



3DGS-based Interactive Scene Object Manipulation and Animation

by

Jiajun Huang

National Centre for Computer Animation

Faculty of Media & Communication

Bournemouth University

A thesis submitted in partial fulfilment of the
requirements of Bournemouth University for the degree of
Doctor of Philosophy

Mar. 2025

Copyright Statement

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Acknowledgements

So, at last, it is finally my turn to put down this section.

I want to thank my supervisors, Dr. Hongchuan Yu, Prof. Jian Jun Zhang, and Prof. Hammadi Nait Charif, for their continuous support throughout my PhD. Especially Dr. Yu, without his patience, time, and guidance, I would never be able to go this far.

I also want to thank my family, especially my mother, for her countless cheers and comfort during the darkest times.

Thanks to my friends and colleagues, especially Xiaoxiao Liu, Mengqing Huang, Kavisha Jayathunge, Sasha Alexdottir, and Armin Sheibanifard, for tolerating my endless interruptions and ramblings. Shuolin Xu, for also being an awesome collaborator!

Thanks to my roommates: Junheng Fang, Nan Xiang, Xingyu Ye, and Abdul Rehman, for picking up the slack. I really should have helped more with the chores!

To Bournemouth University and the Centre of Digital Entertainment, thank you for funding my study. It is truly an honor.

To a feline lady, you have changed my life for the better. It may be just a little, but it matters.

To my loving partner, Xuan, thank you for all your love and company. It is a miracle beyond all imagination to have you by my side.

Finally, to Kirigaya Yui, Ikaruga Atri, and Mizugaki Isla. Thanks for being my final reason to start this journey, to persist, and to push through the end. May my future endeavors not be in vain.

May I still have the courage to continue, and do better.

Abstract

3D Gaussian Splatting (3DGS) has emerged as a premier method for high-fidelity scene reconstruction, yet it lacks a complete workflow for object manipulation. The core challenge lies in the structural incompatibility between the novel volumetric primitives of 3DGS and established editing paradigms found in classical geometry processing and modern 2D computer vision works. The focus of this thesis is to address this gap, namely: how can these disparate methodologies be effectively bridged to enable an end-to-end editing pipeline for 3DGS, comprising segmentation, deformation, and recomposition?

For segmentation, this thesis focuses on how to transfer the powerful, open-world segmentation capabilities of modern 2D vision models to 3DGS. A dual-stage self-prompting cross-view propagation process is proposed to guide 3D segmentation using 2D models. A critical learning from this investigation is that while training extended representations effectively connects 2D models to 3DGS, a dual-stage design is crucial to include primitives inside objects, which is essential for robust manipulation.

For deformation, this thesis focuses on how to adapt classical cage-based deformation techniques to the novel volumetric primitives of 3DGS. The research demonstrates that a proxy point-based representation can bridge this gap and enables real-time, non-rigid deformation on 3DGS or its variants without requiring retraining or architectural modification. An automatic cage-building algorithm is also proposed to produce high-quality cages

for manipulation. A key insight is that such proxy representation could be effective for adapting classical shape manipulation methods to the novel volumetric primitives of 3DGS.

For inpainting and recomposition, this thesis focuses on how to achieve high-fidelity 3D inpainting of 3DGS using 2D image inpainters. This work overcomes the multiview inconsistency issue of 2D inpainters by employing a perceptual-based loss in the scene fine-tuning process. The process is also augmented with a scene content revealing pruning strategy to improve quality. Recomposition is achieved via set union operations. Apart from employing the perceptual loss, a critical learning is that leveraging existing information in the scene is crucial to minimize scene changes and thus improves result quality.

Collectively, these contributions formulate an integrated pipeline for selection, manipulation, and recomposition. By answering these questions, this thesis bridges 3DGS with established paradigms and advancements, enhancing the applicability of captured scenes in creative downstream tasks and achieving state-of-the-art quality in evaluations.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Main Challenge	3
1.3	Aims and Objectives	5
1.4	Contribution	6
1.5	List of Publications	7
1.6	Outline of Thesis	8
2	Literature Review	9
2.1	Scene Representation Methods	9
2.1.1	Traditional Scene Representation Methods	9
2.1.2	Neural Radiance Field (NeRF)	11
2.1.3	3D Gaussian Spalting (3DGS)	13
2.1.4	Summary	16
2.2	Scene Segmentation Methods	17
2.2.1	Traditional Representation Segmentation	17
2.2.2	NeRF Segmentation	19
2.2.3	3DGS Segmentation	22
2.2.4	Summary	24
2.3	Scene Editing Methods	24
2.3.1	Traditional Editing Methods	24
2.3.2	NeRF Editing	26
2.3.3	3DGS Editing	27
2.3.4	Summary	29
2.4	Scene Inpainting Methods	30

2.4.1	Traditional Inpainting Methods	30
2.4.2	Radiance Field Inpainting Methods	31
2.4.3	Summary	33
3	Comprehensive 3DGS Segmentation for Manipulation	35
3.1	Introduction	35
3.2	Pipeline Overview	37
3.3	Methodology	38
3.3.1	Coarse Stage Segmentation	39
3.3.2	Fine Stage Segmentation	42
3.3.3	Merging and Refinement	44
3.4	Experiments	46
3.4.1	Implementation Details	46
3.4.2	Experiment Setups	46
3.4.3	Evaluating Segmentation Quality	47
3.4.4	Ablation Studies	51
3.5	Summary	52
4	Real-time Cage-based Deformation for 3DGS	53
4.1	Introduction	53
4.2	Method	55
4.2.1	Preliminaries	55
4.2.2	Cage-Building Algorithm	56
4.2.3	Deformation Algorithm	59
4.3	Experiments	66
4.3.1	Implementation Details	66
4.3.2	Cage Building Quality	66
4.3.3	Deformation Quality	68
4.3.4	Other Control Methods	71
4.3.5	Training & Deformation Speed	74
4.4	Applications	81
4.4.1	Composition and Animation	81
4.4.2	Combining with Editing Methods	82

4.4.3	Extending to Variants of 3DGS	83
4.4.4	Ablation Studies	84
4.5	Summary	90
5	Targeted Scene Inpainting and Recomposition for 3DGS	91
5.1	Introduction	91
5.2	Methodology	93
5.2.1	Exposed Surface Inpainting	93
5.2.2	Object Recomposition	97
5.3	Experiments	98
5.3.1	Implementation Details	99
5.3.2	Experiment Setups	99
5.3.3	Object Manipulation	100
5.3.4	Evaluating Inpainting Quality	101
5.3.5	Ablation Studies	105
5.4	Summary	107
6	Conclusion and Future Work	108
6.1	Review of Accomplished Research	108
6.2	Future Work	109
	Bibliography	111

List of Figures

3.1	Overview of the proposed approach. The proposed process produces a 3D 3DGS scene segmentation from 2D user prompts.	36
3.2	An overview of the proposed three-stage editing pipeline for 3DGS scenes.	37
3.3	The coarse stage of the proposed segmentation algorithm. . . .	40
3.4	The fine stage of the proposed segmentation algorithm. . . .	43
3.5	The final combination stage of the proposed segmentation algorithm.	45
3.6	Qualitative scene segmentation results on the SPIn-NeRF dataset.	48
3.7	Qualitative results to compare the comprehensiveness of segmentation against SAGA.	51
3.8	Ablation study results on the dual-stage design. $\mathcal{R}_{bg}$ part of segmentation, performed with (left) or without (right) the dual stage design.	52
4.1	Overview of methods proposed in this chapter.	54
4.2	Overview of the cage-building algorithm.	57
4.3	Overview of the deformation algorithm.	60
4.4	The splitting process. This method resolves ill-formed bent Gaussians by splitting the original Gaussian before deformation.	63
4.5	Comparison of intermediate voxel grid and final result of cage building algorithms. Red circles and cyan boxes marks defects.	69
4.6	Quality comparison of deformation methods on selected objects. Red boxes indicate zoomed areas; cyan circles marks defects.	72

4.7	Quality comparison of deformation methods under normal to extreme deformations. Red boxes indicate zoomed areas; cyan circles marks defects.	73
4.8	Control structure and result quality comparison of methods that can be adapted for 3DGS deformation. Red boxes indicate zoomed areas; cyan circles marks defects.	75
4.9	Example of animating and composing scenes using the proposed method.	82
4.10	Integrating the proposed method with GaussianEditor (Chen et al. 2023b).	83
4.11	Integrating the proposed method with GSTex (Rong et al. 2024).	84
4.12	Integrating the proposed method with 2DGS (Huang et al. 2024).	85
4.13	Integrating the proposed method with FLoD (Seo et al. 2024).	85
4.14	Ablation study results for cage building algorithm. The voxel grids before closing simplification are shown.	86
4.15	(a) Ablation study results for deformation algorithm. (b) The zoomed-in view of (a) highlights the deformed Gaussian (cyan diamond) and the actual Gaussians (red diamond).	88
4.16	Ablation study results for the split factor k	89
5.1	An overview of the inpainting and recomposition pipeline proposed in this chapter.	92
5.2	Overview of the proposed 2D-guided 3D inpainting stage.	94
5.3	Qualitative results of the proposed object manipulation pipeline.	101
5.4	Qualitative comparison of inpainting results.	102
5.5	Qualitative ablation of scene content-revealing pruning strategy.	106
5.6	Qualitative ablation of scene fine-tuning, performed with (left) or without (right) reprojection-based initialization.	107

List of Tables

3.1	Quantitative scene segmentation results on the SPIn-NeRF dataset. The best value of each column is bolded for ease of reading.	49
3.2	Per-scene quantitative scene segmentation results on the SPIn-NeRF dataset. The best value of each column is bolded for ease of reading.	50
4.1	Benchmark results comparing training times. Red indicates best values, blue marks second-best.	76
4.2	Detailed per-scene training time of Table 4.1. Red indicates best values, blue marks second-best.	77
4.3	Benchmark results comparing deformation times. Red indicates best values, blue marks second-best.	78
4.4	Detailed per-scene deformation time of Table 4.3 on NeRF scenes.	79
4.5	Detailed per-scene deformation time of Table 4.3 on DeformingNeRF scenes.	80
5.1	Quantitative comparison of inpainting performance. The best value of each column is bolded for ease of reading.	103
5.2	Per-scene PSNR and LPIPS comparison for inpainting. The per-scene best metrics are bolded for ease of reading.	103
5.3	Per-scene FID comparison for inpainting. The per-scene best metrics are bolded for ease of reading.	104
5.4	Quantitative ablation of scene content-revealing pruning strategy.	106

Chapter 1

Introduction

1.1 Motivation

The ability to capture the three-dimensional world in high fidelity and then freely manipulate it has been a long-standing goal of computer graphics and vision. Imagine a film director seamlessly altering a prop in a photorealistic digital set post-production, an architect allowing a client to virtually move walls in a captured building, or a user in an augmented reality application picking up a real-world object and placing it elsewhere. These applications, spanning entertainment, content creation, and virtual and augmented reality, all hinge on a core capability: the intuitive and robust editing of realistic 3D scene representations.

For decades, the creation and manipulation of 3D content have been dominated by explicit geometric representations, primarily polygonal meshes (Botsch et al. 2010) and point clouds (Markus Gross 2007). These representations have fostered a mature ecosystem of modeling and animation techniques (Ju et al. 2005, Sorkine-Hornung and Alexa 2007a, Rusu and Cousins 2011). Their explicit nature, namely a mesh’s interconnected vertices and faces, or a point cloud’s discrete samples of a surface, makes them inherently amenable to direct manipulation, supporting a vast array of well-established editing workflows.

However, reconstructing these representations from real-world imagery, typically through photogrammetry methods (Schönberger and Frahm 2016,

Schönberger et al. 2016), is fraught with challenges. This process often yields incomplete, noisy, or overly-smoothed geometry, struggling particularly with thin structures, textureless regions, and reflective surfaces (Furukawa and Hernández 2015). Furthermore, as fundamentally surface-based approximations, they struggle to represent the full richness of reality. Volumetric phenomena like clouds or smoke, translucent materials such as glass, and complex view-dependent effects like iridescence are difficult to capture and render faithfully with these methods.

The central challenge for traditional methods has thus been achieving high-fidelity, photorealistic capture.

The recent advent of radiance field-based scene representations has dramatically altered this landscape, effectively inverting this classic challenge. Pioneered by Neural Radiance Fields (NeRF) (Mildenhall et al. 2020) and advanced by successors like 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023), these approaches excel at capturing the appearance of real-world objects and environments with unprecedented quality. By learning a scene’s radiance and density directly from images, they can synthesize novel views with great realism and handle the complex materials and lighting that were once difficult to capture. Yet, this success introduces the opposite problem: the very properties that enable their realism create a fundamental barrier to their manipulation.

Unlike a mesh, a 3DGS scene is a disorganized cloud of semi-transparent, view-dependent Gaussian primitives. It does not fit cleanly into the traditional explicit structures that classical techniques have focused on, as it lacks an explicit surface structure and exact semantic organization. This disconnect forms a critical gap: radiance field-based scene representations possess the ability to capture the world with incredible fidelity, but they lack the tools to intuitively and creatively edit these captures.

The focus of this thesis is to bridge this critical gap. By bridging this novel family of scene representations with the well-developed classical shape manipulation methods (Ju et al. 2005), straightforward editing of these representations can be achieved. By further bridging these representations with advancements in 2D vision models, such as interactive segmentation models

(Kirillov et al. 2023) and generative inpainting models (Suvorov et al. 2022), a more intuitive and natural editing experience can be achieved.

In summary, this thesis is motivated by the goal of creating an intuitive and integrated editing experience for 3D Gaussian Splatting. The editing experience consists of the user selecting an object from an existing 3DGS capture, manipulating it, and finally reintegrating the object back into the capture, with newly exposed regions inpainted. In short, this involves segmentation, manipulation, and recomposition. Manipulation of 3DGS is achieved by bridging it with cage-based deformation, a well-studied and widely adopted technique in the animation industry. Intuitive segmentation and natural inpainting of exposed areas after editing are achieved by leveraging rapidly developing 2D vision models, with novel designs that make them more applicable to 3D representations and the editing scenario.

1.2 Main Challenge

While 3D segmentation (He et al. 2021), shape manipulation (Botsch et al. 2010), and inpainting (Tesema et al. 2024) are extensively studied research topics, the recent emergence of radiance field-based scene representations presents fundamental challenges that preclude the direct application of these established techniques. Unlike traditional structured representations like meshes, 3DGS scenes are composed of unstructured primitives with volumes, lacking an explicit notion of surface. This inherent incompatibility prevents the direct application of many existing editing methods (Floater 2003, Kavan 2014). Furthermore, while recent 2D vision models have demonstrated impressive abilities in tasks such as intuitive 2D segmentation (Kirillov et al. 2023) and high-quality 2D inpainting (Suvorov et al. 2022), transferring their 2D capabilities to the 3D context is a non-trivial task.

The main challenges to be tackled as follows:

1. Manipulation-Ready 3D 3DGS Segmentation from 2D Cues.

The power of modern 2D segmentation models lies in their ability to understand intuitive user prompts (e.g, point prompts on an image) and

perform segmentation without limits to specific domains or datasets. The challenge then lies in how to utilize 2D segmentation models and single-view 2D prompts to achieve complete 3D segmentation on a 3DGS capture. It should be noted that lifting 2D segmentations to 3D segmentations is prone to ignoring parts of the object that are below the surface. This superficial result is inadequate for downstream manipulation tasks, which require a comprehensive selection of all primitives, both surface and interior, that constitute the object.

2. Deformation of the Novel Volumetric Primitives of 3DGS.

3DGS represents scenes using a collection of novel volumetric primitives that do not cleanly fit any of the existing geometry representations. This makes existing shape manipulation techniques, designed for traditional geometry representations, not directly applicable, necessitating special designs. Furthermore, to achieve the most straightforward editing experience, the method should not require extensive conversion, ideally run in real-time, and should work with variants of 3DGS, all of which make the design more challenging.

- 3. High-Quality 3DGS Inpainting from Multi-view Inconsistent 2D Inpainters.** Following an object’s removal or significant manipulation, surfaces in captures that were originally covered by the object would be exposed and should be filled for realistic results. While state-of-the-art 2D generative models produce photorealistic completions for a single viewpoint, they are not designed with multiview consistency in mind. This makes using them to guide the inpainting of 3D representations challenging. Additionally, minimizing unnecessary changes in scene captures is also critical for achieving high-quality inpainting, as well-trained scene contents should be kept intact for the best result quality.

1.3 Aims and Objectives

This thesis aims to bridge the critical gap between the high-fidelity capture capabilities of 3D Gaussian Splatting (3DGS) and the ability to perform intuitive, robust scene manipulation. More specifically, it aims to enable users or artists to intuitively segment objects from a trained 3DGS capture, manipulate it in a way that is easy to integrate with the existing mesh-based tools, and naturally inpaint all newly exposed scene surfaces with high quality. This is achieved by bridging the 3DGS scene representation or its variants with classical shape manipulation methods and recent advancements in 2D vision models.

Based on the proposed approach, the research questions are as follows:

- How can the powerful, open-world segmentation capabilities of modern 2D vision models be effectively transferred to 3DGS? How to overcome the existing method’s focus on object surfaces or rendered masks, and comprehensively cover the internals of objects?
- How can the well-established shape manipulation techniques, such as cage-based deformation, be adapted to manipulate the novel unstructured volumetric primitives of 3DGS? How to make this connection without requiring retraining or conversion, runs in real-time, and generalizes to variants of 3DGS?
- How can the photorealistic but view-inconsistent outputs of 2D generative inpainters be utilized to achieve coherent and high-fidelity 3D scene inpainting for 3DGS? How to minimize the amount of unnecessary inpainting or scene changes needed for higher quality results?

With these research questions answered, this thesis aims to achieve three major objectives:

- The first objective is to achieve high-quality 3D segmentation from easily provided single-view 2D point prompts, leveraging the power of state-of-the-art 2D computer vision models. The segmentation will

comprehensively cover the internals of the object to ease manipulation. This objective is resolved in Chapter 3.

- the second objective is to achieve training-free and real-time application of cage-based deformation to the 3DGS family of methods. An algorithm to automatically generate high-quality cages from this family of 3D representation methods should also be proposed. This objective is detailed in Chapter 4.
- the final objective is to achieve high-quality 3D inpainting from high-quality but multiview inconsistent 2D inpainters, as well as 3DGS re-composition. The proposed inpainting algorithm should also be designed to maximize the preservation of existing scene content, ensuring high-fidelity results. This objective is detailed in Chapter 5.

1.4 Contribution

The main contribution of this thesis is a cohesive framework that bridges the novel 3D Gaussian Splatting (3DGS) representation with established editing paradigms and modern 2D vision models. This work provides a complete and intuitive workflow for editing 3DGS scenes by developing new methods for the three core stages of an editing pipeline: segmentation, manipulation, and inpainting. By connecting 3DGS to traditional shape manipulation techniques and leveraging the power of 2D computer vision models, this research makes high-fidelity radiance field captures more accessible for practical scenarios.

The detailed contributions are as follows:

- The first contribution is a new way to perform comprehensive 3D segmentation by effectively lifting the capabilities of 2D segmentation models to 3D. The proposed method utilizes a cross-viewpoint propagation and a two-stage refinement process to convert simple 2D user prompts into complete 3D selections. Unlike methods that only include an object’s visible surface, this approach ensures the selection includes the object’s internal primitives, making it robust and suitable for clean extraction and subsequent manipulation.

- The second contribution connects the traditional and widely-used technique of cage-based deformation to the novel volumetric representation of 3DGS. This is achieved through a novel proxy-based design that transfers deformations to the Gaussian primitives in real-time without requiring scene conversion or retraining. This approach effectively bridges the gap between the mature ecosystem of mesh-based animation tools and the unstructured nature of 3DGS, enabling artists to apply familiar workflows directly to this new class of representations.
- The third contribution is a novel inpainting method for 3DGS that produces high-quality, multi-view consistent results while ensuring minimal changes to the original scene. It bridges the gap between powerful 2D inpainting models, which are inherently view-inconsistent, and the requirements of 3D scene completion. A key aspect of this method is a content-preserving pruning strategy that reveals and preserves existing, well-reconstructed parts of the scene, thereby localizing the inpainting process only to newly exposed regions and maximizing the overall fidelity of the final result.

1.5 List of Publications

- Huang, J., Yu, H., Zhang, J. and Nait-Charif, H., 2024. Point’n move: Interactive scene object manipulation on gaussian splatting radiance fields. *IET Image Processing*, 18 (12), 3507–3511
- Huang, J., Xu, S., Yu, H. and Lee, T.-Y., 2024. GSDeformer: Direct, Real-time and Extensible Cage-based Deformation for 3D Gaussian Splatting. (Under Review at *IEEE Transactions on Visualization and Computer Graphics*)

1.6 Outline of Thesis

The rest of the thesis is organized as follows:

- Chapter 2 reviews advances in scene representations and state-of-the-art approaches for segmentation, editing, and inpainting of 3D scenes. All with a focus on the novel family of radiance field-based scene representations.
- Chapter 3 presents the proposed 3D segmentation method, detailing the dual-stage self-prompting mask propagation process for lifting 2D user prompts to comprehensive 3D object selections.
- Chapter 4 introduces the novel method for applying cage-based deformation to 3DGS, presenting both the cage-building algorithm and the deformation algorithm.
- Chapter 5 details the approach for exposed area inpainting and recomposition, explaining how rapid multiview-consistent 3D inpainting is achieved from 2D inpainters and how to maximize the preservation of original scene content.
- Chapter 6 concludes this thesis by summarizing key contributions and discussing potential directions for future research.

Chapter 2

Literature Review

Segmenting, editing, and inpainting 3D Gaussian Splatting, or more broadly, 3D scene representations, is not an unexplored topic. This chapter conducts a thorough review of 3D scene representation methods and measures for segmenting, editing, and inpainting them, all with a focus on radiance field-based scene representations. This is to set the stage for the proposal and integration of novel methods.

2.1 Scene Representation Methods

2.1.1 Traditional Scene Representation Methods

Traditionally, the creation, representation, and manipulation of 3D scenes have been dominated by explicit geometric representations. Among these, polygonal meshes, point clouds, and voxel grids stand out as the most prominent methods, each offering a distinct approach to encoding 3D structure. While they have fostered a mature ecosystem of tools and techniques, they share fundamental limitations, particularly in the context of reconstructing and representing the full visual complexity of the real world.

Polygonal meshes are arguably the most ubiquitous representation in computer graphics. By defining surfaces as a collection of interconnected vertices, edges, and faces, they provide a structured and explicit description of an object’s shape (Botsch et al. 2010). This structured nature is their core strength; it makes them computationally efficient for rendering

with modern graphics hardware and inherently amenable to a vast array of well-established direct manipulation and animation techniques (Ju et al. 2005, Sorkine-Hornung and Alexa 2007a). However, this contribution is also their primary limitation. As a fundamentally surface-based approximation, meshes struggle to represent volumetric phenomena like smoke, complex materials such as glass, or view-dependent effects. Furthermore, reconstructing high-quality meshes from real-world imagery is notoriously challenging, often resulting in noisy, incomplete, or overly-smoothed geometry, especially for thin structures or non-Lambertian surfaces (Furukawa and Hernández 2015).

Point clouds offer a simpler and more direct alternative, representing scenes as a raw collection of discrete 3D points (Markus Gross 2007). Their primary advantage lies in their ease of acquisition; sensors like LiDAR and depth cameras directly produce point clouds, making them a common first step in many 3D reconstruction pipelines (Rusu and Cousins 2011). This directness, however, comes at the cost of structure. Point clouds lack the explicit connectivity information inherent in meshes, which complicates tasks like rendering solid surfaces and makes direct geometric editing less intuitive. Their unstructured nature also means they often contain significant noise and varying densities, posing challenges for downstream processing.

In contrast to surface-based methods, voxel grids discretize 3D space into a regular grid of volumetric elements, or voxels. This approach is exceptionally well-suited for representing internal structures and volumetric data, making it a staple in fields like medical imaging. Its conceptual simplicity, as a direct extension of 2D pixels to 3D, is a significant advantage. The primary criticism of the voxel grid approach, however, is its prohibitive spatial complexity. The memory consumption scales cubically with resolution, making it impractical for representing large or highly detailed scenes. This trade-off between resolution and memory usage means that voxel-based representations often suffer from either a coarse, blocky appearance or an unmanageable memory footprint, limiting their utility for high-fidelity scene representation.

In summary, while traditional methods have been instrumental in the development of 3D graphics and are invaluable for tasks involving explicit

geometric manipulation, they fundamentally struggle to bridge the gap between geometric structure and photorealistic appearance. Their limitations in representing complex materials and volumetric effects, coupled with the inherent difficulties of high-quality reconstruction from real-world data, highlight the need for representations that prioritize visual fidelity, setting the stage for the emergence of radiance field-based methods.

2.1.2 Neural Radiance Field (NeRF)

Radiance field-based scene representations are a new family of methods that have recently emerged. These approaches model scenes as radiance fields and have demonstrated impressive capabilities in achieving high-fidelity 3D scene reconstruction. Furthermore, they exhibit considerable potential for integrating with other existing artificial intelligence and machine learning works, promising new avenues for research and application.

Neural Radiance Field (NeRF) (Mildenhall et al. 2020), as a pioneering method in its domain, introduced two principal innovations. Firstly, NeRF proposed representing scenes as continuous volumetric radiance fields, wherein every point in 3D space is characterized by an opacity and a view-dependent emitted color. This 3D representation can then be synthesized into 2D images using differentiable volumetric rendering techniques. The differentiability of this rendering process is crucial, as it allows the radiance field to be optimized via standard gradient descent. Consequently, given a set of input images with corresponding camera poses, this optimization procedure facilitates the reconstruction of the underlying 3D scene. However, naively storing such a radiance field using explicit voxel grids would be prohibitively memory-intensive. Therefore, NeRF innovatively proposed to encode this field implicitly using a small neural network. These innovations enable NeRF to achieve high-fidelity 3D scene reconstruction from posed images, establishing it as a compelling alternative to traditional photogrammetry pipelines.

More concretely, NeRF proposes to represent scenes using a radiance field function. The function is defined as $f_{\theta} : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, where θ are the

learnable parameters of function f . Given any point in the 3D space, this function takes its 3D coordinate $\mathbf{x} \in \mathbb{R}^3$ and the viewing direction $\mathbf{d} \in \mathbb{S}^2$ to produce color $\mathbf{c} \in \mathbb{R}^3$ and volume opacity $\sigma \in \mathbb{R}$.

To render an image $\mathbf{I}$ from this function, volumetric rendering is used. For every pixel of the image to render, the process cast a ray $\mathbf{r}(t) = \mathbf{x}_0 + t\mathbf{d}$, where $\mathbf{x}_0$ is the camera’s origin, $\mathbf{d}$ is the ray’s direction that points from the origin of the camera to the pixel on the image plane, and t is the distance of the point along the ray. The color $\mathbf{I}_i(\mathbf{r})$ of the pixel is then obtained by Equation (1) in NeRF (Mildenhall et al. 2020):

$$\mathbf{I}_i(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt \quad (2.1)$$

where $\mathbf{r}$ is the ray of the pixel, $T(t) = \exp(-\int_{t_n}^t \sigma(\mathbf{r}(s))ds)$, t_n and t_f represents the near and far bounds of the ray, $\sigma(\mathbf{r}(t))$ and $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ are the opacity and color value of the radiance field function at position $\mathbf{r}(t)$ and viewing direction $\mathbf{d}$.

However, NeRF is not without its limitations. Representing the radiance field using an opaque neural network poses significant challenges for direct scene editing, as fine-tuning is often the only practical method for modification. Furthermore, the volumetric rendering process inherent to NeRF involves sampling a substantial number of points along viewing rays. Since the radiance field is typically modeled by a Multi-Layer Perceptron (MLP), each of these sample points requires a full forward pass through the network. This leads to prohibitive computational costs in rendering and, consequently, training.

To overcome these challenges, many works have proposed to reintegrate explicit structures back into NeRF, aiming to offload the work from the neural network, making it easier to edit and faster to render. One pioneering work, NSVF (Liu et al. 2020), introduces a sparse voxel octree where each non-empty cell contains a voxel-bounded implicit field, enabling faster rendering by skipping empty regions and facilitating scene editing through its explicit voxel structure. This is followed by EG3D (Chan et al. 2021), which employs a hybrid explicit-implicit tri-plane representation, where 3D features

are explicitly stored on three axis-aligned orthogonal 2D feature planes. The density or color value at a 3D point is calculated by querying the features from the tri-planes, interpolating and aggregating the queried features, and then feeding the features to a lightweight MLP decoder. Furthermore, Instant-NGP (Müller et al. 2022) accelerates NeRF by using a multi-resolution hash grid, where trainable feature vectors are stored in multiple hash tables corresponding to different levels of detail. For 3D point queries, the feature vectors are efficiently retrieved from the hash grids, concatenated, and finally fed into a small MLP to produce the opacity or color values.

The methods above still require neural networks in their representation; a more drastic approach involves removing the neural network completely and attempting to directly represent the radiance field. This unlocks more speed gains, at the cost of using more spaces. Plenoxels (Fridovich-Keil and Yu et al. 2022) achieve this by representing the radiance field using a sparse 3D voxel grid, where each voxel stores opacity and spherical harmonic coefficients, which are then directly optimized. DVGO (Sun et al. 2022) also directly optimizes voxel grids, using one dense grid to model geometry (density) and a second feature grid combined with a shallow MLP to model view-dependent color; it achieves super-fast convergence and high-quality by employing techniques such as post-activation interpolation on the density grid and specific optimization priors. TensorRF (Chen et al. 2022) models the radiance field as a 4D tensor (a 3D voxel grid with a feature channel) which is then factorized into a set of compact low-rank tensor components (e.g., vectors and matrices using CP or a novel vector-matrix decomposition) for efficient and compact scene representation.

2.1.3 3D Gaussian Splatting (3DGS)

Following the line of work that aims to directly represent radiance field using explicit structures, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) proposes a novel scene representation that is fully explicit, with impressive quality and performance.

Departing from existing explicit representations based on grids or factorized tensors, 3D Gaussian Splatting (3DGS) proposes a novel approach to represent the radiance field using an ensemble of 3D Gaussian distributions. Each distribution is characterized by its opacity and view-dependent color information, the latter modeled using Spherical Harmonics. These Gaussian distributions can be seen as 3D ellipsoids, which are then rendered via a splatting process. This splatting process projects the 3D Gaussian ellipsoids into 2D ellipses on the image plane and composes their colors using alpha blending to synthesize the final image. Crucially, this rendering pipeline is differentiable, enabling 3D reconstruction in a manner analogous to Neural Radiance Fields (NeRF). Furthermore, the absence of neural networks and explicit grid-based structures in this process allows for significant compute and memory savings, particularly in representing empty space. Due to its novel formulation, 3DGS achieves impressive results in terms of visual quality, rendering speed, and training efficiency.

More concretely, 3DGS represents a scene using a set of 3D Gaussian distributions $\mathcal{R}$, each with mean $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance $\boldsymbol{\Sigma}$, opacity $\alpha \in \mathbb{R}$, and spherical harmonics parameters $\mathcal{C} \in \mathbb{R}^k$ (k is the degrees of freedom) for modelling view-dependent color. For regularizing optimization, the covariance matrix is further decomposed into a rotation matrix $\boldsymbol{R}$ and a scaling matrix $\boldsymbol{S}$ by $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T$. These matrices are further represented as quaternions $\boldsymbol{q} \in \mathbb{R}^4$ and scaling factor vector $\boldsymbol{s} \in \mathbb{R}^3$.

For 3DGS, view rendering is performed via point splatting (Yifan et al. 2019). Specifically, all 3D Gaussian distributions in the scene are first projected onto the 2D image plane, and their color is computed from spherical harmonic parameters. Then, for every 16x16 pixel patch of the final image, the projected Gaussians that intersect with the patch are sorted by depth. For every pixel in the patch, its color $\boldsymbol{I}_i$ is computed by alpha compositing the opacity and color of all the Gaussians covering this pixel, ordered by depth, as in Equation (2.2):

$$\boldsymbol{I}_i = \sum_{j \in \mathcal{N}_{cov}} \boldsymbol{c}_j \alpha_j \prod_{k=1}^{j-1} (1 - \alpha_k) \quad (2.2)$$

where $\mathcal{N}_{cov}$ represents the splats that cover this pixel, α_j represents the opacity of Gaussian splat $\mathcal{R}_j$ multiplied by the density of its projected 2D Gaussian distribution at the location of the pixel, and $\mathbf{c}_j$ represents the color computed from the spherical harmonics parameters of the splat.

