGB values.
- **Splatting**: The process of projecting 3D Gaussians onto a 2D image plane, where their contributions are blended to form the final pixel colors using the SH-based appearance model.
- **Optimization**: The parameters of the Gaussians (position, covariance, opacity, color/spherical harmonic coefficients) are optimized using multiview image data to minimize reconstruction errors.

4.2.1 Mathematical Formulation

A 3D Gaussian primitive is defined by its probability density function in a 3D space. For a point $\mathbf{x} \in \mathbb{R}^3$, the Gaussian function is given by:

$$\mathcal{G}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{3/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \tag{4.1}$$

where $\boldsymbol{\mu}$ is the mean (center) of the Gaussian, $\boldsymbol{\Sigma}$ is the positive-definite covariance matrix controlling its shape and orientation, and $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$. The covariance matrix $\boldsymbol{\Sigma}$ is typically parameterized as $\boldsymbol{\Sigma} = RSS^\top R^\top$, where R is a rotation matrix and S is a diagonal scaling matrix, allowing anisotropic shapes. These Gaussians are initialized using the sparse structure-from-motion (SfM) point cloud, as shown in Figure 4.2.

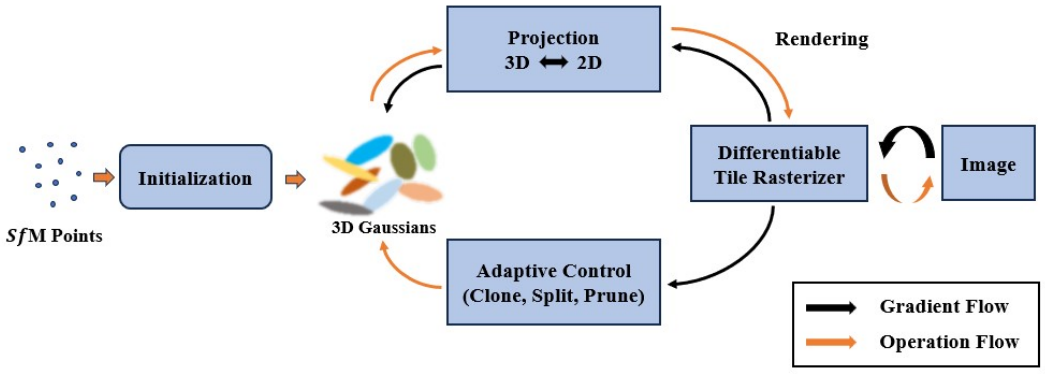

Figure 4.2: The training pipeline of 3D Gaussian Splatting [23] begins with a sparse SfM point cloud [46], which is converted into an initial set of 3D Gaussians. These Gaussians are then iteratively optimized and their density are adaptively updated. A tile-based differentiable renderer enables efficient forward rendering and gradient computation, thus supports real-time rendering of real world scene.

To render a scene, 3D Gaussians are projected onto the 2D image plane using a perspective projection $\mathcal{P}$. For a camera with an intrinsic matrix $K \in \mathbb{R}^{3\times3}$ and extrinsic parameters $[R_c|\mathbf{t}_c]$ (rotation R_c and translation $\mathbf{t}_c$), a 3D point $\mathbf{x}$ is projected onto a 2D point $\mathbf{u} = [u, v]^\top$ as:

$$\mathbf{u} = \mathcal{P}(\mathbf{x}) = K[R_c|\mathbf{t}_c] \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \tag{4.2}$$

where J is the Jacobian of the projection function $\mathcal{P}$ evaluated at $\boldsymbol{\mu}$. This 2D Gaussian is used to compute the contribution of each primitive to the color and opacity of the pixel.

4.2.2 Rendering with Gaussian Splatting

Although both the 2D and 3D Gaussian functions of Equation 4.1 are theoretically defined over the entire domain $(-\infty, \infty)$, their effective contributions decay exponentially with distance from the mean. In practice, this means that only the region close to the Gaussian center contributes significantly to the final image. During rendering, each 3D Gaussian primitive is projected onto the 2D image plane as an elliptical footprint determined by its covariance. Rather than integrating over the entire infinite support, the renderer evaluates contributions within this finite footprint, typically truncated to a few standard deviations (e.g., 3σ), where contributions outside this region are negligible. Rendering involves splatting the 3D Gaussians onto the 2D image plane and blending their contributions. For a pixel at position $\mathbf{u}$, the color $C(\mathbf{u})$ and opacity $\alpha(\mathbf{u})$ are computed by accumulating contributions from all Gaussians, sorted by depth to handle occlusion.

The rendering equation is:

$$C(\mathbf{u}) = \sum_{i=1}^{N} c_i \alpha_i \mathcal{G}_i(\mathbf{u}; \boldsymbol{\mu}_{2D,i}, \boldsymbol{\Sigma}_{2D,i}) \prod_{j<i} (1 - \alpha_j \mathcal{G}_j(\mathbf{u}; \boldsymbol{\mu}_{2D,j}, \boldsymbol{\Sigma}_{2D,j})), \tag{4.3}$$

where c_i is the color of the i-th Gaussian, α_i is its opacity, $\mathcal{G}_i$ is the 2D Gaussian from Equation 4.2, and the product term accounts for occlusion by Gaussians closer to the camera. The opacity $\alpha(\mathbf{u})$ is computed similarly as:

$$\alpha(\mathbf{u}) = 1 - \prod_{i=1}^{N} (1 - \alpha_i \mathcal{G}_i(\mathbf{u}; \boldsymbol{\mu}_{2D,i}, \boldsymbol{\Sigma}_{2D,i})). \tag{4.4}$$

4.2.3 Optimization for Scene Reconstruction

To reconstruct a 3D scene, the parameters of the Gaussians ($\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, α_i, c_i) are optimized to minimize the difference between rendered images and input multi-view images. Given a set of input images $\{I_k\}$ with known camera parameters $\{K_k, R_{c,k}, \mathbf{t}_{c,k}\}$, the optimization minimizes a hybrid loss function. Following the original 3D Gaussian Splatting formulation [23], a weighted combination of the L_1 loss and the Structural Similarity Index (SSIM) [6] loss is used:

$$\mathcal{L}_{\text{rgb}} = (1 - \lambda)\, \mathcal{L}_1(I_k, \hat{I}_k) + \lambda\, \mathcal{L}_{\text{D-SSIM}}(I_k, \hat{I}_k) \tag{4.5}$$

where I_k denotes the ground truth image, $\hat{I}_k$ is the rendered image from the current Gaussian parameters, and λ controls the balance between pixel-wise fidelity and perceptual similarity.

The optimization process for 3D scene reconstruction, also shown in Figure 4.2, involves:

1. **Initialization**: Gaussians are initialized from a SfM point cloud [46], with initial positions $\boldsymbol{\mu}_i$, isotropic covariances, and estimated colors and opacities.

2. **Gradient-Based Optimization**: The parameters are updated using gradient descent method, leveraging the differentiable property of Equations 4.3 and 4.4.

3. **Adaptive Control**: The number and properties of Gaussians are adjusted during optimization, such as **splitting** large Gaussians or **pruning** those with low opacity to improve efficiency and quality.

After introducing the core principles of 3D Gaussian Splatting (3D-GS), we now connect this foundation to our proposed methodology. Building upon this baseline, Section 4.3 extends 3D-GS by incorporating semantic feature-based training, where 2D semantic features are extracted, compressed as explained in Section 3.2, and embedded as an additional parameter into the 3D representation to enrich scene understanding. And Section 5.2 augments the base 3D-GS pipeline with object mask-based 3D-GS training, enabling explicit object-level reasoning and editing in 3D scene.

The pseudocode for 3D Gaussian Splatting optimization process is as follows:

Algorithm 2 Optimization of 3D Gaussian Splatting

Initialize set of Gaussians $\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, c_i\}_{i=1}^{N}$ from point cloud
for each iteration **do**
    for each input image I_k with camera parameters $\{K_k, R_{c,k}, \mathbf{t}_{c,k}\}$ **do**
        Project Gaussians to 2D using Equations 4.2
        Render image using Equations 4.3 and 4.4
        Compute loss $\mathcal{L}$ using Equation 4.5
        Update parameters via gradient descent
    end for
    Adjust Gaussian count (split or prune) based on adaptive criteria
end for

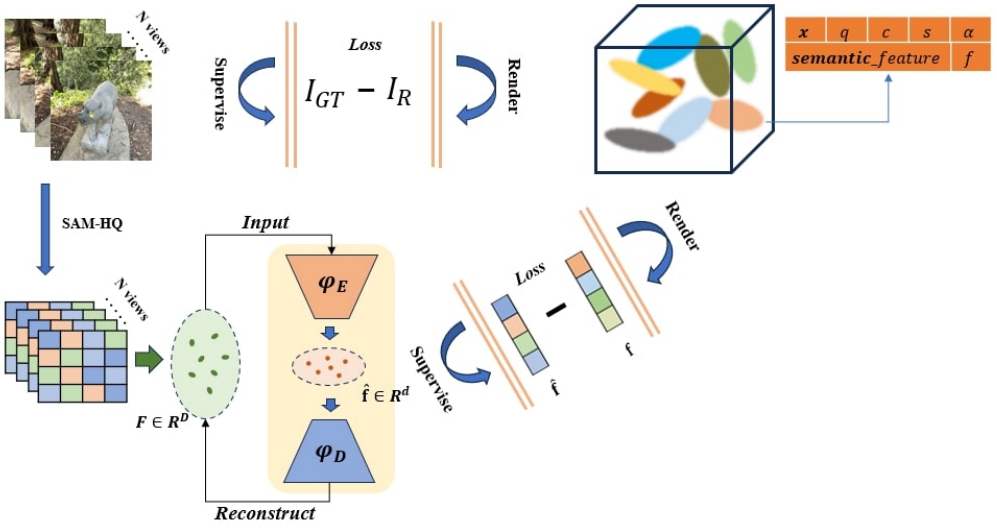

Figure 4.3: The architecture diagram illustrates our semantic feature-based 3D Gaussian Splatting training pipeline. High-dimensional 2D image features are first extracted across multiple views images I_{GT} using SAM-HQ. These features are then compressed into a lower-dimensional representation of dimension d using a scene-specific autoencoder. The compressed features $\hat{\mathbf{f}}$ are subsequently used to supervise the semantic features $\mathbf{f}$ of 3D Gaussians. This enables semantic feature learning to occur in parallel with the reconstruction of the 3D scene, effectively embedding multi-view semantic information into the 3D Gaussian representation.

4.3 Semantic Feature Training and Rendering

The 3D GS framework represents a scene as a collection of 3D Gaussians, each defined by a set of optimizeable parameters as explained in Section 4.2. To enable semantic segmentation, we extend each Gaussian with the compressed feature vector $\mathbf{f}_{d,i} \in \mathbb{R}^d$ extracted from Section 3.2, resulting in the set of parameters for the i-th Gaussian: $\Theta_i = \{\mathbf{x}_i, \mathbf{q}_i, \mathbf{s}_i, \alpha_i, \mathbf{c}_i, \mathbf{f}_{d,i}\}$.

3D Gaussians are initialized using a sparse point cloud generated by Structure from Motion (SfM), typically implemented with COLMAP [46], which provides initial 3D points and camera poses from the input images. Each Gaussian is defined by a 3D covariance matrix Σ, which must be positive semi-definite to ensure physical validity. To enforce this, the covariance is decomposed as follows:

$$\Sigma = \mathbf{RSS}^T\mathbf{R}^T,$$

where $\mathbf{R}$ is a rotation matrix derived from the quaternion $\mathbf{q}_i$, and $\mathbf{S}$ is a diagonal scaling matrix from $\mathbf{s}_i$. During rendering, the 3D Gaussians are projected to 2D image space, transforming the

covariance matrix to camera coordinates as:

$$\Sigma' = \mathbf{JW}\Sigma\mathbf{W}^T\mathbf{J}^T,$$

where $\mathbf{W}$ is the world-to-camera transformation matrix, and $\mathbf{J}$ is the Jacobian of the projective transformation's affine approximation [23]. This projection enables differentiable rendering, crucial for optimizing both visual and semantic attributes. To incorporate compressed semantic features $\mathbf{f}_{d,i}$, the differentiable rasterization pipeline of 3D Gaussian Splatting is modified to jointly render RGB colors and feature maps. The rendering process follows a volumetric approach, where the color $\mathbf{C}$ and the feature value $\mathbf{F}_s$ of a pixel are computed using front-to-back blending, as stated below:

$$\mathbf{C} = \sum_{i \in \mathcal{N}} \mathbf{c}_i \alpha_i T_i, \quad \mathbf{F}_s = \sum_{i \in \mathcal{N}} \mathbf{f}_{d,i} \alpha_i T_i,$$

where $\mathcal{N}$ is the set of Gaussians overlapping the pixel, sorted by depth, and

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

is the transmittance (the product of opacity complements of preceding Gaussians) described in Section 4.2. The quantity $\mathbf{F}_s$ denotes the "student" feature rendered from the 3D Gaussians. The feature rendering leverages the same tile-based rasterization used for RGB images, dividing the screen into 16×16 pixel tiles, with each thread processing one pixel [23]. Gaussians are culled against the view frustum and each tile to optimize performance, ensuring that both color and feature maps are rendered at the same resolution but with different dimensions ($\mathbf{c}_i \in \mathbb{R}^3$ for color, $\mathbf{f}_{d,i} \in \mathbb{R}^d$ for features). This joint optimization approach and the use of CUDA-accelerated rasterization, as implemented in frameworks like gsplat [49], enhances efficiency, allowing real-time rendering of both visual and semantic information.

The rendered feature map $\mathbf{F}_s(\hat{I})$ is supervised by the ground truth feature map $\mathbf{F}_t(I)$, using $\mathcal{L}_1$ loss, also shown in Figure 4.3. This feature loss is combined with the standard photometric loss $\mathcal{L}_{\text{rgb}}$ given by Equation 4.5 of 3D Gaussian Splatting.
The combined loss is given as below :

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_f \mathcal{L}_f, \tag{4.6}$$

where

$$\mathcal{L}_f = \|\mathbf{F}_t(I) - \mathbf{F}_s(\hat{I})\|_1, \tag{4.7}$$

where I is the ground truth image, $\hat{I}$ is our rendered image, and λ_f is a hyperparameter. The latent embedding $\mathbf{F}_t(I)$ is derived from the SAM-HQ [22] by encoding the image I as explained in Section 3.2, while $\mathbf{F}_s(\hat{I})$ represents our rendered feature map. To ensure an identical resolution $H \times W$ for the calculation of the per-pixel $\mathcal{L}_f$ loss, we apply bilinear interpolation to resize $\mathbf{F}_s(\hat{I})$ accordingly.

4.4 Promptable 3D Scene Editing with Semantic Feature-Based 3D-GS

Once the semantic 3D Gaussian Splatting (3D-GS) scene has been trained from above Section 4.3, we present the inference process for interactive scene editing, focusing on object removal and recolorization tasks. This process leverages the semantic features embedded within each Gaussian to enable user-guided editing. The pipeline begins with a user-selected 3D point prompt that specifies the target object in the reconstructed scene. To identify the semantic feature for the query object, the K-Nearest Gaussians around the selected point are retrieved, and their mean semantic feature is computed. Further, a cosine similarity search is then performed between this semantic query feature and all the Gaussians, followed by thresholding to extract all Gaussians corresponding to the selected object. For object removal, the associated Gaussians are deleted from the scene representation, whereas for object recolorization, the RGB values of the identified Gaussians are updated.

Mathematically, given a 3D point $p \in \mathbb{R}^3$ selected by the user as the target object location. The k-nearest Gaussians $\{g_1, g_2, \ldots, g_k\}$ around p are retrieved, and their semantic features $\{f_1, f_2, \ldots, f_k\}$ are averaged to form a query feature:

$$f_q = \frac{1}{k} \sum_{i=1}^{k} f_i.$$

Later, for each Gaussian g_j with semantic feature f_j, cosine similarity is computed as:

$$s_j = \frac{f_q \cdot f_j}{\|f_q\| \, \|f_j\|}.$$

Finally, a threshold τ (set default to 0.3) is applied to select the object set:

$$\mathcal{G}_{\text{obj}} = \{g_j \mid s_j \geq \tau\}.$$

After applying the editing operation, the modified 3D Gaussian scene is re-rendered from the camera viewpoints, producing the final visual results which can be viewed in form of 2D images.

