Efficient Text-Driven 3D Scene Editing Based on Gaussian Splatting

by

Xuanqi Zhang

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Text-guided 3D scene editing has advanced with diffusion models and neural rendering. While Neural Radiance Fields (NeRF) excel at 3D reconstruction and novel view synthesis, they face key challenges: computational inefficiency, multi-view inconsistency, and poor handling of motion blur.

To overcome these limitations, we introduce a 3D Gaussian Splatting (3DGS)-based framework for efficient and consistent 3D editing. 3DGS offers real-time rendering and a more practical alternative to NeRF. We enhance its performance with two key components: (1) a Complementary Information Mutual Learning Network (CIMLN) for refining 3DGS-derived depth maps, enabling depth-aware image editing; and (2) a Wavelet Consensus Attention mechanism that aligns latent codes across views during diffusion denoising, ensuring consistent multi-view outputs.

We also address motion-blurred scene reconstruction by applying low-pass filtering and jointly optimizing Gaussian parameters with camera trajectories.

Experiments show that our method achieves photorealistic, structurally consistent results while maintaining real-time performance and significantly reducing computational overhead compared to prior methods.

Acknowledgements

I would like to sincerely dedicate this work to my supervisors, Professor Wonsook Lee and Professor Chris Joslin, whose invaluable support, guidance, and encouragement have shaped both my research and my personal growth throughout this journey.

To Professor Wonsook Lee, I am deeply grateful for your constant support, especially during the most difficult moments of my academic path. When I faced uncertainty and doubt, it was your belief in my potential that gave me the confidence to continue pursuing research. Your encouragement has meant more to me than words can express. To Professor Chris Joslin, I would like to express my heartfelt appreciation for your meticulous attention to detail and your commitment to helping me refine my work. Your insightful and constructive feedback on my papers and thesis not only improved the quality of my research but also taught me how to think more critically and rigorously.

Additionally, I am deeply grateful to Professor Jieun Lee and Professor Paola Flocchini for your generous help with my research and thesis. I would like to express my heartfelt thanks to my parents for your unwavering support, which has given me the opportunity to pursue my education abroad. Their encouragement and sacrifices have made this journey possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Artificial Intelligence Generated Content (AIGC) has emerged as a revolutionary force in the digital realm, transforming the landscape of content creation across various mediums. This innovative approach harnesses the power of advanced AI models to generate a wide array of digital content, including images, natural language, and music [2,4,33]. The rapid evolution of AIGC technologies has not only streamlined creative processes but also opened up new possibilities for expression and innovation in the digital space.

In recent years, the field of 3D content generation has experienced a particularly dramatic transformation, propelled by the development of sophisticated neural representations and generative models. These advancements have significantly enhanced the quality and diversity of 3D models that can be created through AI-driven processes. The implications of these developments are far-reaching, touching industries ranging from entertainment and gaming to architecture and product design.

Traditionally, 3D content creation has been characterized as a labour-intensive and time-consuming endeavor, requiring specialized skills and substantial resources. However, the advent of AIGC techniques has brought about a paradigm shift in this domain. These

new methodologies have drastically streamlined the workflow of 3D content creation, making it more accessible and efficient. This democratization of 3D content generation has the potential to unleash a new wave of creativity and innovation across various sectors.

Within the broader context of 3D AIGC, the subfield of 3D scene editing (3D editing) has recently garnered significant attention from researchers and practitioners. Unlike generating entirely new scenes, this area focuses on modifying and enhancing existing 3D content. Its applications span virtual reality (VR), augmented reality (AR), and visual effects (VFX) in film and television production [23], offering opportunities for personalization and adaptation in immersive technologies and visual storytelling.

Text-driven 3D editing has emerged as a groundbreaking approach that allows users to manipulate 3D scenes using natural language instructions [36]. By lowering the technical barrier, it enables individuals without expertise in 3D modeling to perform sophisticated edits [28]. This democratization of 3D editing fosters rapid prototyping and creative expression across domains such as game development, VR, and digital art. Its intuitive, language-based interface aligns well with human cognition, making the process more accessible for both professionals and hobbyists [30].