Developing from 3DGS, this idea of using ensembles of 3D primitives to represent scenes and use the splatting process to perform rendering has received significant attention. For example, Gaussian Surfels (Dai et al. 2024) represents scenes using 2D ellipses instead of 3D Gaussians by setting their z-scale to zero, treating the direction of their z-axis as their normal vector, and employing a normal-depth consistency loss to enhance surface alignment. 2DGS (Huang et al. 2024) models scenes using 2D oriented planar Gaussian disks to better represent object surfaces, further utilizing explicit ray-splat intersection for perspective-correct splatting and incorporating depth distortion and normal consistency losses for enhanced reconstruction quality. Quad Splatting (Zhang et al. 2024) (QGS) builds on 2DGS by replacing 2D disks with quadric surfaces (specifically quadratic paraboloids), providing a second-order geometric approximation; this method defines Gaussian distributions on these non-Euclidean primitives using geodesic distances and can render spatial curvature to guide normal consistency, leading to higher-quality results.

On the other hand, many techniques have been proposed to further improve the efficiency of 3DGS. Scaffold-GS (Lu et al. 2024) introduces a hierarchical structure using anchor points from which local 3D Gaussians are spawned and their attributes dynamically predicted on-the-fly based on viewing context. This process helps eliminate redundant Gaussians and reduce per-Gaussian feature storage cost, reducing memory usage and improving render speed. Compact-3DGS (Lee et al. 2024) proposes to employ a learnable masking strategy to reduce Gaussian count, a grid-based neural field for compact representation of view-dependent color, and vector quantization with codebooks for geometric attributes, all reducing storage cost. Light-Gaussian (Fan et al. 2023) proposes to prune Gaussians based on global significance, distill Spherical Harmonics to lower degrees using pseudo-view

augmentation, and apply significance-aware vector quantization to further compress attributes.

Furthermore, a family of work proposes to extend 3DGS to explicitly model textures and materials instead of the general view-dependent colors. Relightable 3D Gaussians (Gao et al. 2023) enhances each Gaussian with normal vectors and BRDF parameters, decomposing material and lighting through physically-based differentiable rendering and employing point-based ray tracing for visibility and shadow effects. Similarly, GS-IR (Liang et al. 2023) extends 3DGS by incorporating BRDF parameters (albedo, metallic, roughness) into each Gaussian and utilizes a baking-based occlusion model for efficient indirect lighting estimation. GSTex (Rong et al. 2024) introduces per-primitive texturing for 2D Gaussian splatting, assigning an RGB texture map for spatially varying albedo to each Gaussian and retaining spherical harmonics for view-dependent effects. This explicit texture map enables albedo texture editing. BBSplat (Svitov et al. 2024) represents scenes using textured planar primitives (billboards), where each billboard has learnable RGB and alpha-map textures for representing appearance and shape, combined with spherical harmonics for view-dependent lighting effects. This formulation of explicit RGB maps for texture representation enables texture editing.

2.1.4 Summary

The landscape of 3D scene representation has evolved significantly. Traditional methods like meshes and point clouds remain ubiquitous and powerful for applications requiring direct geometric control, but they fundamentally struggle with photorealistic reconstruction and the representation of volumetric or complex material effects. The advent of Neural Radiance Fields (NeRF) marked a paradigm shift, demonstrating an unparalleled ability to capture scene appearance with high fidelity directly from images. However, NeRF’s implicit nature and reliance on slow volumetric rendering introduced critical bottlenecks in speed and editability. While numerous successors attempted to accelerate NeRF, often by reintroducing explicit data structures,

they frequently involved trade-offs in quality or memory and did not fully resolve the core performance issues.

3D Gaussian Splatting (3DGS) represents the latest and most impactful iteration of this evolution. By adopting a fully explicit, primitive-based representation, it achieves a remarkable combination of state-of-the-art visual quality and real-time rendering speeds, overcoming the primary limitations of NeRF. Given its superior performance and practicality, this thesis focuses on developing editing methodologies for the core 3DGS representation. The solutions developed here may also be generalized to the rapidly growing ecosystem of 3DGS variants, which build upon the same underlying principles, ensuring the broad relevance and impact of this work.

2.2 Scene Segmentation Methods

Existing scene understanding and segmentation approaches primarily focus on 2D images or explicit 3D representations like voxels, meshes, and point clouds. The emergence of radiance field-based scene representations, as a novel way to capture 3D scenes, presents a paradigm shift, introducing challenges that necessitate new segmentation approaches not bound by the constraints of traditional methods.

2.2.1 Traditional Representation Segmentation

For years, 3D segmentation has been dominated by methods operating on explicit geometric data. These approaches, while powerful in their respective domains, are fundamentally incompatible with implicit or hybrid representations like NeRF and 3DGS. They are architecturally bound to their input data structure and generally share limitations in data dependency and closed-world categorization.

Segmentation of point clouds, as a direct output from LiDAR and other 3D sensors, has received significant attention. Pioneering works like PointNet (Qi et al. 2016) and PointNet++ (Qi et al. 2017) established a paradigm for directly processing unstructured point sets using shared MLPs and hierarchical feature learning. Subsequent methods like KPConv (Thomas et al. 2019)

introduced true 3D convolutions for point clouds, achieving state-of-the-art results. The primary contribution of these methods was enabling deep learning on raw point data, but they require massive, meticulously annotated point cloud datasets for training and are restricted to the fixed, closed-world categories present in those datasets.

Voxel grids offer a structured, 3D analogue to 2D image pixels, allowing for the direct application of 3D convolutional neural networks (CNNs). Early methods like VoxNet (Maturana and Scherer 2015) demonstrated the efficacy of 3D CNNs for object recognition. For segmentation, the main challenge is the cubic growth in memory and computation with resolution. This led to the development of efficient, sparse convolution techniques, such as Submanifold Sparse Convolutional Networks (Graham et al. 2017), which only compute on non-empty voxels. While these approaches successfully leverage the powerful and well-understood CNN framework, they are still constrained by the high memory footprint for detailed scenes and, like point cloud methods, are not easily adaptable to novel representations or open-world semantic categories.

Finally, meshes provide a representation that captures surface topology explicitly through vertices, edges, and faces. This structure is widely adopted in computer graphics and modeling. Methods like MeshCNN (Hanocka et al. 2019) developed specialized convolution and pooling operations that operate directly on the mesh edges. Other works, like ShapePFCN (Kalogerakis et al. 2016), use projective rendering to leverage 2D CNNs. The contribution of these methods is their ability to respect the underlying surface topology, but they are specialized for this representation and often struggle with the noisy, incomplete, and non-manifold meshes that result from real-world 3D reconstruction, further limiting their general applicability.

In summary, these traditional approaches are highly specialized. Their reliance on large-scale, representation-specific datasets for supervised training makes them inflexible. They cannot be applied to the new family of radiance-field representations without a costly and often information-degrading conversion step, and their closed-world design prevents them from segmenting the vast diversity of object categories found in the wild. This inflexibility

motivates the development of new segmentation techniques tailored to the unique properties of NeRF and 3DGS.

2.2.2 NeRF Segmentation

A straightforward way to perform NeRF segmentation is to convert it to an explicit 3D representation and then segment it. NeSF (Vora et al. 2022) does this by transforming NeRF’s implicit density field into an explicit density voxel grid, which is then fed into a 3D UNet to create a feature grid. Combined with a decoder, this feature grid can be seen as a segmentation label field, which can be rendered into 2D segmentation maps. While this approach is intuitive, it has two main drawbacks: First, converting to voxel grids and using 3D UNets requires substantial memory. Second, this classification approach is scenario-specific, meaning the 3D UNet has to be retrained to adapt to new object categories or new types of scenes.

To avoid the memory cost of explicit 3D grids, other methods extend NeRF by incorporating semantic information directly into it, making it represent not only radiance fields, but also semantic label fields. First introduced by Semantic-NeRF (Zhi et al. 2021a), this approach adds a semantic branch to the network and trains it using 2D semantic maps. Panoptic NeRF (Fu et al. 2022) enhanced this approach for autonomous driving by utilizing both 2D segmentation maps and coarse 3D annotations in autonomous driving datasets. JacobiNeRF (Xu et al. 2023) proposed a regularization scheme during training to align the Jacobians of highly correlated scene components, aiming to maximize their mutual information when the scene is randomly perturbed. This design allows it to effectively propagate sparse semantic information, such as segmentation labels.

This family of methods has been demonstrated to achieve consistent semantic labels across different views and can effectively combine multi-view semantic information while being resilient to noise in training data. By relying on external image-based models for segmentation, this approach can leverage existing 2D segmentation models’ capabilities, especially open-world segmentation in recent models such as Segment Anything (Kirillov et al. 2023),

minimizing the need for additional adaptation when encountering new object categories.

However, while this paradigm is effective for semantic segmentation, it has limitations for instance or panoptic segmentation. The off-the-shelf 2D segmentation models do not guarantee the same instance indices across different viewpoints, leading to inconsistency across viewpoints.

Numerous methods have been proposed to tackle the problem of re-establishing cross-view consistent identity, especially when using standard multiview inconsistent 2D panoptic segmentation networks. Panoptic Lifting (Siddiqui et al. 2023) matches 3D identifiers with 2D instances using linear assignment. Contrastive Lift (Bhalgat et al. 2023) proposes to model a low-dimensional pixel similarity field instead of directly modeling the instance segmentation field. Segmentation is then done using clustering over said field. This approach is more efficient and scales to large number of objects (up to 500 per scene). PCFF (Cheng et al. 2023) learns a feature field for scene decomposition and employs metric learning to constrain the distances among rendered panoptic features of pixels in every view, removing the need for consistent identity labels across views. Finally, Instance-NeRF (Liu et al. 2023b) proposes to match 2D masks rendered from 3D masks created by a NeRF-RCNN against multiview inconsistent 2D panoptic segmentation maps produced by image segmentation models, leading to cross-view consistency of labels. The challenge in this approach is that the features are more complicated than those in Semantic-NeRF approaches, as they only encode the segmentation result. In contrast, this approach must encode a full implicit feature vector used for segmentation.

Apart from directly embedding segmentation labels into NeRF, another family of methods focuses on distilling features from existing 2D vision models into NeRFs, and leveraging them for segmentation. Works like DFF (Kobayashi et al. 2022), N3F (Tschernezki et al. 2022), and FeatureNeRF (Ye et al. 2023a) propose to distill features from pre-trained 2D vision models like LSeg (Li et al. 2022), SAM (Kirillov et al. 2023), and DINO (Caron et al. 2021) into NeRF. These learned feature fields can then be used for

downstream applications such as language-guided editing, scene object removal, or segmentation. For instance, 3D-OVS (Liu et al. 2023a) shows that aligning class relevancy distribution against off-the-shelf 2D vision models can lead to precise annotation-free segmentations.

However, a key limitation of this approach is the inherent challenge of lifting 2D features, which are optimized for 2D image planes, into a coherent 3D space. This dimensional misalignment can result in artifacts and challenges in 3D segmentation.

Finally, a branch of NeRF segmentation work focuses on interactive segmentation, namely achieving full 3D object segmentation from sparse user inputs. For example, iLabel (Zhi et al. 2021b) integrates the propagation of semantic labels with an online NeRF SLAM system. This enables coherent segmentation of 3D objects using minimal user click annotations. Baking in the Feature (Blomqvist et al. 2023) and ISRF (Goel et al. 2023) both propose to distill 2D vision model features into extended NeRFs, and utilize the learned feature field for interactive segmentation based on user click inputs. However, distilling 2D vision model features to perform segmentation requires significant preprocessing time for feature training. NVOS (Ren et al. 2022) trains a 3D segmentation network supervised by partial user scribbles to classify voxels as foreground or background. Other methods like SA3D (Cen et al. 2023b) and NOC (Wei et al. 2023) adapt the Segment Anything Model (SAM) (Kirillov et al. 2023) for 3D object extraction. These approaches iteratively build a 3D mask from user prompts on a single view by utilizing mask inverse rendering and cross-view self-prompting to propagate the prompt across views. These strategies achieve better labeling accuracy than standard pre-trained semantic segmentation methods. Building on this, SGISRF (Tang et al. 2023) requires even fewer annotations (single click in a single view) to specify segmentation by employing cross-dimension guidance propagation and concealment-revealed supervised learning schemes. However, these prompt-propagation methods are limited by their dependence on significant cross-view overlap and a specific view-processing order.

2.2.3 3DGS Segmentation

3D Gaussian Splatting (3DGS) provides a faster, higher-quality, and more explicit scene representation than NeRF, leading to increased interest in achieving editing and thus 3D segmentation on it. Current research often adapts NeRF segmentation methods to 3DGS, while also designing to leverage the speed and explicit formulation benefits of 3DGS.

Since 3DGS uses collections of 3D Gaussian ellipsoids to represent scenes, the most straightforward way to perform segmentation is to assign a segmentation label to each Gaussian, usually guided by 2D segmentation masks from 2D image segmentation models. For example, FlashSplat (Shen et al. 2024b) formulates 3D segmentation label assignment as a linear programming (LP) optimization. Capitalizing on the linearity of mask rendering, it derives a globally optimal closed-form solution for assigning labels to Gaussians. It also incorporates a background bias to improve robustness against noisy 2D masks. Lifting by Gaussians (LBG) (Chacko et al. 2025) proposes to lift 2D SAM masks by assigning segmentation masks to the per-pixel maximum-contributor Gaussian. It then uses an incremental merging strategy based on geometric and semantic overlap to combine per-frame 3D object fragments into coherent scene-level objects. Additionally, SAGD (Hu et al. 2024) enhances segmentation boundaries in 3DGS via a novel Gaussian Decomposition scheme within a training-free mask lifting pipeline. However, these approaches focus on segmenting a single object at a time, making them inefficient for multi-class or multi-object tasks, as the entire pipeline must be re-executed for each new target.

Alternatively, instead of segmentation labels, another family of method embeds semantic features into each Gaussians. Segment Any Gaussians (SAGA) (Cen et al. 2023a) attaches scale-gated affinity features to the Gaussians, using scale-aware contrastive learning to capture SAM’s multi-granularity segmentation abilities. A soft scale gate mechanism adapts feature channel magnitudes based on a specified 3D scale, enabling multi-granularity segmentation. Gaussian Grouping (Ye et al. 2023b) extends 3DGS by augmenting each Gaussian with a learnable identity encoding, supervised by SAM masks

and 3D spatial regularization, enabling detailed segmentation and editing. Contrastive Gaussian Clustering (Silva et al. 2024) learns a consistent 3D feature field from inconsistent 2D masks using contrastive learning and spatial regularization for weakly supervised segmentation. CoSegGaussians (Dou et al. 2024) learns compact 3D Gaussian segmentation fields using an efficient feature unprojection/fusion module and a novel loss for robustness against noisy zero-shot labels. Unified-Lift (Zhu et al. 2025) presents an end-to-end object-aware lifting approach. It augments Gaussians with feature vectors and introduces a learnable object-level codebook, associating the two via an object-Gaussians association mechanism. It employs dedicated association learning and noisy label filtering modules for robust codebook optimization from 2D masks. This codebook can then be used to segment rendered Gaussian feature maps. Like their NeRF-based counterparts, these methods are often constrained by a time-consuming feature distillation phase, which is necessary to learn the semantic features for segmentation.

Finally, there is a branch of work focusing on interactive 3DGS segmentation, where users can select specific objects through clicks, scribbles, or text prompts. Click-Gaussian (Choi et al. 2024) achieves interactive segmentation by learning two-level granularity feature fields. The feature field is guided by globally aggregated feature candidates for enhanced consistency. While this enhances consistency by guiding the feature field with globally aggregated candidates, the necessary training process makes the process too slow for single object selection. GaussianCut (Jain et al. 2024) enables interactive multi-view segmentation by constructing a graph from scene Gaussians and partitioning it using graph cuts guided by user input and scene properties. SAMa (Fischer et al. 2024) enables material-aware selection and segmentation by fine-tuning a video segmentation model (Segment Anything 2 (Ravi et al. 2024)) and projecting similarities onto a queryable 3D point cloud. These methods achieve higher-quality and faster segmentation at the cost of forgoing any reusable preprocessing steps, requiring the entire process to be rerun for each new selection.

2.2.4 Summary

Traditional 3D segmentation methods are ill-suited for modern radiance field-based scene representations, as they operate in closed world categories, require substantial annotated data of the representation to segment, and are architecturally incompatible with these representations.

This has spurred new paradigms for NeRF and 3DGS segmentation, primarily revolving around two strategies: embedding a direct segmentation score or distilling complex features from 2D vision models. While feature distillation is powerful, it demands significant upfront time and memory. In contrast, embedding a segmentation score is more efficient and directly aligns with the single object segmentation task. For its efficiency and directness, this thesis adopts the approach of embedding a segmentation score.

The trade-off between pre-computation and on-demand processing is also central to interactive segmentation. One approach involves pre-computing a rich 3D feature field, which is slow initially but enables rapid segmentation of multiple objects afterward. The alternative, cross-view propagation, directly generates a mask on the fly from user input, making it faster and higher-quality for single-object selection but less efficient for segmenting many objects. Given the focus on responsive, single-object manipulation, this thesis focuses on the cross-view propagation strategy.

2.3 Scene Editing Methods

As a novel approach, radiance field-based scene representation is able to reconstruct and represent 3D scenes in impressive quality. Therefore, it has significant potential for downstream applications such as animation, virtual reality, and augmented reality. To make them practical for these applications, it is crucial to enable users to freely edit the captured scenes.

2.3.1 Traditional Editing Methods

Traditionally, the manipulation of 3D content has been dominated by techniques built for explicit geometric representations. While various tools have

been developed for direct point cloud processing (Zwicker et al. 2002) and voxel grid editing (Wang and Kaufman 1995), the vast majority of traditional 3D editing research and workflows have centered on polygonal meshes. For meshes, a rich set of manipulation techniques has been established, with four prominent approaches being skeleton skinning, blendshapes, cage-based deformation, and As-Rigid-As-Possible (ARAP) deformation.

Skeleton skinning is perhaps the most ubiquitous method, especially in character animation. It offers an intuitive workflow where an artist manipulates an underlying skeleton, and the deformation is propagated to the mesh surface (Lewis et al. 2000). Its primary challenge, however, lies in the non-trivial task of assigning appropriate skinning weights to vertices, which often requires significant manual effort or complex automatic solutions (Jacobson et al. 2011).

Blendshapes, or morph targets, enable high-quality, artist-driven deformations by interpolating between pre-defined target shapes. This approach is ideal for detailed facial expressions or corrective shapes but is inherently limited to the span of the pre-authored models (Lewis et al. 2000).

For free-form surface manipulation, cage-based deformation and ARAP are foundational. Cage-based deformation (CBD) provides a simple and exceptionally fast approach, where manipulating a coarse, simple mesh (the "cage") intuitively deforms the complex interior mesh (Sederberg and Parry 1986). The core principle relies on a set of generalized barycentric coordinates, known as cage coordinates, which define the influence of the cage's vertices on the interior points. Numerous coordinate systems have been proposed, including the widely-used Mean Value Coordinates (MVC) (Floater 2003), Harmonic Coordinates (HC) (Joshi et al. 2007), and Green Coordinates (GC) (Lipman et al. 2008), all enabling robust and intuitive free-form manipulation. In contrast, As-Rigid-As-Possible (ARAP) deformation (Sorkine-Hornung and Alexa 2007b) focuses on preserving the local rigidity of the surface, yielding natural-looking results by minimizing an elastic energy. Among these, cage-based deformation is often favored for its simplicity, making it a powerful tool for interactive editing.

These powerful and mature techniques, however, are fundamentally designed for explicit representations. They are not directly applicable to modern radiance-field-based representations like NeRF or the cloud of anisotropic primitives in 3D Gaussian Splatting, which lack a coherent and explicit surface structure. Therefore, a significant research challenge lies in bridging these classical manipulation methods with novel scene representations to unlock their full creative potential.

2.3.2 NeRF Editing

There has been a lot of literature on scene editing based on radiance field representations, including geometry editing (Peng et al. 2022, Xu and Harada 2022, Yuan et al. 2022) and object-centric editing (Liu et al. 2021, Yang et al. 2021).

To achieve editing, they employ many sophisticated techniques. One common strategy involves fine-tuning the underlying neural network based on user edits. For instance, EditNeRF (Liu et al. 2021) enables local editing on category-level radiance fields by propagating sparse 2D user scribbles into 3D and selectively fine-tuning modular network components. Another set of approaches focuses on deforming the rendering rays to achieve the desired geometric changes. Methods like CageNeRF (Peng et al. 2022), DeformingNeRF (Xu and Harada 2022), and NeRFShop (Jambon et al. 2023) have successfully adapted cage-based deformation for radiance fields. These approaches work by using the cage to define a deformation field that warps the 3D sample points along rendering rays. To create the cage itself, these methods often first extract a mesh from the NeRF’s density field via marching cubes, which is then simplified into a coarse control cage (Jambon et al. 2023, Xu and Harada 2022). While effective for NeRF, this ray-warping strategy is fundamentally incompatible with the rasterization-based pipeline of 3DGS. Other methods, like NeRF-Editing (Yuan et al. 2022), establish an explicit correspondence between the radiance field and an extracted mesh, allowing users to edit via standard mesh deformation, which is then transferred back to deform the NeRF’s rendering rays via a tetrahedral mesh

proxy. A third direction involves designing inherently editable or compositional representations from the outset. Yang et al. (Yang et al. 2021) learns a disentangled representation with separate branches for background scene elements and individual objects (conditioned on activation codes), enabling object manipulation like moving, duplicating, or removing within the scene.

However, most works prefer neural networks to represent radiance fields, and the implicit nature of this representation usually requires time-consuming fine-tuning for edit operations. As a result, these methods cannot support practical interactive editing use cases.

To tackle this challenge, another attempt is to represent radiance fields using explicit structures, including Control-NeRF(Lazova et al. 2022) for volumetric grid, PAPR(Zhang* et al. 2023), NeuralEditor(Chen et al. 2023a) and RIP-NeRF(Wang et al. 2023) for point clouds, NeuMesh(Bao and Yang et al. 2022) and Differentiable Blocks World(Monnier et al. 2023) for meshes. The foundation of the employed method, 3D Gaussian Splatting (Kerbl et al. 2023), also falls into this category.

The current body of work falls short in adequately supporting practical interactions, facing two challenges: intuitive 3D segmentation for object selection and inpainting of exposed regions or holes resulting from editing, which is also the focus of this thesis.

2.3.3 3DGS Editing

Many methods have been proposed to edit 3D Gaussian Splatting (3DGS) captures. There are high-level, text-prompt-based editing methods and other lower-level, more explicit editing methods.

Several methods have been proposed for high-level, text-prompt-based editing of 3D Gaussian Splatting (3DGS) captures. GaussianEditor (Chen et al. 2023b) performs text-driven editing by fine-tuning the 3DGS model, employing 2D diffusion models such as Stable Diffusion (Rombach et al. 2021) for guidance. It also introduces Hierarchical Gaussian Splatting (HGS) to improve robustness against the noisy and multiview-inconsistent guidance from these diffusion models. Another approach, VcEdit (Wang et al. 2024),

focuses on resolving multi-view inconsistencies within the guidance images from 2D diffusion models. Finally, GaussCtrl (Wu et al. 2024) explicitly enforces multi-view consistency during editing by leveraging depth-conditioned guidance via ControlNet (Zhang et al. 2023) and employing attention-based latent code alignment across views. However, these text-based approaches often lack the precision required for professional editing, largely due to the inherent ambiguity of natural language.

To enable low-level, explicit editing, one effective approach is binding the Gaussian distributions in 3DGS to a mesh surface. Deforming 3DGS is then achieved by deforming the proxy mesh. SuGaR (Guédon and Lepetit 2023) pioneered this approach by proposing an algorithm that extracts mesh from 3DGS, along with training regularizations to improve the quality of the extracted mesh. GaussianFrosting (Guédon and Lepetit 2024) builds on it by proposing a more flexible way to bind distributions to the mesh. Both GaMeS (Waczyńska et al. 2024) and Gao et al. (Gao et al. 2024a) start with a provided initial mesh and train their models from there. While Mani-GS (Gao et al. 2024b) does not need an initial mesh, it first trains a 3DGS or NeuS (Wang et al. 2021) model to create one, then uses this mesh as a base of its mesh-bounded Gaussian model.

While mesh-bounding methods can be effective, they have a significant drawback: they modify the core 3DGS architecture to handle deformation, requiring costly retraining, and even the initial mesh provided to them. This limitation makes it challenging to use these methods (Guédon and Lepetit 2023 2024, Waczyńska et al. 2024, Gao et al. 2024a b) for modifying existing trained 3DGS scenes or extending them to new variants of the standard 3DGS.

Many work have also explored physics-based simulation to manipulate 3DGS. PhysGaussian (Xie et al. 2023) uses the Material Point Method (MPM) (Jiang et al. 2016) to model object deformation from touch and push interactions on 3DGS. Gaussian Splashing (Feng et al. 2024) integrates position-based dynamics (PBD) (Macklin et al. 2016) with 3DGS for simulations. VR-GS (Jiang et al. 2024) proposes a deformation approach similar to this thesis but relies on tetrahedral mesh grids with a large number of vertices.

While this approach is designed for simulation, it is less suited for direct manual manipulation, which is the focus of this thesis.

Previous work has also explored direct and manual manipulation of vanilla 3DGS models. SC-GS (Huang et al. 2023) enables Gaussian deformation by mapping control point transformations to Gaussians, but it requires video data for learning these mappings, limiting its applicability. Additionally, its control point-based formulation makes it challenging to adjust the scale of Gaussians. D3GA (Zielonka et al. 2023) achieves cage-based deformation on 3DGS but relies on tetrahedral mesh cages and is limited to 3DGS human bodies and garments.

Recent work ARAP-GS (Tong et al. 2023) proposes a 3D Gaussian deformation method that adapts the As-Rigid-As-Possible deformation approach and resolves spike-like artifacts by aligning deformed Gaussians with their corresponding radiance fields. However, this method is ineffective in handling extreme deformations, such as the Gaussian highlighted in the ablation of deformation algorithm in Chapter 4, which shows the bending of a single Gaussian.

2.3.4 Summary

In summary, the landscape of 3D scene editing reveals a critical disconnect. On one hand, traditional mesh editing offers a powerful and mature suite of tools. However, these methods are incompatible with novel representations like 3DGS. Adapting them is a promising direction, with cage-based deformation standing out as a prime candidate due to its simplicity, speed, and straightforward integration with ubiquitous mesh modeling workflows. On the other hand, editing methods for radiance fields have largely focused on NeRF, which, being based on neural networks for encoding fields, suffers from slow optimization and rendering times. While adapting cage-based deformation to warp rendering rays was an effective strategy for NeRF, this mechanism is irrelevant to the rasterization-based 3DGS pipeline.

Within the rapidly growing field of 3DGS editing, current approaches present significant trade-offs. High-level, text-based methods lack the pre-

cision and direct control required for many creative applications due to the inherent ambiguity of language. Low-level methods provide more control but come with their own drawbacks. Mesh-backed approaches require either a pre-existing mesh or costly retraining of a modified architecture, limiting their applicability for 3DGS variants. Physics simulation methods are tailored for physically plausible effects rather than direct, artist-driven manipulation. This analysis reveals a clear and crucial gap in the literature: the absence of a method that enables real-time, low-level editing of vanilla 3DGS captures without requiring architectural modifications, costly retraining and generalizes to variants of 3DGS. This thesis is focused on addressing this gap.

2.4 Scene Inpainting Methods

Finally, object removal or inpainting in radiance field-based scene representations is vital for effective scene recomposition, as the original object might be moved, and the exposed regions or holes caused by movement need to be filled. Another application scenario is to remove undesired content from existing captures for aesthetic or privacy reasons.

2.4.1 Traditional Inpainting Methods

Traditionally, 3D inpainting and completion efforts have centered on explicit geometric representations, primarily polygonal meshes and point clouds. For meshes, these methods can be broadly categorized into three families. The first formulates inpainting as an optimization problem, seeking a mathematically reasonable surface by solving a Partial Differential Equation (PDE) over the hole region, such as using the Poisson equation (Zhao et al. 2007) or minimizing the Total Variation of the surface normals to preserve sharp features (Baumgärtner et al. 2020). The second family reframes the problem using analogies from other fields, such as treating the mesh as a mechanical network of bars and nodes and simulating its equilibrium (Pernot et al. 2006), or assuming the local geometry is low-rank and solving for missing vertices via matrix completion (Arvanitis et al. 2018). The third family uses data-driven, example-based approaches to copy and recombine geometric patches

from the valid parts of the mesh itself to synthesize a coherent fill (Harary et al. 2014).

Similarly, traditional point cloud completion methods are also fundamentally surface-oriented. Early techniques relied on geometric cues like symmetry and regularity to infer missing structures (Rumelshak et al. 2021), while more recent learning-based approaches train on large datasets of 3D shapes (e.g., ShapeNet (Chang et al. 2015)) to learn a prior for completion (Yuan et al. 2018). However, these learning-based methods operate in a "closed world," as their generative capability is confined to the specific object categories and geometric patterns present in their 3D training corpus.

A fundamental limitation unites these traditional approaches: they are intrinsically designed for explicit, surface-based representations, operating directly on entities like vertices, faces, and surface normals. They rely entirely on the local geometry of the input object or on curated 3D datasets, which can be insufficient for complex or novel structures. Crucially, they do not leverage the vast, multimodal knowledge from real-world 2D image datasets that modern generative models use for semantic understanding. This inherent reliance on an explicit surface topology and their "closed-world" data dependency makes them fundamentally incompatible with modern, volumetric, and unstructured radiance-field-based scene representations like 3D Gaussian Splatting.

2.4.2 Radiance Field Inpainting Methods

For Neural Radiance Fields (NeRFs), several approaches have emerged to tackle the challenge of 3D object removal and inpainting. NeRF-In (Shen et al. 2024a) presents a free-form inpainting framework, where a user-drawn mask on a single rendered view is first propagated to a series of sampled views guided by a video object segmentation model. The masked regions are then inpainted by 2D inpainting models to generate the guiding color and depth images, which are used to optimize the NeRF parameters for object removal or retouching. Similarly, SPIn-NeRF (Mirzaei et al. 2023b) initiates its process with multiview segmentation derived from sparse 2D user annotations.

It first employs an interactive segmentation model and a video instance segmentation model to generate initial multiview masks. A semantic NeRF is then trained on the initial masks to produce multiview-consistent segmentation masks. For inpainting, it employs 2D inpainting models to inpaint the original RGB and depth maps, and employs a special multiview patch-wise perceptual loss in masked regions to achieve multiview-consistent 3D object removal. Building upon SPIn-NeRF, OR-NeRF (Yin et al. 2023) enhances segmentation efficiency by propagating sparse user annotations (points or text) from a single view to all views via 3D reprojection, and employing Segment Anything Model (Kirillov et al. 2023) for mask generation. Following this, 2D inpainted RGB and depth maps are created. The NeRF representation is then fine-tuned for object removal using these inpainted maps, supervised by RGB, depth, and multiview patch-level perceptual losses. A key innovation in these works is the use of a multiview perceptual loss to encourage 3D consistency, cleverly addressing artifacts that might arise from the multiview inconsistencies of 2D inpainting models. Despite this, a core limitation remains: these methods directly optimize the NeRF to fit pre-generated 2D inpainted images. While more explicit representations, like the 3D Gaussian Splatting (3DGS) adopted in this thesis, enables additional pruning and initialization opportunities that can improve quality and speed.