5 LEARNED PRIOR-BASED OBJECT LABEL REASSIGNMENT FOR 3D-GS

5.1 Introduction

This chapter explains how high-quality object masks obtained from Section 3.3, are trained and integrated into the 3D Gaussian Splatting framework to create object-aware 3D scenes. Unlike the semantic feature-based approach described earlier in Section 4.3, this formulation leverages object masks directly, providing explicit object-level supervision rather than relying on compressed semantic features. This distinction enables a more targeted and interpretable integration, ensuring that the reconstructed 3D scenes capture both geometric fidelity and object-level consistency. Further, the chapter also describes a novel method for reassignment of object labels to 3D Gaussians using learned priors, enabling robust segmentation even when a Gaussian contributes to multiple objects across different views. A high-level overview diagram for high-quality object mask-based training pipeline is shown in Figure 5.1.

Section 5.2 outlines the methodology for training and rendering **high-quality object masks** from Section 3.3, to integrate into 3D Gaussian Splatting (3D-GS) for object segmentation. A 16-dimensional object-specific feature vector is incorporated into each 3D Gaussian as a trainable parameter. The training produces object-aware 3D Gaussian splatting scene.

Section 5.3 presents a novel approach for reassigning object labels to 3D Gaussians in the 3D Gaussian Splatting scene, leveraging learned object priors from Section 5.2 to achieve robust 3D scene segmentation. The proposed approach acknowledges that a Gaussian may contribute to multiple object classes across different views due to the non-exclusive nature of 3D-GS projections. By employing prior-based label reassignment, it **circumvents the need for heuristic background bias** used in previous work [47].

5.2 Training and Rendering Object feature for 3D-GS Segmentation

Given a set of images $\mathcal{I}$ of the scene and its segmentation mask prepared using the method from Section 3.3, we aim to enable direct object segmentation in 3D scenes. The 3D Gaussian Splatting framework from Section 4.2 is extended by embedding a 16-dimensional object feature vector, $\mathbf{f}_{o,i} \in \mathbb{R}^{16}$, into each Gaussian, referred to as the ***Object Feature***. This vector encodes the instance ID of the object to which the Gaussian belongs, distinguishing up to 256 unique objects in a scene.

Figure 5.1: Overview diagram of the Object feature-Based Training Pipeline with High-Quality Object Masks from Section 3.3, Prior-Guided Label Reassignment, and Promptable Inference Using User-Selected 3D Points.

Unlike the view-dependent spherical harmonics (SH) coefficients used for color rendering, the Object Feature is view-invariant, ensuring consistent object identification across all viewpoints. This design choice aligns with the need for multiview consistent segmentation, as highlighted in Section 3.3.

The 3D-GS framework represents a scene as a collection of 3D Gaussians as described in Section 4.2. The addition of the Object Feature $\mathbf{f}_{o,i} \in \mathbb{R}^{16}$ expands the Gaussians parameter set as:

$$\Theta_i = \{\mathbf{x}_i, \mathbf{q}_i, \mathbf{s}_i, \alpha_i, \mathbf{c}_i, \mathbf{f}_{o,i}\}.$$

The 16-dimensional feature size is chosen to balance computational efficiency with the capacity to represent a large number of objects, as validated in Gaussian Grouping [56], where a similar dimensionality effectively distinguished scene instances. The view-invariant property is enforced by setting the spherical harmonics(SH) degree of $\mathbf{f}_{o,i}$ to zero, modeling only the direct-current component. This ensures that the object ID remains consistent regardless of the viewing angle, unlike the view-dependent color attributes.

The initialization of the 3D Gaussians is performed using a sparse point cloud generated by Structure from Motion (SfM) with COLMAP [46], which provides initial 3D points and camera poses from the input images, similar original 3D-GS training described in Figure 4.2. Each Gaussian's 3D covariance matrix Σ^{3D} is parameterized to ensure positive semi-definiteness:

$$\Sigma^{3D} = \mathbf{RSS}^T\mathbf{R}^T,$$

where $\mathbf{R}$ is the rotation matrix derived from $\mathbf{q}_i$, and $\mathbf{S}$ is the diagonal scaling matrix from $\mathbf{s}_i$.

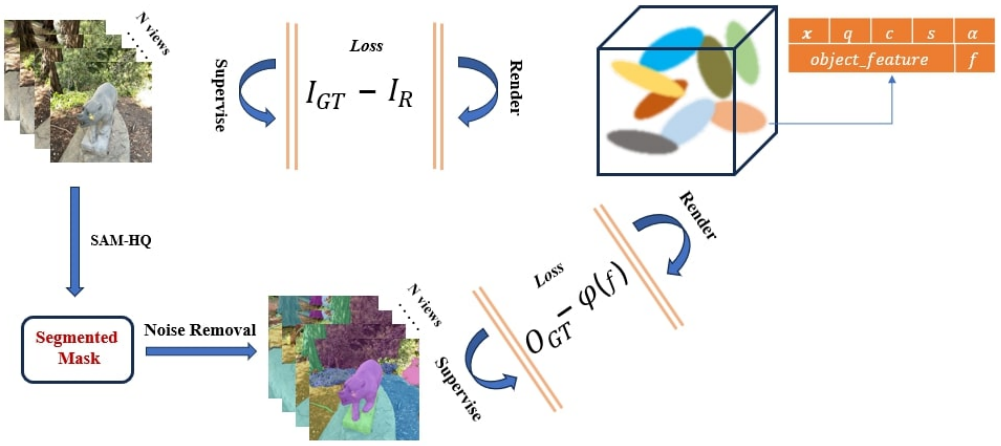

Figure 5.2: Overview of the object feature-based 3D Gaussian Splatting training, where high-quality object masks are generated for each image I_{GT} across N views using SAM-HQ assisted video tracking DEVA, followed by preprocessing to ensure cleaner and multi-view consistent masks. During training, per-Gaussian object features of dimension 16 are jointly trained with other Gaussian parameters, mapped via a linear layer $\phi(f)$ to 256-dimensional class probabilities, and supervised using the ground truth masks labels O_{GT}. This enables label-aware 3D Gaussian splatting for interactive scene editing.

For rendering, the covariance is projected to 2D image space:

$$\Sigma^{2D} = \mathbf{JW}\Sigma^{3D}\mathbf{W}^T\mathbf{J}^T,$$

where $\mathbf{W}$ is the world-to-camera transformation matrix, and $\mathbf{J}$ is the Jacobian of the affine approximation of the projective transformation [23]. This projection enables differentiable rendering, allowing simultaneous optimization of both visual and object-specific attributes.

The Object Feature is initialized randomly or with a heuristic based on initial SfM point correspondences, as suggested in Click-Gaussian [9], to provide a starting point for optimization. As shown in Figure 5.2, the 3D-GS rasterization pipeline is adapted to jointly render RGB colors and Object Features, while ensuring not to reduce the original scene reconstruction quality. The rendering process employs a neural point-based alpha-blending approach described in Section 4.2.2, where the influence of each Gaussian on a pixel is determined by its 2D projection. For a given pixel, the rendered color $\mathbf{C}$ and object feature $\mathbf{F}_0$ are computed as:

$$\mathbf{C} = \sum_{i \in \mathcal{N}} \mathbf{c}_i \, \alpha_i' \, T_i, \quad \mathbf{F}_{\text{o}} = \sum_{i \in \mathcal{N}} \mathbf{f}_{o,i} \, \alpha_i' \, T_i, \tag{5.1}$$

where $\mathcal{N}$ is the set of Gaussians overlapping the pixel, sorted by depth, and

$$T_i = \prod_{j=1}^{i-1}(1 - \alpha'_j), \tag{5.2}$$

is the transmittance. The term α'_i given below

$$\alpha'_i = \alpha_i \cdot \mathcal{G}(\Sigma_i^{2D}) \tag{5.3}$$

is the influence weight, computed as the product of the learned opacity α_i and a 2D Gaussian distribution with covariance Σ_i^{2D} [23].

The object feature $\mathbf{F}_{\text{o}} \in \mathbb{R}^{16}$ represents the aggregated Object Feature for the pixel, encoding the object mask $\mathbf{O}$. This joint rendering approach, inspired by Gaussian Grouping [56], ensures that the object feature map maintains high fidelity, crucial for accurate object segmentation. Later, the rendered 16-dimensional object feature $\mathbf{F}_o$ for each pixel is passed through a trainable linear layer that maps it to a 256-dimensional space, corresponding to the maximum number of object classes in the scene:

$$\mathbf{Z} = \phi_L(\mathbf{F}_o), \tag{5.4}$$

where ϕ_L is the single perceptron layer, and $\mathbf{Z} \in \mathbb{R}^{256}$ is the logit output vector. Later, a softmax function converts $\mathbf{Z}$ into a probability distribution over the 256 object classes:

$$\mathbf{P}(k) = \frac{\exp(Z_k)}{\sum_{j=0}^{255}\exp(Z_j)}, \tag{5.5}$$

where $\mathbf{P}(k)$ is the predicted probability for class k. The predicted object class is obtained by taking $\arg\max$ of $\mathbf{P}$.

Linear layer and object features are optimized using a cross-entropy loss [60], which supervises the predicted probabilities against the ground-truth object mask ids from SAM-HQ [22]:

$$\mathcal{L}_{\text{obj\_seg}} = -\frac{1}{N_p}\sum_{p=1}^{N_p}\sum_{k=0}^{255}\mathbf{1}\{y_p = k\}\log\big(\mathbf{P}_p(k)\big), \tag{5.6}$$

where N_p is the number of pixels, $y_p \in [0, 255]$ is the ground-truth object mask id for pixel p, $\mathbf{1}\{y_p = k\}$ is the indicator function (equal to 1 if $y_p = k$, and 0 otherwise), and $\mathbf{P}_p(k)$ is the predicted probability for class k at pixel p.

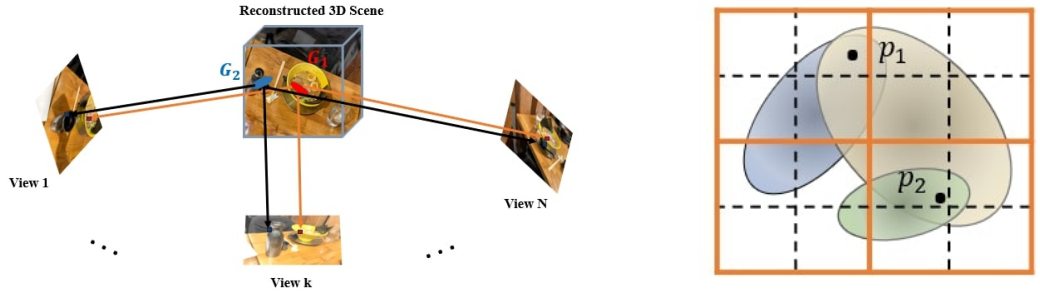

Figure 5.3: Overview of 3D to 2D projection and GS rasterization. Left: Illustration of 3D Gaussian Splatting scene where Gaussians G_1 (for object **bottle**) and G_2 (for object **bowl**) project differently across views: in view k they correspond to distinct objects, in view 1 they show partial overlap, and in view N both contribute to the same object (**bowl**). Right: Visual representation of projected 2D Gaussians in the 3DGS rasterization process, where each tile is shown as 2×2 for illustration. Gaussians are shared across tiles and can overlap between different object instances for example, near p_1 and p_2.

This segmentation loss is chosen for its effectiveness in multiclass classification [20], encouraging the model to assign high probabilities to the correct object mask ids. The overall training objective combines this classification loss with the standard 3D Gaussian Splatting losses $\mathcal{L}_{\text{rgb}}$ as described in Section 4.5:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rgb}} + \gamma_{\text{obj\_seg}} \mathcal{L}_{\text{obj\_seg}}, \tag{5.7}$$

where $\gamma_{\text{obj\_seg}}$ are weighting factors controlling the importance of feature consistency and mask classification accuracy relative to RGB reconstruction.

5.3 Object Label Reassignment Using Object Priors

The 3D Gaussian Splatting framework from above represents a scene as a collection of 3D Gaussians, each with attributes including position, covariance, opacity, color (via spherical harmonics), and a 16-dimensional Object Feature $\mathbf{f}_{o,i} \in \mathbb{R}^{16}$, as described in the previous Section 5.2. These features, trained to encode mask ids for up to 256 object classes, provide a prior probability distribution over object classes for each Gaussian. However, the projection of 3D Gaussians to 2D views can result in a single Gaussian contributing to multiple objects due to occlusions or overlapping projections, as illustrated in works such as Gaussian Grouping [56]. Figure 5.3 clearly shows how the particular Gaussian can point to different objects when projected from different camera views. To address this, we propose a label reassignment strategy that uses the learned priors to guide the assignment of labels $P_i \in \{0, 1, \ldots, 255\}$ to each Gaussian (i), where one Gaussian can have labels for multiple objects.

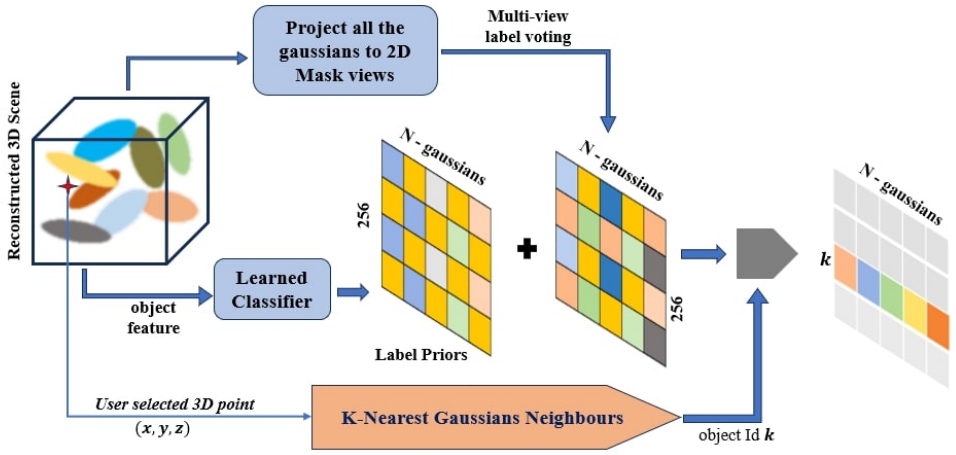

Figure 5.4: **Prior-based label reassignment**: The figure illustrates our novel pipeline for label reassignment, where a user-selected point (x, y, z) in the reconstructed 3D Gaussian Splatting scene is used to collect K-nearest Gaussian neighbors for object identification. In parallel, 3D Gaussians are projected onto the ground-truth object masks to compute label voting, which is combined with learned label priors to form a label matrix $R^{256\times \text{num\_gaussians}}$; the final binary mask for object k is then extracted by selecting all Gaussians that maximally contribute to the chosen object k.

The reassignment process constructs a contribution matrix $\mathbf{S} \in \mathbb{R}^{L\times N}$, where $L = 256$ is the number of object classes, and N is the number of Gaussians. This matrix captures the contribution of each Gaussian to each object class across multiple views, guided by 2D segmentation masks $M_v(j,k) \in \{0, 1, \ldots, 255\}$ from view v. The learned priors of Equation 5.5 described in Section 5.2, represented as a probability distribution $p_{l,i}$, where $p_{l,i}$ is the probability that Gaussian i belongs to class l, are incorporated to regularize the assignment, ensuring robustness against noisy or incomplete masks.

This research work [47] suggests that the contribution of Gaussian (i) to class (l) in view (v) can be computed by projecting the Gaussian onto the 2D image plane and aggregating its influence, weighted by its opacity and transmittance:

$$A_{l,i} = \sum_{v,j,k} \alpha_i(j,k) \cdot T_i(j,k) \cdot \mathbb{I}(M_v(j,k) = l), \tag{5.8}$$

where $\alpha_i(j,k)$ (from Equation 5.3) is the influence weight of Gaussian (i) on pixel (j,k), $T_i(j,k)$ (from Equation 5.2) is the transmittance accounting for occlusions, and $\mathbb{I}(M_v(j,k) = l)$ is an indicator function (1 if pixel (j,k) in view v has label l, 0 otherwise).

In addition, a simple majority voting approach [47] is employed to assign the label as:

$$P_i = \arg\max_l A_{l,i}, \tag{5.9}$$

However, this can be sensitive to noisy or incomplete masks, leading to incorrect assignments, especially in scenes with occlusions or overlapping objects. To improve robustness, we incorporate the learned object priors $p_{l,i}$, derived from the trained Object Features $\mathbf{f}_{o,i}$. These priors provide a probability distribution for the 256 classes.