The growing interest in this field highlights its potential to transform 3D content creation [62]. As text-driven techniques evolve, they promise to bridge the gap between conceptual ideas and their 3D realization, enhancing productivity and encouraging innovation in immersive and interactive design.

1.2 Problem Overview

Text-driven 3D editing, while promising, faces several significant challenges:

Time-intensive Processing: The time-intensive nature of text-driven 3D editing poses a significant obstacle to its widespread adoption, largely due to several intertwined challenges. First, achieving consistent 3D outputs from 2D image-based generative models often in-

volves multiple iterative refinements, as the initial results frequently require adjustments to meet the desired quality. Additionally, the computational demands are considerable, with state-of-the-art methods typically requiring optimization times ranging from tens of minutes to several hours when using a single GPU. Furthermore, some approaches exacerbate these demands by necessitating model retraining for each specific 3D scene, resulting in substantial computational and memory overhead. These factors collectively highlight the current inefficiencies and limitations of text-driven 3D editing workflows. [71]

Multi-view Inconsistency: Multi-view inconsistency is a critical challenge in text-driven 3D editing, particularly when relying on 2D image-based generative models to manipulate 3D scenes [61]. Such inconsistency manifests as visual artifacts—e.g., blurring, ghosting, and mismatched details—when viewing the scene from different angles. This issue arises because 2D models lack explicit awareness of 3D geometry, often resulting in outputs that are plausible in isolation but misaligned across views. Multi-view consistency refers to the alignment of several key aspects across all viewpoints, including:

- **Geometry:** ensuring that shapes and spatial structures remain coherent from all perspectives.
- **Texture:** maintaining consistent surface details and patterns.
- **Color:** preserving uniform color distribution and shading across views.

Ensuring consistency in these aspects is essential for generating photorealistic and geometrically sound 3D edits.

Moreover, the inability of current text-to-image (T2I) models to ensure consistent editing effects across multiple views further diminishes the performance and reliability of 3D edits. Adding to this complexity, state-of-the-art techniques frequently rely on multiple synchronized edited images of the same scene to achieve effective results that remains difficult to fulfill with existing T2I models. These limitations collectively underscore the

critical need for more robust solutions to address multi-view inconsistency in 3D editing workflows. [66].

Motion Blur in Real-World Scenes: In real-world multi-view capture scenarios, motion blur frequently occurs due to fast camera motion or dynamic environments. However, most existing 3D datasets assume ideal, blur-free conditions and fail to account for this degradation. This discrepancy limits the generalization of current 3D editing or reconstruction models when deployed in practical settings. Future work should focus on constructing realistic datasets that incorporate motion blur, or developing learning frameworks that can explicitly handle blurred inputs during training and inference.

1.3 Research Contributions

1.3.1 Efficient 3D Editing Based on Gaussian Splatting with Complementary and Consensus Information Learning

We proposed an efficient 3DGS editing system as shown in Fig. 1.1. Compared with NeRF-based 3D editing methods, our system reduces training and rendering time with the help of 3DGS as 3D representation. Enhancing depth map quality is essential for achieving superior image editing results in 3D scenes. Inspired by the previous cross-modal enhancement research [16], we developed the Complementary Information Mutual Learning Network (CIMLN). This network leverages complementary information from multi-view images to refine depth maps, enabling precise depth information which is used as an additional condition for diffusion-based image editing in ControlNet. We further propose a Wavelet Consensus Attention (WCA) module to ensure advanced multi-view consistency and extract consensus information from both the spatial and frequency domains. The WCA module enhances multi-view consistency by replacing ControlNet's self-attention with wavelet-based attention, enabling latent code alignment during the diffusion denoising process. This architectural modification facilitates seamless coordination across viewpoints, resulting in