Apart from utilizing perceptual losses to achieve 3D consistency, reference-guided controllable inpainting (Mirzaei et al. 2023a) leverages a single user-provided inpainted reference view to ensure consistency and controllability. This method employs monocular depth estimation and bilateral solvers to reconstruct view-dependent effects and fill disoccluded regions in non-reference views, thereby offering detailed control over the final inpainted scene. However, its reliance on a single viewpoint means it can struggle with scenes involving significant camera motion, where large disoccluded regions may be revealed that cannot be supervised by the reference view, potentially leading to artifacts. The approach proposed by Weder et al. (Weder et al. 2023) introduces a novel confidence-score-based scheme. This iterative process filters view-inconsistent 2D inpainted images from the optimization based on their confidence scores and retrain the model, continually updating these scores

to improve overall 3D consistency. A key limitation of this approach is that discarding views can lead to data sparsity, resulting in insufficient geometric constraints for the NeRF optimization and potentially causing holes or degraded quality in areas of the scene that were only visible in the discarded frames.

Concurrent with the development of this thesis, several pioneering methods have begun to address inpainting for 3D Gaussian Splatting (3DGS). InFusion (Liu et al. 2024), for instance, proposes to inpaint 3DGS by fine-tuning it on inpainted 2D images. However, it focuses on guiding the initialization of inpainted Gaussians by restoring the depth map of a reference view using a specially trained image-conditioned depth completion model, then unprojecting these points into the scene. VISTA (Cui et al. 2025) employs a diffusion model refined by scene concept learning to inpaint masked regions in rendered 2D images and proposes a method that measures the visibility uncertainties of 3D points across different input views to guide the 3D fine-tuning process. RefFusion (Mirzaei et al. 2024) introduces a 3D inpainting method based on an image inpainting diffusion model personalized on a given reference view, effectively adapting the prior distribution for more effective score distillation sampling, which is used to supervise the scene representation to fill missing scene contents. These concurrent efforts demonstrate a clear trend towards personalizing diffusion models to integrate powerful generative priors, yielding superior quality and control. This benefit, however, comes at the cost of increased complexity and potential instabilities from sensitive optimization techniques, such as Score Distillation Sampling (Poole et al. 2022).

2.4.3 Summary

The review of existing work highlights a significant challenge: traditional 3D inpainting and completion methods are incompatible with novel volumetric primitives employed by novel radiance-field based scene representations, such as 3D Gaussian Splatting. These classic techniques are designed for surface-oriented representations (e.g., meshes, point clouds), and their learning-based variants are often closed-world, requiring large datasets of

a specific representation that do not yet exist for 3DGS. In contrast, an effective and pragmatic alternative has emerged from the radiance field literature: leveraging powerful, pre-trained 2D inpainting models and enforcing 3D consistency through a multiview perceptual loss, as demonstrated by SPIn-NeRF. This approach is simple, fast, and circumvents the need for 3D training data.

This thesis focuses on adapting and advancing this 2D-guided inpainting paradigm specifically for 3DGS. This direction is particularly promising because the explicit, point-like nature of Gaussians offers unique opportunities for geometric control, such as direct pruning and intelligent initialization that are unavailable in implicit NeRF representations. It is noteworthy that concurrent research has also begun to leverage personalized diffusion models, achieving superior generative quality. However, these methods often suffer from instability and complexity due to sensitive optimization techniques like Score Distillation Sampling. Therefore, this thesis build upon the more robust 2D-guided framework, viewing the integration of advanced diffusion priors as a valuable direction for future investigation.

Chapter 3

Comprehensive 3DGS Segmentation for Manipulation

3.1 Introduction

In this chapter, a novel method is presented for achieving intuitive object selection and 3D semantic segmentation on 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023).

As illustrated in Figure 3.1, the proposed method generates a comprehensive 3D segmentation of a target object from a trained 3DGS scene representation, initiated by only a few user point clicks from a single viewpoint. This is achieved by integrating the capabilities of state-of-the-art 2D segmentation models with a novel cross-view prompt propagation process. To ensure a complete segmentation that includes all constituent Gaussians, a two-stage design is introduced that explicitly models both the target object and the background. A final merging step then produces a comprehensive result. This fusion and initialization process is uniquely enabled by the explicit, point cloud-like formulation of 3DGS, which allows the method to perform segmentation and refinement directly on the 3D primitives. This explicit structure also provides a critical foundation for the efficient inpainting and manipulation techniques detailed in subsequent chapters.

The effectiveness of the proposed method is demonstrated by segmenting objects in 360 and forward-facing scenes. The method is also benchmarked against existing methods, achieving competitive or even superior

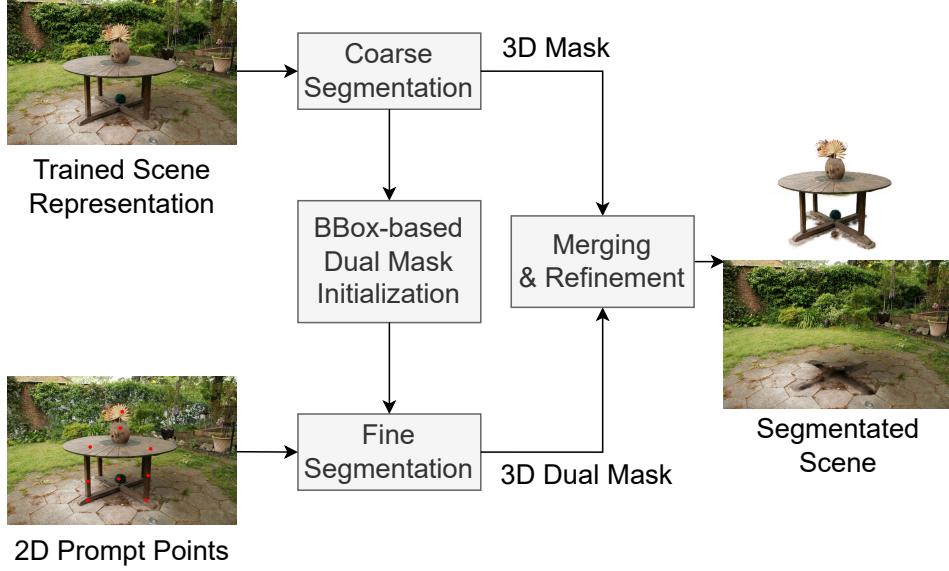


Figure 3.1: Overview of the proposed approach. The proposed process produces a 3D 3DGS scene segmentation from 2D user prompts.

results. Crucially, this segmentation approach is designed not as an end in itself, but as the foundational first step in a complete manipulation workflow. Before delving into the technical methodology, the following section will situate this segmentation stage within the broader, three-part editing pipeline that motivates this work, clarifying how it enables the subsequent stages of object manipulation and recomposition.

In summary, the specific contributions of this chapter are as follows:

- A novel technique is presented, designed for intuitive scene object segmentation within the framework of 3D Gaussian Splatting (3DGS).
- For intuitive selection, a dual-stage self-prompting mask propagation process is proposed that produces high-quality 3D semantic segmentation masks from 2D image prompt points.
- Evaluations are performed on the method, demonstrating its performance is comparable, if not superior to existing models, and is designed for downstream manipulations rather than just segmentation.

3.2 Pipeline Overview

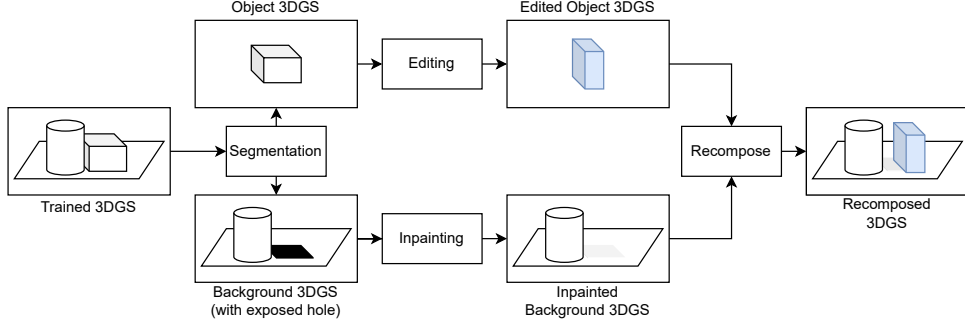


Figure 3.2: An overview of the proposed three-stage editing pipeline for 3DGS scenes.

The segmentation method detailed in this chapter is the foundational first step of the integrated editing pipeline proposed by this thesis. To clarify its role and importance, this full three-stage workflow is now outlined (Figure 3.2). While subsequent chapters will address manipulation and re-composition, the focus here is on achieving a robust and comprehensive segmentation, which is the critical prerequisite for any downstream editing. The three stages are as follows:

1. **Intuitive Object Segmentation:** An object is first segmented from the 3D scene. This process is initiated by simple 2D point prompts provided by the user on a single viewpoint. The method developed for this stage lifts these 2D cues to a complete 3D segmentation, separating the 3DGS scene into distinct foreground (object) and background representations.
2. **Interactive Object Manipulation:** The segmented foreground object is then manipulated. A novel method is introduced to bridge traditional, intuitive cage-based deformation techniques with the novel volumetric primitives of 3DGS. This allows for real-time, free-form editing and animation of the captured object.

3. **Scene Inpainting and Recomposition:** After the object is edited or moved, the newly exposed regions of the background are filled. A rapid inpainting method is proposed that leverages 2D vision models to generate high-quality, multi-view consistent results. Finally, the edited object is recomposed into the inpainted scene.

The decision to adopt this modular, three-stage approach, rather than pursuing a single end-to-end model, is deliberate. An end-to-end learning approach would necessitate a large-scale dataset of 3DGS captures, which is not currently available. While alternative end-to-end text prompt based editing approaches have been employed for editing, they often suffer from instability due to the optimization procedure employed and lack the precision required for fine-grained artistic control.

The staged pipeline, in contrast, allows each sub-problem to be tackled by strategically integrating established techniques with novel solutions tailored to 3DGS. As established in the Literature Review, traditional 3D methods for segmentation, deformation, and inpainting are designed for surface-based representations and are incompatible with the volumetric primitives of 3DGS. Therefore, each stage of the pipeline necessitates a novel contribution. This thesis draws upon recent advances in the NeRF and 2D computer vision communities, adapting and extending them to the unique architectural properties of 3DGS to achieve a robust and high-quality editing workflow. The technical details of each stage are presented in the subsequent sections and chapters.

3.3 Methodology

The segmentation stage starts with a trained 3D Gaussian Splatting model (Kerbl et al. 2023), which is denoted as $\mathcal{R}$. A collection of cameras $\mathcal{C}$ that view this scene is also provided. Additionally, there are some 2D point annotations $\mathcal{P}$ that select an object in an image rendered from $\mathcal{R}$ at a specific camera pose $\mathbf{C}_0 \in \mathcal{C}$. The goal is to divide the scene into two parts: the desired object ($\mathcal{R}_{\text{obj}}$) and everything else in the background ($\mathcal{R}_{\text{bg}}$).

To achieve this, a dual-stage segmentation method shown in Figure 3.1 is employed. The method augments each 3D Gaussian with a trainable mask score. The coarse stage trains this score by distilling masks from a 2D segmentation model in a self-prompting loop: render the current 3D mask, derive 2D prompt points, and generate a pseudo-ground truth mask for supervision. Following the coarse stage, the fine stage introduces and trains a dual mask score to explicitly model the background. Jointly optimizing both scores refines the final 3D segmentation of the target object.

After the fine stage, the final segmentation result is obtained by combining both the masks and the dual masks. The combination strategy is designed to include Gaussians that are not visible from render, which is likely object internals.

3.3.1 Coarse Stage Segmentation

The core of the proposed 3D segmentation approach builds upon the cross-view self-prompting process proposed by SA3D (Cen et al. 2023b).

The first stage of the proposed segmentation process is shown in Figure 3.3. It starts by augmenting 3DGS for segmentation by adding a new feature to each 3D Gaussian $\mathcal{R}_i$: a zero-initialized mask confidence score $m_{\mathcal{R}_i}$. Unlike SA3D (Cen et al. 2023b), which uses a grid-based approach, the proposed approach works directly with the 3D Gaussians since they are the basic building blocks in 3DGS. This means the approach can simply classify each Gaussian as either foreground or background to achieve segmentation.

Under this design, the new score attributes can be rendered into 2D masks by treating the score as color then render using the splatting process presented in Equation (2.2). Using this method, instead of the dense grid approach, leads to lower memory usage and reduced computational costs. This improvement comes from eliminating the need for a dense grid to store the segmentation scores and removing the extra queries needed during rendering to retrieve the score for 3D Gaussians.

Then, starting from the camera with initial prompt points, the algorithm iterates through every camera to train the score attribute.

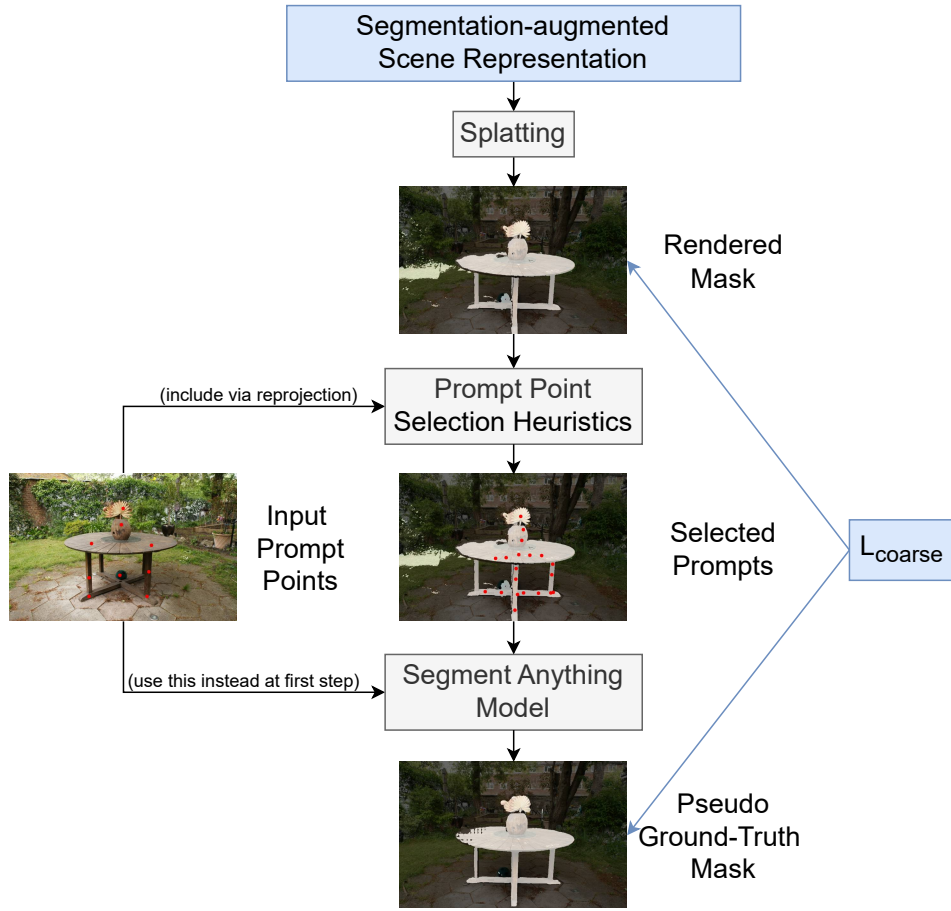


Figure 3.3: The coarse stage of the proposed segmentation algorithm.

Following SA3D (Cen et al. 2023b), each training step starts by rendering images and segmentation score map from a chosen camera pose using the augmented representation, passing them through a heuristic algorithm to extract prompt points, and feed the prompts as well as the rendered image to Segment Anything Model (SAM)(Kirillov et al. 2023) to produce a more accurate pseudo ground truth mask. The pseudo-ground truth is then compared to the rendered mask for loss. In the first iteration, the initial view $\mathbf{C}_0$ and the input prompt points $\mathcal{P}$ are used instead of the heuristically selected points.

This propagation process leverages the overlap between camera views to gradually expand and refine the 3D segmentation. After initial scores are trained into the scene representation from the starting viewpoint, the method moves to a subsequent view with significant overlap. In this new view, a selection heuristic identifies points on the object that were also visible in the previous frame based on the mask score map rendered on the new view. These points then serve as prompts for the Segment Anything Model to reselect the entire object again at the new view, effectively propagating the selection across different viewpoints.

The selection heuristic first chooses the point with the highest rendered score. To guide subsequent choices, the algorithm then modifies the score map by down-weighting points far from the new selection and forbidding the selection of points in its immediate vicinity. This strategy balances two objectives: it prioritizes points that are most likely part of the target object (those with the highest scores) while ensuring that new prompts remain reasonably close to existing ones, so they are more likely to be on the object.

The loss consists of the projection loss and the negative refinement term proposed in SA3D. The loss for coarse stage $\mathcal{L}_{\text{coarse}}$ is Equation (5) in SA3D (Cen et al. 2023b):

$$\begin{aligned} \mathcal{L}_{\text{coarse}} = & - \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{I})} M_{\text{SAM}}(\mathbf{r}) M(\mathbf{r}) \\ & + \lambda \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{I})} (1 - M_{\text{SAM}}(\mathbf{r})) M(\mathbf{r}), \end{aligned} \tag{3.1}$$

where $\mathcal{R}(\mathbf{I})$ are all rays belong to image $\mathbf{I}$, $\mathbf{M}_{\text{SAM}}(\mathbf{r})$ is the value of pixel corresponding to ray $\mathbf{r}$ in the pseudo ground truth mask produced by SAM, $\mathbf{M}(\mathbf{r})$ is the value of pixel corresponding to ray $\mathbf{r}$ in the rendered score map, and λ is a hyperparameter that controls the strength of the refinement term.

This loss consists of two terms: the projection loss term and the negative refinement term weighted by λ . The projection loss is to encourage the alignment between the pseudo-ground truth $\mathbf{M}_{\text{SAM}}$ and the rendering result $\mathbf{M}$, expressed using the negative products. The refinement term is introduced to encourage the process to only label a 3D region as foreground when SAM consistently identifies it as foreground across multiple viewpoints. This is to handle the inaccuracy in the SAM pseudo-ground truth and the geometry of 3DGS.

3.3.2 Fine Stage Segmentation

To improve segmentation quality and identify object internals for comprehensive segmentation, the process is expanded by adding a fine stage and a merging process. After iterating over all the provided camera poses $\mathcal{C}$, the process enters the fine stage of segmentation. The process is outlined in Figure 3.4.

In the same way as adding segmentation scores, the representation is further augmented with dual segmentation scores, aiming to capture scene contents that should not be selected.

The dual scores are then initialized by computing a 3D bounding box containing all Gaussians with high scores in the first stage. Every Gaussian outside the bounding box has its dual score set close to one, while Gaussians inside the bounding box have their dual score set to zero. This is a conservative initialization strategy as it only considers Gaussians outside the bounding box as impossible to be the object, while leaving Gaussians inside the bounding box to be neutral, allowing the fine stage to refine and adjust.

The process is then continue for another full iteration, training the segmentation score and the newly added dual score, with the dual score attribute following the same self-prompting process and a new loss.

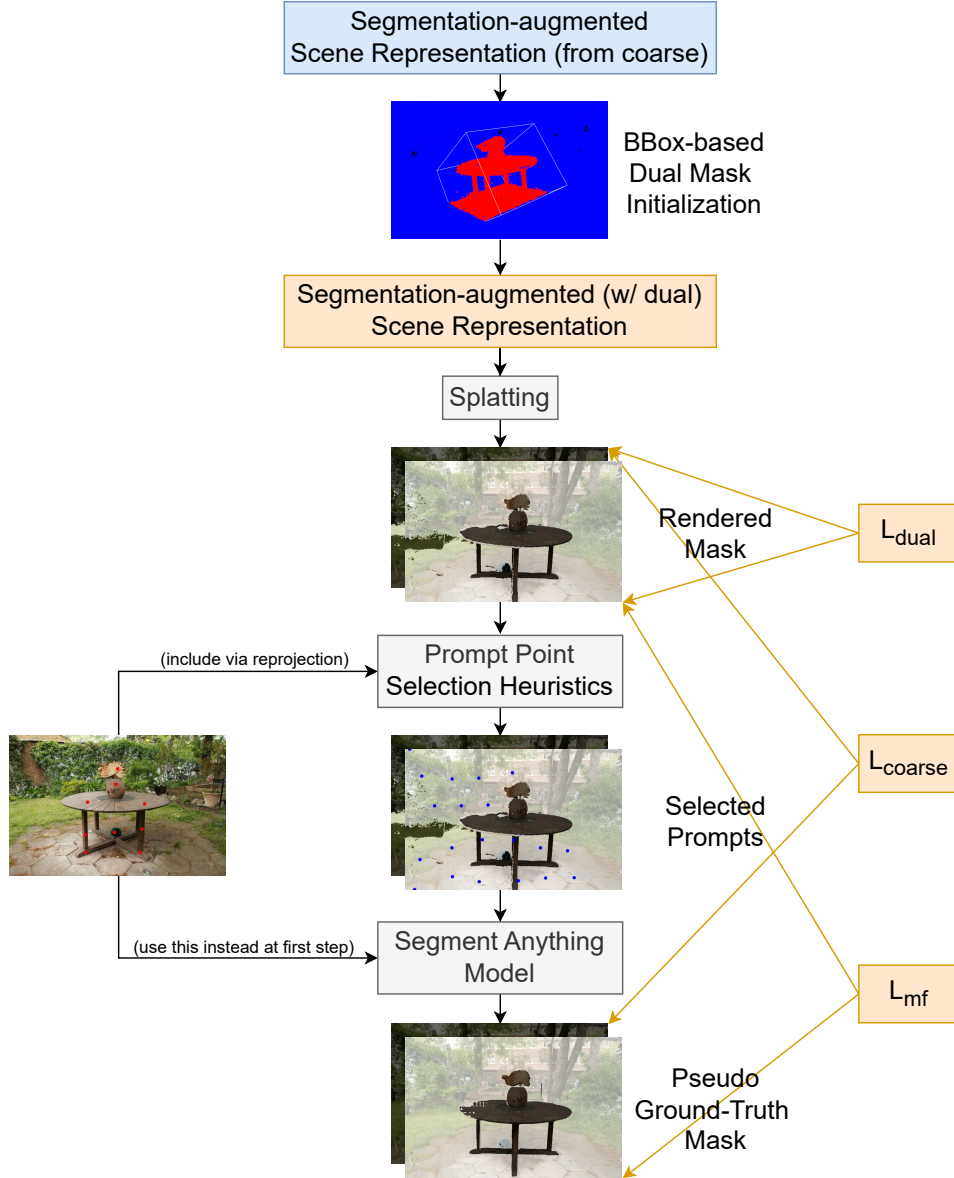


Figure 3.4: The fine stage of the proposed segmentation algorithm.

This fine-stage segmentation is designed to identify parts of the scene that are definitively not the target object. It achieves this by applying the same rendering-based training procedure to train a new "dual score." The key insight is that combining the original object score and this dual score reveals under-trained splats, namely those that are neither confidently part of the object nor confidently part of the background. Such splats are often located in occluded areas like the interior of an object, and the later merging step ensures that they are correctly included in the final segmentation.

The loss terms for the fine stage are presented in Equation (3.2). $\mathcal{L}_{\text{mf}}$ (Equation (3.3)) is the loss for the dual mask, where $\mathbf{M}_{\text{DSAM}}(\mathbf{r})$ is the value of pixel corresponding to ray $\mathbf{r}$ in the pseudo ground truth dual mask produced by SAM, and $\mathbf{M}_{\text{D}}(\mathbf{r})$ is the value of pixel corresponding to ray $\mathbf{r}$ in the rendered dual score map. An additional term is also added, $\mathcal{L}_{\text{dual}}$ (Equation (3.4)), that encourages masks and dual masks to have no intersections.

$$\mathcal{L}_{\text{fine}} = \mathcal{L}_{\text{coarse}} + \mathcal{L}_{\text{mf}} + \mathcal{L}_{\text{dual}} \quad (3.2)$$

$$\mathcal{L}_{\text{mf}} = - \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{I})} \mathbf{M}_{\text{DSAM}}(\mathbf{r}) \mathbf{M}_{\text{D}}(\mathbf{r}) \quad (3.3)$$

$$\mathcal{L}_{\text{dual}} = - \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{I})} -\mathbf{M}(\mathbf{r}) \mathbf{M}_{\text{D}}(\mathbf{r}) + \lambda_{\text{dd}} \left(- \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{I})} -\mathbf{M}_{\text{D}}(\mathbf{r}) \mathbf{M}(\mathbf{r}) \right) \quad (3.4)$$

3.3.3 Merging and Refinement

Finally, as shown in Figure 3.5, the scores and dual scores are merged to create the final segmentation. This stage compute the 3D bounding box in the same way as in the initialization process. Then, in addition to the Gaussians with a high enough segmentation score, this stage also include every Gaussian in the bounding box whose dual score is below a threshold into the final segmentation, effectively accepting those "rejected" by the dual mask. This rejection inclusion scheme aims to cover all Gaussians that are difficult to train, such as those hiding under the surface of the object. The

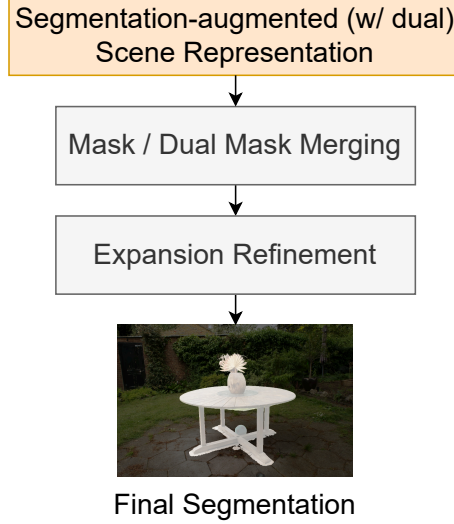


Figure 3.5: The final combination stage of the proposed segmentation algorithm.

selection is also expanded by including Gaussians whose mean is close enough to the already selected Gaussians.

The dual-stage design, which explicitly learns what constitutes the object versus the background, is critical for achieving a comprehensive segmentation. This approach allows the method to identify and incorporate Gaussians that are otherwise hard to reach by the rendering-based segmentation loop, such as those occluded or internal to the object. Thus improving over other existing segmentation methods that employs similar strategy such as SPIn-NeRF (Mirzaei et al. 2023b) and SAGA (Cen et al. 2023a). The final merging process is also made possible and efficient by the fact that the 3D Gaussian Splatting representation is composed of discrete, explicit primitives.

With $\mathcal{R}_{\text{obj}}$ and $\mathcal{R}_{\text{bg}}$ segmented, rudimentary editing can then be performed by editing $\mathcal{R}_{\text{obj}}$ and re-composing it back into $\mathcal{R}_{\text{bg}}$. This is the foundation for later stages of manipulation. This capability will be explored in Chapter 4 and Chapter 5.

3.4 Experiments

The proposed method was then tested on both 360-degree and forward-facing scene datasets to evaluate its effectiveness. In this chapter, effectiveness is defined by two primary criteria: the ability to produce high-quality segmentation masks and the ability to comprehensively cover the interiors of objects. The central hypothesis is that the proposed method can achieve segmentation quality comparable to state-of-the-art methods while providing a more complete 3D segmentation. Additional ablation studies were also conducted to highlight the importance of the proposed dual-stage design.

3.4.1 Implementation Details

Both stages of the segmentation training were performed using the SGD optimizer, with a learning rate of 1.0. λ_{dd} was set to 0.1. λ for the negative refinement term was set to 0.15. All experiments were conducted using a single A5000.

3.4.2 Experiment Setups

The proposed method was evaluated on two datasets chosen to assess different capabilities: SPIn-NeRF (Mirzaei et al. 2023b) and MipNeRF360 (Baron et al. 2022). The SPIn-NeRF dataset is a ubiquitous benchmark for 3D scene segmentation and was selected because it includes ground truth masks, enabling quantitative evaluation of segmentation quality. For this dataset, the point annotations from OR-NeRF (Yin et al. 2023) were used as input prompts for the segmentation methods. In contrast, the MipNeRF360 dataset, featuring 360-degree scenes, was chosen to better test propagation abilities. Such scenes require propagating prompts from the front of an object to its unseen back, a challenge not present in forward-facing datasets such as the SPIn-NeRF dataset.

Segmentation performance is measured by the mean accuracy and IoU (Intersection over Union) score of the rendered 2D segmentation masks across

all images. The accuracy metric measures the percentage of pixels the segmentation model classifies correctly. However, for scenes where the object to segment is small, predicting everything as background could yield high accuracy as well. To address this limitation, the IoU metric was also calculated, which focuses on how well the predicted foreground mask overlaps with the ground truth mask instead.

To evaluate the segmentation ability of the proposed approach, it was compared against state-of-the-art methods in 3D segmentation for object removal, as well as recent state-of-the-art 3DGS segmentation methods. Methods for scene object removal were included because they represent the closest application, focusing on segmenting objects from user prompts for subsequent manipulation. The comparisons in this category were against SPIn-NeRF (Mirzaei et al. 2023b) and OR-NeRF (Yin et al. 2023). More concretely, the proposed method was compared against the TensoRF variant of OR-NeRF, which is the fastest and the best overall result quality variant presented in the original paper. Additionally, the proposed approach was compared with a recent state-of-the-art 3DGS segmentation method, Segment Any 3D Gaussians (SAGA) (Cen et al. 2023a). This comparison serves to demonstrate the superior quality of the proposed method and to highlight its distinct ability to correctly include the internal, non-visible parts of the target object.

3.4.3 Evaluating Segmentation Quality

The proposed approach was compared against the aforementioned baseline methods (SPIn-NeRF (Mirzaei et al. 2023b), OR-NeRF (Yin et al. 2023) and SAGA (Cen et al. 2023a)) on the SPIn-NeRF (Mirzaei et al. 2023b) dataset. For this experiment, effectiveness is evaluated based on the quantitative accuracy and IoU of rendered segmentation masks. The hypothesis is that the proposed method will achieve competitive performance against specialized 2D methods while significantly outperforming other 3D segmentation baselines. Qualitative results were all produced from the published source code. For quantitative results, results for SPIn-NeRF and OR-NeRF were taken from their original papers, while results for SAGA were reproduced from

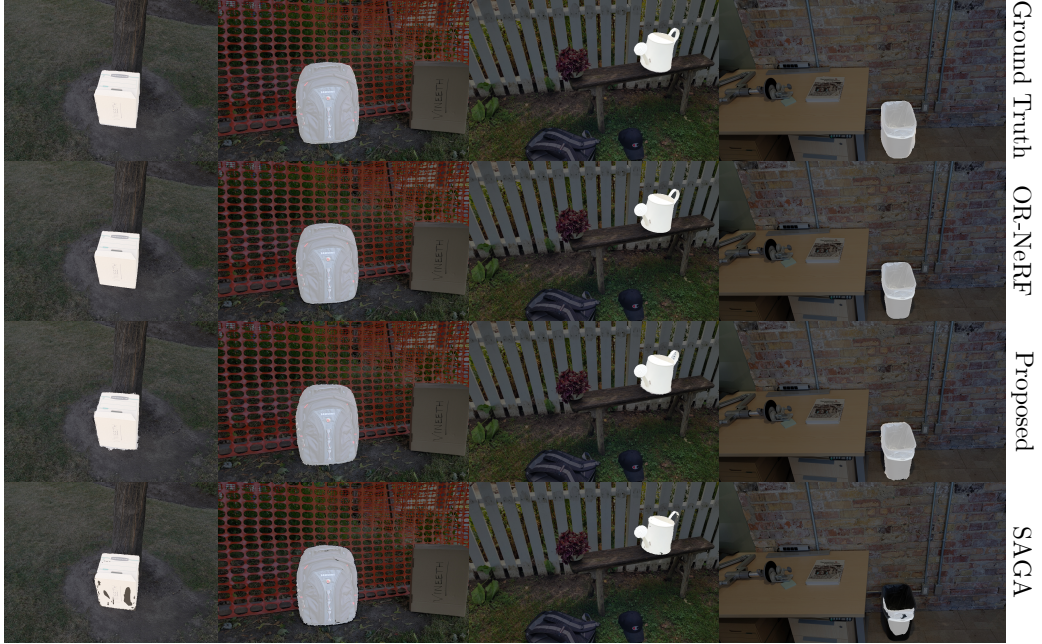


Figure 3.6: Qualitative scene segmentation results on the SPIn-NeRF dataset.

their published code. SPIn-NeRF did not provide per-scene metrics; hence, only the mean metrics are provided.