Our label reassignment approach is formulated as an optimization problem, minimizing the error between rendered and ground-truth 2D masks while regularizing with the learned priors. As shown in Figure 5.4, to incorporate the learned object priors, a regularization term is added **while keeping the optimization objective a linear program**, solvable efficiently as noted in FlashSplat [47]. Thus, we get the modified majority voting approach that incorporates the priors directly into the contribution score.

$$A_{l,i}^{new} = \sum_{v,j,k} \alpha_i(j,k) \cdot T_i(j,k) \cdot \mathbb{I}(M_v(j,k) = l) + \gamma_p \cdot p_{i,l_i} \cdot \mathbb{I}(l_i = l), \tag{5.10}$$

where γ_p is a hyperparameter controlling the regularization strength and the new label assignment is done as below:

$$P_i^{new} = \arg\max_l A_{l,i}^{new} \tag{5.11}$$

This approach increases the confidence of the voting towards the prior label l_i, leveraging its score p_{i,l_i} to improve robustness against noisy masks. The resulting segmentation is represented as a matrix $\mathbf{S} \in \mathbb{R}^{256 \times N}$ where $S_{l,n} \in \{0, 1\}$ indicates whether Gaussian n belongs to the class l, also shown in Figure 5.4. Our approach enhances traditional label voting methods by integrating learned priors, **eliminating the reliance on heuristic background bias**, and improving robustness and multiview consistency.

5.4 Promptable Scene Editing Using Object-Specific Features in 3D-GS

The Inference process for the object feature-based 3D-GS framework with label reassignment, enables interactive object manipulation such as removal and recolorization. The pipeline begins with a user-selected 3D point prompt that specifies the target object within the reconstructed scene.

The K-Nearest Gaussians (default $K = 20$) around this point are retrieved, and their predicted object labels are aggregated using majority voting to obtain an initial object label, denoted as k.

Mathematically, given a user-selected 3D point $p \in \mathbb{R}^3$ from the reconstructed scene and K-Nearest Gaussians, where each Gaussian G_i in this neighborhood is associated with a predicted object label $\ell_i \in [0, 255]$. To infer the most likely object label for the query point p, we aggregate these labels using majority voting.

Formally, the initial object label assigned to p is defined as follows:

$$k = \text{Mode}\left(\{\ell_1, \ell_2, \ldots, \ell_K\}\right),$$

where $\text{Mode}(\cdot)$ returns the most frequent label among the K retrieved neighbors. This strategy ensures robustness against noisy or ambiguous predictions by exploiting local spatial consistency in the Gaussian representation.

Since each Gaussian in the 3D-GS representation is enriched with object features and corresponding 256-dimensional label probabilities, we refine this initial prediction using our label reassignment strategy as shown in figure 5.4. Specifically, Gaussians are projected onto ground-truth object masks to compute label voting, which is combined with learned label priors to form a label matrix $\mathbf{R} \in \mathbb{R}^{256 \times N}$, where N is the number of Gaussians. The final binary mask for the selected object is then extracted by identifying all Gaussians that maximally contribute to label k. For object removal, these Gaussians are removed from the scene representation, while for object recolorization, their RGB values are modified accordingly. After applying the editing operation, the updated 3D Gaussian scene is re-rendered from the camera viewpoints, allowing visualization of the manipulated results.

6 EXPERIMENTS AND RESULTS

In this section, we evaluate our method on quantitative and qualitative benchmarks for 3D object segmentation and editing tasks, specifically object removal and recolorization. Section 6.1 and 6.2 present the datasets and evaluation metrics used for evaluating our proposed framework. Section 6.3 provides implementation details for experiments conducted. Section 6.4 reports various quantitative results including metrics comparisons on the standard dataset and our new high-quality mask dataset, demonstrating competitive accuracy against state-of-the-art methods. Section 6.5 provides qualitative evaluations of extracted semantic features, mask preprocessing quality and practical editing tasks, including object recolorization and object removal, highlighting the effectiveness and limitations of our per-Gaussian representation. Further, Section 6.6 provides a few ablation studies to understand the behaviour of our method in different scenarios.

6.1 Datasets

We evaluate the performance on data from multiple datasets, containing synthetic and real-world scenes. To ensure comprehensive evaluation, we select six scenes from three widely recognized datasets: LeRF [24], Mip-NeRF [1], and LLFF [32, 56]. These datasets collectively provide a mix of synthetic and real-world scenes, capturing a variety of object types, lighting conditions, and geometric complexities, which are essential for testing the robustness of our 3D scene segmentation and editing approach. The dataset from the paper Gaussian Grouping [56] already contains the multiview-consistent mask generated using SAM [27] and DEVA [8], but the segmentation quality of mask is not superior. To ensure high-quality segmentation, we preprocess the 2D masks generated by DEVA with SAM-HQ [22] for each image in the selected scenes. The preprocessed masks significantly improve upon the quality as shown in Figure 3.3. By leveraging SAM-HQ's high-quality output and our preprocessing pipeline, we establish a new benchmark dataset, referred to as the High-Quality 3D Segmentation Dataset, for evaluating 3D segmentation methods and helping future research on high-quality 3D scene segmentation. More details about the new high-quality dataset are provided in Appendix 7.2.

6.2 Metrics

We adopted a combination of pixel-level, perceptual, and semantic metrics to evaluate the reconstruction quality of 3D Gaussian Splatting (3D-GS) scene. For scene reconstruction, we report

Peak Signal-to-Noise Ratio (PSNR) [54], Structural Similarity Index (SSIM) [6], and Learned Perceptual Image Patch Similarity (LPIPS) [48], also used in the original 3D-GS paper [23]. PSNR measures pixel-wise intensity differences between reconstructed and ground-truth images, with higher values reflecting improved fidelity. SSIM captures structural and perceptual similarity, providing a more reliable indicator of visual quality beyond pixel accuracy. LPIPS complements these metrics by assessing perceptual similarity in a learned deep feature space, aligning more closely with human visual perception. For evaluating scene editing and segmentation tasks, we employ mean Intersection over Union (mIoU) and mean Accuracy (mAcc). mIoU measures the overlap between predicted and ground-truth segmentation masks, while mAcc quantifies the proportion of correctly classified pixels. Together, these metrics assess both the quality of reconstructed views and the precision of object-level editing in 3D-GS, such as object removal and recolorization.

6.3 Implementation Details

The implementation of our proposed framework for feature-based 3D scene segmentation and editing leverages the official 3D GS codebase [23] with default parameters.

Semantic Feature and Autoencoder Training: We added features extracted from SAM-HQ [22] to each Gaussian. These semantic features, compressed from 256 dimensions to d-dimensional ($d = 16$ by default) using a scene-specific autoencoder from Section 3.2, are added to each Gaussian's parameters $\Theta_i = \{\mathbf{x}_i, \mathbf{q}_i, \mathbf{s}_i, \alpha_i, \mathbf{c}_i, \mathbf{f}_{d,i}\}$.The autoencoder is trained with a combined L2 and cosine similarity loss with $\lambda_{\text{L2}} = 1.0, \quad \lambda_{\cos} = 0.1$, using a batch size of 256, 50 epochs, and the Adam [26] optimizer with a learning rate of 0.001. For 3D-GS training, the feature map is supervised by an L1 loss with $\lambda_f = 1.0$. All parameters are jointly optimized using the Adam [26] optimizer with a default learning rate [23] and semantic features with a learning rate of 0.001, over $30{,}000$ iterations. These training follow a progressive spherical harmonics schedule with zeroth-order for the first $1{,}000$ iterations, adding one band every $1{,}000$ iterations up to the third order, as mentioned in the original 3D-GS paper [23]. The entire training process is executed on NVIDIA RTX A4000 GPU with 16 GB RAM, leveraging CUDA-accelerated rasterization to ensure efficient rendering.

Object Feature Training and Label Reassignment: For object feature training, we extends the official 3D GS codebase [23] by incorporating a 16-dimensional object feature vector $\mathbf{f}_{o,i} \in \mathbb{R}^{16}$ into each Gaussian's parameters, $\Theta_i = \{\mathbf{x}_i, \mathbf{q}_i, \mathbf{s}_i, \alpha_i, \mathbf{c}_i, \mathbf{f}_{o,i}\}$. Additionally, a trainable linear

Table 6.1: Details of Scene Preprocessing for Generating Refined High-Quality Object Masks.

Scenes	**Number of images**	**Num. of Objects** (segmented objects)	**mIOU** (after preprocessing)
bear	96	105	0.856
ramen	135	63	0.874
horns	62	57	0.905
teatime	180	220	0.801
fortress	42	10	0.991

layer for classification is added for mapping 16-dimensional input channel to 256-dimensional output channels. The rendered object features are passed through the linear layer to produce class probabilities, supervised by a cross-entropy loss with a weight $\lambda_{\text{obj}} = 1.0$. All parameters are jointly optimized using the Adam [26] optimizer with learning rates of 0.005 for object features and 0.0005 for the linear layer, over 30,000 iterations across all scenes.

For label reassignment, the contribution matrix $\mathbf{S} \in \mathbb{R}^{256\times L}$ is computed using tile-based rasterization, focusing on Gaussian attributes, $G_i = \{\mathbf{x}_i, \mathbf{q}_i, \mathbf{s}_i, \alpha_i\}$, excluding the color component, and performing alpha composition with learned label priors p_{i,l_i} as mentioned in equation 5.10. The default prior weight is set to $\gamma_{\text{prior}} = 0.10$, which can be adjusted to influence the prior knowledge in label reassignment.The entire 3D-GS training and reassignment process is executed on an NVIDIA RTX A4000 GPU with 16GB of RAM, leveraging CUDA-accelerated rasterization to ensure faster rendering.

6.4 Quantitative Results

6.4.1 High-Quality Mask Curation: Multi-View Consistency and Impact on Reconstruction

To study the effectiveness of our preprocessing pipeline on new high quality dataset, we performed analysis of new high-quality object mask dataset consisting of several representative scenes, as shown in Table 6.1. Each scene contains a varying number of input images and segmented objects, and the quality of the object masks was measured using the mean Intersection-over-Union (mIOU) metric after preprocessing. Our preprocessing step, which is based on SAM-HQ [22] segmentation followed by noise boundary removal and multiview consistency check, substantially improves overall mask quality. For example, *bear*, and *teatime* scenes, which contain a large num-

Table 6.2: Quantitative Analysis of Scene Reconstruction with Different Object Mask Generation method. Here, "Ours" refer to SAM-HQ + Noise Removal masks.

Scene	**PSNR ↑**			**SSIM ↑**			**LPIPS ↓**		
	SAM	SAM-HQ	Ours	SAM	SAM-HQ	Ours	SAM	SAM-HQ	Ours
bear	28.53	**28.85**	28.83	0.907	0.907	**0.908**	0.135	0.135	**0.135**
ramen	27.71	28.48	**28.49**	0.903	0.906	**0.907**	0.167	0.165	**0.161**
horns	-	23.74	**24.11**	-	0.855	**0.863**	-	0.213	**0.181**
teatime	30.09	30.29	**30.42**	0.913	0.915	**0.918**	0.141	0.137	**0.130**
fortress	-	33.31	**33.31**	-	0.932	**0.933**	-	0.111	**0.111**
ship	-	31.56	**31.64**	-	0.909	**0.912**	-	0.142	**0.137**

ber of segmented objects, tend to have more artifacts and noisy boundaries. After preprocessing, these scenes achieve mIOU values in the range of 0.80 - 0.85, demonstrating robustness even in highly cluttered environments. On the other hand, scenes such as *fortress* and *horns*, which contain significantly fewer segmented objects, achieve mIOU values greater than 0.90. By removing artifacts and merging fragmented segments, our preprocessing ensures that the masks not only serve current editing tasks but also act as a reliable high-quality benchmark dataset for future 3D scene editing research.

We further evaluated the impact of improved object masks on 3D Gaussian Splatting (3D-GS) scene reconstruction quality, we compare masks generated by SAM [27], SAM-HQ [22], and our proposed SAM-HQ with noise removal pipeline using PSNR [54], SSIM [6], and LPIPS [48] metrics, as reported in Table 6.2. Across multiple scenes such as bear, ramen, teatime, horns, fortress, and ship, SAM-HQ consistently outperforms SAM, achieving higher PSNR and SSIM and lower LPIPS, due to its sharper boundary predictions. Our preprocessing step which incorporates noise removal, further enhances performance, particularly in the ramen, horns, and teatime scenes, where it reduces label fragmentation and artifacts, yielding improvements in PSNR, for example, 30.42 compared to 30.29 in teatime, SSIM, for example, 0.918 compared to 0.915 in teatime, and LPIPS, for example, 0.130 compared to 0.137 in teatime. Notably, the fortress and ship scenes lack SAM masks, but exhibit slight improvements with our SAM-HQ + noise removal pipeline compared to SAM-HQ alone. These results suggest the positive impact of our noise removal pipeline in enhancing reconstruction quality. Overall, our new dataset with cleaner

Table 6.3: Performance Results on the New High-Quality Object Mask Dataset

Scenes with HQ mask	**mIoU (%) ↑**	**mAcc (%) ↑**
teatime	78.37	98.19
ramen	61.39	98.84
bear	89.98	98.36
horns	85.24	98.43
fortress	91.03	98.54
Overall	**81.20**	**98.47**

Table 6.4: Quantitative Results on the NVOS Dataset with 8 Front-Facing Scenes.

Method	**mIoU (%) ↑**	**mAcc (%) ↑**
NVOS	39.4	73.6
ISRF	70.1	92.0
FlashSplat	91.8	98.6
Ours	91.0	98.4

and more consistent object boundaries achieves improved reconstruction quality across scenes, confirming its effectiveness for high-quality 3D Gaussian Splatting scene.

6.4.2 Robust 3D Scene Segmentation on High Quality Masks

After creating the new high-quality object mask dataset, we evaluated the performance of our prior-guided label reassignment strategy for robust per-Gaussian segmentation in 3D Gaussian Splatting. For evaluation, we use the test set containing ground-truth (GT) object masks across scenes. Each test example provides a GT mask of the target object from one camera view. During testing, the GT mask from the reference view is used to generate corresponding object masks in all other viewpoints. The object is then removed via the rendering process, and the extracted object is projected back to the reference view to produce a predicted mask. Finally, the segmentation quality is quantified using the mean Intersection-over-Union (mIoU) and mean pixel Accuracy (mAcc) between the predicted and GT masks.

The results, summarized in Table 6.3, show strong performance across all five representative scenes in terms of both mIoU and mAcc. Our pipeline achieves very high pixel accuracy, consistently above 98%, demonstrating that the combination of high-quality masks and prior-guided

Table 6.5: Quantitative Study of 16-Dimensional Object Feature Integration in 3D Gaussian Splatting

Scenes	PSNR ↑		SSIM ↑		LPIPS ↓	
	w/o	with object	w/o	with object	w/o	with object
bear	29.23	28.23	0.906	0.908	0.134	0.135
horns	26.60	24.11	0.871	0.863	0.183	0.184
ramen	29.00	28.49	0.906	0.907	0.161	0.161
teatime	30.35	30.42	0.918	0.918	0.131	0.130
fortress	34.86	33.31	0.934	0.933	0.111	0.111
ship	33.49	31.64	0.921	0.912	0.134	0.137

label reassignment produces reliable and consistent labels at the per-Gaussian level. Dense and complex scenes, such as *teatime* and *ramen*, naturally present more challenging overlaps and occlusions, yet our method effectively mitigates fragmentation and noisy labeling, achieving mIoU values between 61–78%. In scenes with simpler object distributions, including *bear*, *horns*, and *fortress*, the prior-guided reassignment ensures nearly complete recovery of correct Gaussian labels, resulting in mIoU above 85% and pixel accuracy close to 99%. Overall, these results validate that integrating learned label priors with the curated high-quality masks enables robust per-Gaussian label assignment, thus, producing a 3D Gaussian Splatting representation that is suitable for downstream object-level editing tasks such as object removal and recolorization.