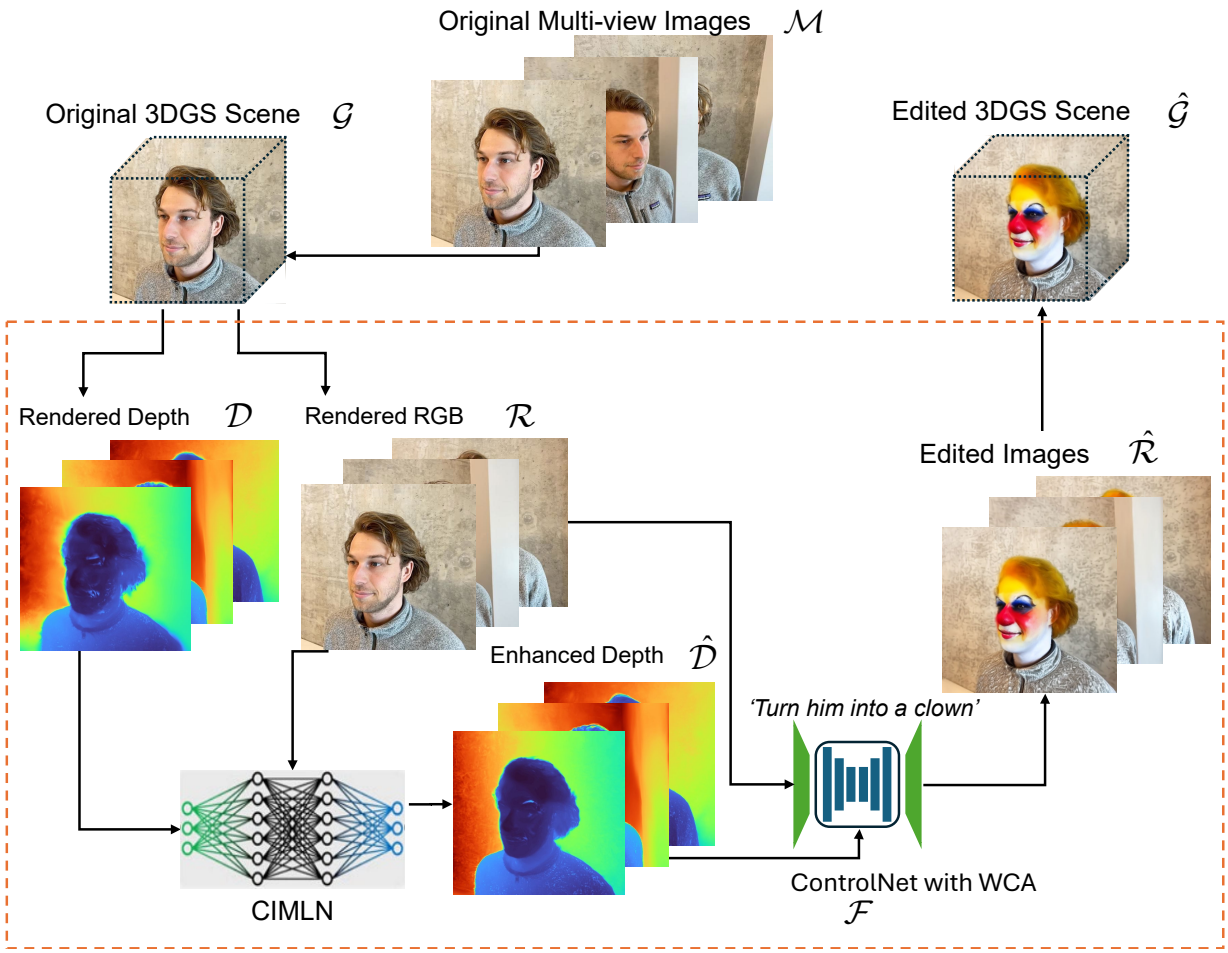


Figure 1.1: Overview of the proposed 3D Gaussian splatting editing system. Multi-view images $\mathcal{M}$ are used to train a 3DGS model, from which the rendered depth and RGB images can be obtained. Rendered depth maps and RGB images are optimized through CIMLN respectively. The framework replaces ControlNet's image self-attention with WCA to better align images with reference views.

more coherent multi-view editing results.