The qualitative results are presented in Figure 3.6. The proposed method achieves superior quality compared to SAGA. This is especially obvious from the box scene in the first column and the trash bin scene in the last column. The proposed method is also comparable to OR-NeRF (which is superior to SPIn-NeRF) despite producing 3D segmentation masks instead of 2D segmentation masks for every view.

This observation can also be seen from the quantitative results, with mean metrics presented in Table 3.1 and per-scene breakdowns in Table 3.2. The mean results show the proposed method achieving competitive performance against the 2D segmentation only method OR-NeRF, while outperforming 3D segmentation methods such as SPIn-NeRF and SAGA. The per-scene table further demonstrates that this method is consistently on-par with OR-NeRF, better than SPIn-NeRF and surpasses SAGA by a large margin, especially

method	Segmentation Metrics	
	Acc($\uparrow$)	IoU($\uparrow$)
Proposed	99.51	92.71
OR-NeRF	99.71	95.42
SPIn-NeRF	98.91	91.66
SAGA	90.46	57.32

Table 3.1: Quantitative scene segmentation results on the SPIn-NeRF dataset. The best value of each column is bolded for ease of reading.

in IoU.

The performance relative to the baselines is explained by methodological differences. While competitive with OR-NeRF (Yin et al. 2023), the proposed method achieves this by performing a full 3D segmentation, unlike OR-NeRF which only operates in 2D and is thus only applicable to perform scene object removal by inpainting instead of full 3D manipulation. The advantage over other 3D methods is twofold. Compared to SPIn-NeRF (Mirzaei et al. 2023b), this method leverages the higher-fidelity representation capabilities of 3DGS, as well as capability improvements in the employed 2D segmentation model. The substantial performance gap with the 3DGS segmentation method SAGA (Cen et al. 2023a) is due to a fundamental architectural choice: the proposed approach is optimized to segment one specific object with high accuracy, while SAGA is designed for generalizability by embedding features for all future selections. This focus on a single target allows the proposed method to achieve superior segmentation quality.

The proposed method also provides a more comprehensive 3D segmentation, a key aspect of its overall effectiveness. To test this, a comparison against SAGA (Cen et al. 2023a) was conducted on the MipNeRF360 (Barron et al. 2022) garden scene, as shown in Figure 3.7. The hypothesis is that the proposed method’s design results in a more complete 3D object mask, including its non-visible interior parts. Although SAGA produces reasonable rendered masks, its underlying 3D segmentation fails to include large portions of the vase, leaving many of its constituent Gaussians behind in the

Scene	Method			
	Proposed	OR-NeRF	SPIIn-NeRF	SAGA
Acc($\uparrow$)				
2	99.80	99.82	-	97.47
3	99.79	99.73	-	99.43
4	99.74	99.79	-	50.28
7	99.59	99.81	-	86.50
10	99.84	99.87	-	99.68
12	98.72	99.30	-	95.63
book	99.01	99.51	-	97.51
trash	99.56	99.51	-	97.19
mean	99.51	99.71	98.91	90.46
IoU($\uparrow$)				
2	96.16	96.47	-	65.79
3	98.09	97.48	-	94.78
4	98.15	98.50	-	20.85
7	94.68	97.43	-	29.11
10	94.67	95.47	-	89.44
12	86.51	91.73	-	66.03
book	83.45	88.68	-	58.27
trash	89.99	88.68	-	34.25
mean	92.71	95.42	91.66	57.32

Table 3.2: Per-scene quantitative scene segmentation results on the SPIIn-NeRF dataset. The best value of each column is bolded for ease of reading.



Figure 3.7: Qualitative results to compare the comprehensiveness of segmentation against SAGA.

background. Such an incomplete selection is undesirable for manipulation tasks, and in contrast, is not an issue for the proposed approach.

This limitation in SAGA likely stems from its learning and feature projection process. The method employs a training-based design to embed SAM-like features into individual Gaussian splats. Consequently, Gaussians in areas that are difficult to see during rendering, such as the object’s interior, may remain untouched by the training process and are thus not included from the final selection.

3.4.4 Ablation Studies

Finally, an ablation study was conducted to validate the effectiveness of the dual-stage design. The hypothesis is that this design is critical for achieving a comprehensive segmentation by correctly classifying occluded, internal parts of the object.

To validate the importance of the dual-stage design, an ablation study was conducted by disabling the fine stage of the segmentation process. The resulting background set, $\mathcal{R}_{bg}$, was then rendered for evaluation.

As shown in Figure 3.8, segmentation without the fine stage introduces ”floaters” (highlighted with red rectangles), namely internal parts of the object that are incorrectly left in the background. The full pipeline resolves this issue. These floating internal parts of the table typically receive neither

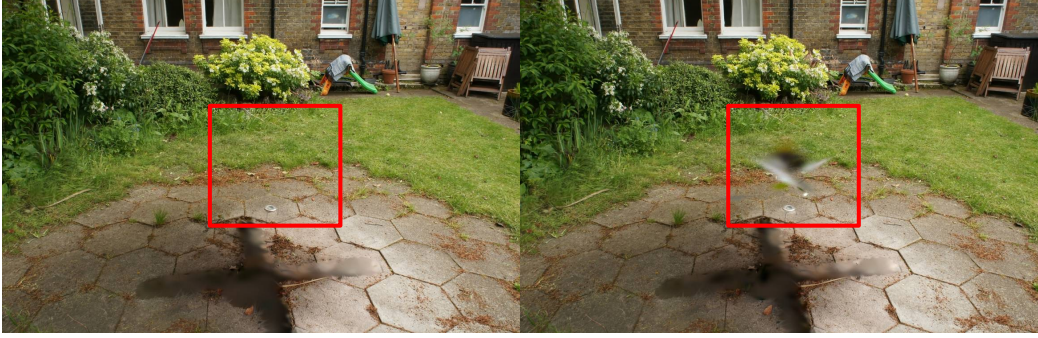


Figure 3.8: Ablation study results on the dual-stage design. $\mathcal{R}_{bg}$ part of segmentation, performed with (left) or without (right) the dual stage design.

a high object score nor a high dual score due to occlusion. The final merging stage is designed to handle this ambiguity by assigning such Gaussians to the object set, $\mathcal{R}_{obj}$. This ensures a complete segmentation, which is crucial for preventing artifacts in downstream manipulation tasks.

3.5 Summary

This chapter presents a method for segmenting 3D Gaussian Splatting (3DGS) scenes for manipulation, given 2D user point prompts as input. Building upon the state-of-the-art 2D segmentation model Segment Anything (Kirillov et al. 2023) and leveraging the explicit, point cloud-like formulation offered by 3D Gaussian Splatting (Kerbl et al. 2023), the proposed method achieves intuitive object selection and segmentation that properly includes the internals of objects. These capabilities make it a solid foundation for downstream manipulation and recomposition.

Chapter 4

Real-time Cage-based Deformation for 3DGS

4.1 Introduction

After segmenting the object from the 3DGS scene representation, this chapter explores how to manipulate the segmented object. Specifically, the focus is on ways to edit or animate 3DGS scene representations (or variants of 3DGS) in real-time, without needing lengthy training or fine-tuning processes.

To address these challenges, GSDeformer is proposed, a method that enables cage-based deformation on trained vanilla 3DGS models and their variants. The overall approach is illustrated in Figure 4.1. This approach operates directly on trained 3DGS without requiring extensive retraining, performs deformation in real time, and can be easily extended and integrated with other methods that enhance or build upon 3DGS.

The proposed approach leverages Cage-Based Deformation (CBD), which uses a coarse mesh (cage) to control deformations of the finer geometry within it. To deform the Gaussian distributions that make up the 3DGS representation, a proxy point cloud is generated from the Gaussians and CBD is applied to deform the point cloud. The deformed proxy points then drive the transformations applied to the Gaussians. To handle potential bending caused by deformation, a splitting process is introduced for the relevant Gaussians.

This direct deformation approach enables editing of any trained vanilla 3DGS model without requiring architectural modifications or retraining. It

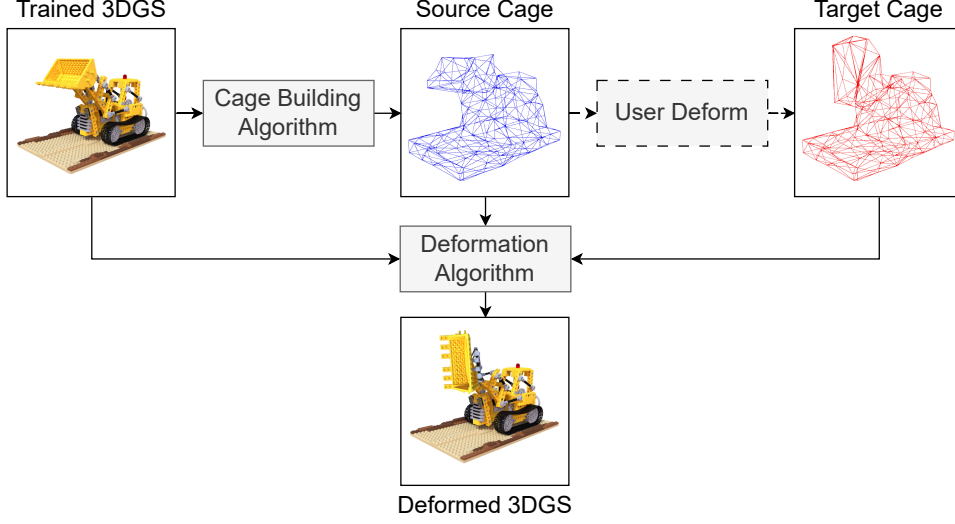


Figure 4.1: Overview of methods proposed in this chapter.

also ensures compatibility with other 3DGS variants. Additionally, by using standard triangular cage meshes to control deformation, the proposed method seamlessly integrates with existing animation software and pipelines.

Cages for deformation can be created either manually or automatically from 3DGS using an automated algorithm. The automatic cage-building algorithm supports not only 3DGS but also its variants, such as 2DGS (Huang et al. 2024), through a render-reconstruct-simplify approach.

The effectiveness of the proposed method is evaluated on object datasets. The results demonstrate that the deformation algorithm delivers superior quality compared to existing methods, particularly under extreme deformations. Among methods that enable explicit control, the employed control structure is the easiest to integrate with existing software. Furthermore, the algorithm achieves real-time performance ($\sim 60FPS$), renders deformed representations at high speeds ($> 200FPS$), and can be extended to other 3DGS variants.

In summary, the contributions in this chapter include:

- A method, GSDeformer, that achieves real-time cage-based deformation on any trained 3D Gaussian Splatting (3DGS) model without re-training or altering its core architecture.
- A robust automatic cage-generation algorithm that works on 3DGS as well as its variants due to its render-reconstruct-simplify approach.
- An extensive experimental evaluation demonstrating the proposed method’s ease of integration with other work extending 3DGS, as well as its superior quality and ease-of-control against existing methods under normal and extreme scenarios.

4.2 Method

This section details the proposed method for deforming trained 3D Gaussian Splatting (3DGS) scenes. To establish the foundation for this approach, the principles of cage-based deformation, a traditional but ubiquitous technique for shape manipulation, are first reviewed. A challenge in applying this technique to 3DGS is the generation of a suitable cage. To address this, a novel render-reconstruct-simplify cage-building algorithm is proposed, which leverages the depth information from the 3DGS scene to create a robust and watertight cage. Finally, the core deformation algorithm is detailed, demonstrating how the classical cage-based framework can be effectively adapted to manipulate the underlying novel primitives of the modern 3DGS representation.

4.2.1 Preliminaries

Cage-based Deformation Cage-based deformation is a technique that enables the manipulation of a complex object by deforming a simple, coarse polygon mesh that encloses it, known as a cage. To deform a fine mesh using a cage, consider a cage $\mathcal{C}_s$ with vertices $\{\mathbf{v}_j\}$, points $\mathbf{x} \in \mathbb{R}^3$ inside the cage $\mathcal{C}_s$ can then be represented by cage coordinates $\{\omega_j\}$ (e.g., mean value coordinates (Floater 2003)). These coordinates define the position of $\mathbf{x}$ relative

to the cage vertices. The position of $\mathbf{x}$ is calculated as the weighted sum of cage vertex positions:

$$\mathbf{x} = \sum_j \omega_j \mathbf{v}_j \quad (4.1)$$

After deforming the cage from $\mathcal{C}_s$ to $\mathcal{C}_d$ with vertices $\{\mathbf{v}'_j\}$, the new position $\mathbf{x}'$ of $\mathbf{x}$ can be computed using the calculated cage coordinates:

$$\mathbf{x}' = \sum_j \omega_j \mathbf{v}'_j \quad (4.2)$$

This process establishes a continuous deformation field within the cage’s volume. Consequently, any enclosed geometry, whether a fine mesh or, as is relevant to this work, a set of discrete points, can be smoothly deformed by manipulating the cage’s vertices.

4.2.2 Cage-Building Algorithm

Cage-based deformation requires a cage mesh to manipulate the underlying geometry. Given a trained 3DGS (or its variants) model $\mathcal{R}_s$, the goal is to generate a simple, low-polygon cage $\mathcal{C}_s$ that robustly encloses the object of interest. This cage can then be used to define and apply deformations.

Existing methods for cage generation, such as those used in DeformingNeRF (Xu and Harada 2022) and SuGaR (Guédon and Lepetit 2023), typically follow a direct "marching-cube-then-simplify" approach. They interpret the NeRF or 3DGS scene as an opacity field, extract a dense mesh using the marching cubes algorithm, and then decimate this mesh to create a cage. However, this direct extraction is sensitive to noise and artifacts in trained 3DGS models, especially floaters. This often results in cages with undesirable floating components. Furthermore, naive decimation, which optimizes for polygon count alone, can introduce new geometric defects or exacerbate existing ones, such as self-intersections and sharp concavities. These flaws degrade the quality of MVC coordinates and ultimately harm the quality of final deformation.

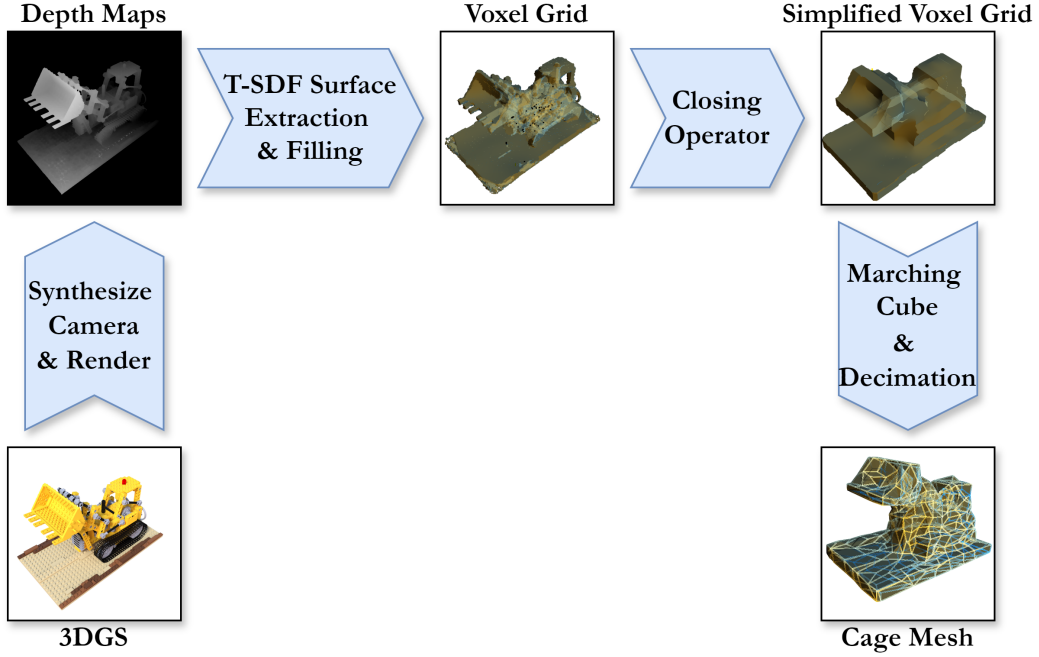


Figure 4.2: Overview of the cage-building algorithm.

Furthermore, such volumetric interpretation is incompatible with 3DGS variants that use non-volumetric primitives, like the 2D planar Gaussians in 2DGS (Huang et al. 2024).

To overcome these limitations, a more robust "render-reconstruct-simplify" pipeline is employed, as illustrated in Figure 4.2. Instead of interpreting the 3DGS scene directly as an opacity field, this method extracts the underlying geometry by rendering 2D depth maps from multiple views. The full process is broken down into the following steps, following Figure 4.2.

Step 1: Synthesize Camera and Render The algorithm begins by synthesizing a set of virtual cameras arranged on a sphere that encompasses the 3DGS scene. Depth maps are then rendered from each of these camera positions. This initial rendering step serves two crucial purposes. First, it aligns better with the native, rendering-based nature of 3DGS, making the geometry extraction less susceptible to artifacts like low-opacity floaters than methods that treat the scene as a density field. Second, by converting the scene into a standardized format of depth maps, the pipeline becomes

agnostic to the specifics of the underlying 3DGS representation, ensuring its applicability to both standard 3DGS and its non-volumetric variants.

Step 2: T-SDF Surface Extraction and Filling The collection of rendered depth maps is then fused into a single, coherent 3D volume using the Truncated Signed Distance Field (T-SDF) integration algorithm from KinectFusion (Newcombe et al. 2011). By integrating depth information from multiple viewpoints, the T-SDF reconstruction naturally averages out inconsistencies and suppresses noise caused by render errors or floaters, resulting in a significantly cleaner surface representation.

To produce a solid volume, this process is followed by a space-carving step. Clean depth maps are re-rendered from the newly created T-SDF volume and used to carve away empty space, filling the interior of the surface. This process yields a high-resolution, solid voxel grid that represents the object’s geometry without the artifacts present in the original 3DGS scene.

Step 3: Voxel Grid Simplification While the reconstructed voxel grid is geometrically accurate, its high level of detail is unsuitable for a deformation cage. To generate a simplified, coarse envelope, a morphological closing operator, as described in Bounding Proxy (Calderon and Boubekeur 2017), is applied. This operator first performs a dilation, which expands the voxel grid to fill out small holes and connect nearby components. It then performs an erosion to shrink the grid back towards its original size, which has the effect of smoothing the surface. The net result is a simplified, solid volumetric shape that smoothly envelops the original geometry, making it an ideal precursor to the final cage.

Step 4: Marching Cube and Decimation In the final stage, the simplified voxel grid is converted into a polygonal mesh, as done in Bounding Proxy (Calderon and Boubekeur 2017). First, the Marching Cubes algorithm is used to extract a dense isosurface from the volume. Subsequently, this dense mesh is simplified using an edge collapse algorithm to reduce its polygon count. The result is the final low-polygon cage mesh, which is lightweight, manifold, and well-suited for interactive deformation.

In summary, the proposed method achieves cage extraction by utilizing depth maps and T-SDF integration for robust geometry information from

3DGS, combined with morphological closing to remove details for final conversion to the coarse cage mesh. This approach effectively addresses floaters and artifacts in 3DGS models and supports non-volumetric 3DGS variants.

4.2.3 Deformation Algorithm

The proposed deformation algorithm takes a trained 3DGS scene $\mathcal{R}_s$ along with source and target cages, $\mathcal{C}_s$ and $\mathcal{C}_d$. These cages define a deformation for part or all of the scene. The objective is to produce the deformed scene $\mathcal{R}_d$, a 3DGS representation with the specified deformation applied.

The core of this method is a per-Gaussian transformation process, which consists of four main steps. First, each 3D Gaussian is converted into a geometric proxy, namely an ellipsoid represented by a sparse point cloud. Second, this point cloud proxy is deformed using cage-based deformation, which transfers the transformation from the cage to the proxy. Third, a splitting mechanism is introduced to handle cases where the deformation is non-rigid and would require Gaussians to bend. Finally, the transformation of the proxy points is transferred to the Gaussians to achieve deformation. This entire process is visualized in Figure 4.3.

Step 1: Gaussian to Ellipsoid Proxy To geometrically manipulate the 3D Gaussians, the algorithm starts by converting them into explicit geometries, namely ellipsoids. The ellipsoids are represented using point clouds so they can be deformed by cage-based deformation.

According to the definition, the probability density function (pdf) of a 3D Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ is defined as:

$$G(x) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

Given a specific value of $G(x)$, the isosurface of this function can be seen as an ellipsoid. As defined by 3DGS (Kerbl et al. 2023), in quadric form, the ellipsoid is:

$$(\mathbf{x} - \boldsymbol{\mu}_s)^\top \boldsymbol{\Sigma}_s^{-1}(\mathbf{x} - \boldsymbol{\mu}_s) = 1$$

Here, $\boldsymbol{\mu}_s$ is the ellipsoid’s center $\mathbf{c}$, $\boldsymbol{\Sigma}_s$ is the covariance attribute stored by 3DGS (encoded by a rotation matrix and scaling matrix).

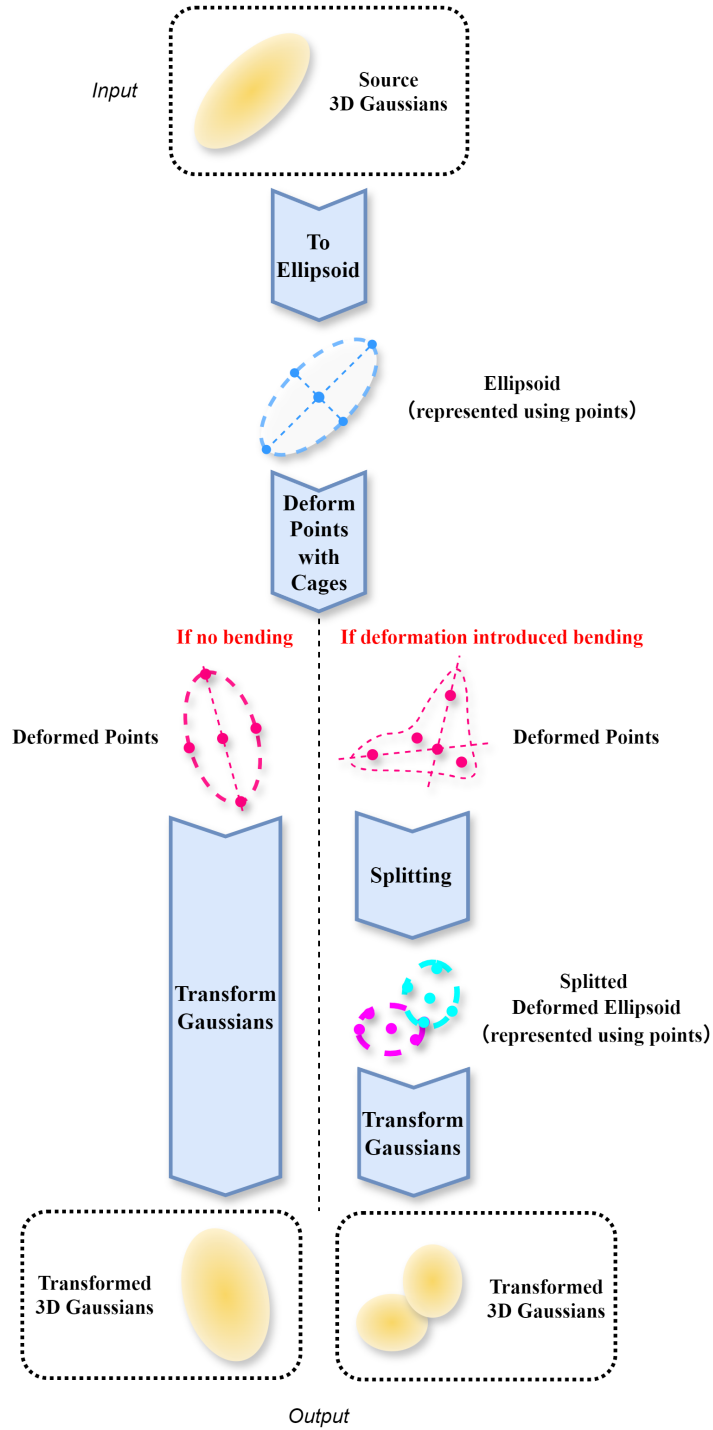


Figure 4.3: Overview of the deformation algorithm.

The principal axes and their lengths are derived from the eigenvalue decomposition of Σ_s^{-1} , namely the three principal axes (e.g., x, y, z) correspond to the eigenvectors, the axis lengths are inversely proportional to the eigenvalues of Σ_s^{-1} .

This connection is derived from the eigendecomposition of the shape matrix Σ_s^{-1} . Let $\Sigma_s^{-1} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$, where the columns of the orthogonal matrix $\mathbf{Q}$ are the eigenvectors and $\mathbf{D}$ is the diagonal matrix of corresponding eigenvalues λ_i . Substituting this into the ellipsoid equation yields:

$$(\mathbf{x} - \boldsymbol{\mu}_s)^\top (\mathbf{Q}\mathbf{D}\mathbf{Q}^\top) (\mathbf{x} - \boldsymbol{\mu}_s) = 1$$

By defining a new coordinate system $\mathbf{z} = \mathbf{Q}^\top (\mathbf{x} - \boldsymbol{\mu}_s)$, which geometrically represents a rotation that aligns the ellipsoid with the coordinate axes, the equation simplifies to its canonical form:

$$\mathbf{z}^\top \mathbf{D} \mathbf{z} = 1 \quad \implies \quad \sum_{i=1}^3 \lambda_i z_i^2 = 1 \quad \implies \quad \sum_{i=1}^3 \frac{z_i^2}{(1/\sqrt{\lambda_i})^2} = 1$$

This final expression describes an axis-aligned ellipsoid in the $\mathbf{z}$ -coordinates with semi-axis lengths of $1/\sqrt{\lambda_i}$. Since the transformation back to the original coordinates is given by $(\mathbf{x} - \boldsymbol{\mu}_s) = \mathbf{Q}\mathbf{z}$, the columns of $\mathbf{Q}$, namely the eigenvectors, define the orientation of these principal axes in the original space.

To facilitate deformation, this continuous ellipsoid shape is discretized into a sparse set of seven proxy points. This set includes the center point $\mathbf{c}$ and the six points where the three principal axes intersect the ellipsoid's surface. This point set $\mathcal{P}$ is defined as:

$$\mathcal{P} = \{\mathbf{c}, \mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1, \mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_2\} \quad (4.3)$$

where $\mathbf{x}_{1,2}$ denote the two intersection points on the x -axis, with similar definitions for y and z . This minimal set of points $\mathcal{P}$ effectively captures the ellipsoid's position, orientation, and scale, making it a suitable proxy for the cage-based deformation stage.

Step 2: Deform Points with Cages With the proxy point cloud $\mathcal{P}_s$ established for a source Gaussian, the deformation defined by the source

cage $\mathcal{C}_s$ and target cage $\mathcal{C}_d$ is applied. This method employs Mean Value Coordinates (MVC) (Floater 2003) for this task.

More concretely, the deformation proceeds in two stages. First, each point in the proxy set $\mathcal{P}_s$ is converted from its standard Euclidean coordinates into Mean Value Coordinates with respect to the source cage $\mathcal{C}_s$. Second, these MVCs, which encode the relative position of the points within the cage, are used to reconstruct the points in Euclidean space, but this time using the vertex positions of the deformed target cage $\mathcal{C}_d$. This process effectively transfers the deformation from the cage to the proxy points, resulting in a new, deformed point set denoted as $\mathcal{P}_d$.

Step 3: Splitting Bent Gaussians Cage-based deformation can induce non-rigid transformations that cause elongated Gaussians to bend. A single Gaussian, being an ellipsoid, cannot accurately represent such a curved shape, which leads to noticeable visual artifacts. To address this, a splitting procedure is introduced to replace a single, ill-formed bent Gaussian with multiple smaller Gaussians that can better approximate the deformed shape, as illustrated in Figure 4.4.

This step starts by identifying the bent Gaussians to split. Given a Gaussian with deformed points $\mathcal{P}_d$, its center is $\mathbf{c}'_d \in \mathcal{P}_d$. The deformed points of an axis (e.g., x-axis) are $\mathbf{x}'_1, \mathbf{x}'_2 \in \mathcal{P}_d$. The angle between vectors $\mathbf{x}'_1 - \mathbf{c}'$ and $\mathbf{x}'_2 - \mathbf{c}'$ are calculated. Splitting is required if the angle falls below a threshold.

The splitting method divides a pre-MVC Gaussian ellipsoid into two smaller ellipsoids along a chosen axis. Each new ellipsoid has its axis lengths reduced by a factor k , and both are positioned to meet at the center of the original ellipsoid along the splitting axis. Finally, both resulting ellipsoids undergo MVC transformation again for further processing. The resulting two smaller ellipsoids are more tolerable to deformation along that axis, reducing the risk of bending or distortion when applying MVC directly to elongated shapes.

The axis scaling factor k is determined by analyzing the optimal value that can maximally preserve the volume and shape of the original ellipsoid. More

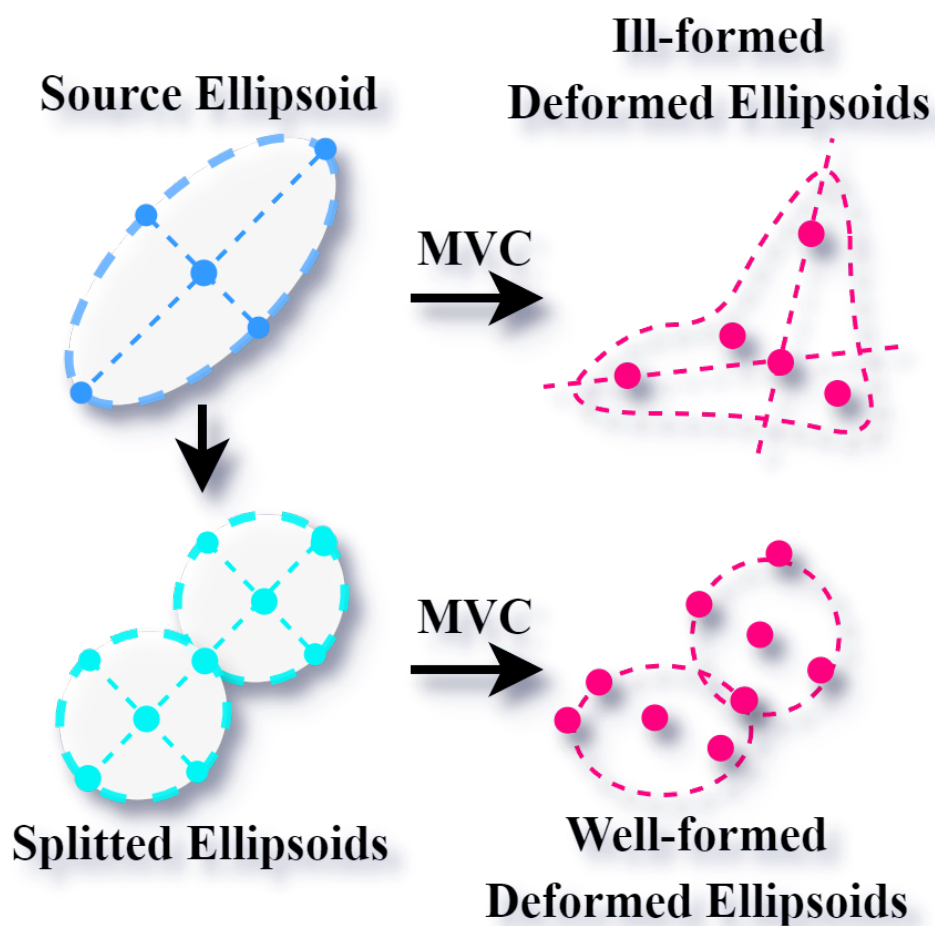


Figure 4.4: The splitting process. This method resolves ill-formed bent Gaussians by splitting the original Gaussian before deformation.

concretely, this scaling factor should satisfy two key constraints: Volume Preservation and Shape Preservation.