6.4.3 3D Object Segmentation Comparison on Existing Dataset

We conducted a comparison study with existing methods on object-level 3D segmentation on the available NVOS dataset [43, 56]. Table 6.4 reports the evaluation results on the NVOS dataset containing eight front-facing scenes, where our method achieves an overall mIoU of 91.0% and mAcc of 98.4%. These results are highly competitive with the state-of-the-art approach, which obtains 91.8% mIoU and 98.6% mAcc. In comparison, earlier methods such as NVOS [43] and ISRF [13] show a substantially low segmentation accuracy, highlighting the effectiveness of our robust object feature learning and prior-guided label assignment in our pipeline. It is important to note that, unlike FlashSplat [47], our method jointly learns per-Gaussian object features along with other Gaussian parameters, thereby enabling label-aware 3D Gaussian splatting. Results from Table 6.5 indicates that the additional feature slightly reduces 3D scene reconstruction quality, where our method exhibits a marginal drop in both PSNR and SSIM compared to FlashSplat [47]. However,

Table 6.6: Quantitative Evaluation of Object Feature Joint Training in 3D Gaussian Splatting: Impact on Model Size and Convergence at 30K Iterations.(M=million, GB=GigaBytes)

Scenes	Num of Gaussians		Loss		Peak Memory (GB)	
	w/o	with object	w/o	with object	w/o	with object
bear	2.64M	3.33M	0.023	0.025	8.80	14.18
horns	0.97M	1.21M	0.024	0.033	9.13	14.78
ramen	0.72M	1.05M	0.022	0.023	6.52	9.80
teatime	2.32M	2.09M	0.020	0.020	9.79	13.48
fortress	0.77M	1.15M	0.012	0.013	9.90	13.56
ship	0.26M	1.19M	0.010	0.013	7.74	9.87

the **gain in object awareness more than compensates** for this trade-off, since FlashSplat relies solely on precomputed object masks without embedding any object-level features in the 3D representation. In contrast, our framework enriches the scene with explicit object features, ensuring that segmentation remains consistent across views and directly supporting editing tasks such as selective object removal and recolorization. Overall, these results demonstrate that our method achieves segmentation accuracy on par with the state-of-the-art while additionally producing a semantically structured 3D representation. This balance between high-quality reconstruction and label-aware Gaussians marks a crucial step forward in enabling interactive and editable 3D Gaussian splatting scenes.

6.4.4 Analysis of Computational Costs for Object Feature Joint Training in 3D-GS

We carried out a quantitative analysis to assess the impact of jointly training object-level semantic features with 3D Gaussian Splatting (3D GS) in terms of computational scalability, as summarized in Table 6.6. We observed that **joint training increases Gaussian counts** in most scenes, reflecting the complexity of encoding semantic boundaries. For scenes such as *bear*, number of Gaussians increases from 2.64 million to 3.33 million, while the *teatime* scene, with its cluttered fine structures, shows a reduction in Gaussians from 2.32 million to 2.09 million, suggesting improved point allocation efficiency. Convergence, measured by reconstruction loss, remains stable, with only marginal differences observed. This indicates that integrating object features does not substantially hinder photometric reconstruction quality. However, peak memory usage rises significantly (40-60%), with *bear* and *horns* requiring over 14 GigaBytes (GB) versus 9 GB of

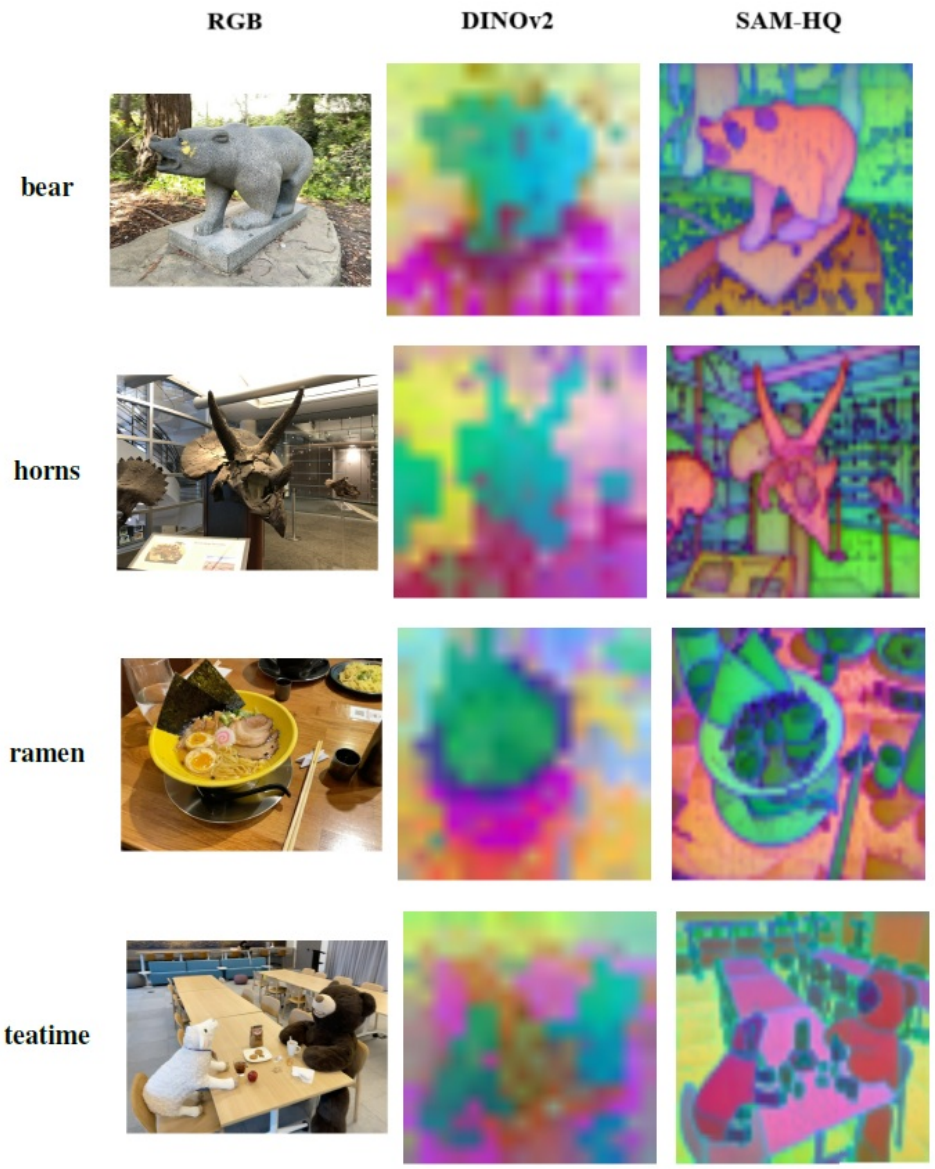

Figure 6.1: Qualitative feature comparison. From left to right: input RGB image, DINOv2 [3] extracted feature map, and SAM-HQ [22] extracted feature map. SAM-HQ produces sharper and more semantically meaningful features with clear object boundaries, whereas DINOv2 appears noisy and fails to preserve fine structure.

Graphical Processing Unit (GPU) memory without feature training, highlighting scalability challenges. The analysis suggests that while feasible for moderate-scale scenes, practical deployment at larger scales will require memory-efficient designs or adaptive strategies that balance semantic richness with computational overhead.

6.5 Qualitative Results

6.5.1 Extracted Semantic Feature Comparison

To better understand the quality of the features used for semantic supervision, we provide a qualitative comparison between the features extracted using foundation models, DINOv2 [3] and SAM-HQ [27], as shown in Figure 6.1. We observe that DINOv2 features tend to appear glittered and noisy, failing to provide sharp boundaries around objects. Although DINOv2 captures some degree of global semantic information, it struggles to preserve fine-grained details such as object contours and local boundaries. This limitation becomes problematic where accurate boundary alignment is essential for consistent segmentation across views.

In contrast, the SAM-HQ [27] features exhibit significantly higher quality, with clear object delineation and well-preserved structural boundary features, clearly visible in Figure 6.1. The extracted feature maps from SAM-HQ are able to robustly distinguish semantic objects within the scene, providing sharper and more interpretable representations. This qualitative evidence highlights the superiority of SAM-HQ over DINOv2 for our task, justifying its use as the backbone feature extractor to generate reliable and consistent semantics.

6.5.2 Object Mask Quality Comparison

For our high-quality dataset and its preprocessing pipeline, we present a qualitative comparison of segmented masks before and after preprocessing, as shown in Figure 6.2. While SAM-HQ [22] segmentation already provides reasonably accurate masks, they often contain noisy regions, fragmented segments, and small boundary artifacts. These problems are particularly noticeable in cluttered scenes, where artifact regions of a very small area ($area\_threshold$ = 500, by default) appear around object boundaries or in disconnected background regions. Such artifacts, if left uncorrected, can degrade the quality of object feature learning in 3D Gaussian Splatting. Our preprocessing pipeline effectively removes these unwanted components by cleaning noisy boundaries and small artifacts. As illustrated in the Figure 6.2, the refined masks are visually cleaner and more structurally consistent with the shapes of the ground-truth masks. This visualization of improvement and quantitative results from Table 6.2, directly supports our proposition that higher quality 2D masks lead to strong supervisor, allowing more accurate per-Gaussian object feature learning and high-quality 3D segmentation performance.

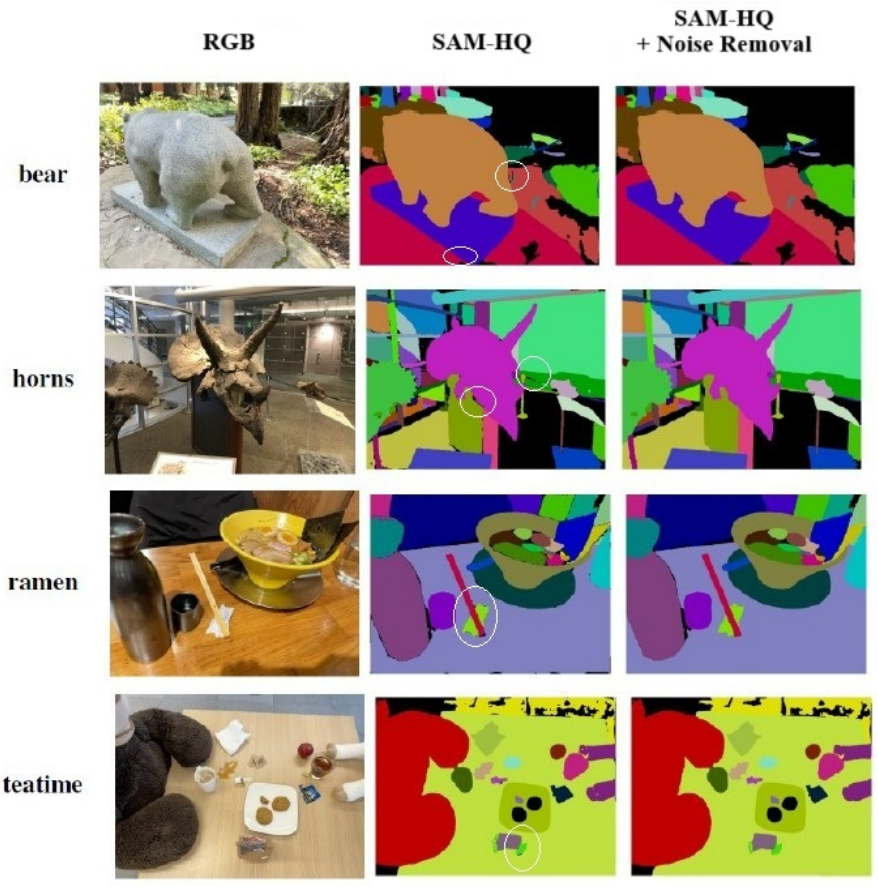

Figure 6.2: Qualitative comparison of segmented masks. From left to right: RGB image, SAM-HQ mask, and SAM-HQ mask after preprocessing. Our preprocessing step removes small noisy regions (< 500 pixels) and boundary artifacts, leading to high-quality masks for training object features in 3D Gaussian Splatting. **Areas circled in white color** show small boundary artifacts in each scene.

6.5.3 Object Re-colorization Task

In addition to evaluating segmentation quality, we demonstrate the effectiveness of our method in real editing scenarios through an object re-colorization task. The qualitative results shown in shown in Figure 6.3 and 6.4, highlight the superior performance of our approach to accurately segment and recolor user-selected objects in 3D Gaussian Splatting scene. For most scenes, the re-colorization is visually consistent and produces sharp boundaries around the target objects, clearly demonstrating the utility of per-Gaussian object features in practical 3D scene editing. The overall quality remains high, which indicates that our method is robust enough to support real-time applications, enabling high-quality object re-colorization as a step toward more advanced 3D scene editing tasks.

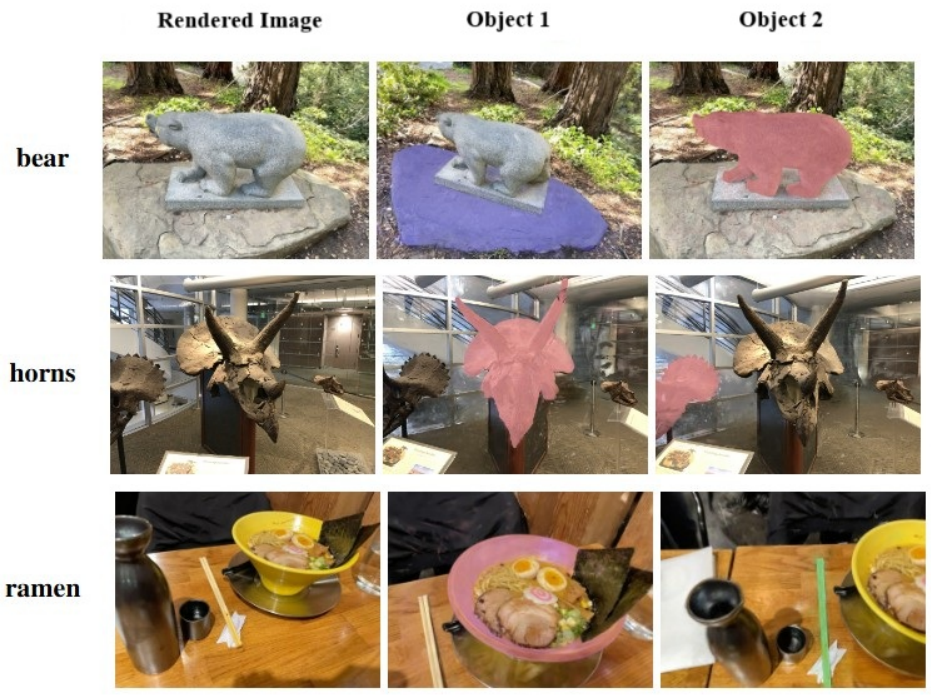

Figure 6.3: Qualitative results for the object re-colorization task, [**part 1**]. From left to right: RGB rendered image, rendered view with re-colored object, and rendered view with another re-colored object. The results show sharp object boundaries and high visual consistency, enabling effective real-time scene recolorization.

Further, results shown in Figure 6.5 demonstrates the quality of our recolorization results across multiple viewpoints of the scenes. The recolorization tasks performed at the level of 3D Gaussians rather than on individual 2D frames, remains consistent when rendering from different viewpoints. Importantly, object boundaries are sharply preserved, as label assignment and recolorization are constrained to the Gaussians lying exactly at the object edges. As a result, even fine structures, thin surfaces, and occlusion boundaries retain their integrity without color bleeding into neighboring regions. For instance, in ramen scene, the chopsticks, while occluded by a steel bottle, shows no artifacts in other views. These results highlight the effectiveness of our method for producing high-quality, multiview-consistent object edits.

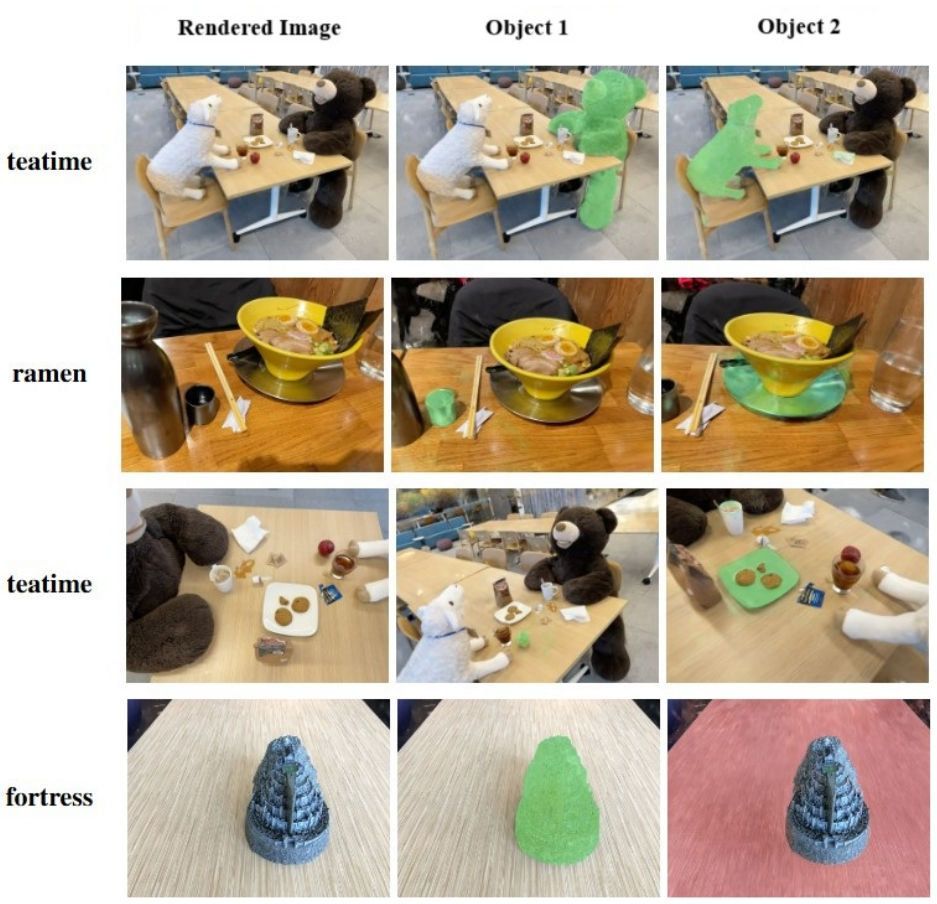

Figure 6.4: Qualitative results for the object re-colorization task, [**part 2**]. From left to right: RGB rendered image, rendered view with re-colored object, and rendered view with another re-colored object. The results show sharp object boundaries and high visual consistency, enabling effective real-time scene recolorization.