1.3.2 3D Gaussian Splatting from Motion Blur Images via Low-pass Filter

Reconstructing high-quality 3D representations from motion-blurred images is a challenging problem in computer vision and computational photography. Motion blur occurs when a camera captures a scene while either the camera or objects within the scene move during exposure. This leads to the loss of sharp details and introduces temporal integration effects. Such degradations pose significant challenges for traditional 3D reconstruction pipelines, particularly for structure-from-motion (SfM) [46], which relies on precise feature matching and correspondence estimation across views. Similarly, neural reconstruction techniques also struggle to recover fine-grained geometry from severely blurred inputs.

To address this, we explore a novel approach to reconstructing 3D Gaussian representations from motion-blurred images. Our method leverages the inherent structure of Gaussian splatting while incorporating a motion-aware deblurring mechanism. By modeling motion trajectories and optimizing Gaussian attributes jointly with deblurring constraints, we achieve high-fidelity 3D reconstructions even from blurred inputs. Unlike naive deblurring-then-reconstruction pipelines, which suffer from accumulated errors, our framework jointly estimates sharp scene representations and corrects for motion blur during optimization. We also introduce a low-pass filter to eliminate the high-frequency artifacts during the training of 3DGS. Extensive experiments demonstrate that our proposed framework achieves superior performance compared to state-of-the-art methods in both qualitative and quantitative results.

In summary, our main contributions are as follows:

- We proposed an efficient 3D editing method based on 3DGS that incorporates complementary and consensus information learning for improved performance with less

editing time.

- We developed a self-supervised complementary information learning network that extracts complementary details from 3D depth information and multi-view RGB images to refine depth map for image editing, which improves multi-view consistency for our 3D editing.

- We designed a wavelet consensus attention for a latent code alignment module that extracts consensus information from both spatial and frequency domains and ensures consistent editing results in the diffusion process.

- We proposed a novel approach to reconstructing 3D Gaussian representations from motion-blurred images via low-pass filter, which aims to solve motion blur problem for real-captured 3D data. **This work is accepted by Canadian Artificial Intelligence 2025**.

1.4 Publication and Preprint

[1] Xuanqi Zhang, Wonsook Lee, Chris Joslin. 3D Gaussian Splatting and Camera Poses Joint Optimization for Motion Blur Images via Low-pass Filter. The 38th Canadian Conference on Artificial Intelligence, 2025. Location: Calgary, Canada. Conference dates: May 26-29, 2025. [68]

[2] Xuanqi Zhang, Jieun Lee, Chris Joslin, Wonsook Lee. Advancing 3D Gaussian Splatting Editing with Complementary and Consensus Information. *arXiv preprint* arXiv:2503.11601. [67]

1.5 Thesis Structure

This section is a brief description of the structure of this thesis. This thesis has 5 chapters.

- Chapter 2 is the literature review. Our model includes a 2D editing phase so we introduce conceptions of diffusion models. Then we discuss the NeRF and 3DGS for 3D representation from different aspects. In our main task, we introduce two kinds of 3D editing: 2D-3D lifting 3D editing and direct 3D editing and corresponding works. Furthermore, we review depth map enhancement related works which are beneficial for 3D awareness and multi-view consistency in our 3D editing. Finally, we summarize existing motion deblurring methods.

- Chapter 3 is the detailed methodology of our 3D editing system. In this section, we start with our module's architecture, and we explicitly describe each step of the data flow and calculations from end to end. We also explicitly describe the motivation of our designed modules for improving 3D editing.

- Chapter 4 covers the experimental setup details. We introduce the datasets, loss function and preprocess details. Then, we specify how we construct our training strategy. We analyze different 3D editing methods results in different scenes. Ablation studies are provided to prove the effects of our proposed modules.

- Chapter 5 is the conclusion and future work. We summarize the work that we have done and analyze our model's advantages and disadvantages for our future research.