Volume Preservation Constraint: The volume of the two smaller ellipsoids combined should approximate the volume of the original ellipsoid. This can be expressed as:

$$L_V = \frac{2}{3}\pi abc - 2\frac{2}{3}\pi k_1 a k_2 b k_3 c \quad (4.4)$$

where, a, b, c are the semi-principal axes of the original ellipsoid, k_1, k_2, k_3 are the individual scaling factors for the smaller ellipsoids. Simplifying this equation leads to:

$$k_1 k_2 k_3 = \frac{1}{2} \quad (4.5)$$

Shape Preservation Constraint: Each smaller ellipsoid should maintain a shape similar to the original ellipsoid. This constraint can be expressed as:

$$\frac{k_1 a}{k_2 b} \approx \frac{a}{b} \quad \frac{k_2 b}{k_3 c} \approx \frac{b}{c} \quad \frac{k_1 a}{k_3 c} \approx \frac{a}{c} \quad (4.6)$$

From these constraints, the scaling factor for all axes becomes:

$$k_1 = k_2 = k_3 = \sqrt[3]{\frac{1}{2}} \quad (4.7)$$

Thus, the scaling factor k for splitting should be $\sqrt[3]{\frac{1}{2}}$. The factor is further studied and tuned in the ablation study.

Step 4: Infer Transform and Apply to Gaussians After obtaining the original proxy points $\mathcal{P}_s$ and their deformed counterparts $\mathcal{P}_d$ (either from direct deformation or after the splitting process), the final step is to infer an affine transformation that maps one set to the other. This transformation is then applied to the parameters of the source Gaussian $\mathbf{s}$ to complete the deformation.

Given the original and deformed points $\mathcal{P}_s$ and $\mathcal{P}_d$, transformation $\mathbf{T} \in \mathbb{R}^{3 \times 3}$ is computed by minimizing:

$$\min_{\mathbf{T}} \|\mathbf{D}_d - \mathbf{T}\mathbf{D}_s\|^2 \quad (4.8)$$

where $\mathbf{D} = [\mathbf{x}_1 - \mathbf{c}, \mathbf{y}_1 - \mathbf{c}, \mathbf{z}_1 - \mathbf{c}, \mathbf{x}_2 - \mathbf{c}, \mathbf{y}_2 - \mathbf{c}, \mathbf{z}_2 - \mathbf{c}] \in \mathbb{R}^{3 \times 6}$ for $\mathcal{P}_s$ or $\mathcal{P}_a$. The least squares solution for $\mathbf{T}$ is given by:

$$\mathbf{T} = \mathbf{D}_d \mathbf{D}_s^\top (\mathbf{D}_s \mathbf{D}_s^\top)^{-1} \quad (4.9)$$

The 3D Gaussian $\mathbf{s}$ can then be transformed using $\mathbf{T}$, while using the center of deformed points $\mathcal{P}_a$ as the new mean:

$$\boldsymbol{\mu}_d = \mathbf{c}_d \quad (4.10)$$

$$\boldsymbol{\Sigma}_d = \mathbf{T} \boldsymbol{\Sigma}_s \mathbf{T}^\top \quad (4.11)$$

This step is crucial to the proposed algorithm as it updates the Gaussians' covariance, which models their shape and orientation. Not updating the covariance would lead to severe artifacts, which is demonstrated in ablation studies.

According to the implementation of 3DGS (Kerbl et al. 2023), the transformed mean and covariance can be directly used for rendering. Optionally, to recover rotation and scaling from the covariance, use SVD decomposition $\boldsymbol{\Sigma}_d = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top$, where $\mathbf{U}$ gives rotation and $\sqrt{\boldsymbol{\Sigma}}$ gives scaling.

In summary, the proposed algorithm deforms 3DGS by first converting 3D Gaussians into ellipsoids represented using points. Deformations of the points can then be transferred back to the underlying 3D Gaussians. An additional splitting step is also introduced to handle the non-rigid deformation required by cage-based deformation. This proxy point approach and the splitting process are what bridge the traditional cage-based deformation editing method with the novel 3DGS scene representation.

This approach directly operates on the primitives of 3DGS; thus, no architectural changes or retraining are needed. Furthermore, with caching, this direct transform approach can work in real-time. Finally, since it operates directly on the primitives, the proposed method can be easily extended to work with 3DGS variants without concerns on training (such as FLoD (Seo et al. 2024)) and integrate with other 3DGS editing work (such as GaussianEditor (Chen et al. 2023b)) with minimal changes.

4.3 Experiments

4.3.1 Implementation Details

For cage building, the voxel grid’s resolution was set to 128. The cage building algorithm employed the T-SDF integration and carving procedure from Open3D (Zhou et al. 2018), using their default parameters.

For deformation, the splitting threshold was set at 175 degrees and avoided splitting axes with lengths below $1e-2$ for numerical stability. For cages that only encompass part of the scene, the algorithm computed the convex hull of the cage and only deformed gaussians that fell inside it for speed and stability.

To achieve real-time deformation, extensive caching was performed in the implementation of the deformation algorithm. Note that MVC coordinates in step 2 and $\mathbf{D}_s$ in step 4 can be precomputed and reused when only the target cage is manipulated.

All experiments were run using an NVIDIA A5000 GPU and an AMD Ryzen Threadripper PRO 3975WX processor (32 cores).

4.3.2 Cage Building Quality

The evaluation of the proposed cage-building algorithm was designed to test the hypothesis that the proposed render-reconstruct approach produces higher-quality cages and is more robust to different 3DGS representations, particularly surface-like variants such as 2DGS. The effectiveness of the cage building process was therefore evaluated on three main criteria: (1) mesh quality, specifically the smoothness and regularity of the surface, which is critical for effective user manipulation, (2) structural integrity, meaning the cage is manifold and free of unwanted floating artifacts, and (3) algorithm robustness, defined as the ability to generate coherent cages from 3DGS variants that lack volumetric primitives, such as 2DGS.

The proposed approach was evaluated against the traditional marching-cube-then-simplify process used in NeRFShop (Jambon et al. 2023) and DeformingNeRF (Xu and Harada 2022). However, since the process used by

existing methods is designed for radiance fields and cannot directly work with 3DGS, they were adapted based on SuGaR’s (Guédon and Lepetit 2023) approach, detailed below. Additionally, this experiment tested a variant of this baseline that used the proposed method’s voxel grid smoothing and simplification process instead of direct edge collapse to highlight the strength of the proposed method’s render-reconstruct design.

To provide a comprehensive evaluation, two distinct scenes were chosen to test the algorithm’s performance on different geometric characteristics. The Lego scene from the NeRF Synthetic Dataset (Mildenhall et al. 2020) was selected as it is a standard benchmark featuring complex geometry with both sharp edges and flat surfaces. To complement this synthetic example, a flower vase was extracted from the MipNeRF360 dataset’s garden scene (Barron et al. 2022) to evaluate the algorithm’s performance on objects with the smooth, curved surfaces more typical of real-world captures.

SuGaR-inspired Baseline Inspired by the 3DGS marching cube method proposed by SuGaR (Guédon and Lepetit 2023), the marching cube baseline created a cage by first converting 3DGS into a binary occupancy voxel grid through opacity thresholding. The process then applied marching cubes to create meshes and simplified them through edge collapse, as proposed by NeRFShop (Jambon et al. 2023).

3DGS was transformed into a binary voxel grid by thresholding each voxel’s opacity. A voxel became one if its opacity exceeded the threshold and zero otherwise. The baseline calculated voxel opacity $d(\mathbf{v})$ by summing contributions from the K-nearest 3D Gaussians from the voxel center:

$$d(\mathbf{v}) = \sum_g \alpha_g \exp\left(-\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu}_g)^\top \boldsymbol{\Sigma}_g^{-1}(\mathbf{v} - \boldsymbol{\mu}_g)\right) \quad (4.12)$$

For each Gaussian g near voxel center $\mathbf{v}$, the baseline used its opacity α_g , mean $\boldsymbol{\mu}_g$, and covariance $\boldsymbol{\Sigma}_g$. The baseline used a threshold of 1e-6 and K=16 nearest Gaussians. With the occupancy voxel grid, the process meshed it using marching cubes and simplified it using edge collapse to create the final cage.

This cage-building method was later extended to work with 2DGS (Huang et al. 2024) by converting 2DGS to 3DGS: For every flat ellipse in 2DGS, the process computed its third axis as the cross product of the first two axes and set its length to $1e-5$, converting 2D ellipses into 3D ellipsoids.

In Figure 4.5, the output of the proposed method and the output of the marching cube (MC) baseline are compared, showing both the raw voxel grids (before any smoothing operation) and the final meshes. The proposed method yielded cleaner voxel grids for Lego and vase scenes, leading to higher-quality cages. In contrast, the baseline’s noisy voxel grids led to a highly dense and noisy mesh that could not be used. Even using the smoothing and simplification process in the proposed pipeline, the MC baseline still created unwanted bubbles in the Lego cage and complex edges in the vase cage, as marked by the red circles. This is attributed to the fact that the proposed method uses rendering rather than direct opacity field interpretation, which is closer to how 3DGS is trained and viewed, as well as the noise-removal effect of the T-SDF integration algorithm.

The proposed cage-building algorithm also extends naturally to 3DGS variants. When applied to a trained 2DGS representation of the vase (third and final row of Figure 4.5), the MC baseline method failed to produce a coherent voxel grid and cage. Even with the voxel grid smoothing process, the cage was still hollow, while the proposed approach still generated accurate voxel grids and cages suitable for cage-based deformation. This difference occurs because 2DGS uses flat 2D ellipses with minimal volume instead of volumetric 3D ellipsoids, which causes the baseline methods to fail as it requires primitives to have volume. The proposed method remained effective regardless of this distinction.

4.3.3 Deformation Quality

The deformation quality of the proposed approach was then evaluated against existing methods to test the hypothesis that the algorithm, particularly with its splitting mechanism, produces smoother and more plausible results with

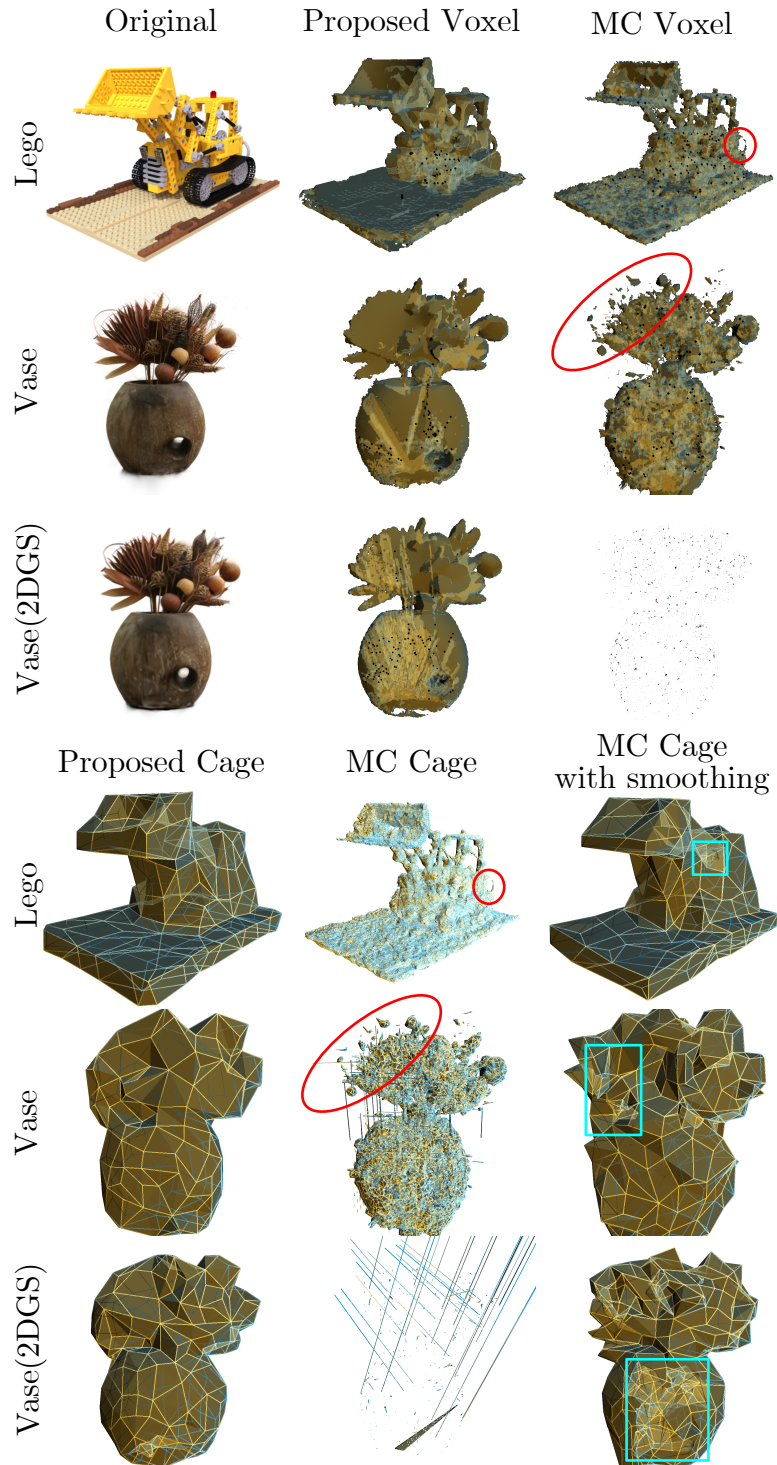


Figure 4.5: Comparison of intermediate voxel grid and final result of cage building algorithms. **Red circles** and **cyan boxes** marks defects.

fewer artifacts, especially under extreme transformations. For this comparison, an effective deformation is one that preserves fine details, maintains surface continuity without tearing, and avoids visual artifacts such as blurriness or spikes. The evaluation started with pre-trained 3D Gaussian Splatting (Kerbl et al. 2023) models, applied the proposed cage construction algorithm for cages, manually deformed the cages, and ran the proposed deformation algorithm. For comparison, the experiment trained and deformed DeformingNeRF (Xu and Harada 2022), GaMeS (Waczyńska et al. 2024), SuGaR (Guédon and Lepetit 2023), and Gaussian Frosting (Guédon and Lepetit 2024) on the same objects as well. Cages from the proposed cage-building algorithm were used to deform their underlying mesh or triangle soup for fair comparisons.

To rigorously test these criteria, a diverse set of scenes was selected to challenge different aspects of the deformation algorithms. The microphone scene was chosen for its fine, repeating geometric grid, which is difficult to preserve without introducing artifacts. The ficus scene, with its thin branches and delicate leaves, tested the method’s ability to maintain topological integrity. The hotdog scene provided a case for preserving the continuity of smooth surfaces under manipulation. Finally, the Lego bulldozer scene was used to evaluate performance under extreme articulation, specifically large-angle rotations that often cause catastrophic failures in other methods.

Normal Deformations Results are presented in Figure 4.6. In the microphone scene, DeformingNeRF failed to preserve the detailed grid structure of the microphone’s mesh, while Gaussian Frosting produced holes and spiky artifacts on it. GaMeS and SuGaR also showed spiky artifacts. For the ficus scene, DeformingNeRF and Gaussian Frosting created artifacts on the unedited flower pot. In the expanded upper part, Gaussian Frosting and SuGaR struggled with details, breaking connections between branches. In the hotdog scene, there were wrinkles on the plate with DeformingNeRF, severe tearing with Gaussian Frosting, and spiking artifacts with SuGaR and GaMeS due to the lack of a splitting process. Across all scenes, the proposed approach was the only method that consistently produced the smoothest and most plausible results.

Extreme Deformations How well these methods handle challenging deformations was further evaluated. In Figure 4.7, the bulldozer’s head in the Lego scene from the NeRF Synthetic Dataset was rotated by 90, 135, and 180 degrees. As can be seen, DeformingNeRF failed to handle twisting. Gaussian Frosting produced blurry artifacts across all angles. GaMeS showed spiky artifacts. While SuGaR worked reasonably well at 90 degrees, it created artifacts at larger angles. In contrast, the proposed method remained stable and generated reasonable results even under extreme rotations.

The artifact-free result of the proposed method stems from its splitting process, which accurately divides Gaussians in areas that undergo bending, thus removing the spiking artifacts that plague other methods. Its superiority over DeformingNeRF stems from the fundamental difference in scene representation, as 3DGS employs an ensemble of primitives rather than an implicit field encoded as neural network weights. This allows for direct deformation of the scene geometry instead of the rendering rays, which prevents blurriness in the microphone scene and complete breakdown in the 180-degree Lego deformation, both of which likely result from extreme distortion in the rendering process.

4.3.4 Other Control Methods

The proposed approach was further compared against alternative approaches to test the hypothesis that the adopted control structure, namely triangular mesh cages, offers a more effective balance of control, simplicity, and quality. In this context, effectiveness was evaluated on (1) ease of integration with standard 3D software and animation pipelines, and (2) the ability to achieve deformation quality that is comparable or superior to methods using more complex control structures like dense tetrahedral grids (VR-GS (Jiang et al. 2024)) or sparse control points (SC-GS (Huang et al. 2023)).

Since VR-GS and SC-GS were designed for different scenarios, they were adapted for deformation by directly manipulating their control structure. For this evaluation, three scenes from the D-NeRF dataset (Pumarola et al. 2020)-mutant, jumpingjacks, and lego-were selected. This dataset serves as a

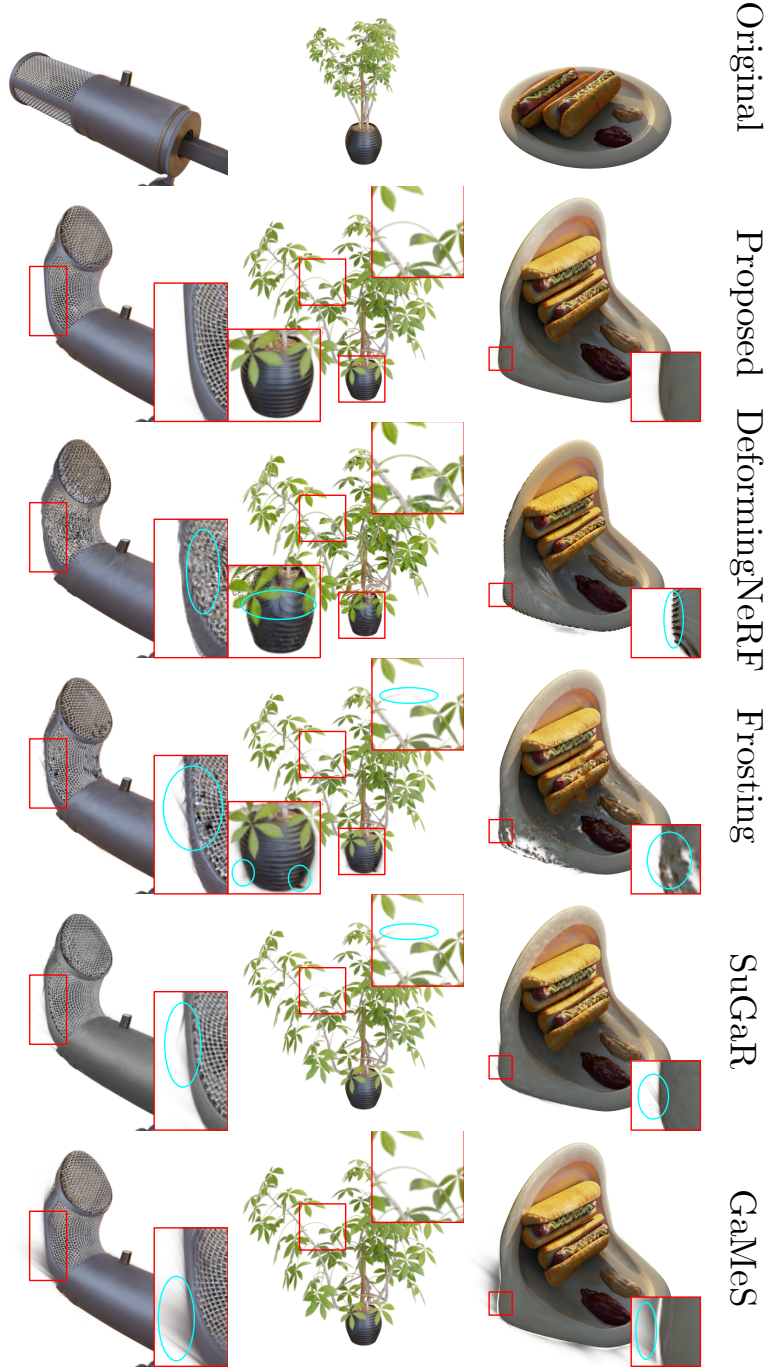


Figure 4.6: Quality comparison of deformation methods on selected objects. **Red boxes** indicate zoomed areas; **cyan circles** marks defects.

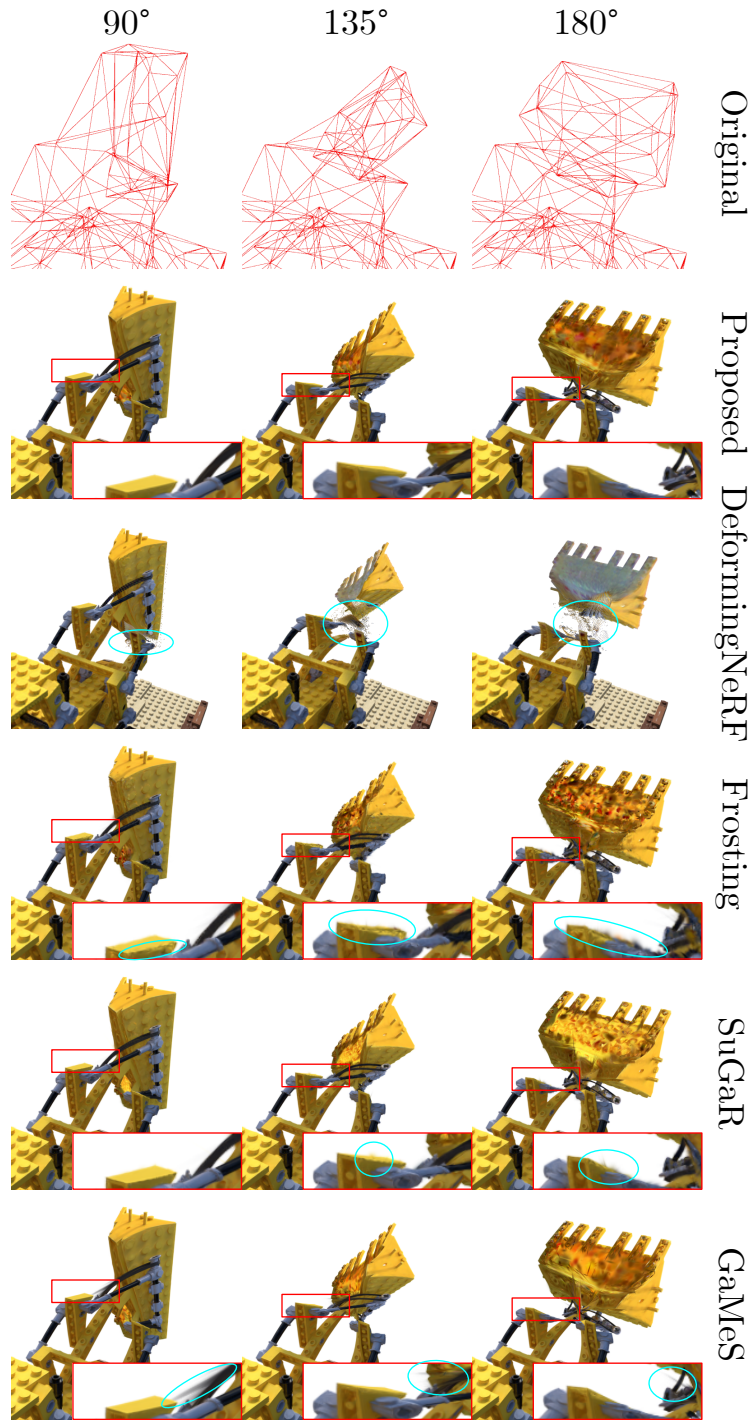


Figure 4.7: Quality comparison of deformation methods under normal to extreme deformations. **Red boxes** indicate zoomed areas; **cyan circles** marks defects.

standard benchmark for dynamic scenes, making it well-suited for comparing the effectiveness of different deformation control structures that may require videos for training. For fair comparison, SC-GS was trained on these dynamic scenes and extracted models at a single time frame for testing VR-GS and the proposed method. This is because SC-GS requires training on video data, while other methods do not. Finally, each method’s control structure for controlling deformation and deformation results was analyzed.

As shown in Figure 4.8, the proposed method controls deformation using standard triangle meshes, while VR-GS uses dense tetrahedral grids and SC-GS uses sparse control points. The proposed approach offers a simpler structure than VR-GS’s simulation-focused design and integrates better with existing animation software than both SC-GS and VR-GS. In terms of result quality, the proposed method demonstrated clear advantages over SC-GS, producing smoother shoulder deformations in the jumpingjacks and mutant scenes, while avoiding the disconnection artifacts present in the lego scene. When compared to VR-GS, the proposed method produced smoother details on the lego and jumpingjacks scenes, and the results were comparable in the mutant scene. This showed that the proposed method is more effective in handling complex deformations while using a simpler and easier-to-integrate control method.

4.3.5 Training & Deformation Speed

Finally, a benchmark of training and deformation times across all methods was done to test the hypothesis that the proposed approach is significantly more efficient, eliminating retraining and enabling real-time deformation. Here, effectiveness was measured by (1) training time, which is posited to be zero for the proposed method, and (2) deformation and rendering speeds, measured in frames per second (FPS), which were expected to be interactive.

The evaluation was conducted on two distinct sets of scenes and cages to ensure a comprehensive and fair benchmark. The first set included the lego, chair, ficus, and hotdog scenes from the NeRF Synthetic Dataset (Mildenhall et al. 2020), paired with complex cages automatically generated by the

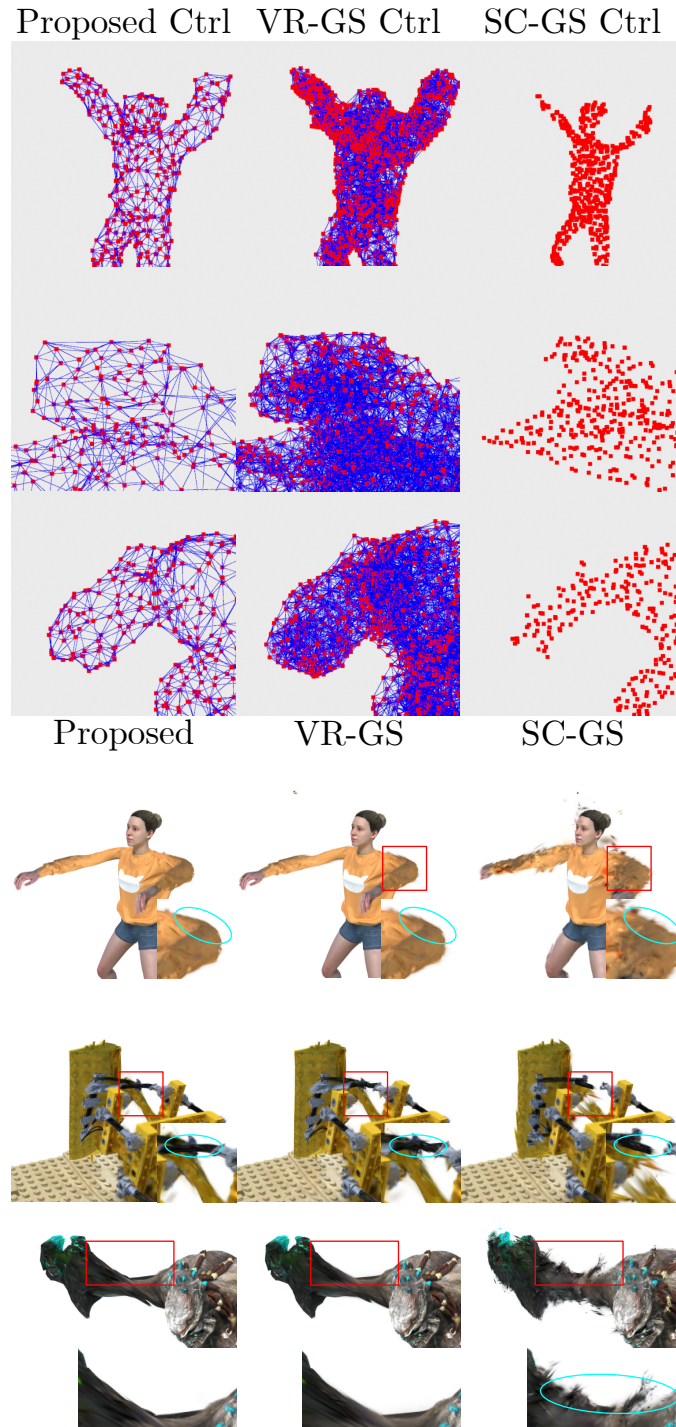


Figure 4.8: Control structure and result quality comparison of methods that can be adapted for 3DGS deformation. **Red boxes** indicate zoomed areas; **cyan circles** marks defects.

Method	NeRF (Mildenhall et al. 2020) Scenes	DeformingNeRF (Xu and Harada 2022) Scenes
	training (sec.↓)	training (sec.↓)
DeformingNeRF (Xu and Harada 2022)	491.33	479.18
SuGaR (Guédon and Lepetit 2023)	3234.30	3166.88
GaMeS (Waczyńska et al. 2024)	432.02	469.65
Frosting (Guédon and Lepetit 2024)	2649.78	2673.52
Vanilla 3DGS (Kerbl et al. 2023)	460.19	462.38
Proposed Method	N/A	N/A

Table 4.1: Benchmark results comparing training times. **Red** indicates best values, **blue** marks second-best.

proposed algorithm. This selection was designed to test performance on challenging, extensive deformations and includes scenes with fine details (ficus) and smooth surfaces (hotdog) to assess robustness. The second set comprised the scenes and simpler, manually created cages from the benchmark setup of DeformingNeRF (Xu and Harada 2022), including chair, lego, robot, and toad. Using this set enabled a direct, standardized comparison with the original DeformingNeRF setup and evaluated performance on milder, artist-authored deformations.

In the first set, the ficus and hotdog scenes were chosen over the robot and toad scenes to test the method’s performance when scaling objects with significant details or splitting Gaussians representing flat surfaces. In terms of cages, the selected scenes’ cages were automatically generated and underwent extensive deformations, while DeformingNeRF’s cages were manually created and had milder deformations.

Training Speed Table 4.1 presents the average training time for all methods, while the detailed per-scene results are shown in Table 4.2. Training times of vanilla 3DGS were provided for reference. With a pre-trained vanilla 3DGS model or its variants, the proposed method can directly deform it without retraining or conversion, hence no training would be needed. In contrast, other approaches require retraining or conversion because they altered the architecture of 3DGS for editability.