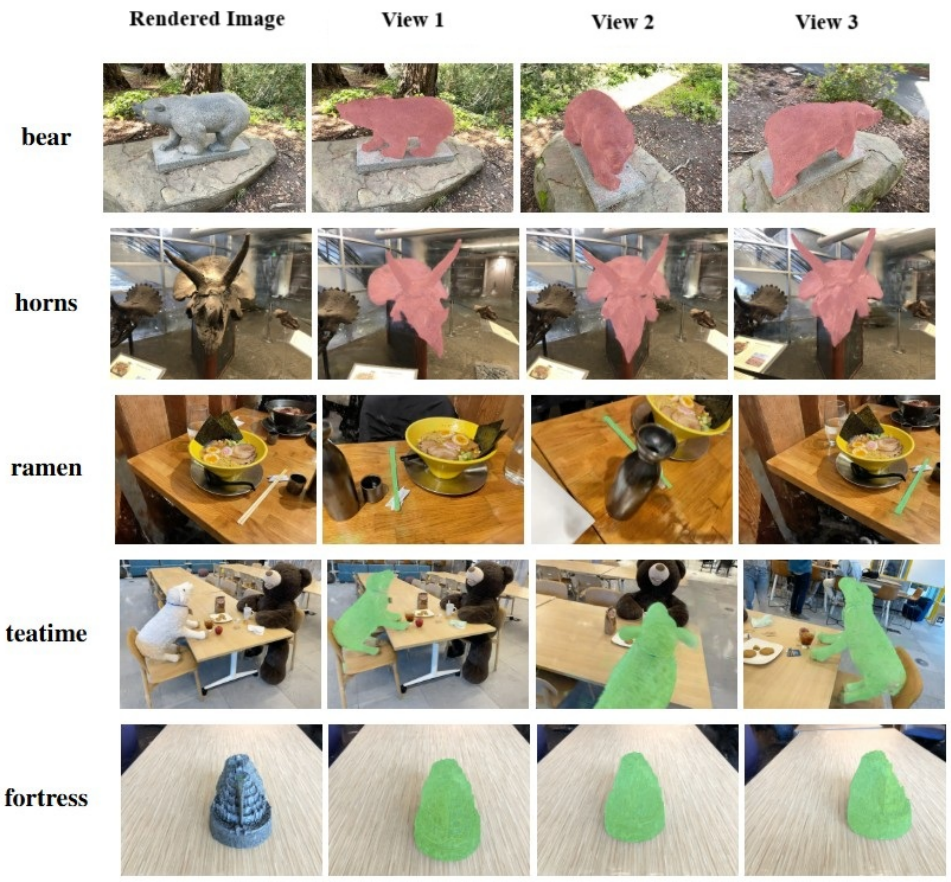

Figure 6.5: Qualitative results for recolorization task across multiple viewpoints of the given object within the scene.

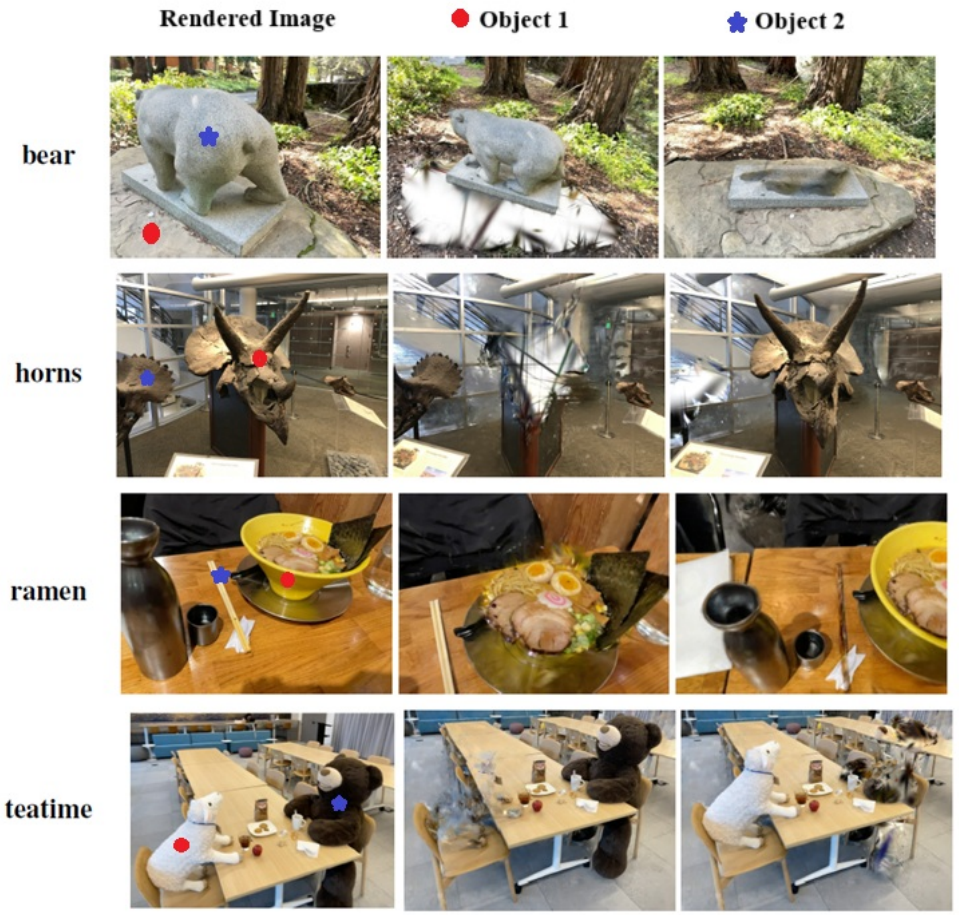

Figure 6.6: Qualitative results for the object removal task. From left to right: RGB rendered image, rendered view with the first object (red) removed, and rendered view with a second object (blue) removed. The figure shows the selected objects are successfully removed from the 3D Scene.

6.5.4 Object Removal Task

We further demonstrate the capability of our approach for interactive object removal within 3D Gaussian Splatting scenes. In this task, the target object is identified from the user-selected 3D Point, and Gaussians for the selected object is subsequently removed from the 3D scene representation. The updated 3D scene is then rendered from a chosen camera viewpoint, resulting in a new rendered image without the selected object. The qualitative results shown in Figure 6.6, clearly indicate that our method successfully eliminates the entire set of Gaussians corresponding to the selected object, thus consistently removing it from all views. For better visualizations, Figure 6.7 presents qualitative results of object removal from multiple viewpoints performed on different scenes. For each scene, the target object is removed from the reconstructed 3D Gaussian representation, and the rendered outputs from multiple viewpoints are shown. This consistency emphasizes

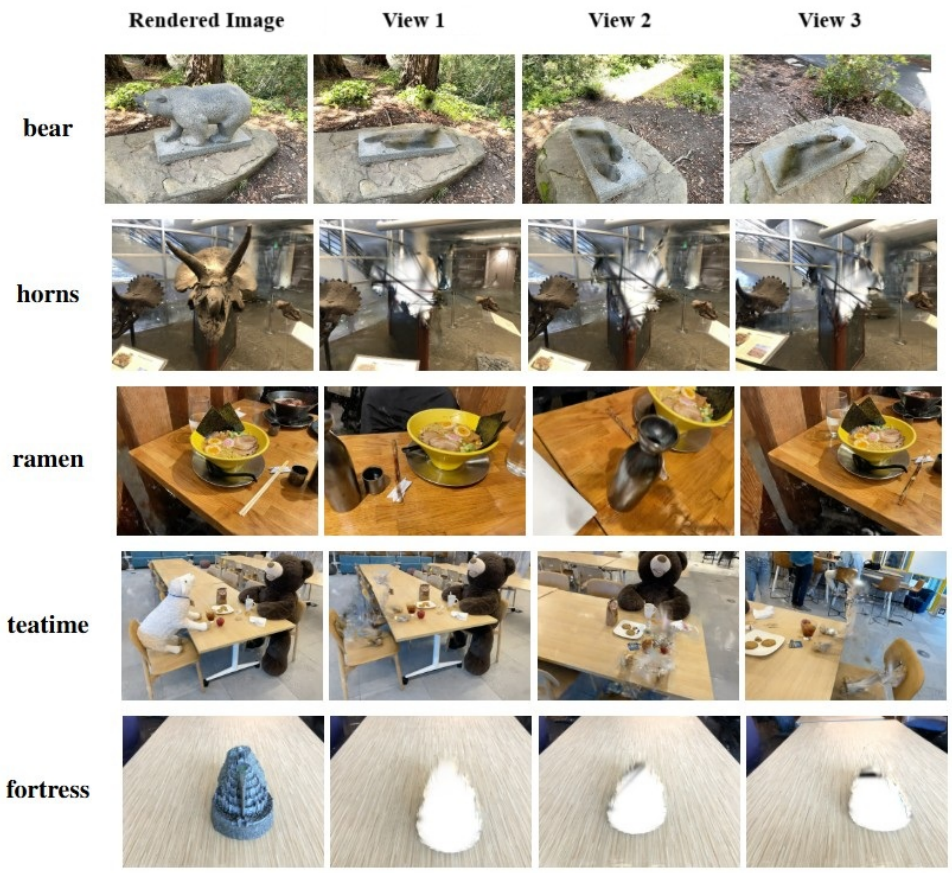

Figure 6.7: Qualitative results for object removal task across multiple viewpoints of the given object within the scene.

that the Gaussians in the 3D scene have successfully learned object-level understanding, enabling robust and seamless object removal task.

However, we observe that the removal process introduces empty voids in the 3D space previously occupied by the selected object, which appear as unnatural gaps in the rendered images. In addition, minor boundary artifacts such as small spikes occasionally emerged around the boundaries of the removed object. This happens because individual Gaussians may contribute to multiple objects depending on the viewpoint as mentioned in Figure 5.3. These limitations are expected in direct object removal pipelines. Despite these challenges, the results demonstrate the feasibility of per-Gaussian editing for interactive scene manipulation. The void regions could be plausibly filled using generative inpainting techniques [55] or Gaussian primitives replacement [53] with another object, which we identify as a promising future direction beyond the scope of this dissertation.

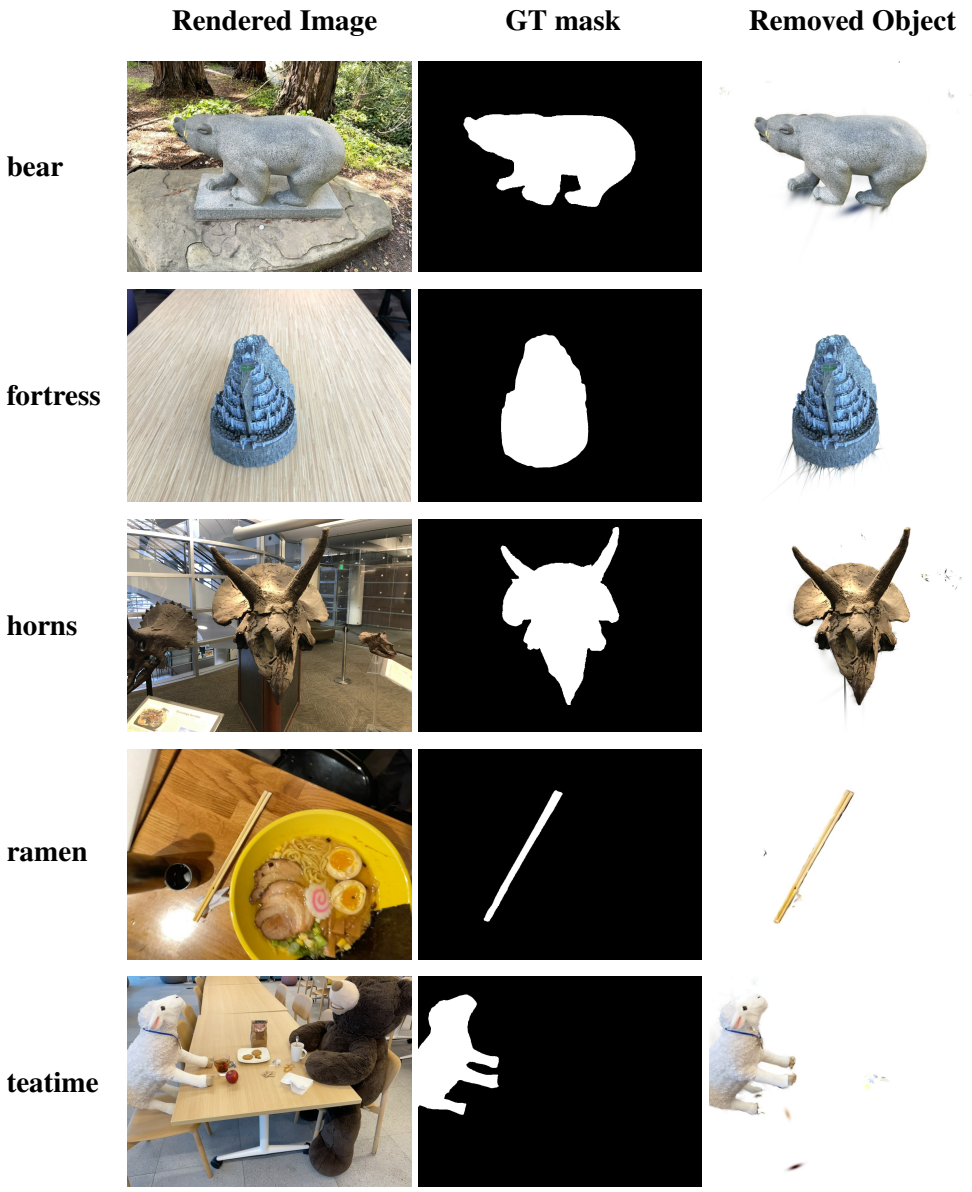

Figure 6.8: Qualitative visualization of 3D object extraction. Given a ground-truth mask in a single view (middle), our method produces a segmented 3D object (right) that can be rendered from novel viewpoints.

In addition, we visualize the extracted 3D object when provided with a ground-truth mask in a single input view. Figure 6.8 shows the segmented object obtained by propagating the mask supervision through the 3D Gaussian Splatting representation. The results clearly illustrate how our method can isolate the full 3D geometry of the selected object, rather than limiting segmentation. To provide better visualization of these results, we have included our project link in Appendix 7.2.