Chapter 2

Literature Review

In this chapter, we provide a comprehensive review of foundational and recent advances related to our work on 3D Gaussian Splatting (3DGS)-based text-driven 3D editing. We begin by introducing the denoising diffusion probabilistic models (DDPMs), which form the basis of many recent generative techniques. We then explore deep learning-based image editing methods, including both GAN- and diffusion-based approaches. Next, we review common 3D representations such as meshes, neural radiance fields (NeRF), and the recently proposed 3DGS. Introduction of COLMAP is given to help understand camera poses estimation. We then categorize and compare existing text-driven 3D editing methods, distinguishing between direct 3D editing and 2D-3D lifting strategies. Finally, we discuss two critical auxiliary tasks—depth enhancement and motion deblurring—that play essential roles in improving 3D editing performance under challenging conditions such as sparse or blurred multi-view data.

2.1 Basic Modules

2.1.1 Denoising Diffusion Probabilistic Model

Before investigating existing 3D editing methods, we first review the basis of current AIGC techniques, particularly diffusion models, as they play a critical role in text-guided image generation.

Denoising Diffusion Probabilistic Models (DDPM) proposed by Ho et al. [22, 49], or simply diffusion models, are a class of probabilistic generative models that turn noise into a representative data sample and thus are mainly used for generation tasks [14, 44]. For example, Stable Diffusion [44] is a state-of-the-art deep learning model for text-to-image generation released in 2022. It uses a latent diffusion model (LDM) architecture, which consists of three main components: a variational autoencoder (VAE) [31], a U-Net [45], and an optional text encoder [44]. The model works by gradually denoising a latent representation of an image, guided by text prompts or other conditioning inputs. The generation results are shown in Fig. 2.1.

2.1.2 3D Representations

Meshes

Polygonal meshes are non-Euclidean data structures that represent the surfaces of shapes using vertices, edges, and faces. Unlike voxels [5], which represent 3D volumes, and point clouds, which consist of unstructured sets of 3D points, meshes provide explicit connectivity between surface points, enabling the modeling of geometric relationships. This compact yet expressive representation makes meshes a popular choice in traditional computer graphics tasks such as geometry processing, animation, and rendering.

However, applying deep neural networks to meshes is more complex than to point clouds or voxels, as it requires accounting for both vertex positions and edge connectivity.