Deformation Speed Table 4.3 presents the average deformation time for all methods, and the detailed per-scene results are provided in Table 4.4 and Table 4.5. As Table 4.3 shows, the proposed method, on average,

NeRF (Mildenhall et al. 2020) Scenes			DeformingNeRF (Xu and Harada 2022) Scenes		
Scene	Method	training (sec.↓)	Scene	Method	training (sec.↓)
chair	DeformingNeRF (Xu and Harada 2022)	451.15	chair	DeformingNeRF (Xu and Harada 2022)	451.15
	SuGaR (Guédon and Lepetit 2023)	3154.14		SuGaR (Guédon and Lepetit 2023)	3154.14
	GaMeS (Waczyńska et al. 2024)	427.86		GaMeS (Waczyńska et al. 2024)	427.86
	Frosting (Guédon and Lepetit 2024)	2651.98		Frosting (Guédon and Lepetit 2024)	2651.98
	Vanilla 3DGS (Kerbl et al. 2023)	417.16		Vanilla 3DGS (Kerbl et al. 2023)	417.16
	Proposed Method	N/A		Proposed Method	N/A
lego	DeformingNeRF (Xu and Harada 2022)	515.22	lego	DeformingNeRF (Xu and Harada 2022)	515.22
	SuGaR (Guédon and Lepetit 2023)	3264.54		SuGaR (Guédon and Lepetit 2023)	3264.54
	GaMeS (Waczyńska et al. 2024)	462.17		GaMeS (Waczyńska et al. 2024)	462.17
	Frosting (Guédon and Lepetit 2024)	2726.72		Frosting (Guédon and Lepetit 2024)	2726.72
	Vanilla 3DGS (Kerbl et al. 2023)	457.76		Vanilla 3DGS (Kerbl et al. 2023)	457.76
	Proposed Method	N/A		Proposed Method	N/A
hotdog	DeformingNeRF (Xu and Harada 2022)	621.38	robot	DeformingNeRF (Xu and Harada 2022)	439.98
	SuGaR (Guédon and Lepetit 2023)	3475.71		SuGaR (Guédon and Lepetit 2023)	3060.61
	GaMeS (Waczyńska et al. 2024)	401.03		GaMeS (Waczyńska et al. 2024)	407.78
	Frosting (Guédon and Lepetit 2024)	2606.45		Frosting (Guédon and Lepetit 2024)	2603.48
	Vanilla 3DGS (Kerbl et al. 2023)	535.54		Vanilla 3DGS (Kerbl et al. 2023)	400.25
	Proposed Method	N/A		Proposed Method	N/A
figus	DeformingNeRF (Xu and Harada 2022)	377.58	toad	DeformingNeRF (Xu and Harada 2022)	510.36
	SuGaR (Guédon and Lepetit 2023)	3042.79		SuGaR (Guédon and Lepetit 2023)	3188.21
	GaMeS (Waczyńska et al. 2024)	437.01		GaMeS (Waczyńska et al. 2024)	580.80
	Frosting (Guédon and Lepetit 2024)	2613.99		Frosting (Guédon and Lepetit 2024)	2711.88
	Vanilla 3DGS (Kerbl et al. 2023)	430.30		Vanilla 3DGS (Kerbl et al. 2023)	574.36
	Proposed Method	N/A		Proposed Method	N/A

Table 4.2: Detailed per-scene training time of Table 4.1. **Red** indicates best values, **blue** marks second-best.

Method	NeRF (Mildenhall et al. 2020) Scenes (Automatic Cages)		
	preprocess	deform	render
	(ms↓)	(ms↓/FPS↑)	(ms↓/FPS↑)
DeformingNeRF* (Xu and Harada 2022)	3530.10	3933.93 / 0.25FPS	4970.13 / 0.20FPS
SuGaR (Guédon and Lepetit 2023)	1197.03	1483.37 / 0.67FPS	17.26 / 57.94FPS
GaMeS (Waczyńska et al. 2024)	1517.75	8.34 / 119.90FPS	5.87 / 170.36FPS
Frosting (Guédon and Lepetit 2024)	1163.60	125.83 / 7.95FPS	17.27 / 57.90FPS
Proposed Method	3565.11	16.42 / 60.90FPS	4.12 / 242.72FPS

Method	DeformingNeRF (Xu and Harada 2022) Scenes (Manual Cages)		
	preprocess	deform	render
	(ms↓)	(ms↓/FPS↑)	(ms↓/FPS↑)
DeformingNeRF* (Xu and Harada 2022)	2642.48	2420.32 / 0.41FPS	3441.58 / 0.29FPS
SuGaR (Guédon and Lepetit 2023)	746.24	1474.52 / 0.68FPS	16.78 / 59.59FPS
GaMeS (Waczyńska et al. 2024)	1183.13	6.56 / 152.44FPS	6.24 / 160.26FPS
Frosting (Guédon and Lepetit 2024)	715.36	122.48 / 8.16FPS	16.62 / 60.17FPS
Proposed Method	2744.05	12.33 / 81.10FPS	3.97 / 251.89FPS

* DeformingNeRF performs deformation during rendering. Note that the render time involves the deformation time.

Table 4.3: Benchmark results comparing deformation times. **Red** indicates best values, **blue** marks second-best.

Scene	Method	NeRF Mildenhall et al. (2020) Scenes		
		preprocess (ms↓)	deform (ms↓/FPS↑)	render (ms↓/FPS↑)
chair	DeformingNeRF Xu and Harada (2022)	3361.99	2882.62 / 0.35FPS	3908.45 / 0.26FPS
	SuGaR Guédon and Lepetit (2023)	1055.86	1528.79 / 0.65FPS	17.04 / 58.69FPS
	GaMeS Waczyńska et al. (2024)	1355.31	7.25 / 137.93FPS	5.42 / 184.50FPS
	Frosting Guédon and Lepetit (2024)	1023.09	125.62 / 7.96FPS	16.36 / 61.12FPS
	Proposed Method	3124.88	13.89 / 71.99FPS	3.43 / 291.55FPS
lego	DeformingNeRF Xu and Harada (2022)	4075.17	4633.86 / 0.22FPS	5668.55 / 0.18FPS
	SuGaR Guédon and Lepetit (2023)	1510.96	1511.84 / 0.66FPS	16.91 / 59.14FPS
	GaMeS Waczyńska et al. (2024)	2188.09	12.09 / 82.71FPS	6.43 / 155.52FPS
	Frosting Guédon and Lepetit (2024)	1495.65	133.17 / 7.51FPS	16.10 / 62.11FPS
	Proposed Method	5148.76	23.71 / 42.18FPS	4.12 / 242.72FPS
hotdog	DeformingNeRF Xu and Harada (2022)	3500.92	3989.47 / 0.25FPS	5028.07 / 0.20FPS
	SuGaR Guédon and Lepetit (2023)	1234.86	1477.10 / 0.68FPS	17.46 / 57.27FPS
	GaMeS Waczyńska et al. (2024)	803.10	4.43 / 225.73FPS	4.30 / 232.56FPS
	Frosting Guédon and Lepetit (2024)	1156.40	122.38 / 8.17FPS	17.01 / 58.79FPS
	Proposed Method	1918.72	9.53 / 104.93FPS	3.25 / 307.69FPS
figus	DeformingNeRF Xu and Harada (2022)	3182.31	4229.77 / 0.24FPS	5275.44 / 0.19FPS
	SuGaR Guédon and Lepetit (2023)	986.44	1415.76 / 0.71FPS	17.61 / 56.79FPS
	GaMeS Waczyńska et al. (2024)	1724.51	9.59 / 104.28FPS	7.34 / 136.24FPS
	Frosting Guédon and Lepetit (2024)	979.27	122.16 / 8.19FPS	19.61 / 50.99FPS
	Proposed Method	4068.09	18.53 / 53.97FPS	5.68 / 176.06FPS

Table 4.4: Detailed per-scene deformation time of Table 4.3 on NeRF scenes.

Scene	Method	DeformingNeRF Xu and Harada (2022) Scenes		
		preprocess (ms↓)	deform (ms↓/FPS↑)	render (ms↓/FPS↑)
chair	DeformingNeRF Xu and Harada (2022)	3422.40	2910.13 / 0.34FPS	3936.68 / 0.25FPS
	SuGaR Guédon and Lepetit (2023)	1061.50	1486.58 / 0.67FPS	16.91 / 59.14FPS
	GaMeS Waczyńska et al. (2024)	1353.42	7.49 / 133.51FPS	5.43 / 184.16FPS
	Frosting Guédon and Lepetit (2024)	1018.24	126.04 / 7.93FPS	16.98 / 58.89FPS
	Proposed Method	3139.67	14.31 / 69.88FPS	3.71 / 269.54FPS
lego	DeformingNeRF Xu and Harada (2022)	2084.41	1197.90 / 0.83FPS	2219.81 / 0.45FPS
	SuGaR Guédon and Lepetit (2023)	441.99	1532.87 / 0.65FPS	16.83 / 59.42FPS
	GaMeS Waczyńska et al. (2024)	593.17	3.36 / 297.62FPS	6.17 / 162.07FPS
	Frosting Guédon and Lepetit (2024)	424.85	125.03 / 8.00FPS	15.84 / 63.13FPS
	Proposed Method	1378.66	6.18 / 161.81FPS	3.94 / 253.81FPS
robot	DeformingNeRF Xu and Harada (2022)	1895.59	1441.66 / 0.69FPS	2440.74 / 0.41FPS
	SuGaR Guédon and Lepetit (2023)	517.81	1421.84 / 0.70FPS	16.43 / 60.86FPS
	GaMeS Waczyńska et al. (2024)	706.71	3.86 / 259.07FPS	5.68 / 176.06FPS
	Frosting Guédon and Lepetit (2024)	510.92	117.94 / 8.48FPS	17.00 / 58.82FPS
	Proposed Method	1664.74	7.19 / 139.08FPS	3.94 / 253.81FPS
toad	DeformingNeRF Xu and Harada (2022)	3167.54	4131.60 / 0.24FPS	5169.09 / 0.19FPS
	SuGaR Guédon and Lepetit (2023)	963.66	1456.78 / 0.69FPS	16.95 / 59.00FPS
	GaMeS Waczyńska et al. (2024)	2079.23	11.51 / 86.88FPS	7.69 / 130.04FPS
	Frosting Guédon and Lepetit (2024)	907.43	120.92 / 8.27FPS	16.67 / 59.99FPS
	Proposed Method	4793.12	21.63 / 46.23FPS	4.29 / 233.10FPS

Table 4.5: Detailed per-scene deformation time of Table 4.3 on DeformingNeRF scenes.

achieved 60FPS on the simpler DeformingNeRF cages and the more challenging NeRF scene cages, hence achieving real-time deformation. In terms of once-per-scene preprocessing, the proposed approach was slower compared to mesh-based 3DGS methods like SuGaR (Guédon and Lepetit 2023), GaMeS (Waczyńska et al. 2024), and Gaussian Frosting (Guédon and Lepetit 2024). This is because the proposed method needs to process more points for deformation and splitting, while mesh-based methods can simply deform the underlying mesh or triangle soup. However, the proposed method was the fastest in rendering, as it uses the unmodified vanilla 3DGS rendering process.

This could also be seen from the per-scene results shown in Table 4.4 and Table 4.5. As can be seen, the proposed method was consistently the second-fastest in deformation and fastest in rendering.

4.4 Applications

The proposed method can be extended to work with other methods or 3DGS variants, making it valuable for many application scenarios. The following applications tested the hypothesis that the proposed method is a general and practical tool for the 3DGS ecosystem. Effectiveness was evaluated qualitatively based on successful integration with other methods and the preservation of high visual fidelity.

4.4.1 Composition and Animation

Since the proposed deformation algorithm enables cage-based deformation on 3DGS, it can be used to animate and compose different 3DGS models by animating their cages and placing the cages into another scene. This animation and composition can be done using existing mesh editing and animation software, such as Blender (Blender Online Community 2025).

Figure 4.9 shows animated 3DGS versions of Mixamo (Adobe 2024) characters Sophie and Mutant, composited into the MipNeRF360 (Barron et al. 2022) garden scene. This was done by animating the cage of the characters using the automatic rigging and animation functionality of Mixamo, placing

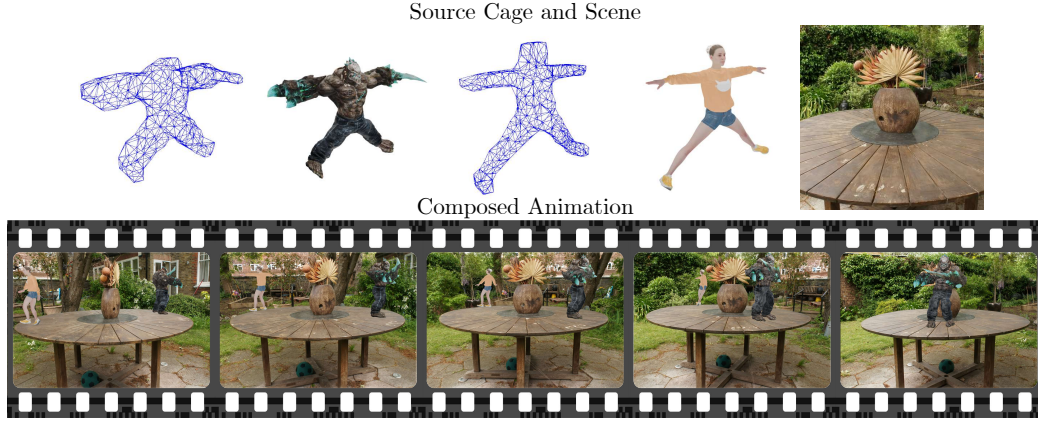


Figure 4.9: Example of animating and composing scenes using the proposed method.

the cages within the garden scene, and applying the proposed algorithm. As can be seen, the proposed deformation algorithm achieved high-quality animation and composition of 3DGS models, which is useful for animators and artists intending to work with 3DGS.

4.4.2 Combining with Editing Methods

The proposed deformation algorithm achieves low-level, direct shape editing and can be integrated with other work on editing or editable 3DGS to complement their editing abilities.

The experiment started by integrating with GaussianEditor (Chen et al. 2023b), a text-prompt-based 3DGS editing method. As shown in Figure 4.10, in the face scene from the Instruct-NeRF2NeRF Dataset (Haque et al. 2023), GaussianEditor was used to select the scene’s facial region and edit its appearance with the text prompt ”turn him into a clown”. The proposed deformation algorithm was then applied to stretch the nose of the edited face. As can be seen, the proposed method successfully stretched the nose of the edited face. This combination demonstrated how the proposed approach can be used to offer direct low-level control in addition to high-level text-prompt-based editing.



Figure 4.10: Integrating the proposed method with GaussianEditor (Chen et al. 2023b).

The proposed method can also be integrated with GSTex(Rong et al. 2024), a 2DGS variant that allows texture editing. This integration was tested using a chair from the NeRF Synthetic Dataset(Mildenhall et al. 2020). As shown in Figure 4.11, the chair’s texture was modified using GSTex, then the proposed deformation algorithm was applied to deform the result. This combination demonstrated how the proposed approach can be used to enhance the overall editing capabilities for both shape and color editing.

4.4.3 Extending to Variants of 3DGS

The proposed method can also be extended to work with other 3DGS variants.

This was first demonstrated by integrating with 2DGS (Huang et al. 2024), a method that uses flattened 2D Gaussian disks instead of 3D Gaussian balls for improved geometry, depth rendering, and normal reconstruction. This was shown by deforming a 2DGS capturing the lego scene from the NeRF Synthetic Dataset (Mildenhall et al. 2020). Results were shown in Figure 4.12; note that the depth and normal renders of the deformed model were of high quality, which demonstrated the proposed method’s effectiveness when applied to 3DGS variants.

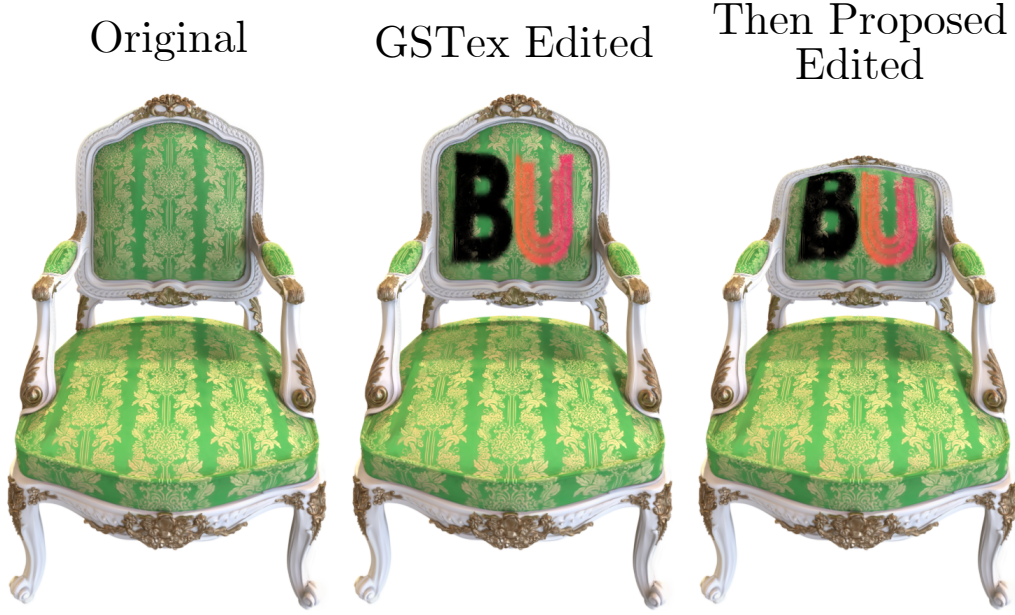


Figure 4.11: Integrating the proposed method with GSTex (Rong et al. 2024).

This was further shown by integrating with FLoD (Seo et al. 2024), a method that adds Level-of-Details(LoD) to 3DGS. The integration was tested by editing a FLoD capturing the garden scene from the MipNeRF360 dataset (Barron et al. 2022). As shown by Figure 4.13, deformation worked well across all LoDs, demonstrating the proposed method’s adaptability to these variants of 3DGS. This adaptability could allow the proposed method to utilize the available information across different models for different applications.

4.4.4 Ablation Studies

Cage-building Algorithm The ablation study started by analyzing the key steps in the proposed cage-building algorithm, using the vase segmented from the garden scene in the MipNeRF360(Barron et al. 2022) dataset. This analysis tested the hypotheses that T-SDF integration is critical for noise reduction and that the depth carving (filling) step is essential for creating a solid structure; the evaluation criteria were therefore the absence of noise

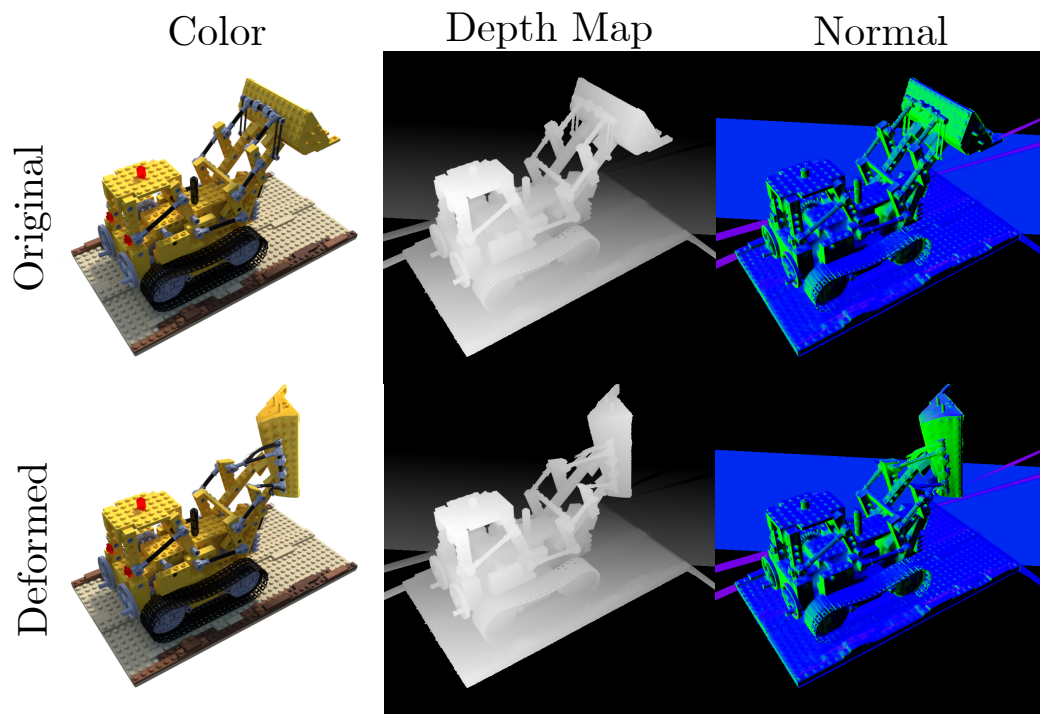


Figure 4.12: Integrating the proposed method with 2DGS (Huang et al. 2024).

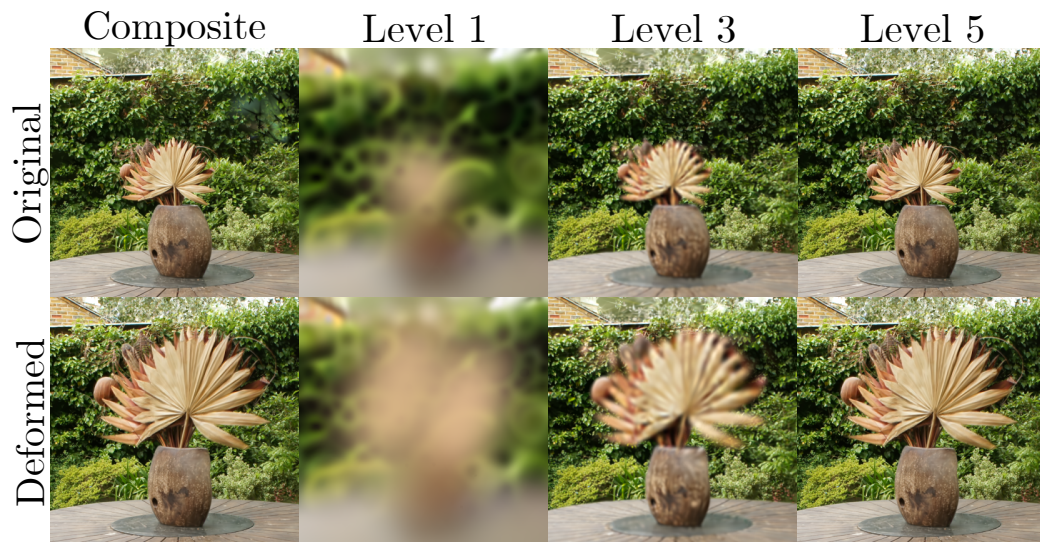


Figure 4.13: Integrating the proposed method with FLoD (Seo et al. 2024).

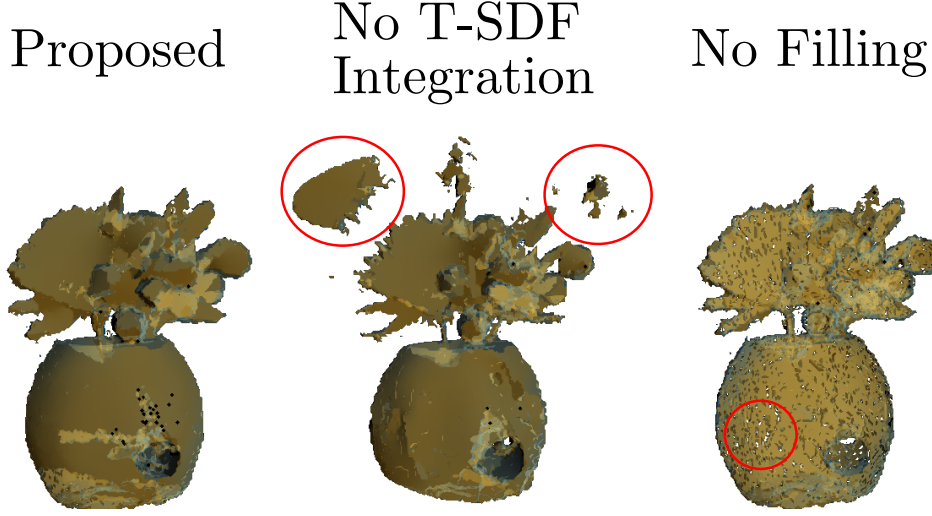


Figure 4.14: Ablation study results for cage building algorithm. The voxel grids before closing simplification are shown.

and hollow regions in the resulting voxel grid. The proposed method’s voxel grid extraction method was compared against the two baselines: a naive approach that skips T-SDF integration and directly constructs voxel grids by performing depth carving using 3DGS-rendered depth maps (“No T-SDF Integration”), and a version of the proposed algorithm that skips depth carving and uses the surface points extracted from T-SDF integration instead (“No Filling”).

Figure 4.14 shows the produced voxel grids prior to applying the closing operator. Using depth maps directly for voxel carving without T-SDF integration created noise on the vase’s top. Without filling, the resulting voxel grid was porous and hollow, which closing operations cannot effectively remedy. Both approaches produced suboptimal results and could harm the cage’s quality.

Deformation Algorithm To evaluate the design choices in the deformation algorithm, the proposed algorithm was compared with two simpler variants: one without splitting and another that directly applies cage-based deformation to the position (mean) of the gaussians. This ablation tested two hypotheses: that deforming the full covariance is essential, and that the

splitting procedure is crucial for sharp bends. The criteria for success were the absence of specific visual artifacts: striping for the covariance test, and spiking for the splitting test. These algorithms were tested on the lego and hotdog scenes from the NeRF Synthetic Dataset (Mildenhall et al. 2020). Figure 4.15 shows the results.

As can be seen, the simpler mean-only variant of the proposed algorithm produced significant striping artifacts. This is prevented in the proposed algorithm by transforming the covariance (thus, rotation and shape) of the Gaussians as well. Furthermore, as shown in the highlighted and zoomed-in regions, using the proposed algorithm without splitting resulted in spiking artifacts in bent regions. This occurs because the long belt is represented by a few thin, elongated ellipsoids. When bent, noticeable spiking artifacts appeared, as shown in Figure 4.15 (b). Note that when the deformed Gaussian was approximated using only one Gaussian, rather than splitting, noticeable spiking artifacts appeared as bending is challenging to approximate using a single Gaussian ellipsoid. This demonstrates that the splitting procedure is essential for handling large deformations. Simply optimizing the existing Gaussians, as proposed by ARAP-GS (Tong et al. 2022) or VR-GS (Jiang et al. 2024), is insufficient for accurately modeling bending.

Split Scaling Factor Finally, the choice of scaling factor k in the splitting step was analyzed, using the Lego scene from the NeRF Synthetic Dataset (Mildenhall et al. 2020). This ablation tested the hypothesis that an optimal value of k must balance two competing factors. The evaluation criteria were therefore (1) the avoidance of spiking artifacts, which indicates the factor is not too large, and (2) the maintenance of a continuous surface, which indicates the factor is not too small.

As shown in Figure 4.16, the scaling factor k significantly impacted results. A large k (e.g., 1.0) created spikes on the bulldozer’s belt due to oversized split Gaussians failing to model the intricate bending deformation. Conversely, a small k (e.g., 0.1) produced fragmented surfaces as the split Gaussians became too small to cover the belt. While the theoretical analysis suggests k should be $\sqrt[3]{\frac{1}{2}}$ (denoted as $\text{cbt}(0.5)$), this experiment showed $\frac{1}{2}$ appeared to produce superior results. This is likely caused by the fact that

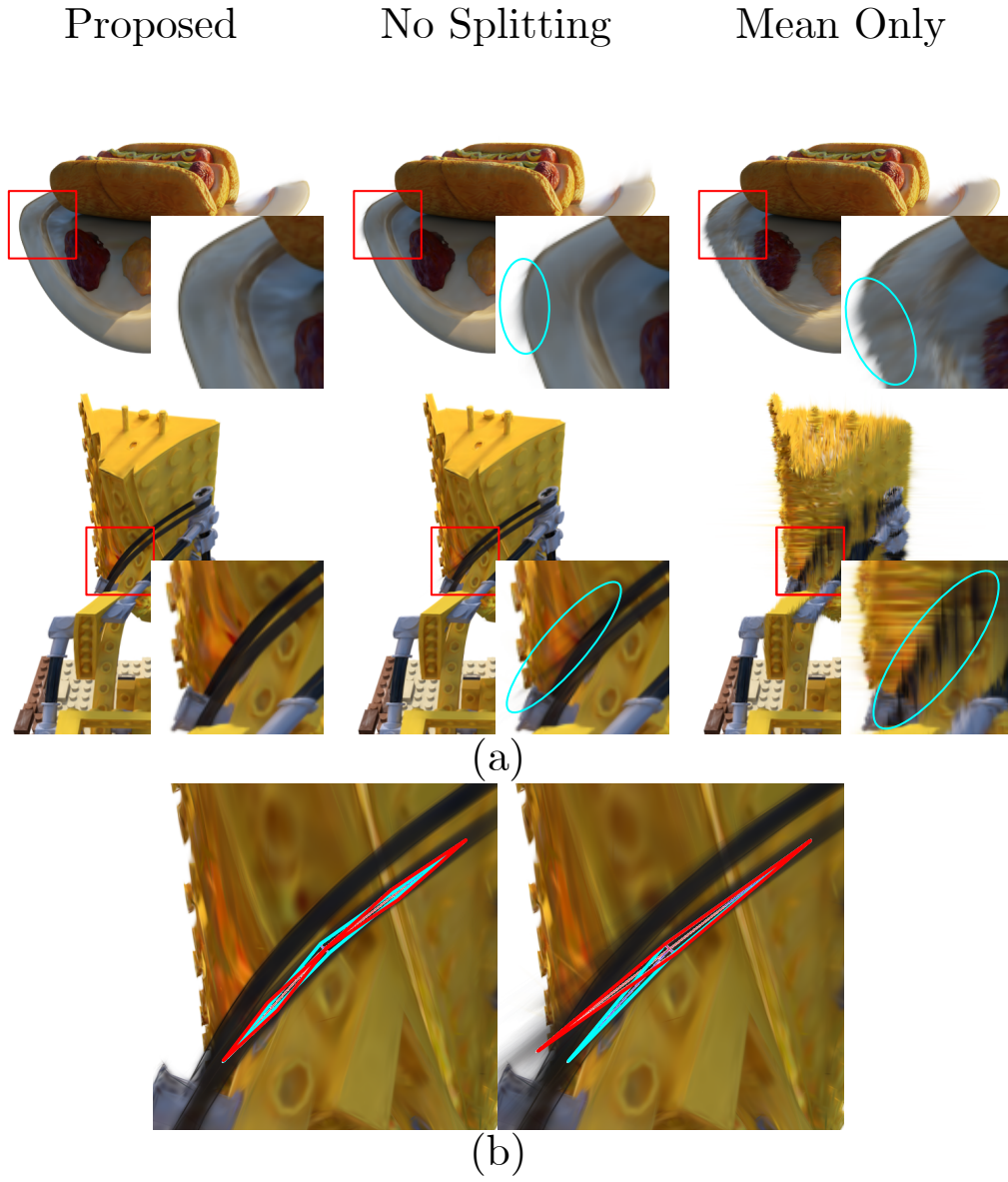


Figure 4.15: (a) Ablation study results for deformation algorithm. (b) The zoomed-in view of (a) highlights the deformed Gaussian (**cyan diamond**) and the actual Gaussians (**red diamond**).