These qualitative examples highlight two key aspects. First, the extracted 3D object closely aligns with the ground-truth mask from the reference view while maintaining multiview consistency,

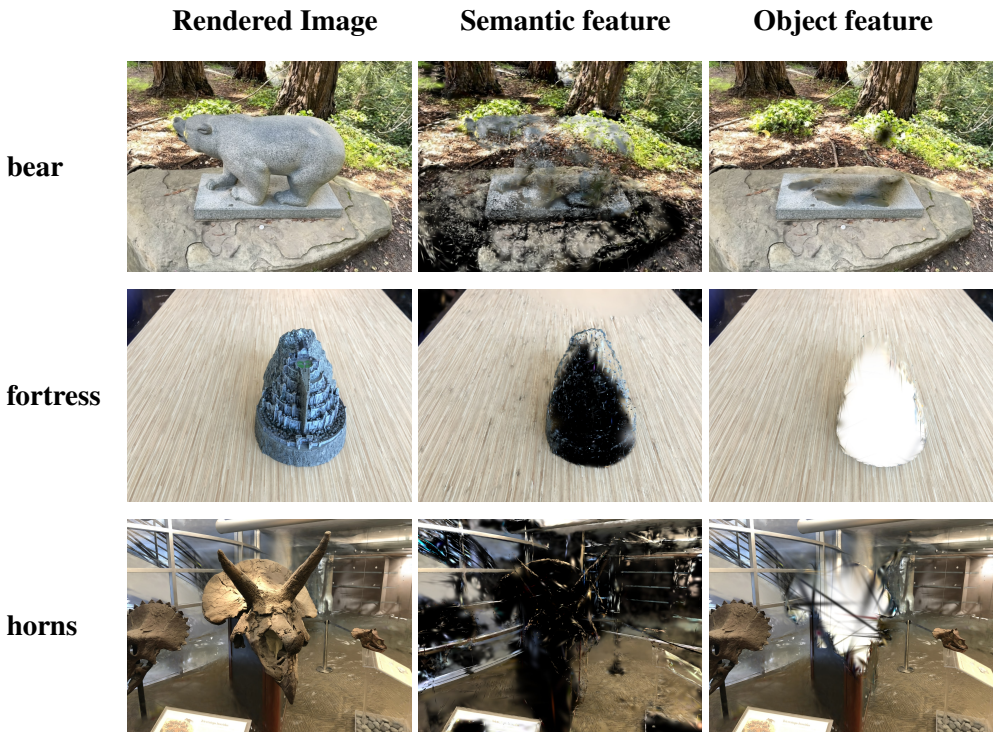

Figure 6.9: Qualitative comparison of object removal between the semantic feature–based method and the proposed object feature with prior-guided label reassignment method on bear, fortress, and horns scenes.

demonstrating that the label reassignment and preprocessing pipeline effectively transfer reliable supervision into the 3D domain. Second, the segmented 3D object can be directly rendered from arbitrary viewpoints, opening up a wide range of applications. These include object-level editing tasks such as recolorization, removal, or replacement, as well as higher-level downstream uses such as instance-level scene composition, and synthetic 3D dataset generation.

6.5.5 Semantic and Object feature method comparison

The qualitative results shown in Figure 6.9 illustrate a comparative evaluation of the object removal task in two approaches, one based on semantic feature only and the other on object feature with label reassignment strategy. The semantic feature–based method demonstrates the ability to remove the user-selected object from the 3D scene, but the quality of object removal remains average. Residual Gaussians often remain within the target region, and the extracted boundaries are very irregular and noisy, leading to incomplete removal. In contrast, the object feature with the label reassignment method achieves significantly superior performance. It ensures that the selected object is removed completely, producing clean boundaries with negligible noise, thereby aligning with the true objective of the task. Furthermore, the semantic feature–based method exhibits inher-

Table 6.7: Ablation results for the "bear" scene across different feature dimensions. (OOM = Out of Memory)

Dimension	**8**	**16**	**32**	**64**	**128**	**256**
Autoencoder Loss	0.474	0.387	0.325	0.274	0.150	0.0
PSNR (7k)	25.93	**26.12**	26.10	26.11	26.08	OOM
SSIM (7k)	0.887	**0.894**	0.889	0.893	0.887	OOM
LPIPS (7k)	0.177	**0.176**	0.177	0.177	0.178	OOM

Table 6.8: Ablation results for the "ramen" scene across different feature dimensions.

Dimension	**8**	**16**	**32**	**64**	**128**	**256**
Autoencoder Loss	0.413	0.316	0.257	0.195	0.195	0.0
PSNR (7k)	26.22	**26.23**	26.13	26.17	26.13	26.11
SSIM (7k)	0.894	**0.894**	0.891	0.893	0.892	0.892
LPIPS (7k)	0.178	**0.178**	0.179	0.178	0.178	0.178

ent limitations in handling task where multiple objects with similar semantics are present across the scene. For instance, in the horns scene example, the method fails to isolate the user-selected central horn, erroneously removing an additional horn due to feature overlap. However, the object label reassignment technique, on the other hand, consistently maintains object instance specificity across all tested cases. These results highlight the robustness and precision of our proposed object fetaure-based approach for high-quality scene editing.

6.6 Ablation Studies

6.6.1 Study on varying dimension of semantic features

We conducted an ablation study on the semantic feature dimension $d \in \{8, 16, 32, 64, 128, 256\}$ for five representative scenes such as, *bear*, *fortress*, *horns*, *ramen*, *teatime*. We evaluated three primary axes of behavior starting with (1) reconstruction quality measured by PSNR at 7k and 30k iterations, (2) autoencoder reconstruction loss, and (3) computational cost in training time and memory. We observed several consistent patterns that highlight the trade-offs between reconstruction quality, autoencoder compression effects, and computational cost. As expected, autoencoder loss decreased steadily with increasing d, indicating that higher-dimensional embeddings can more correctly reconstruct the original SAM-HQ features as presented in Table 6.7 and 6.8. However, this improvement in high-dimensional feature quality did not translate into higher photometric reconstruction quality, as PSNR values at 7k iterations remained essentially flat or even degraded slightly as feature dimension increased. For instance, in the *bear* scene comprising more than 2 million Gaussians, PSNR and SSIM peaked around $d = 16$ and declined at $d \geq 32$, a trend

Table 6.9: Training time for 30k iterations across different feature dimensions (d).(OOM = Out of Memory)

Scene	**3D-GS training time (h=hour, m=minute)**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	4h 06m	4h 39m	5h 04m	5h 41m	OOM	OOM
fortress	2h 29m	2h 34m	2h 41m	3h 06m	OOM	OOM
horns	2h 07m	2h 26m	2h 51m	3h 45m	4h 28m	OOM
ramen	1h 43m	1h 57m	2h 39m	3h 27m	4h 13m	OOM
teatime	2h 47m	3h 07m	3h 41m	4h 27m	OOM	OOM

observed across other scene like *ramen* which consist of less than 1 million Gaussian primitives. This suggests that beyond a modest embedding size, additional dimensions add redundancy or noise rather than useful signal for scene reconstruction. In stark contrast, the computational costs, both training time and memory usage, rose sharply with dimension, often causing out-of-memory ($\geq$16GB) errors for 30k iterations when $d \geq 128$, as presented in Table 6.9.

Notably, while small and simple scenes tolerated larger dimensions with little PSNR and SSIM variation, larger and more cluttered scenes such as *bear* and *teatime* showed greater sensitivity, reinforcing that scene complexity exacerbates the inefficiency of high-dimensional embeddings. These findings imply that compact embeddings ($d = 16$ or $d = 32$) provide the best balance, with $d = 16$ requiring less memory usage and training time. In summary, increasing feature dimensionality improves semantic information but has little to very less effect on rendered quality, instead inflating time and memory requirements. Please see Appendix 7.2 for additional details.

6.6.2 Study on learned prior influence during label reassignment

To investigate the impact of the learned label prior weight, γ_p, on label reassignment framework, we conducted a qualitative ablation study on the scene *bear*, focusing on the object removal task. We varied γ_p across values of -0.9, -0.5, 0.2 (default), 0.5, and 1.0, and evaluated the quality of both the remaining scene and the removed object. The results, visualized in the rendered images as shown in Figure 6.10, reveal different trends. In the negative domain, a moderately negative $\gamma_p = -0.5$ produces a smoother removed object by suppressing erroneous Gaussian assignments around the boundaries. However, further decreasing to $\gamma_p = -0.9$ degrades performance, as some Gaussians persist in the remaining scene, introducing artifacts in both the remaining scene and the removed object. Conversely, in the positive domain, $\gamma_p = 0.5$ yields a smoother remaining scene by leveraging confident priors, but increasing to $\gamma_p = 1.0$ introduces multiple artifacts in the removed object due to outlier Gaussians being incorrectly included. The default $\gamma_p = 0.2$

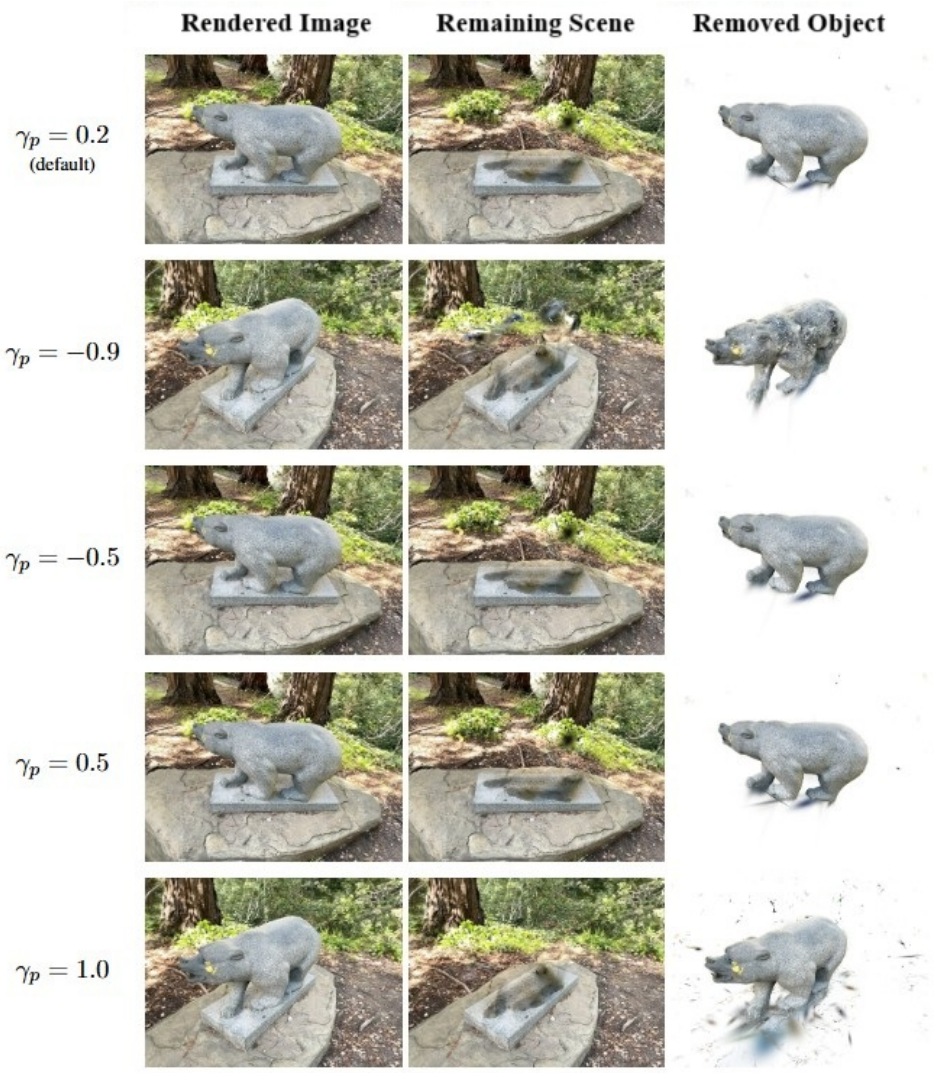

Figure 6.10: **Qualitative Results:** Effect of varying γ_p which controls the contribution of prior learned labels during label reassignment for scene *bear*.

strikes an optimal balance, achieving high-quality segmentation with minimal artifacts in both the remaining scene and the removed object. Therefore, our study demonstrates the importance of carefully tuning the prior weight γ_p to balance prior influence in our 3D Gaussian Splatting segmentation framework.

7 CONCLUSIONS

Our dissertation work aimed to enable precise object removal and recolorization tasks in static 3D Gaussian Splatting (3D-GS) scenes, ensuring semantic consistency and visual coherence across multiple views. Experimental results across methodological stages demonstrate that robust preprocessing, semantic feature integration, and object-aware training are critical for achieving our goal.

In Chapter 3, we created preprocessing pipelines to support object-level editing in 3D Gaussian Splatting (3D-GS) by generating high-quality object masks. For semantic feature extraction, we utilized SAM-HQ[31] to produce robust features, which were embedded into 3D-GS to create semantically aware scenes. An autoencoder-based compression strategy preserved discriminative information while reducing computational overhead and enhancing scalability for complex scenes. Notably, results show that increasing feature dimensionality did not consistently improve reconstruction or editing quality. Separately, for object mask generation, we employed SAM-HQ[31] with DEVA [8] for view-consistent segmentation via bidirectional temporal propagation, ensuring robust multiview masks with fine boundaries. A noise removal preprocessing module further enhanced mask quality by eliminating artifacts and repairing broken boundaries, as evidenced by improved reconstruction metrics like PSNR and SSIM metrics values. Further, this pipeline created a high-quality mask dataset, forming a reliable foundation for accurate and efficient object-aware 3D-GS editing tasks.

In Chapter 4, we established the deliberate choice of 3D Gaussian Splatting (3D-GS) as the foundational 3D scene representation for this dissertation, owing to its unique suitability for interactive 3D scene editing tasks. Unlike mesh or voxel-based approaches, 3D-GS offers a continuous and explicit point-based representation in which each Gaussian is parameterized by position, scale, opacity, and appearance attributes. Its differentiable and efficient rasterization process enables high-quality novel view synthesis while maintaining real-time rendering performance, which is essential for interactive applications. More importantly, this rasterization pipeline extends naturally beyond photometric rendering, allowing Gaussians to encode additional attributes, such as semantic or object-level features. This extensibility is central to the dissertation's contributions, as it enabled both semantic feature-based training and object mask-driven training, thus supporting

object-aware operations such as removal, recolorization, and potentially more advanced editing tasks. To demonstrate this, we integrated semantic features into 3D-GS training by embedding Gaussians with semantic attributes alongside their photometric and geometric properties. This enriched representation improved scene interpretability, particularly in textureless or cluttered regions, without compromising reconstruction fidelity. Experimental results confirmed that semantic feature-based training enhanced interpretability while preserving overall reconstruction quality. However, while this method proved effective in distinguishing semantically similar regions, it exhibited limitations in boundary precision. Specifically, Gaussians near object edges were not sharply separated, which hindered fine-grained editing tasks such as object removal. These insights motivated the exploration of object feature-based training in subsequent chapters to address the boundary consistency challenge and improve object-specific editing performance.

In Chapter 5, we presented a targeted integration of object mask supervision into the 3D Gaussian Splatting (3D-GS) framework, offering a more interpretable and controllable alternative to the semantic feature-based approach introduced in Chapter 4. This object feature-based method was designed to leverage the high-quality multiview SAM-HQ [22] object masks curated in Chapter 3, thereby ensuring that object-aware 3D scenes could be reconstructed with both geometric fidelity and consistent object-level semantics. Central to this approach was the introduction of a 16-dimensional object feature vector assigned to each Gaussian, a view-invariant representation that enabled stable multiview identification of objects across diverse perspectives. By balancing compactness with representational power, this embedding facilitated efficient learning while maintaining scalability to scenes with many distinct objects. To incorporate this object supervision into the 3D-GS pipeline, we extended the rasterization process to jointly render both RGB appearance and object features, projecting them into a higher-dimensional probability space via a linear mapping. The resulting logits were normalized with softmax and optimized using a cross-entropy objective against SAM-HQ object masks, in combination with the standard photometric loss, thereby encouraging each Gaussian to learn both its visual and categorical identity. A major contribution of this chapter was the novel approach of a prior-based object label reassignment strategy, which addressed the inherent ambiguity of Gaussians contributing to multiple objects across views due to occlusions and overlaps. By constructing a contribution matrix that quantified Gaussian-to-object associations and regularizing it with learned label priors, the method extended simple majority voting into a principled probabilistic assignment, avoiding ad hoc heuristics like background bias. This innovation markedly improved robustness and ensured consistent label as-

signments across multiple views. During inference, the framework enabled promptable editing by allowing users to select a 3D point, retrieve its nearest Gaussians, and assign an initial object label that was subsequently refined through prior-based reassignment. This process produced a set of Gaussians that accurately corresponded to the user's chosen object, which could then be manipulated through deletion of Gaussians for object removal or RGB value adjustment of Gaussians for recolorization task. Our label reassignment approach circumvents the heuristic-based background bias and produces robust prior-guided method with perfomance matching state-of-the-art (SOTA) results on standard NVOS [43] dataset. Further, qualitative results showed superior performance over semantic feature-based training in object removal and recolorization tasks, with sharper boundaries and fewer artifacts, particularly in challenging scenes such as *horns* scene.