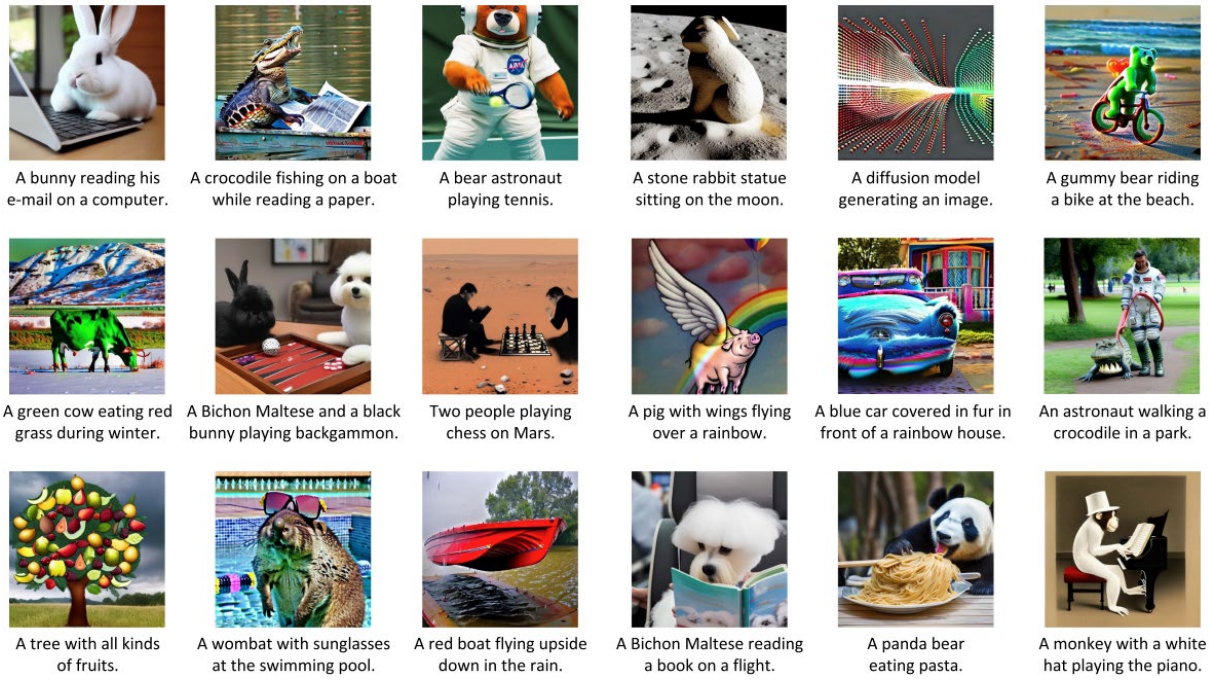


Figure 2.1: Images generated by Stable Diffusion [44] based on various text prompts, via the https://beta.dreamstudio.ai/dreamplatform. reprinted from [12]

This structural complexity makes meshes more difficult to edit using learning-based 3D approaches.

Neural Radiance Fields

Neural Radiance Fields (NeRF) proposed by Mildenhall et al. [38] has emerged as a groundbreaking technique in computer graphics and 3D vision since their introduction in 2020. This novel method enables photorealistic rendering of 3D scenes from sparse 2D input views, revolutionizing applications in novel view synthesis, scene reconstruction, and reflectance modeling. A neural field is a continuous neural implicit representation that models the complete or partial geometry and appearance of a scene or object using a neural network. Instead of storing explicit geometry like meshes or voxels, it encodes this information as a function—typically approximated by a multilayer perceptron (MLP)—that maps spatial coordinates (and sometimes viewing directions) to physical properties such as color and density. This is similar to a "neural blueprint" of the scene: given any coordinate in space, the network predicts what exists there and how it should look from a specific viewpoint.

The overall NeRF volume rendering and training process is shown in Fig. 2.2. However, the baseline NeRF models are known for their slow training and inference speed due to the need to densely sample points along every camera ray and evaluate a large multilayer perceptron (MLP) at each sampled location. Additionally, the volumetric rendering process requires computing transmittance and color for many such samples, leading to high computational and memory costs. These limitations hinder the real-time applicability of NeRFs and motivate the development of faster alternatives such as 3D Gaussian Splatting.

3D Gaussian Splatting

Despite significant efforts [10,40] to accelerate NeRFs for practical use on everyday devices like smartphones and laptops, achieving both fast training on consumer-grade GPUs and