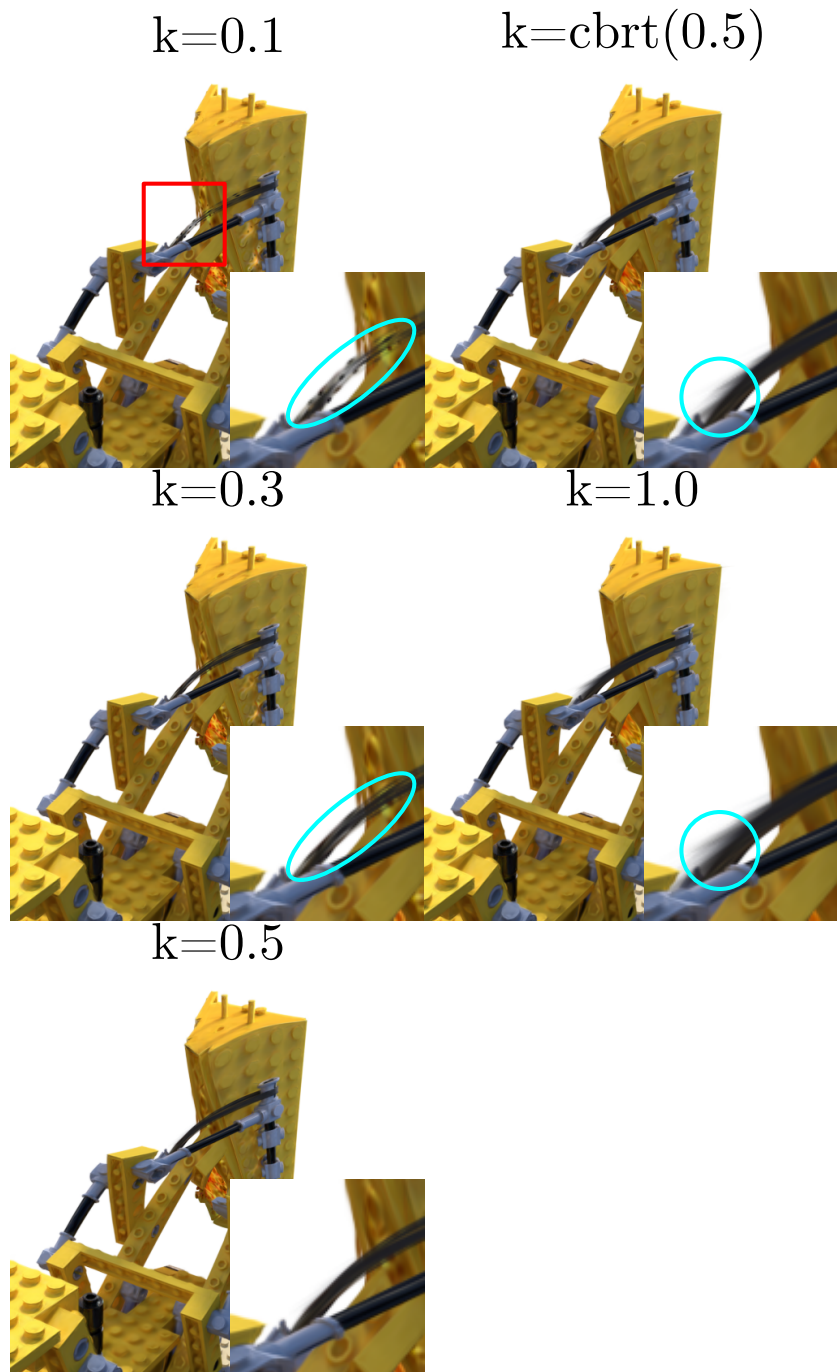


Figure 4.16: Ablation study results for the split factor k .

the spiking artifacts caused by overly long Gaussians are a more severe issue in practice than the minor imperfections in volume and shape preservation that result from using a smaller scaling factor. Thus, naively splitting the original Gaussians in half would be sufficient.

4.5 Summary

This chapter introduces GSDeformer, a cage-based deformation algorithm for 3D Gaussian Splatting (3DGS). The proposed approach can directly deform existing trained vanilla 3DGS in real time and can be easily extended to its variants. It adapts cage-based deformation for 3DGS by first building a proxy point cloud from the Gaussians and then transferring the point cloud’s deformation back to 3DGS, splitting the relevant Gaussians to handle bending. This approach requires no additional training data or architectural changes. This chapter also presents an algorithm that automatically constructs cages for 3DGS deformation.

Chapter 5

Targeted Scene Inpainting and Recomposition for 3DGS

5.1 Introduction

Following the segmentation of the target object in Chapter 3 and its subsequent manipulation in Chapter 4, the final step involves reintegrating the edited object back into the scene representation. This process, termed re-composition, often reveals previously occluded or under-trained surfaces at the object’s original position. Effectively addressing these exposed regions through inpainting is crucial for maintaining visual consistency and achieving realistic editing results.

This chapter addresses the challenges of object recomposition and, more significantly, the inpainting of exposed regions within the 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) framework. An overview of this process is presented in Figure 5.1, given a segmented scene, the segmentation is used to derive inpainted 2D scene images with a scene-content revealing pruning strategy. After using inpainted 2D images to fine-tune $\mathcal{R}_{bg}$, the edited $\mathcal{R}_{obj}$ can be composited back into $\mathcal{R}_{bg}$ in real-time. The explicit and point-based nature of 3DGS facilitates a straightforward recomposition process by combining the Gaussian primitives of the edited object and the background. More importantly, this representation enables tailored strategies for rapid and high-quality inpainting. Specifically, a scene-content revealing pruning strategy and a reprojection-based initialization scheme are introduced, both

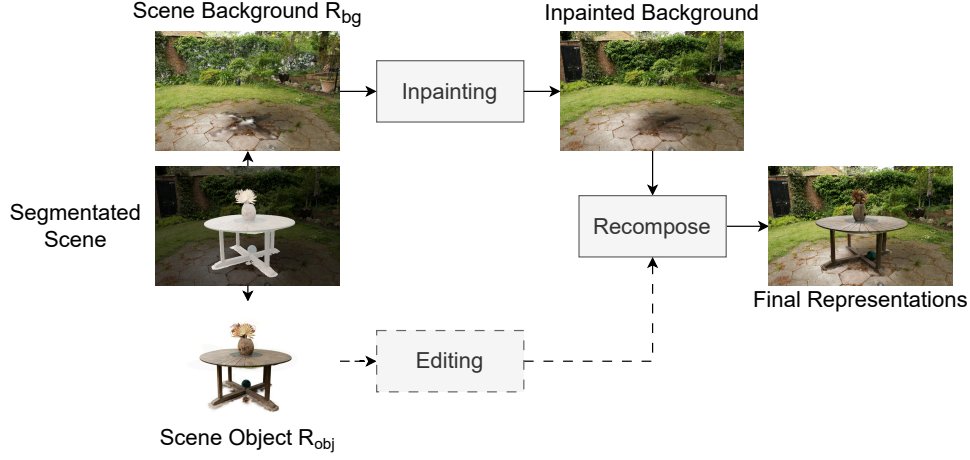


Figure 5.1: An overview of the inpainting and recomposition pipeline proposed in this chapter.

capitalizing on the structure of 3DGS to improve the quality and efficiency of exposed region inpainting. The inherent speed of 3DGS further provides a significant advantage, enabling rapid inpainting and a real-time editing experience.

The effectiveness of the proposed approach is demonstrated by editing objects in 360 and forward-facing scenes. The proposed method is also benchmarked against existing object removal and state-of-the-art inpainting techniques, showcasing competitive or superior performance in terms of visual quality and speed, despite being able to achieve free manipulation rather than just object removal.

In summary, the contributions specific to this chapter are as follows:

- A method for high-quality exposed region inpainting and object reposition on 3D Gaussian Splatting (3DGS) is presented, specifically designed for the scenario of arbitrary object manipulation.
- A rapid inpainting procedure is proposed, incorporating a scene-content revealing pruning strategy to minimize unnecessary changes as well as a

reprojection-based initialization scheme to enhance inpainting quality for exposed regions. Both contribute to high-quality results.

- Evaluations are performed to demonstrate the superior performance of the proposed inpainting approach compared to existing relevant methods.

5.2 Methodology

As demonstrated in Figure 5.1, after segmenting a trained 3D Gaussian Splatting scene $\mathcal{R}$ into the object component $\mathcal{R}_{\text{obj}}$ and the background component $\mathcal{R}_{\text{bg}}$, and subsequent editing operations applied to $\mathcal{R}_{\text{obj}}$, two challenges remain for achieving seamless scene manipulation. First, the removal or movement of $\mathcal{R}_{\text{obj}}$ leaves behind holes or newly exposed under-trained surfaces within $\mathcal{R}_{\text{bg}}$ that disrupt visual coherence. Second, the edited $\mathcal{R}_{\text{obj}}$ must be integrated back into the modified $\mathcal{R}_{\text{bg}}$ to form the final composed scene.

This chapter addresses these challenges by detailing the proposed approach for exposed surface inpainting within $\mathcal{R}_{\text{bg}}$ and the object recomposition process. The approach first focuses on repairing the background $\mathcal{R}_{\text{bg}}$ by generating inpainted 2D views and leveraging them to fine-tune the 3D representation. Subsequently, the edited object $\mathcal{R}_{\text{obj}}$ is recomposed into the inpainted background $\mathcal{R}_{\text{bg}}$. This approach leverages the explicit nature of the 3D Gaussian Splatting representation throughout the process for efficiency and effectiveness.

5.2.1 Exposed Surface Inpainting

After separating $\mathcal{R}_{\text{obj}}$ from $\mathcal{R}_{\text{bg}}$, the background component $\mathcal{R}_{\text{bg}}$ will have regions exposed by the removal of $\mathcal{R}_{\text{obj}}$. These regions appear as holes or under-trained surfaces, and part of them might remain exposed after editing and recomposing the object back. To ensure realistic rendering results of the final recomposed scene, these defects within $\mathcal{R}_{\text{bg}}$ must be inpainted.

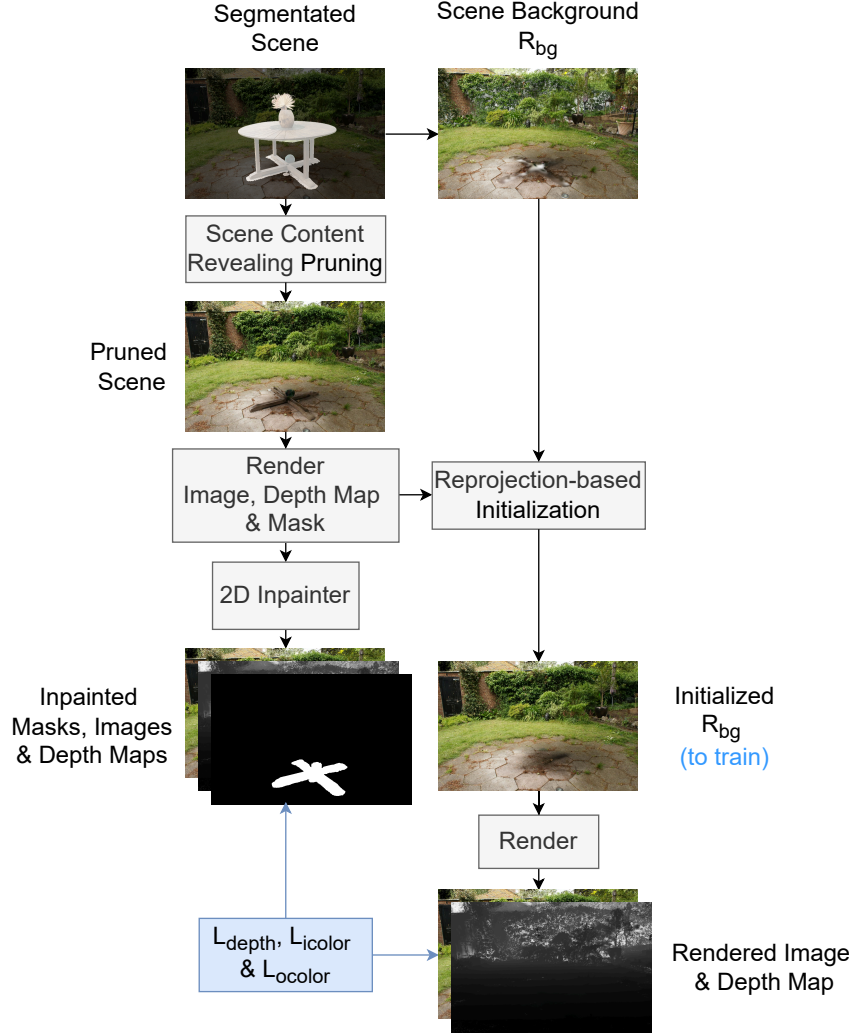


Figure 5.2: Overview of the proposed 2D-guided 3D inpainting stage.

As illustrated in Figure 5.2, the proposed strategy starts by performing scene content revealing pruning to reveal already captured backgrounds, minimizing the areas to inpaint. Segmentation masks, depth maps, and images of the pruned scene are then rendered, and the images and depth maps are inpainted using 2D inpainting models and the rendered masks. With the inpainted results, one pair of the inpainted depth map and color images is used

to initialize additional Gaussians in $\mathcal{R}_{\text{bg}}$, which is then fine-tuned using the full set of inpainted depth maps and images, supervised using the proposed losses.

Scene Content-Revealing Pruning & 2D Inpainting Direct 3D inpainting of volumes and surfaces in $\mathcal{R}_{\text{bg}}$ is challenging. Instead, the approach presented in this chapter guides the 3D inpainting process using information derived from inpainted 2D images and depth maps rendered from multiple viewpoints. The generation of this 2D guidance data involves careful mask creation and the application of a 2D inpainting model.

To minimize the area requiring inpainting and preserve as much of the original scene content as possible, this chapter proposes the "scene content-revealing pruning" mask generation strategy. First, within the object Gaussians $\mathcal{R}_{\text{obj}}$, Gaussians belonging to $\mathcal{R}_{\text{obj}}$ that are distant from the $\mathcal{R}_{\text{bg}}$ Gaussians are identified and temporarily removed. This step effectively selects those object Gaussians situated near the object-background interface. Rendering the segmentation mask from this pruned representation helps identify the primary regions requiring inpainting, namely those that are close to the contact area of the object and the background, while not touching background areas that are simply occluded by the object. Second, this mask is augmented by identifying pixels in $\mathcal{R}_{\text{bg}}$ RGB renders that exhibit characteristics of holes or exposed areas, such as low accumulated alpha (opacity) or color values close to a predefined background color.

Finally, these initial masks may contain small holes or spurious unmasked regions within the main area intended for inpainting. To ensure a contiguous and well-covered mask suitable for the 2D inpainter, the masks are refined using an algorithm inspired by the refinement procedure proposed in OR-NeRF (Yin et al. 2023). The process involves dilating the mask slightly to close small gaps, identifying all distinct contours within the dilated mask, selecting the contour that encloses the largest area (most likely the main object region), and finally creating a solid mask by filling this largest contour. This procedure effectively eliminates internal holes and produces a clean mask $\mathbf{M}_i$ for each view i .

This mask generation strategy, combined with mask refinement, ensures that the subsequent 2D inpainting focuses only on the necessary, object-scene contact areas, maximizing the retention of existing scene details, which leads to more realistic and coherent results.

Along with the final refined masks, corresponding RGB images and depth maps are rendered from the background 3DGS scene model $\mathcal{R}_{\text{bg}}$ (merged with the pruned $\mathcal{R}_{\text{obj}}$). These images, depths, and masks are then fed into a standard 2D inpainting network (e.g., LaMa (Suvorov et al. 2022)) to produce inpainted images and depth maps.

3D Rapid Fine-tuning With the set of inpainted 2D images $\{\mathbf{I}_i\}$ and depth maps $\{\mathbf{D}_i\}$ corresponding to various camera views $\{\mathbf{C}_i\}$, the background representation $\mathcal{R}_{\text{bg}}$ is subsequently fine-tuned to incorporate the inpainted information.

Initialization To accelerate fine-tuning and improve the quality of the fine-tuned geometry and appearance, the process is initialized by adding new Gaussians to $\mathcal{R}_{\text{bg}}$. As depicted in Figure 5.2, these new Gaussians are created by reprojecting the pixel information (color from $\mathbf{I}_i$, depth from $\mathbf{D}_i$) within the masked regions $\mathbf{M}_i$ of a selected view back into 3D space. This provides a reasonable starting point for the optimization, aligning with the principle that good initialization is crucial for 3DGS training (Kerbl et al. 2023).

Optimization Losses The parameters of the Gaussians in $\mathcal{R}_{\text{bg}}$ (including the backprojected ones) are then optimized using a composite loss function. The loss functions are designed to carefully integrate the inpainted information, ensuring multi-view inconsistent inpainted images lead to consistent 3D results, and maximally preserve the existing background structure.

Outside Mask Color Loss ($\mathcal{L}_{\text{ocolor}}$) For regions outside the inpainted mask ($1 - \mathbf{M}_i$), consistency with the original rendered colors is enforced. This is achieved using a combination of L1 loss and the Structural Similarity Index Measure (SSIM) (Wang et al. 2004), balancing pixel-level accuracy and perceptual similarity:

$$\begin{aligned} \mathcal{L}_{\text{ocolor}} = & (1 - \lambda_{\text{ssim}}) \text{L1}(\mathbf{I}_i(1 - \mathbf{M}_i), \mathbf{I}_r(1 - \mathbf{M}_i)) \\ & + \lambda_{\text{ssim}} \text{SSIM}(\mathbf{I}_i(1 - \mathbf{M}_i), \mathbf{I}_r(1 - \mathbf{M}_i)) \end{aligned} \quad (5.1)$$

where $\mathbf{I}_r$ is the image rendered from the current state of $\mathcal{R}_{\text{bg}}$, and λ_{ssim} is a weighting factor.

Depth Loss ($\mathcal{L}_{\text{depth}}$) To ensure geometric consistency, particularly within the inpainted regions, the rendered depth map $\mathbf{D}_r$ is supervised against the inpainted depth map $\mathbf{D}_i$ using an L1 loss across the entire view:

$$\mathcal{L}_{\text{depth}} = \lambda_{\text{depth}} \text{L1}(\mathbf{D}_i, \mathbf{D}_r) \quad (5.2)$$

where λ_{depth} is the loss weight. This guides the 3D geometry to conform to the structure suggested by the 2D inpainted depth.

Inside Mask Color Loss ($\mathcal{L}_{\text{color}}$) For the color within the inpainted region (defined by mask $\mathbf{M}_i$), directly enforcing pixel-wise consistency (e.g., L1 or L2 loss) can be detrimental. This is because 2D inpainting methods may not produce multi-view consistent results. Strict enforcement could lead to blurry or inconsistent results. Therefore, the LPIPS (Zhang et al. 2018) perceptual loss is adopted, which measures perceptual similarity rather than exact pixel values:

$$\mathcal{L}_{\text{color}} = \lambda_{\text{lips}} \text{LPIPS}(\text{Box}(\mathbf{I}_i, \mathbf{M}_i), \text{Box}(\mathbf{I}_r, \mathbf{M}_i)) \quad (5.3)$$

where $\text{Box}(\mathbf{I}, \mathbf{M})$ extracts the portion of image $\mathbf{I}$ within the bounding box of mask $\mathbf{M}$. Applying the loss over the bounding box rather than strictly within the mask encourages better blending between the inpainted region and its immediate surroundings. λ_{lips} weights this perceptual loss term.

The total loss for fine-tuning $\mathcal{R}_{\text{bg}}$ is the sum of these components. The representation is trained over all the inpainted views, one view at a step. Optimizing $\mathcal{R}_{\text{bg}}$ with this loss and data effectively transfers the inpainted appearance and geometry from the 2D guidance data to the 3D representation.

5.2.2 Object Recomposition

Once the background $\mathcal{R}_{\text{bg}}$ has been inpainted using the fine-tuning process described above, the object $\mathcal{R}_{\text{obj}}$ being manipulated as desired, the final step is to recompose the scene.

Leveraging the explicit nature of 3D Gaussian Splatting, editing is straightforward and computationally inexpensive. Apart from the editing operations detailed in Chapter 4, simple affine transformations (translation $\mathbf{t}$, rotation $\mathbf{Q}$) can be applied directly to the parameters of the Gaussians constituting $\mathcal{R}_{\text{obj}}$. Specifically, the mean (position) $\boldsymbol{\mu}_j$ and rotation quaternion $\mathbf{q}_j$ of each Gaussian $j \in \mathcal{R}_{\text{obj}}$ are updated according to the desired transformation. Let the transformed object representation be denoted as $\mathcal{R}'_{\text{obj}}$.

The final composite scene representation $\mathcal{R}_{\text{final}}$ is simply obtained by taking the union of the Gaussians from the inpainted background $\mathcal{R}_{\text{bg}}$ and the transformed object $\mathcal{R}'_{\text{obj}}$:

$$\mathcal{R}_{\text{final}} = \mathcal{R}_{\text{bg}} \cup \mathcal{R}'_{\text{obj}} \quad (5.4)$$

Crucially, this recomposition step requires no further training or optimization. Because the background $\mathcal{R}_{\text{bg}}$ has already been inpainted, and the transformations on $\mathcal{R}_{\text{obj}}$ are applied directly to the Gaussian parameters, the combined representation $\mathcal{R}_{\text{final}}$ can be rendered immediately using the standard 3DGS rasterization pipeline (Equation 2.2). This allows for real-time interaction and visualization of the edited scene.

In summary, the methods presented in this chapter, namely the scene content-revealing pruning strategy, the 2D-guided 3D inpainting of the background, and the simple set union for rapid recomposition, fully exploit the explicit nature of 3D Gaussian Splatting. This enables seamless reintegration of the edited object into a realistically completed background, contributing to achieving high-quality, real-time 3D scene manipulation.

5.3 Experiments

The experiments in this section were designed to evaluate the effectiveness of the proposed method. Here, effectiveness was defined as the ability to perform high-quality, view-consistent object manipulation and inpainting in complex 3D scenes, while maintaining a significant speed advantage over NeRF-based approaches. The central hypothesis was that by leveraging a 2D inpainting model for initialization and a tailored fine-tuning process on

the 3D Gaussian Splatting representation, the method could achieve superior inpainting quality compared to existing state-of-the-art methods. The proposed method was tested on both 360 and forward-facing scene datasets. The method was also compared to existing scene object removal methods, reporting competitive performance in inpainting, despite being able to do more than just removal and having a speed advantage. A comparison was also made against state-of-the-art 3DGS inpainting methods. Finally, this section concludes with ablation studies to highlight the key contributions of the proposed method.

5.3.1 Implementation Details

For inpainting, a state-of-the-art 2D image inpainting model, Lama (Suvorov et al. 2022), was employed for inpainting the color image and depth map. The scene inpainting training process also used the SGD optimizer with learning rates equal to what was specified in the original 3D Gaussian Splatting paper (Kerbl et al. 2023). λ_{ssim} was 0.2, λ_{depth} was 1.0, λ_{lpiips} was 1.0. All experiments were conducted using a single A5000.

5.3.2 Experiment Setups

To ensure a comprehensive evaluation, the experiments were conducted on diverse datasets that tested different aspects of the proposed method. For diversity in evaluation, the proposed method was tested on the MipNeRF360 (Barron et al. 2022) dataset for 360 scenes and the SPIn-NeRF (Mirzaei et al. 2023b) dataset for forward-facing scenes. The MipNeRF360 dataset was chosen as it is a standard and challenging benchmark for 360-degree, unbounded scenes, enabling evaluation of the method’s robustness in complex environments. The SPIn-NeRF dataset was selected because it is specifically designed for the task of scene object removal, providing ground truth data that is crucial for quantitative comparison. Both datasets consist of captured images of a scene with their associated camera parameters, which can be used to train the model to edit. As a dataset for evaluating object removal methods, the SPIn-NeRF dataset also contains scene images without the

objects to remove and object segmentation masks for all captured images. For input prompt points, the point annotations from OR-NeRF (Yin et al. 2023) were used.

To quantitatively assess the performance of inpainting, standard image quality metrics were adopted. For the image quality of inpainted scenes, the mean PSNR and LPIPS (Zhang et al. 2018) scores between the rendered image of the inpainted scene representation and the ground truth scene image where the object is removed are reported. The Frechet inception distance (FID) (Heusel et al. 2017) between all rendered images and ground truth images taken without the target object was also calculated for image quality.

To contextualize the performance and evaluate the inpainting ability of the proposed method, it was compared against several state-of-the-art baselines in scene object removal: SPIn-NeRF (Mirzaei et al. 2023b) and OR-NeRF (Yin et al. 2023). More concretely, a comparison was made against the TensorRF variant of OR-NeRF, which is the fastest and the best overall result quality variant presented in the paper. A comparison was also made against recent state-of-the-art 3DGS inpainting methods, namely InFusion (Liu et al. 2024) for inpainting.

5.3.3 Object Manipulation

To demonstrate the target effect of the entire process detailed in the three chapters, manipulation was performed with the proposed method on scenes from the forward-facing SPIn-NeRF dataset and the 360 MipNeRF360 dataset. This qualitative evaluation tested the hypothesis that the complete pipeline was effective in producing visually coherent and high-quality manipulation results across different scene complexities. The qualitative results are presented in Figure 5.3. The figure shows the input prompts, the rendered segmentation mask, and the final editing results of the whole pipeline on both 360 and forward-facing scenes. It can be noted that the proposed method can perform high-quality selection and editing on various scenes, with exposed regions properly inpainted. The proposed method produces sensible results even for cases with extreme input, such as the red camping chair scene in



Figure 5.3: Qualitative results of the proposed object manipulation pipeline.

Figure 5.3. Scene changes were also minimized due to the scene content-revealing pruning strategy; note how the scene content behind the vase in the leftmost column is not inpainted, leading to higher quality manipulation results.

5.3.4 Evaluating Inpainting Quality

The inpainting ability of the proposed method was also evaluated on the SPIn-NeRF dataset. For this evaluation, effectiveness was defined by quantitative image quality metrics and qualitative visual realism, with the hypoth-

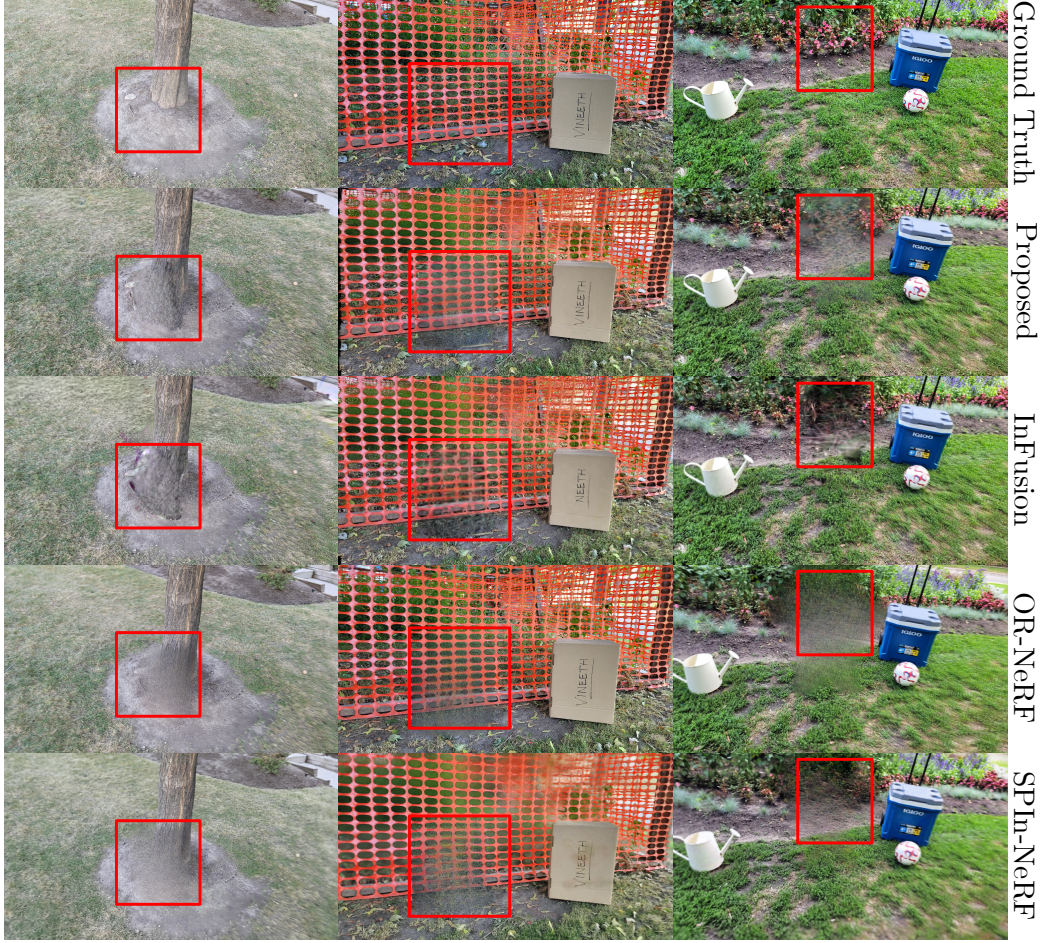


Figure 5.4: Qualitative comparison of inpainting results.

esis that the method would outperform state-of-the-art baselines. In addition to comparing with NeRF object removal methods such as SPIn-NeRF (Mirzaei et al. 2023b) and OR-NeRF (Yin et al. 2023), a comparison was also made against Infusion (Liu et al. 2024), a state-of-the-art 3DGS inpainting method. Qualitative results are presented in Figure 5.4, the quantitative results are in Table 5.1, and the per-scene quantitative results are in Table 5.2 and Table 5.3.

As presented in Table 5.1, the proposed method achieved superior quality compared to all existing methods; the best value of each column is bolded for ease of reading. For the per-scene results, where the best metrics are also

method	Inpainting Metrics			
	PSNR($\uparrow$)	FID($\downarrow$)	LPIPS($\downarrow$)	Inpainting Time($\downarrow$)
Proposed	19.83	40.33	0.2684	20.41 mins
InFusion	19.08	80.61	0.3424	<1 min
OR-NeRF	13.94	49.91	0.6162	168.97 mins
SPIn-NeRF	14.83	67.26	0.6506	58.25 mins

Table 5.1: Quantitative comparison of inpainting performance. The best value of each column is bolded for ease of reading.

Scene	PSNR($\uparrow$)				LPIPS($\downarrow$)			
	Proposed	OR-NeRF	SPIn-NeRF	InFusion	Proposed	OR-NeRF	SPIn-NeRF	InFusion
2	18.48	15.94	16.69	18.28	0.4544	0.7909	0.8489	0.5377
3	18.04	11.42	12.08	16.98	0.2217	0.4937	0.5472	0.2829
4	20.88	13.02	14.90	19.68	0.3229	0.6684	0.6815	0.3851
7	21.40	14.37	15.34	19.27	0.2858	0.6445	0.6552	0.3308
10	19.75	12.89	12.73	19.04	0.2264	0.6165	0.7003	0.2765
12	16.62	11.40	12.39	16.04	0.3352	0.7179	0.7518	0.3610
book	22.28	15.88	17.84	21.59	0.2147	0.5094	0.4226	0.2729
trash	21.18	16.63	16.70	21.79	0.2301	0.4882	0.5972	0.2921
mean	19.83	13.94	14.83	19.08	0.2864	0.6162	0.6506	0.3424

Table 5.2: Per-scene PSNR and LPIPS comparison for inpainting. The per-scene best metrics are bolded for ease of reading.

Scene	FID(↓)			
	Proposed	OR-NeRF	SPIn-NeRF	InFusion
2	53.60	72.10	71.75	223.79
3	36.39	34.72	68.35	62.00
4	51.78	74.04	61.10	67.84
7	22.48	38.66	43.95	60.77
10	21.67	43.89	91.73	28.32
12	26.23	38.02	50.52	32.20
book	81.68	64.91	102.71	114.02
trash	28.84	32.96	47.98	55.92
mean	40.33	49.91	67.26	80.61

Table 5.3: Per-scene FID comparison for inpainting. The per-scene best metrics are bolded for ease of reading.

bolded, results for SPIn-NeRF and OR-NeRF were taken from their original papers, while results for InFusion were reproduced from their published code, and the proposed method demonstrated the best performance across most scenes.

The qualitative results in Figure 5.4 present the ground truth and the output of all methods, with the inpainted region marked with red rectangles. As can be seen, the proposed method produces more plausible results than existing methods. For example, in the right-most column, the proposed method produces results closest to ground truth, while InFusion and SPIn-NeRF fill it with a dark hole, and OR-NeRF produces a blurry patch. In the middle column, the proposed method is the only one that correctly and sharply reproduces the structure of the plastic net, while InFusion fails, and OR-NeRF and SPIn-NeRF produce blurrier results.

The proposed method also had a significant speed advantage when compared against NeRF-based methods. As presented in the rightmost column of Table 5.1, the proposed method was about three times faster than the fastest NeRF-based method despite achieving superior result quality. The proposed method is slower than InFusion, but this is compensated by better

quality and only needs to be done once before all editing operations.

In this experiment, the superiority of the proposed method is attributed to the adopted representation’s superior representation capability and the proposed initialization and fine-tuning scheme. Good initialization and representation give clearer results and enables faster fine-tuning. Extensive fine-tuning allows the proposed method to achieve quality superior to InFusion at the cost of speed, as InFusion relies on reprojection using predicted depth.

5.3.5 Ablation Studies

Content-Revealing Scene Pruning This study investigated the effectiveness of the scene content-revealing pruning strategy, where effectiveness was defined by its ability to preserve the quality of unedited scene regions. The hypothesis was that this strategy was a critical component for preventing visual degradation. The importance of scene content-revealing pruning was validated by running the proposed algorithm with and without this process and comparing the final result against a pseudo-ground truth. The pseudo-ground truth was created by just removing the segmented object from the scene without any additional inpainting or fine-tuning.