Collectively, this dissertation presents a comprehensive 3D Gaussian Splatting framework where preprocessing ensures high-quality inputs, semantic training enhances scene understanding, and object mask supervision enables precise editing. These contributions align with the goal of enabling interactive, object-aware 3D editing especially object removal and recolorization, demonstrating that combining robust mask generation and prior-guided label reassignment is essential for advancing 3D scene manipulation.

7.1 Evaluation

Here, we have evaluated the outcomes of the dissertation against the objectives outlined in Section 1.2, examining how each goal is addressed and the extent to which it is achieved.

The first objective was a comprehensive investigation of existing methods for 3D scene reconstruction and editing to identify a suitable backbone for 3D scene editing. We addressed this systematically by surveying both traditional point clouds, meshes, and voxels, as well as recent neural scene representations, including NeRF [32] and 3D Gaussian Splatting. By analyzing their strengths and weaknesses, we justified our choice of 3D-GS over others like NeRFs beacuse 3D-GS provides an explicit, point-based representation that allows efficient, tile-based differentiable rasterization and real-time rendering, while avoiding heavy computation-based rendering in NeRFs. More importantly, the rasterization pipeline extends naturally beyond photometric rendering, allowing Gaussians to encode additional attributes, such as semantic or object-level features. The decision was validated through its interactive real-time performance, making it a practical foundation for subsequent semantic and object-level extensions.

The second objective was to design and implement a method to integrate semantic features into Gaussian splats, enabling semantically enriched 3D representations. We met this objective by extracting high-quality 2D semantic features with SAM-HQ [22], compressing them with a scene-specific autoencoder, and embedding the resulting low-dimensional features into each Gaussian. The joint optimization of photometric and feature losses produced a semantically informative 3D representation that improved interpretability in ambiguous or textureless regions while maintaining reconstruction quality. The PSNR and LPIPS remained comparable to baseline 3D-GS training and qualitative inspections showed clear semantic object removal task. Importantly, ablation studies on feature dimension revealed a practical 16-dimension feature that preserves semantic fidelity with acceptable training time and memory.

The third objective focused on generating high-quality, noise-free, and multiview-consistent segmentation masks to serve as supervision for object-level training. We selected SAM-HQ [22] for its superior performance in producing high-quality and robust segmentation masks across diverse scenes. We created multiview- consistent object mask by combining SAM-HQ segmentation with DEVA [8] object tracking for temporal propagation and a targeted preprocessing pipeline for the removal of small artifacts via connected component labeling. The resulting High-Quality Mask Dataset substantially reduced boundary noise and fragmented segments relative to baseline SAM [27] outputs, producing modest gains in PSNR and SSIM metrics in reconstruction experiments. These improvements were directly supported by quantitative results and qualitative visualizations through cleaner masks translated into more stable and reliable supervision for both semantic and object-feature training.

The fourth objective aimed to implement a robust prior-guided label reassignment method for Gaussians in 3D-GS to produce consistent per-Gaussian object labels. Our prior-guided reassignment approach successfully met this goal by combining multiview label contribution from simple majority voting with learned per-Gaussian label priors. This approach circumvents the need of heuristic background bias. Experimental results show that this approach yields substantial gains in label consistency and robustness with pixel accuracy remained near 98% on new high-quality object mask dataset and mean intersection-over-union (mIoU) matching state-of-the-art method on NVOS [43] dataset. The results demonstrates that learned priors effectively correct the failure labels of pure count-based voting and eliminate the need for heuristic background biases used in prior work.

The fifth objective concerned designing and evaluating interactive object removal and recolorization enabled by user-specified 3D point selection. Our pipeline enables promptable editing for both semantic feature-based and object feature-based methods by accepting user-selected 3D points as input, performing nearest-Gaussian retrieval, refining labels, and directly manipulating Gaussian parameters. Qualitative and quantitative evaluations indicate that the object-feature training combined with prior-guided reassignment provides the most reliable editing where object removal and recolorization tasks produce minimal artificats. At the same time, we documented trade-offs about how joint object feature training increases model size and peak memory.

Overall, this dissertation has been carried out in a systematic and professional manner, guided by well-defined objectives and supported by transparent reporting of both strengths and limitations. Through this structured approach, the work delivers a cohesive and impactful contribution to the field of interactive, object-aware 3D scene editing, with particular emphasis on advancing the tasks of object removal and recolorization.

7.2 Future Work

Although our dissertation work has made substantial progress in advancing robust interactive object-level editing within the 3D Gaussian Splatting (3D-GS) framework, several promising areas remain for future exploration that could further enhance the quality, flexibility, and applicability of the proposed methods. While our framework successfully removes targeted objects, the resulting voids often reduce visual plausibility. To address this, future work could investigate the integration of generative inpainting strategies, particularly diffusion-based methods, which have recently shown remarkable success in generating high-quality, context-aware content in both image and video domains. The challenge in extending such approaches to 3D-GS lies in moving beyond 2D image completion and developing mechanisms to generate entirely new Gaussian primitives with appropriate positions, scales, opacities, and radiance properties that seamlessly match the geometry and texture of the given scene. By leveraging multi-view consistency inherent in 3D-GS and conditioning generative models on rendered void masks from multiple viewpoints, it may be possible to synthesize new Gaussians that plausibly fill the missing regions, leading to visually coherent edits with minimal artifacts. Such an approach would transform object removal from a purely subtractive operation into a reconstructive process that preserves both structural and photometric continuity.

Beyond object removal, a natural extension of this work lies in object replacement, where instead of leaving voids or filling them generatively, the framework could seamlessly insert new objects into the scene. The foundations for this task are already established in the current framework, which provides robust mechanisms for selecting, labeling, and removing Gaussian clusters corresponding to target objects. Building on this, future research could develop methods for introducing new Gaussian splats representing entirely different objects, acquired from external 3D-GS reconstructions. Critical challenges here would include estimating the appropriate pose, orientation, and scale of the inserted object relative to the scene, ensuring semantic and contextual alignment. This capability would significantly expand the utility of the framework for applications in 3D content creation, simulation, and virtual environment design.

Another important direction concerns the extension of the current framework from static to dynamic 3D scenes. Although our dissertation work achieves strong multiview consistency for static scenes, real-world environments are often dynamic, with objects moving, deforming, or interacting over time. Supporting editing in such scenarios requires not only spatial coherence but also temporal consistency. Building upon our multiview high-quality mask generation pipeline, future work could extend temporal mask propagation to full spatiotemporal tracking of Gaussian-based object features. This would involve augmenting Gaussian parameters with temporal dynamics, allowing their attributes, such as position, orientation, scale, and object embeddings, to evolve smoothly over time. Moreover, techniques such as flow matching could be integrated to model scene motion and predict the trajectories of Gaussians across frames, enabling accurate tracking of both rigid and deformable objects. Such a framework would allow not only static object edits but also dynamic scene editing, such as removing a moving object, recoloring it across an animation sequence, or even replacing it with a temporally consistent alternative, thereby broadening the scope of interactive 3D scene editing toward real-time applications in dynamic virtual and augmented reality.

Finally, our proposed framework opens promising avenues for advancing high-quality scene editing that go beyond the current focus on object removal and recolorization. Future research could investigate more sophisticated operations such as object deformation, texture stylization, and generative inpainting, applied within both static and dynamic 3D scene environments. Pursuing these directions would enable the framework introduced in this dissertation to evolve from a foundation for object-aware static scene editing into a more comprehensive paradigm for dynamic, intelli-

gent, and interactive 3D scene manipulation. Such advancements have the potential to establish new possibilities for applications across computer vision, computer graphics, and immersive technologies, significantly broadening the impact and utility of 3D Gaussian Splatting scene editing in both research and practical domains.

BIBLIOGRAPHY

[1] Barron, J. T. , Mildenhall, B. , Verbin, D. , Srinivasan, P. P. , and Hedman, P. . Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, June 2022.

[2] Carmigniani, J. , Furht, B. , Anisetti, M. , Ceravolo, P. , Damiani, E. , and Ivkovic, M. . Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51(1):341–377, 2011.

[3] Caron, M. , Touvron, H. , Misra, I. , Jégou, H. , Mairal, J. , Bojanowski, P. , and Joulin, A. . Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, October 2021.

[4] Cen, J. , Fang, J. , Yang, C. , Xie, L. , Zhang, X. , Shen, W. , and Tian, Q. . Segment any 3d gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 1971–1979. AAAI Press, 2025. doi: 10.1609/aaai.v39i2.32193.

[5] Cen, J. , Fang, J. , Zhou, Z. , et al. Segment anything in 3d with radiance fields. *International Journal of Computer Vision*, 133:5138–5160, 2025. doi: 10.1007/s11263-025-02421-7.

[6] Chen, M.-J. and Bovik, A. C. . Fast structural similarity index algorithm. *Journal of Real-Time Image Processing*, 6(4):281–287, 2011.

[7] Cheng, H. K. and Schwing, A. G. . Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In Avidan, S. , Brostow, G. , Cissé, M. , Farinella, G. M. , and Hassner, T. , editors, *Computer Vision – ECCV 2022*, volume 13688 of *Lecture Notes in Computer Science*, pages 640–658. Springer, Cham, 2022. doi: 10.1007/978-3-031-19815-1_37.

[8] Cheng, H. K. , Oh, S. W. , Price, B. , Schwing, A. , and Lee, J.-Y. . Tracking anything with decoupled video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1316–1326, October 2023.

[9] Choi, S. , Song, H. , Kim, J. , Kim, T. , and Do, H. . Click-gaussian: Interactive segmentation to any 3d gaussians. In Leonardis, A. , Ricci, E. , Roth, S. , Russakovsky, O. , Sattler, T. , and

Varol, G. , editors, *Computer Vision – ECCV 2024*, volume 15061 of *Lecture Notes in Computer Science*, pages 277–294. Springer, Cham, 2025. doi: 10.1007/978-3-031-72646-0_17.

[10] Chou, Z.-T. , Huang, S.-Y. , Liu, I.-J. , and Wang, Y.-C. F. . Gsnerf: Generalizable semantic neural radiance fields with enhanced 3d scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20806–20815, June 2024.

[11] Dosovitskiy, A. , Beyer, L. , Kolesnikov, A. , Weissenborn, D. , Zhai, X. , Unterthiner, T. , Dehghani, M. , Minderer, M. , Heigold, G. , Gelly, S. , Uszkoreit, J. , and Houlsby, N. . An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. URL `https://api.semanticscholar.org/CorpusID:225039882`.

[12] Fei, B. , Xu, J. , Zhang, R. , Zhou, Q. , Yang, W. , and He, Y. . 3d gaussian splatting as a new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 31(8): 4429–4449, 2025. doi: 10.1109/TVCG.2024.3397828.

[13] Goel, R. , Sirikonda, D. , Saini, S. , and Narayanan, P. J. . Interactive segmentation of radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4201–4211, June 2023.

[14] Gonzalez, R. C. and Woods, R. E. . *Digital Image Processing*. Prentice Hall, 3 edition, 2008.

[15] Guo, J. , Ma, X. , Fan, Y. , Liu, H. , and Li, Q. . Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting. *ArXiv*, abs/2403.15624, 2024. URL `https://api.semanticscholar.org/CorpusID:268680548`.

[16] Guo, M.-H. , Lu, C.-Z. , Hou, Q. , Liu, Z. , Cheng, M.-M. , and Hu, S.-M. . Segnext: Rethinking convolutional attention design for semantic segmentation. *Advances in neural information processing systems*, 35:1140–1156, 2022.

[17] Han, J. , Cao, Y. , Xu, L. , Liang, W. , Bo, Q. , Wang, J. , Wang, C. , Kou, Q. , Liu, Z. , and Cheng, D. . 3d reconstruction method based on medical image feature point matching. *Computational and Mathematical Methods in Medicine*, 2022:9052751, 2022. doi: 10.1155/2022/9052751.

[18] He, L. , Ren, X. , Gao, Q. , Zhao, X. , Yao, B. , and Chao, Y. . The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43, 2017. ISSN 0031-3203. doi: 10.1016/j.patcog.2017.04.018. URL https://www.sciencedirect.com/science/article/pii/S0031320317301693.

[19] He, Y. , Yu, H. , Liu, X.-Y. , Yang, Z. , Sun, W. , Wang, Y. , Fu, Q. , Zou, Y. , and Mian, A. S. . Deep learning based 3d segmentation: A survey. *ArXiv*, abs/2103.05423, 2021. URL https://api.semanticscholar.org/CorpusID:232168831.

[20] Heaton, J. . Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618. *Genetic Programming and Evolvable Machines*, 19, 10 2017. doi: 10.1007/s10710-017-9314-z.

[21] Hui, J. . Approach to the interior design using augmented reality technology. In *2015 Sixth International Conference on Intelligent Systems Design and Engineering Applications (ISDEA)*, pages 163–166. IEEE, 2015.

[22] Ke, L. , Ye, M. , Danelljan, M. , Tai, Y.-W. , Tang, C.-K. , Yu, F. , et al. Segment anything in high quality. *Advances in Neural Information Processing Systems*, 36:29914–29934, 2023.

[23] Kerbl, B. , Kopanas, G. , Leimkühler, T. , and Drettakis, G. . 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[24] Kerr, J. , Kim, C. M. , Goldberg, K. , Kanazawa, A. , and Tancik, M. . Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19729–19739, October 2023.

[25] Khan, S. , Naseer, M. , Hayat, M. , Zamir, S. W. , Khan, F. S. , and Shah, M. . Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[26] Kingma, D. P. and Ba, J. . Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

[27] Kirillov, A. , Mintun, E. , Ravi, N. , Mao, H. , Rolland, C. , Gustafson, L. , Xiao, T. , Whitehead, S. , Berg, A. C. , Lo, W.-Y. , Dollar, P. , and Girshick, R. . Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[28] Kobayashi, S. , Matsumoto, E. , and Sitzmann, V. . Decomposing nerf for editing via feature field distillation. *Advances in neural information processing systems*, 35:23311–23330, 2022.

[29] Lhuillier, M. . Improving thin structures in surface reconstruction from sparse point cloud. In Leal-Taixé, L. and Roth, S. , editors, *Computer Vision – ECCV 2018 Workshops*, volume 11129 of *Lecture Notes in Computer Science*, pages 441–457. Springer, Cham, 2019. doi: 10.1007/978-3-030-11009-3_27.

[30] Li, B. , Weinberger, K. Q. , Belongie, S. J. , Koltun, V. , and Ranftl, R. . Language-driven semantic segmentation. *ArXiv*, abs/2201.03546, 2022. URL https://api.semanticscholar.org/CorpusID:245836975.

[31] Liu, Y. , Hu, B. , Tang, C.-K. , and Tai, Y.-W. . Sanerf-hq: Segment anything for nerf in high quality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3216–3226, June 2024.

[32] Mildenhall, B. , Srinivasan, P. P. , Tancik, M. , Barron, J. T. , Ramamoorthi, R. , and Ng, R. . Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[33] Müller, T. , Evans, A. , Schied, C. , and Keller, A. . Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. ISSN 1557-7368. doi: 10.1145/3528223.3530127. URL http://dx.doi.org/10.1145/3528223.3530127.

[34] Ngo, T. D. , Hua, B.-S. , and Nguyen, K. . Isbnet: A 3d point cloud instance segmentation network with instance-aware sampling and box-aware dynamic convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13550–13559, June 2023.

[35] Yang, Y. , nuo, Huang, Y. , Guo, Y. , Lu, L. , Wu, X. , Lam, E. Y. , Cao, Y.-P. , and Liu, X. . Sampart3d: Segment any part in 3d objects. *ArXiv*, abs/2411.07184, 2024. URL https://api.semanticscholar.org/CorpusID:273963848.

[36] Pan, L. , Baráth, D. , Pollefeys, M. , and Schönberger, J. L. . Global structure-from-motion revisited. In Leonardis, A. , Ricci, E. , Roth, S. , Russakovsky, O. , Sattler, T. , and Varol,

G. , editors, *Computer Vision – ECCV 2024*, volume 15098 of *Lecture Notes in Computer Science*, pages 60–78. Springer, Cham, 2025. doi: 10.1007/978-3-031-73661-2_4.

[37] Qin, M. , Li, W. , Zhou, J. , Wang, H. , and Pfister, H. . Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20051–20060, June 2024.

[38] Qiu, S. , Xie, B. , Liu, Q. , and Heng, P.-A. . Advancing extended reality with 3d gaussian splatting: Innovations and prospects. In *2025 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 203–208. IEEE, 2025.

[39] 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`.

[40] 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*, pages 8748–8763. PmLR, 2021.