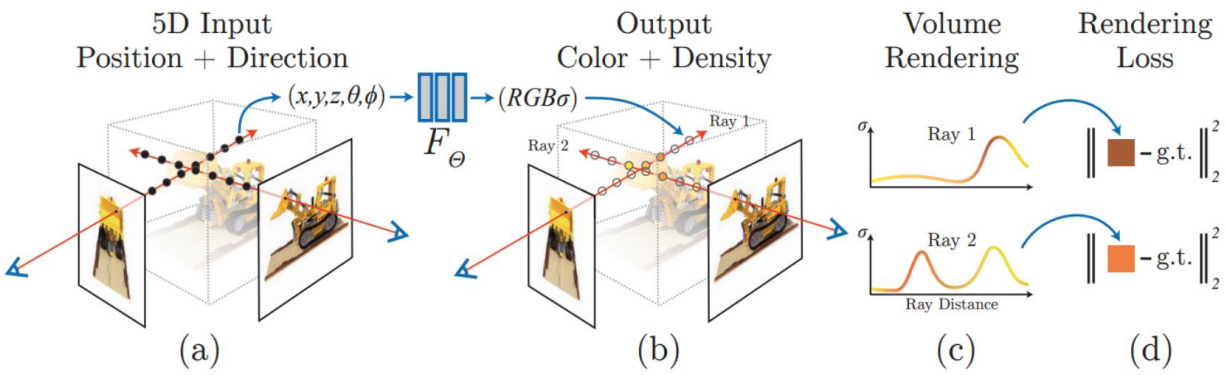


Figure 2.2: The NeRF volume rendering and training process reprinted from [19]. (a) how sampling points are selected along camera rays for each pixel in the synthesized image. (b) how a NeRF MLP network is used to compute densities and colors at these sampled points. (c) the volume rendering process, where accumulated color and density values along a ray are integrated to produce a final pixel color. (d) the rendered pixel colors with ground truth values to compute photometric loss for supervision.

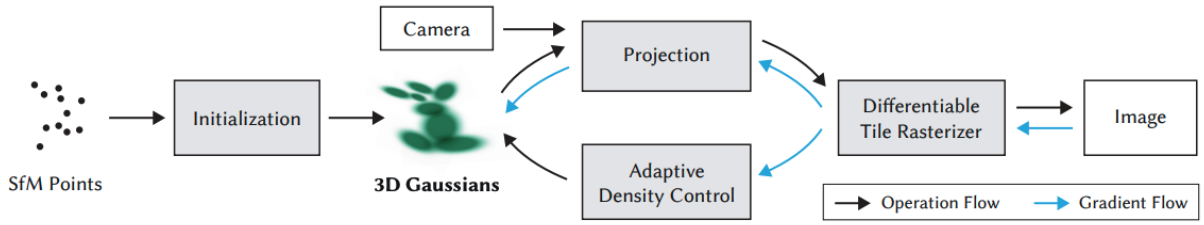


Figure 2.3: Optimization overview reprinted from [29]. It starts with the sparse SfM point cloud and creates a set of 3D Gaussians.

interactive rendering speeds on standard devices remains a challenge. To address these performance limitations, 3D Gaussian Splatting (3DGS) from Kerbl et al. [29] proposes rasterizing a collection of Gaussian ellipsoids to approximate the visual appearance of a 3D scene. This approach not only delivers comparable quality for novel view synthesis but also achieves rapid convergence of training loss (typically within 30 minutes) enabling fast model optimization and real-time rendering. As a result, 3DGS enables low-cost 3D content creation, requiring only consumer-grade GPUs for training and rendering.

To accelerate both training and rendering, 3D Gaussian Splatting (3DGS) eliminates the need for neural networks that predict color and density at every sample point. Instead, it directly optimizes a set of Gaussian ellipsoids, each associated with attributes such as position $\mu \in \mathbb{R}^3$, rotation $\mathbf{R} \in \mathrm{SO}(3)$, scale $\mathbf{s} \in \mathbb{R}^3$, opacity α, and spherical harmonics (SH) coefficients that model view-dependent radiance.

The optimization process is illustrated in Fig. 2.3. Each 3D Gaussian is represented by a mean μ and a full 3D covariance matrix $\mathbf{\Sigma}$ defined in world space [72]:

$$\mathcal{G}(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right). \tag{2.1}$$

For rendering, these 3D Gaussians are projected onto the 2D image plane using a

camera view transformation. The projected 2D covariance $\mathbf{\Sigma}'$ is computed as:

$$\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^\top \mathbf{J}^\top, \tag{2.2}$$

where:

- $\mathbf{\Sigma} = \mathbf{RSS}^\top \mathbf{R}^\top$ is the world-space covariance matrix, with $\mathbf{S}$ as the diagonal matrix constructed from scale vector $\mathbf{s}$,
- $\mathbf{W}$ is the world-to-camera transformation matrix,
- $\mathbf{J}$ is the Jacobian of the perspective projection.

The color of each pixel is determined by rasterizing these projected Gaussians and compositing them using their opacity α and SH-based radiance.