This was tested by removing the table in the MipNeRF360 (Barron et al. 2022) garden scene. The results are presented in Figure 5.5. The figure shows the results of the proposed method with (bottom left) and without (bottom right) pruning, along with the pseudo ground truth (top). As can be seen from the region highlighted by red rectangles, not performing pruning leads to blurry results, especially in places that do not require inpainting. The quantitative results are also presented in Table 5.4, which show that pruning makes the result much more similar to the pseudo-ground truth and leads to better quality.

Inpainting Optimization Initialization This ablation analyzed the effectiveness of the proposed initialization scheme, based on the assumption that it was essential for guiding the optimization to produce artifact-free results. The effect of initialization was analyzed by disabling it and directly



Figure 5.5: Qualitative ablation of scene content-revealing pruning strategy.

method	Image Quality Metrics		
	PSNR($\uparrow$)	FID($\downarrow$)	LPIPS($\downarrow$)
w/ pruning	27.90	75.58	0.0911
w/o pruning	25.70	97.00	0.1561

Table 5.4: Quantitative ablation of scene content-revealing pruning strategy.

training the pruned scene. Figure 5.6 provides a close-up view of the output, comparing fine-tuning performed with the proposed reprojection-based initialization (left) versus without it (right). As shown in the figure, without proper initialization, the fine-tuning process left a cloud of tiny floating gaussians, highlighted by red rectangles, instead of completely optimizing them away.



Figure 5.6: Qualitative ablation of scene fine-tuning, performed with (left) or without (right) reprojection-based initialization.

5.4 Summary

This chapter focused on the final stage of the proposed 3DGS scene editing pipeline: exposed region inpainting and object recomposition. For exposed region inpainting, this chapter proposes the scene-content revealing pruning process to minimize changes to the scenes and a 2D-guided 3D fine-tuning process to leverage state-of-the-art 2D inpainting models to perform 3D inpainting of the exposed region. For object recomposition, a simple set-union operation is proposed to integrate the edited object back into the scene. Crucially, both the inpainting and recomposition processes fully leverage the efficiency and the explicit nature of 3DGS (Kerbl et al. 2023), enabling efficient optimization, high-quality inpainted results, and rapid integration of the edited object. These facets collectively deliver a user-friendly interactive editing experience with high-quality results.

Chapter 6

Conclusion and Future Work

Finally, this thesis concludes with a summary of the accomplished research work and a discussion of possible directions for future study.

6.1 Review of Accomplished Research

On a high level, this thesis aims to bridge the critical gap between the novel scene representation named 3D Gaussian Splatting (3DGS) and the ability to perform intuitive, robust scene manipulation. To this end, the 3DGS representation is connected with classical shape manipulation methods and recent advancements in modern 2D vision models.

The objectives proposed in Chapter 1 are achieved through the following key contributions in the respective chapters:

- Chapter 3 focuses on segmentation; the proposed segmentation method transfers the powerful, open-world segmentation capabilities of modern 2D vision models for 3DGS segmentation via a cross-view self-prompting mask propagation process. A dual-stage design was employed to ensure comprehensive coverage of object internals. A critical learning is that the proposed dual-stage design enables the segmentation process to effectively include primitives inside objects, which is essential for manipulation.
- Chapter 4 focuses on deformation; it presents a cage-based deformation framework for 3DGS. Classical cage-based deformation techniques are

connected to 3DGS through a proxy point cloud representation. The proposed framework performs deformation without retraining or conversion, runs in real-time, and generalizes to variants of 3DGS. An automatic cage-building algorithm based on a render-reconstruct-simplify paradigm is also proposed to produce high-quality cages for manipulation. A key insight is that such proxy representation could be effective for adapting classical shape manipulation methods to the novel volumetric primitives of 3DGS.

- Chapter 5 focuses on inpainting and recomposition, it leverages 2D inpainting models to achieve coherent and high-fidelity 3D scene inpainting for 3DGS, overcoming their multiview inconsistencies via a perceptual-based loss. A scene content revealing pruning strategy is also proposed to minimize unnecessary scene changes and thus improving quality. High-quality recomposition is enabled through a set union operation. A critical learning is that leveraging existing scene information in the trained representation is crucial to minimizing unnecessary scene changes during inpainting and thus improves quality.

Collectively, the contributions presented in this thesis provide a cohesive and effective toolkit for intuitive segmentation, real-time deformation, and high-quality inpainting and recomposition within the 3D Gaussian Splatting framework. The proposed methods also bridges 3DGS with classical shape manipulation methods and advancements in 2D vision models, opening up avenues for further research and development in this area.

6.2 Future Work

While this thesis presents a comprehensive pipeline for 3DGS scene capture manipulation and explores connections between 3DGS, classical shape manipulation methods, and 2D vision models, several promising avenues for future research remains:

- For segmentation, the proposed method relies on iterative gradient descent over rendered masks to train segmentation scores and assumes that prompt points broadly cover the object structure. Methods that directly model this process as an optimization problem, such as Flash-Splat (Shen et al. 2024b), may offer significant advantages in both speed and quality. Furthermore, exploring segmentation methods that utilize alternative or more robust prompting modalities could eliminate the need for precise prompt point selection.
- For deformation, the proposed method primarily targets large-scale changes via cage-based manipulation and utilizes a heuristic splitting mechanism to address potential spiking artifacts. Future work could integrate point-based deformation techniques to offer granular control, which would be particularly beneficial for subtle edits. Additionally, adopting a more rigorous optimization-based approach to address the bending Gaussian requirement behind spiking artifacts could yield further quality improvements over the proposed splitting heuristic.
- For inpainting and recomposition, the current pipeline does not explicitly account for shadows casted by the manipulated object. This can lead to overly dark inpainted regions, as existing shadows near the exposed areas can mislead 2D inpainting models. Future work could address this by incorporating shadow-aware segmentation or advanced inpainting strategies capable of inferring and reconstructing consistent illumination and material properties within the inpainted areas.
- Finally, conducting evaluations on larger and more diverse datasets would further analyze the effectiveness and generalizability of the proposed methods.

References

- Adobe, I., 2024. Mixamo: Animate 3d characters for games, film, and more. <https://www.mixamo.com>.
- Arvanitis, G., Moustakas, K., Fakotakis, N. and Lalos, A. S., 2018. 3d mesh inpainting using matrix completion via augmented lagrange multiplier method. *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, 1–5. URL <https://api.semanticscholar.org/CorpusID:52147652>.
- Bao and Yang, Junyi, Z., Hujun, B., Yinda, Z., Zhaopeng, C. and Guofeng, Z., 2022. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. *European Conference on Computer Vision (ECCV)*.
- Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P. and Hedman, P., 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*.
- Baumgärtner, L., Bergmann, R., Herzog, R., Schmidt, S., Vidal-N'unez, J. and Weiß, M., 2020. Mesh denoising and inpainting using the total variation of the normal and a shape newton approach. *SIAM J. Sci. Comput.*, 47, 300–. URL <https://api.semanticscholar.org/CorpusID:268363272>.
- Bhalgat, Y., Laina, I., Henriques, J. F., Zisserman, A. and Vedaldi, A., 2023. Contrastive lift: 3d object instance segmentation by slow-fast contrastive fusion. *arXiv preprint arXiv:2306.04633*.

- Blender Online Community, 2025. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam. URL <http://www.blender.org>.
- Blomqvist, K., Ott, L., Chung, J. J. and Siegwart, R., 2023. Baking in the feature: Accelerating volumetric segmentation by rendering feature maps. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 7629–7634.
- Botsch, M., Kobbelt, L., Pauly, M., Alliez, P. and Levy, B., 2010. *Polygon Mesh Processing*. A K Peters/CRC Press, 0 edition. URL <https://www.taylorfrancis.com/books/9781439865316>.
- Calderon, S. and Boubekur, T., 2017. Bounding proxies for shape approximation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36 (5).
- Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P. and Joulin, A., 2021. Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE/CVF international conference on computer vision*, 9650–9660.
- Cen, J., Fang, J., Yang, C., Xie, L., Zhang, X., Shen, W. and Tian, Q., 2023a. Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*.
- Cen, J., Zhou, Z., Fang, J., Shen, W., Xie, L., Zhang, X. and Tian, Q., 2023b. Segment anything in 3d with nerfs. *arXiv:2304.12308*.
- Chacko, R., Haeni, N., Khaliullin, E. N., Sun, L. and Lee, D., 2025. Lifting by gaussians: A simple, fast and flexible method for 3d instance segmentation. *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3497–3507. URL <https://api.semanticscholar.org/CorpusID:276094652>.
- Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., Mello, S. D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T. and Wetzstein, G., 2021. Efficient geometry-aware 3D generative adversarial networks. *arXiv*.

- Chang, A. X., Funkhouser, T. A., Guibas, L. J., Hanrahan, P., Huang, Q.-X., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L. and Yu, F., 2015. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012. URL <https://api.semanticscholar.org/CorpusID:2554264>.
- Chen, A., Xu, Z., Geiger, A., Yu, J. and Su, H., 2022. Tensorf: Tensorial radiance fields. *European Conference on Computer Vision (ECCV)*.
- Chen, J.-K., Lyu, J. and Wang, Y.-X., 2023a. NeuralEditor: Editing neural radiance fields via manipulating point clouds. *CVPR*.
- Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H. and Lin, G., 2023b. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. *ArXiv*, abs/2311.14521.
- Cheng, X., Wu, Y., Jia, M., Wang, Q. and Zhang, J., 2023. Panoptic compositional feature field for editable scene rendering with network-inferred labels via metric learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4947–4957.
- Choi, S., Song, H. J., Kim, J., Kim, T. and Do, H., 2024. Click-gaussian: Interactive segmentation to any 3d gaussians. *ArXiv*, abs/2407.11793. URL <https://api.semanticscholar.org/CorpusID:271217895>.
- Cui, M., Guo, Q., Wang, Y., Yu, H., Lin, D., Zou, Q., Cheng, M.-M. and Li, X., 2025. Visibility-uncertainty-guided 3d gaussian inpainting via scene conceptional learning. URL <https://api.semanticscholar.org/CorpusID:278129747>.
- Dai, P., Xu, J., Xie, W., Liu, X., Wang, H. and Xu, W., 2024. High-quality surface reconstruction using gaussian surfels. *ACM SIGGRAPH 2024 Conference Papers*, Association for Computing Machinery.
- Dou, B., Zhang, T., Wang, Z., Ma, Y. and Yuan, Z., 2024. Learning segmented 3d gaussians via efficient feature unprojection for zero-shot

- neural scene segmentation. URL <https://api.semanticscholar.org/CorpusID:266933636>.
- Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D. and Wang, Z., 2023. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps.
- Feng, Y., Feng, X., Shang, Y., Jiang, Y., Yu, C., Zong, Z., Shao, T., Wu, H., Zhou, K., Jiang, C. and Yang, Y., 2024. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting.
- Fischer, M., Georgiev, I., Groueix, T., Kim, V. G., Ritschel, T. and Deschaintre, V., 2024. Sama: Material-aware 3d selection and segmentation. *ArXiv*, abs/2411.19322. URL <https://api.semanticscholar.org/CorpusID:274423161>.
- Floater, M. S., 2003. Mean value coordinates. *Comput. Aided Geom. Des.*, 20.
- Fridovich-Keil and Yu, Tancik, M., Chen, Q., Recht, B. and Kanazawa, A., 2022. Plenoxels: Radiance fields without neural networks. *CVPR*.
- Fu, X., Zhang, S., Chen, T., Lu, Y., Zhu, L., Zhou, X., Geiger, A. and Liao, Y., 2022. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. *International Conference on 3D Vision (3DV)*.
- Furukawa, Y. and Hernández, C., 2015. *Multi-view stereo: a tutorial*. Number 9,1/2 in Foundation and trends in computer graphics and vision, Boston Delft: Now.
- Gao, J., Gu, C., Lin, Y., Zhu, H., Cao, X., Zhang, L. and Yao, Y., 2023. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv:2311.16043*.
- Gao, L., Yang, J., Zhang, B.-T., Sun, J., Yuan, Y.-J., Fu, H. and Lai, Y.-K., 2024a. Mesh-based gaussian splatting for real-time large-scale deformation. *ArXiv*, abs/2402.04796.

- Gao, X., Li, X., Zhuang, Y., Zhang, Q., Hu, W., Zhang, C., Yao, Y., Shan, Y. and Quan, L., 2024b. Mani-gs: Gaussian splatting manipulation with triangular mesh. *arXiv preprint arXiv:2405.17811*.
- Goel, R., Sirikonda, D., Saini, S. and Narayanan, P., 2023. Interactive segmentation of radiance fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4201–4211.
- Graham, B., Engelcke, M. and van der Maaten, L., 2017. 3d semantic segmentation with submanifold sparse convolutional networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9224–9232. URL <https://api.semanticscholar.org/CorpusID:10154243>.
- Guédon, A. and Lepetit, V., 2023. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*.
- Guédon, A. and Lepetit, V., 2024. Gaussian frosting: Editable complex radiance fields with real-time rendering. *arXiv preprint arXiv:2403.14554*.
- Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S. and Cohen-Or, D., 2019. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38, 1 – 12. URL <https://api.semanticscholar.org/CorpusID:52286469>.
- Haque, A., Tancik, M., Efros, A., Holynski, A. and Kanazawa, A., 2023. Instruct-nerf2nerf: Editing 3d scenes with instructions. *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- Harary, G., Tal, A. and Grinspun, E., 2014. Context-based coherent surface completion. *ACM Transactions on Graphics (TOG)*, 33, 1 – 12. URL <https://api.semanticscholar.org/CorpusID:264877938>.
- He, Y., Yu, H., Liu, X.-Y., Yang, Z., Sun, W., Wang, Y., Fu, Q., Zou, Y. and Mian, A. S., 2021. Deep learning based 3d segmentation: A survey. *ArXiv*, abs/2103.05423. URL <https://api.semanticscholar.org/CorpusID:232168831>.

- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S., 2017. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, 6629–6640.
- Hu, X., Wang, Y., Fan, L., Luo, C., Fan, J., Lei, Z., Li, Q., Peng, J. and Zhang, Z., 2024. Sagd: Boundary-enhanced segment anything in 3d gaussian via gaussian decomposition. URL <https://api.semanticscholar.org/CorpusID:267335161>.
- Huang, B., Yu, Z., Chen, A., Geiger, A. and Gao, S., 2024. 2d gaussian splatting for geometrically accurate radiance fields. *SIGGRAPH 2024 Conference Papers*, Association for Computing Machinery.
- Huang, Y.-H., Sun, Y.-T., Yang, Z., Lyu, X., Cao, Y.-P. and Qi, X., 2023. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*.
- Jacobson, A., Baran, I., Popovic, J. and Sorkine-Hornung, O., 2011. Bounded biharmonic weights for real-time deformation. *ACM SIGGRAPH 2011 papers*. URL <https://api.semanticscholar.org/CorpusID:267856545>.
- Jain, U., Mirzaei, A. and Gilitschenski, I., 2024. Gaussiancut: Interactive segmentation via graph cut for 3d gaussian splatting. *ArXiv*, abs/2411.07555. URL <https://api.semanticscholar.org/CorpusID:273970243>.
- Jambon, C., Kerbl, B., Kopanas, G., Diolatzis, S., Leimkühler, T. and Drettakis, G., 2023. Nerfshop: Interactive editing of neural radiance fields”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6 (1). URL <https://repo-sam.inria.fr/fungraph/nerfshop/>.
- Jiang, C., Schroeder, C., Teran, J., Stomakhin, A. and Selle, A., 2016. The material point method for simulating continuum materials. *ACM SIGGRAPH 2016 Courses*, New York, NY, USA: Association for Computing Machinery, SIGGRAPH ’16. URL <https://doi.org/10.1145/2897826.2927348>.

- Jiang, Y., Yu, C., Xie, T., Li, X., Feng, Y., Wang, H., Li, M., Lau, H., Gao, F., Yang, Y. and Jiang, C., 2024. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663*.
- Joshi, P., Meyer, M., DeRose, T., Green, B. and Sanocki, T., 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26 (3), 71–es. URL <https://doi.org/10.1145/1276377.1276466>.
- Ju, T., Schaefer, S. and Warren, J. D., 2005. Mean value coordinates for closed triangular meshes. *ACM SIGGRAPH 2005 Papers*.
- Kalogerakis, E., Averkiou, M., Maji, S. and Chaudhuri, S., 2016. 3d shape segmentation with projective convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6630–6639. URL <https://api.semanticscholar.org/CorpusID:5217417>.
- Kavan, L., 2014. Siggraph course 2014 — skinning: Real-time shape deformation part i: Direct skinning methods and deformation primitives.
- Kerbl, B., Kopanas, G., Leimkühler, T. and Drettakis, G., 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4). URL <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.
- Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P. and Girshick, R., 2023. Segment Anything. URL <http://arxiv.org/abs/2304.02643>.
- Kobayashi, S., Matsumoto, E. and Sitzmann, V., 2022. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*.
- Lazova, V., Guzov, V., Olszewski, K., Tulyakov, S. and Pons-Moll, G., 2022. Control-nerf: Editable feature volumes for scene rendering and manipulation. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 4329–4339.

- Lee, J. C., Rho, D., Sun, X., Ko, J. H. and Park, E., 2024. Compact 3d gaussian representation for radiance field. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 21719–21728.
- Lewis, J. P., Cordner, M. and Fong, N., 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. URL <https://api.semanticscholar.org/CorpusID:12672235>.
- Li, B., Weinberger, K. Q., Belongie, S., Koltun, V. and Ranftl, R., 2022. Language-driven semantic segmentation. *International Conference on Learning Representations*.
- Liang, Z., Zhang, Q., Feng, Y., Shan, Y. and Jia, K., 2023. Gs-ir: 3d gaussian splatting for inverse rendering. *arXiv preprint arXiv:2311.16473*.
- Lipman, Y., Levin, D. and Cohen-Or, D., 2008. Green coordinates. *ACM SIGGRAPH 2008 papers*.
- Liu, K., Zhan, F., Zhang, J., Xu, M., Yu, Y., El Saddik, A., Theobalt, C., Xing, E. and Lu, S., 2023a. Weakly supervised 3d open-vocabulary segmentation. *Thirty-seventh Conference on Neural Information Processing Systems*.
- Liu, L., Gu, J., Lin, K. Z., Chua, T.-S. and Theobalt, C., 2020. Neural sparse voxel fields. *NeurIPS*.
- Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.-Y. and Russell, B., 2021. Editing conditional radiance fields.
- Liu, Y., Hu, B., Huang, J., Tai, Y.-W. and Tang, C.-K., 2023b. Instance neural radiance field. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 787–796.

- Liu, Z., Ouyang, H., Wang, Q., Cheng, K. L., Xiao, J., Zhu, K., Xue, N., Liu, Y., Shen, Y. and Cao, Y., 2024. Infusion: Inpainting 3d gaussians via learning depth completion from diffusion prior. *arXiv preprint arXiv:2404.11613*.
- Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D. and Dai, B., 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20654–20664.
- Macklin, M., Müller, M. and Chentanez, N., 2016. Xpbd: position-based simulation of compliant constrained dynamics. *Proceedings of the 9th International Conference on Motion in Games*.
- Markus Gross, H. P., 2007. *Point-Based Graphics*. Elsevier. URL <https://linkinghub.elsevier.com/retrieve/pii/B9780123706041X50007>.
- Maturana, D. and Scherer, S. A., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928. URL <https://api.semanticscholar.org/CorpusID:14620252>.
- Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R. and Ng, R., 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, 405–421.
- Mirzaei, A., Aumentado-Armstrong, T., Brubaker, M. A., Kelly, J., Levinshstein, A., Derpanis, K. G. and Gilitschenski, I., 2023a. Reference-guided controllable inpainting of neural radiance fields. *ICCV*.
- Mirzaei, A., Aumentado-Armstrong, T., Derpanis, K. G., Kelly, J., Brubaker, M. A., Gilitschenski, I. and Levinshstein, A., 2023b. SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. *CVPR*.

- Mirzaei, A., de Lutio, R., Kim, S., Acuna, D., Kelly, J., Fidler, S., Gilitschenski, I. and Gojcic, Z., 2024. Reffusion: Reference adapted diffusion models for 3d scene inpainting. *ArXiv*, abs/2404.10765. URL <https://api.semanticscholar.org/CorpusID:269157652>.
- Monnier, T., Austin, J., Kanazawa, A., Efros, A. A. and Aubry, M., 2023. Differentiable blocks world: Qualitative 3d decomposition by rendering primitives. *ArXiv*, abs/2307.05473.
- Müller, T., Evans, A., Schied, C. and Keller, A., 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41 (4), 102:1–102:15. URL <https://doi.org/10.1145/3528223.3530127>.
- Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S. and Fitzgibbon, A. W., 2011. Kinectfusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 127–136. URL <https://api.semanticscholar.org/CorpusID:11830123>.
- Peng, Y., Yan, Y., Liu, S., Cheng, Y., Guan, S., Pan, B., Zhai, G. and Yang, X., 2022. Cagenerf: Cage-based neural radiance fields for genrenlized 3d deformation and animation. *Thirty-Sixth Conference on Neural Information Processing Systems*.
- Pernot, J.-P., Moraru, G. and Véron, P., 2006. Filling holes in meshes using a mechanical model to simulate the curvature variation minimization. *Comput. Graph.*, 30, 892–902. URL <https://api.semanticscholar.org/CorpusID:11329309>.
- Poole, B., Jain, A., Barron, J. T. and Mildenhall, B., 2022. Dreamfusion: Text-to-3d using 2d diffusion. *ArXiv*, abs/2209.14988. URL <https://api.semanticscholar.org/CorpusID:252596091>.
- Pumarola, A., Corona, E., Pons-Moll, G. and Moreno-Noguer, F., 2020. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*.

- Qi, C., Su, H., Mo, K. and Guibas, L. J., 2016. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 77–85. URL <https://api.semanticscholar.org/CorpusID:5115938>.
- Qi, C., Yi, L., Su, H. and Guibas, L. J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *ArXiv*, abs/1706.02413. URL <https://api.semanticscholar.org/CorpusID:1745976>.
- Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P. and Feichtenhofer, C., 2024. Sam 2: Segment anything in images and videos. URL <https://arxiv.org/abs/2408.00714>.
- Ren, Z., Agarwala, A., Russell, B., Schwing, A. G. and Wang, O., 2022. Neural volumetric object selection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6133–6142.
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B., 2021. High-resolution image synthesis with latent diffusion models.
- Rong, V., Chen, J., Bahmani, S., Kutulakos, K. N. and Lindell, D. B., 2024. Gstex: Per-primitive texturing of 2d gaussian splatting for decoupled appearance and geometry modeling. *arXiv preprint arXiv:2409.12954*.
- Rumezhak, T., Dobosevych, O., Hryniv, R. O., Selotkin, V., Karpiv, V. and Maksymenko, M., 2021. Towards realistic symmetry-based completion of previously unseen point clouds. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2542–2550. URL <https://api.semanticscholar.org/CorpusID:244476541>.
- Rusu, R. B. and Cousins, S., 2011. 3D is here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China: IEEE.

- Schönberger, J. L. and Frahm, J.-M., 2016. Structure-from-motion revisited. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Schönberger, J. L., Zheng, E., Pollefeys, M. and Frahm, J.-M., 2016. Pixel-wise view selection for unstructured multi-view stereo. *European Conference on Computer Vision (ECCV)*.
- Sederberg, T. W. and Parry, S. R., 1986. Free-form deformation of solid geometric models. *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. URL <https://api.semanticscholar.org/CorpusID:207174668>.
- Seo, Y., Choi, Y. S., Son, H. S. and Uh, Y., 2024. Flod: Integrating flexible level of detail into 3d gaussian splatting for customizable rendering. URL <https://arxiv.org/abs/2408.12894>.
- Shen, I.-C., Liu, H.-K. and Chen, B.-Y., 2024a. Nerf-in: Free-form nerf inpainting with rgb-d priors. *Computer Graphics and Applications (CG&A)*.
- Shen, Q., Yang, X. and Wang, X., 2024b. Flashsplat: 2d to 3d gaussian splatting segmentation solved optimally. *European Conference on Computer Vision*. URL <https://api.semanticscholar.org/CorpusID:272599869>.
- Siddiqui, Y., Porzi, L., Bulò, S. R., Müller, N., Nießner, M., Dai, A. and Kotschieder, P., 2023. Panoptic lifting for 3d scene understanding with neural fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9043–9052.
- Silva, M. C., Dahaghin, M., Toso, M. and Bue, A. D., 2024. Contrastive gaussian clustering: Weakly supervised 3d scene segmentation. *ArXiv*, abs/2404.12784. URL <https://api.semanticscholar.org/CorpusID:269282819>.
- Sorkine-Hornung, O. and Alexa, M., 2007a. As-rigid-as-possible surface modeling. *Eurographics Symposium on Geometry Processing*. URL <https://api.semanticscholar.org/CorpusID:11952089>.

- Sorkine-Hornung, O. and Alexa, M., 2007b. As-rigid-as-possible surface modeling. *Eurographics Symposium on Geometry Processing*. URL <https://api.semanticscholar.org/CorpusID:11952089>.
- Sun, C., Sun, M. and Chen, H., 2022. Direct voxel grid optimization: Superfast convergence for radiance fields reconstruction. *CVPR*.
- Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K. and Lempitsky, V., 2022. Resolution-robust Large Mask Inpainting with Fourier Convolutions. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3172–3182.
- Svitov, D., Morerio, P., Agapito, L. and Del Bue, A., 2024. Billboard splatting (bbsplat): Learnable textured primitives for novel view synthesis. *arXiv preprint arXiv:2411.08508*.
- Tang, S., Pei, W., Tao, X., Jia, T., Lu, G. and Tai, Y.-W., 2023. Scene-generalizable interactive segmentation of radiance fields. *Proceedings of the 31st ACM International Conference on Multimedia*, 6744–6755.
- Tesema, K. W., Hill, L., Jones, M. W., Ahmad, M. I. and Tam, G. K., 2024. Point cloud completion: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 30 (10), 6880–6899.
- Thomas, H., Qi, C., Deschaud, J.-E., Marcotegui, B., Goulette, F. and Guibas, L. J., 2019. Kpconv: Flexible and deformable convolution for point clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 6410–6419. URL <https://api.semanticscholar.org/CorpusID:121328056>.
- Tong, X., Shao, T., Weng, Y., Yang, Y. and Zhou, K., 5555. As-Rigid-As-Possible Deformation of Gaussian Radiance Fields . *IEEE Transactions on Visualization & Computer Graphics*, (01), 1–13. URL <https://doi.ieeecomputersociety.org/10.1109/TVCG.2025.3555404>.

- Tschernezki, V., Laina, I., Larlus, D. and Vedaldi, A., 2022. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. *2022 International Conference on 3D Vision (3DV)*, IEEE, 443–453.
- Vora, S., Radwan, N., Greff, K., Meyer, H., Genova, K., Sajjadi, M. S. M., Pot, E., Tagliasacchi, A. and Duckworth, D., 2022. NeSF: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *Transactions on Machine Learning Research*.
- Waczyńska, J., Borycki, P., Tadeja, S., Tabor, J. and Spurek, P., 2024. Games: Mesh-based adapting and modification of gaussian splatting.
- Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T. and Wang, W., 2021. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*.
- Wang, S. W. and Kaufman, A. E., 1995. Volume sculpting. *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, New York, NY, USA: Association for Computing Machinery, I3D '95, 151–ff. URL <https://doi.org/10.1145/199404.199430>.
- Wang, Y., Wang, J., Qu, Y. and Qi, Y., 2023. Rip-nerf: Learning rotation-invariant point-based neural radiance field for fine-grained editing and compositing. *Proceedings of the 2023 ACM International Conference on Multimedia Retrieval*.
- Wang, Y., Yi, X., Wu, Z., Zhao, N., Chen, L. and Zhang, H., 2024. View-consistent 3d editing with gaussian splatting. *ArXiv*, abs/2403.11868.
- Wang, Z., Bovik, A. C., Sheikh, H. R. and Simoncelli, E. P., 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13, 600–612.
- Weder, S., Garcia-Hernando, G., Monszpart, A., Pollefeys, M., Brostow, G. J., Firman, M. and Vicente, S., 2023. Removing objects from neural radiance fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16528–16538.

- Wei, X., Zhang, R., Wu, J., Liu, J., Lu, M., Guo, Y. and Zhang, S., 2023. Noc: High-quality neural object cloning with 3d lifting of segment anything. *arXiv preprint arXiv:2309.12790*.
- Wu, J., Bian, J., Li, X., Wang, G., Reid, I. D., Torr, P. and Prisacariu, V. A., 2024. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. *ArXiv*, abs/2403.08733.
- Xie, T., Zong, Z., Qiu, Y., Li, X., Feng, Y., Yang, Y. and Jiang, C., 2023. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*.
- Xu, T. and Harada, T., 2022. Deforming radiance fields with cages. *ECCV*.
- Xu, X., Yang, Y., Mo, K., Pan, B., Yi, L. and Guibas, L., 2023. Jacobinerf: Nerf shaping with mutual information gradients. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16498–16507.
- Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G. and Cui, Z., 2021. Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 13759–13768.
- Ye, J., Wang, N. and Wang, X., 2023a. Featurenerf: Learning generalizable nerfs by distilling pre-trained vision foundation models. *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- Ye, M., Danelljan, M., Yu, F. and Ke, L., 2023b. Gaussian grouping: Segment and edit anything in 3d scenes. *ArXiv*, abs/2312.00732.
- Yifan, W., Serena, F., Wu, S., Öztireli, C. and Sorkine-Hornung, O., 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38 (6), 1–14.
- Yin, Y., Fu, Z., Yang, F. and Lin, G., 2023. Or-nerf: Object removing from 3d scenes guided by multiview segmentation with neural radiance fields.

- Yuan, W., Khot, T., Held, D., Mertz, C. and Hebert, M., 2018. Pcn: Point completion network. *2018 International Conference on 3D Vision (3DV)*, 728–737. URL <https://api.semanticscholar.org/CorpusID:51908879>.
- Yuan, Y.-J., Sun, Y.-T., Lai, Y.-K., Ma, Y., Jia, R. and Gao, L., 2022. NeRF-Editing: Geometry Editing of Neural Radiance Fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 18332–18343.
- Zhang, L., Rao, A. and Agrawala, M., 2023. Adding conditional control to text-to-image diffusion models.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E. and Wang, O., 2018. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*.
- Zhang*, Y., Peng*, S., Moazeni, A. and Li, K., 2023. Papr: Proximity attention point rendering. *Advances in Neural Information Processing Systems*.
- Zhang, Z., Huang, B., Jiang, H., Zhou, L., Xiang, X. and Shen, S., 2024. Quadratic gaussian splatting for efficient and detailed surface reconstruction. URL <https://arxiv.org/abs/2411.16392>.
- Zhao, W., Gao, S. and Lin, H., 2007. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23, 987–997. URL <https://api.semanticscholar.org/CorpusID:3365551>.
- Zhi, S., Laidlow, T., Leutenegger, S. and Davison, A. J., 2021a. In-place scene labelling and understanding with implicit scene representation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 15838–15847.
- Zhi, S., Sucar, E., Mouton, A., Haughton, I., Laidlow, T. and Davison, A. J., 2021b. ilabel: Interactive neural scene labelling. *arXiv preprint arXiv:2111.14637*.

- Zhou, Q.-Y., Park, J. and Koltun, V., 2018. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.
- Zhu, R., Qiu, S., Liu, Z., Hui, K.-H., Wu, Q., Heng, P.-A. and Fu, C.-W., 2025. Rethinking end-to-end 2d to 3d scene segmentation in gaussian splatting. *arXiv preprint arXiv:2503.14029*.
- Zielonka, W., Bagautdinov, T., Saito, S., Zollhöfer, M., Thies, J. and Romero, J., 2023. Drivable 3d gaussian avatars.
- Zwicker, M., Pauly, M., Knoll, O. and Gross, M., 2002. Pointshop 3d: an interactive system for point-based surface editing. *ACM Trans. Graph.*, 21 (3), 322–329. URL <https://doi.org/10.1145/566654.566584>.