[41] Rajič, F. , Ke, L. , Tai, Y.-W. , Tang, C.-K. , Danelljan, M. , and Yu, F. . Segment anything meets point tracking. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 9302–9311. IEEE, 2025.

[42] Reiser, C. , Peng, S. , Liao, Y. , and Geiger, A. . Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14335–14345, October 2021.

[43] Ren, Z. , Agarwala, A. , Russell, B. , Schwing, A. G. , and Wang, O. . Neural volumetric object selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6133–6142, June 2022.

[44] Rong, F. , Xie, D. , Zhu, W. , Shang, H. , and Song, L. . A survey of multi view stereo. In *2021 International Conference on Networking Systems of AI (INSAI)*, pages 129–135, 2021. doi: 10.1109/INSAI54028.2021.00033.

[45] Ronneberger, O. , Fischer, P. , and Brox, T. . U-net: Convolutional networks for biomedical image segmentation. In Navab, N. , Hornegger, J. , Wells, W. , and Frangi, A. , editors,

Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, Cham, 2015. doi: 10.1007/978-3-319-24574-4_28.

[46] Schönberger, J. L. and Frahm, J.-M. . Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. doi: 10.1109/CVPR.2016.445.

[47] Shen, Q. , Yang, X. , and Wang, X. . Flashsplat: 2d to 3d gaussian splatting segmentation solved optimally. In Leonardis, A. , Ricci, E. , Roth, S. , Russakovsky, O. , Sattler, T. , and Varol, G. , editors, *Computer Vision – ECCV 2024*, volume 15080 of *Lecture Notes in Computer Science*, pages 421–439. Springer, Cham, 2025. doi: 10.1007/978-3-031-72670-5_26.

[48] Snell, J. , Ridgeway, K. , Liao, R. , Roads, B. D. , Mozer, M. C. , and Zemel, R. S. . Learning to generate images with perceptual similarity metrics. In *2017 IEEE international conference on image processing (ICIP)*, pages 4277–4281. IEEE, 2017.

[49] Tancik, M. , Weber, E. , Ng, E. , Li, R. , Yi, B. , Wang, T. , Kristoffersen, A. , Austin, J. , Salahi, K. , Ahuja, A. , et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–12, 2023.

[50] Tian, Z. , Shen, C. , and Chen, H. . Conditional convolutions for instance segmentation. In Vedaldi, A. , Bischof, H. , Brox, T. , and Frahm, J.-M. , editors, *Computer Vision – ECCV 2020*, volume 12346 of *Lecture Notes in Computer Science*, pages 282–298. Springer, Cham, 2020. doi: 10.1007/978-3-030-58452-8_17.

[51] Tschernezki, V. , Laina, I. , Larlus, D. , and Vedaldi, A. . Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453. IEEE, 2022.

[52] Tucker, R. and Snavely, N. . Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[53] Violante, N. , Meuleman, A. , Gauthier, A. , Durand, F. , Groueix, T. , and Drettakis, G. . Splat and replace: 3d reconstruction with repetitive elements. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12, 2025.

[54] Wang, Z. , Wang, E. , and Zhu, Y. . Image segmentation evaluation: A survey of methods. *Artificial Intelligence Review*, 53:5637–5674, 2020. doi: 10.1007/s10462-020-09830-9.

[55] Yan, Z. , Li, L. , Shao, Y. , Chen, S. , Wu, Z. , Hwang, J.-N. , Zhao, H. , and Remondino, F. . 3dsceneeditor: Controllable 3d scene editing with gaussian splatting. *ArXiv*, abs/2412.01583, 2024. URL https://api.semanticscholar.org/CorpusID:274436310.

[56] Ye, M. , Danelljan, M. , Yu, F. , and Ke, L. . Gaussian grouping: Segment and edit anything in 3d scenes. In Leonardis, A. , Ricci, E. , Roth, S. , Russakovsky, O. , Sattler, T. , and Varol, G. , editors, *Computer Vision – ECCV 2024*, volume 15087 of *Lecture Notes in Computer Science*, pages 149–167. Springer, Cham, 2025. doi: 10.1007/978-3-031-73397-0_10.

[57] Yunus, R. , Lenssen, J. E. , Niemeyer, M. , Liao, Y. , Rupprecht, C. , Theobalt, C. , Pons-Moll, G. , Huang, J.-B. , Golyanik, V. , and Ilg, E. . Recent trends in 3d reconstruction of general non-rigid scenes. In *Computer Graphics Forum*, volume 43, page e15062. Wiley Online Library, 2024.

[58] Zhang, C. , Zhou, Y. , and Zhang, L. . Voxel-mesh hybrid representation for real-time view synthesis by meshing density field. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[59] Zhang, H. , Wang, C. , Tian, S. , Lu, B. , Zhang, L. , Ning, X. , and Bai, X. . Deep learning-based 3d point cloud classification: A systematic survey and outlook. *Displays*, 79: 102456, Sept. 2023. ISSN 0141-9382. doi: 10.1016/j.displa.2023.102456. URL http://dx.doi.org/10.1016/j.displa.2023.102456.

[60] Zhang, Z. and Sabuncu, M. . Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

[61] Zhou, S. , Chang, H. , Jiang, S. , Fan, Z. , Zhu, Z. , Xu, D. , Chari, P. , You, S. , Wang, Z. , and Kadambi, A. . Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21676–21685, June 2024.

[62] Zhu, R. , Qiu, S. , Liu, Z. , Hui, K.-H. , Wu, Q. , Heng, P.-A. , and Fu, C.-W. . Rethinking end-to-end 2d to 3d scene segmentation in gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3656–3665, June 2025.

APPENDIX

This appendix provides additional information on the implementation resources, datasets, experimental results, and code repositories developed in this work. These resources are provided to facilitate reproducibility, allow further exploration, and support future research in segmentation and editing in 3D Gaussian Splatting (3D-GS) scene.

A.1 High-Quality Multi-View Object Mask Dataset

The new high-quality dataset, referred to as 3D Scene Segmentation Dataset, contains preprocessed multiview segmentation masks for synthetic and real-world scenes from datasets such as LeRF [24], Mip-NeRF [1], and LLFF [56]. This dataset captures high-quality object boundaries and artifact-free masks suitable for object-level 3D Gaussian Splatting training. The 3D Segmentation high-quality dataset is a curated collection of 5 real-world scenes with high-quality object segmentation masks designed for research in 3D scene understanding, editing, and rendering. This dataset improves upon existing benchmarks by providing cleaner and more consistent object masks across multiple views, enabling reliable evaluation and training for 3D scene editing tasks. Key features of the dataset include:

- Consistent multiview object labels for all frames of a scene.
- Cleaned masks with small artifacts removed and boundary refinement applied.
- Preprocessed using SAM-HQ [22] segmentations enhanced by our pipeline.

The high-quality object mask dataset has been uploaded and made publicly available on official HuggingFace website at the following link:
`https://huggingface.co/datasets/joshir/3D-Scene-Segmentation-HQ`

A.2 Rasterization and training process of 3D Gaussian Splatting

The rasterization process in 3D Gaussian Splatting (3DGS) is a highly efficient, differentiable pipeline for rendering 3D scenes as collections of anisotropic Gaussians. It bridges the gap between a set of 3D point primitives and the final 2D images used for supervision. Unlike mesh-based rasterization, 3D-GS treats each primitive as a Gaussian ellipsoid parameterized by position,

scale, opacity, and color (or spherical harmonics coefficients). Rendering is achieved through tile-based splatting that ensures real-time efficiency.

In tile-based rasterization, the multiview image screen is divided into 16×16 tiles, and Gaussians are culled against the view frustum and tile bounds. Only relevant Gaussians are kept, with unstable ones near the near plane discarded. Each Gaussian is then assigned a depth and tile key, and all Gaussians are globally sorted using a fast GPU radix sort. This avoids per-pixel ordering and enables approximate but visually accurate α-blending. For rasterization, a thread block processes each tile, loading Gaussians into shared memory and blending them front-to-back. Processing stops when all pixels in a tile reach full opacity ($\alpha \approx 1$), boosting parallelism and efficiency. Unlike prior work [1], the method places no limit on how many splats contribute gradients, making it robust to complex scenes. In backpropagation, the sorted Gaussians are traversed back-to-front, reusing the same tile lists. Gradients are efficiently computed by storing only the final accumulated opacity during the forward pass, which is then decomposed to recover per-step blending coefficients.

When extended with a semantic feature of dimension d, each Gaussian carries both RGB and latent semantic vectors. The rasterization pipeline blends these features using the same α-compositing weights. The rendered per-pixel semantic field can then be supervised with external signals such as segmentation masks or SAM features, enabling Gaussians to learn in parallel appearance and semantic consistency across views. Full details are provided as psuedocode in Algorithm 3

In the course of this dissertation work, we have modified rasterization code to incorporate additional semantic or object-specific feature dimensions alongside the standard radiance attributes. This modification emphasizes our deep understanding of the core design of 3D-GS rasterization and highlights how it can be extended beyond photometric rendering. Through the original RGB rasterization implementation, we learned how the tile-based α-compositing pipeline can be generalized to simultaneously blend both appearance (RGB or spherical harmonics) and latent features across multiple views. In our dissertation, this step was crucial because it allowed each gaussian primitive to carry not only color information but also object or semantic features. By integrating these features into the forward rasterization and backpropagation pipeline, the Gaussians become directly trainable for both visual fidelity and semantic consistency.

Algorithm 3 Parallel Feature-based Gaussian Rasterization

PointCloud ← StructureFromMotion() ▷ Obtain point cloud
(X, C) ← PointCloud ▷ Extract positions and colors
(Σ, A, F) ← InitAttributes() ▷ Initialize covariances, opacities, semantic features
$F_t(I)$ ← ApplyFoundationModel(I) ▷ **Compute feature map**
$i \leftarrow 0$ ▷ Initialize iteration counter
repeat
  (V, I, F_t) ← GetTrainingView() ▷ Get camera pose, image, and feature map
  $(\hat{I}, F_s)$ ← ParallelRasterizer(X, C, Σ, A, F, V) ▷ Rasterization
  $L \leftarrow$ Loss$(I, \hat{I}) + \lambda$**Loss**$(\mathbf{F_t}, \mathbf{F_s})$ ▷ **Loss calculation**
  (X, Σ, C, A, F) ← Adam(L) ▷ Backpropagation and optimization step
  if IsRefinementStep(i) **then**
    for each Gaussian (x, q, c, α, f) **do**
      if $\alpha < \epsilon$ or IsTooLarge(x, q) **then**
        RemoveGaussian() ▷ Remove low-opacity or oversized Gaussians
      end if
      if $\nabla_p L > \tau_p$ **then**
        if $\|S\| > \tau_S$ **then**
          SplitGaussian(x, q, c, α, f) ▷ Handle over-reconstruction
        else
          CloneGaussian(x, q, c, α, f) ▷ Handle under-reconstruction
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$ ▷ Increment counter
until Convergence

This exercise served as **proof of a strong conceptual and practical understanding of how differentiable rasterization operates at scale**, and how subtle design choices (e.g., tile size, sorting, shared-memory usage) directly affect efficiency, scalability, and gradient stability. Such knowledge is critical for future research in 3DGS, as new application domains (e.g., semantic editing, robotics, AR/VR) will demand similar extensions of the rasterization process. **The modified implementation for object-aware rasterization** has been made publicly available and can be found at our github repository:

`https://github.com/joshir199/object-feature-rasterization`.

This repository provides a reproducible reference for how one can adapt the official 3D Gaussian Splatting rasterizer to embed higher-level features and design new loss functions around them.

A.3 More Results For Feature-Based 3D-GS Training

In this appendix, we also provide details of the GitHub repository that accompanies this work. The repository includes implemented codes, rasterization submodule, and precomputed predictions on

the high-quality dataset along with multiview visualizations of object removal and recolorization results. Full details of code and its step-by-step process for building and execution of project is mentioned in README file. Given the large number of views per 3D scene, including all visual results in this document is impractical, as it would require excessive space and reduce overall clarity. To address this, a video version of the results is provided in the repository, offering a more comprehensive and intuitive demonstration of the multiview editing results. The github repo for implementated code and details can be found at: `https://github.com/joshir199/Object-Removal-And-Recolorization-Within-3D-Gaussian-Splatting`

Further, this section presents additional experimental results that evaluate the impact of feature dimensionality on training efficiency, reconstruction quality, and semantic embedding performance within the proposed 3D Gaussian Splatting framework. Tables 7.1 and 7.2 report the training times for 7k and 30k iterations, respectively, across feature dimensions ranging from 8 to 256. The results clearly show that increasing the feature dimension significantly raises training time, with dimensions above 128 often leading to out-of-memory (OOM) errors, especially in larger or more complex scenes. Tables 7.3 and 7.4 provide the corresponding PSNR values at 7k and 30k iterations. Interestingly, PSNR remains largely stable across dimensions, suggesting that increasing feature dimensionality does not directly enhance photometric reconstruction quality. This highlights the fact that the benefits of higher-dimensional features lie not in image fidelity but in enabling semantic interpretability and downstream editing tasks. Finally, Table 7.5 reports the autoencoder training loss across different dimensions, showing a consistent decrease in loss as feature dimensionality increases, with near-zero values at the highest dimensions. This shows that higher-dimensional features retain semantic richness more effectively, though at the cost of training efficiency and memory consumption. Collectively, these results reinforce the design choice of balancing feature dimensionality to achieve a practical trade-off between semantic capability, computational cost, and reconstruction fidelity.

Table 7.1: Training time for 7k iterations across different feature dimensions (d). (OOM = Out of Memory)

Scene	**3D-GS training time (h=hour, m=minute)**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	60m	60m	1h 31m	2h 05m	3h 07m	OOM
fortress	37m	45m	57m	1h 37m	2h 08m	OOM
horns	33m	37m	49m	1h 13m	1h 36m	2h 21m
ramen	22m	26m	38m	1h 04m	1h 28m	2h 11m
teatime	44m	43m	1h 07m	1h 33m	2h 16m	OOM

Table 7.2: PSNR at 7k iterations for different feature dimensions (d). (OOM = Out of Memory)

Scene	**PSNR**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	25.93	26.12	26.10	26.11	26.08	OOM
fortress	29.49	29.24	29.19	29.21	29.19	29.16
horns	24.74	25.04	25.09	25.26	25.19	25.08
ramen	26.22	26.23	26.13	26.17	26.13	26.11
teatime	26.13	26.02	26.09	26.11	26.03	OOM

Table 7.3: Training time for 30k iterations across different feature dimensions (d). (OOM = Out of Memory)

Scene	**3D-GS training time (h=hour, m=minute)**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	4h 06m	4h 39m	5h 04m	5h 41m	OOM	OOM
fortress	2h 29m	2h 34m	2h 41m	3h 06m	OOM	OOM
horns	2h 07m	2h 26m	2h 51m	3h 45m	4h 28m	OOM
ramen	1h 43m	1h 57m	2h 39m	3h 27m	4h 13m	OOM
teatime	2h 47m	3h 07m	3h 41m	4h 27m	OOM	OOM

Table 7.4: PSNR at 30k iterations for different feature dimensions (d). (OOM = Out of Memory)

Scene	**PSNR**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	29.27	29.31	29.31	28.93	OOM	OOM
fortress	31.71	31.74	31.70	31.28	OOM	OOM
horns	26.78	26.82	26.33	26.28	26.11	OOM
ramen	28.27	28.41	28.31	28.19	28.01	OOM
teatime	28.64	28.63	28.49	28.37	OOM	OOM

Table 7.5: Autoencoder loss for different feature dimensions (d).

Scene	**Autoencoder training loss**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	0.474	0.387	0.325	0.274	0.150	0.0
fortress	0.496	0.401	0.372	0.315	0.287	0.0
horns	0.743	0.623	0.536	0.423	0.360	0.0
ramen	0.413	0.316	0.257	0.195	0.195	0.0
teatime	0.406	0.308	0.262	0.198	0.198	0.0

Table 7.6: Number of Gaussian points for different feature dimensions (d). (OOM = Out of Memory)

Scene	**Num. of Gaussians (in million)**					
	d = 8	**16**	**32**	**64**	**128**	**256**
bear	2.51M	2.78M	2.81M	2.82M	OOM	OOM
fortress	0.90M	0.90M	0.91M	0.93M	0.94M	OOM
horns	0.81M	0.84M	0.85M	0.86M	0.88M	OOM
ramen	0.70M	0.72M	0.74M	0.77M	0.83M	OOM
teatime	2.11M	2.19M	2.21M	2.33M	OOM	OOM