3DGS shares a similar rendering process with NeRFs. **Why is 3DGS faster than NeRF?**

1. 3DGS models opacity values directly, whereas NeRF first estimates density and then transforms it into opacity. This simplification reduces computational overhead and accelerates convergence—typically to within 30 minutes, compared to several hours required by NeRF in NeRF360 [3] dataset.
2. 3DGS uses rasterization-based rendering, which avoids the need for dense sampling and neural network queries. In contrast, NeRF performs volumetric rendering by densely sampling 3D points and evaluating a multi-layer perceptron (MLP) at each location.

Without neural sampling and querying, 3DGS is highly efficient, achieving real-time rendering, compared to NeRF implementations—while still providing comparable rendering quality.

2.1.3 COLMAP

To recover the camera poses and scene geometry from multi-view images, we rely on classical 3D reconstruction techniques, namely Structure-from-Motion (SfM) and Multi-View Stereo (MVS). SfM estimates both camera intrinsics and extrinsics from unordered image sets while simultaneously reconstructing a sparse 3D point cloud. MVS then refines this sparse reconstruction to obtain dense depth information.

In our system, we adopt COLMAP [46], a widely-used open-source SfM and MVS pipeline, for its robustness and automation in large-scale image-based 3D reconstruction tasks.

COLMAP is an open-source SfM and MVS framework that provides an automated pipeline for recovering accurate camera poses and dense 3D geometry from multi-view image collections. The software is built on robust feature extraction, image matching, and bundle adjustment techniques, making it a popular choice in computer vision and graphics applications.

COLMAP's SfM module estimates camera intrinsics and extrinsics while simultaneously reconstructing a sparse 3D point cloud. It consists of the following key steps:

- Feature Extraction: Detects distinctive keypoints using algorithms such as SIFT (Scale-Invariant Feature Transform).
- Feature Matching: Establishes correspondences between image pairs using exhaustive or vocabulary-based matching.
- Incremental and Global SfM: COLMAP supports both incremental (sequential) and global (simultaneous) reconstruction strategies.
- Bundle Adjustment: Optimizes camera parameters and 3D structure via nonlinear least squares minimization.

Once the sparse reconstruction is obtained, COLMAP's MVS module refines the geometry to generate a dense 3D model:

- Depth Map Estimation: Predicts per-pixel depth using patch-based stereo matching.
- Fusion and Meshing: Aggregates depth maps into a dense point cloud and optionally converts it into a mesh.

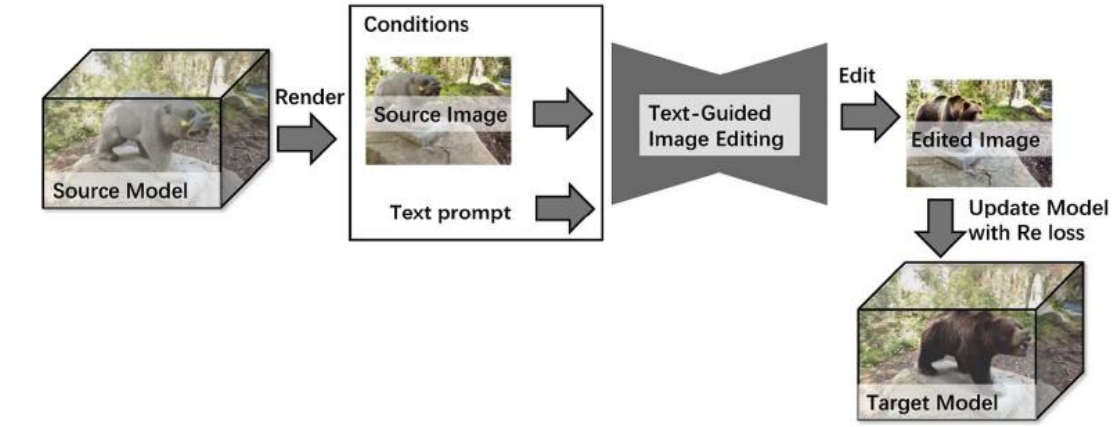
This recovered camera and depth information serves as the geometric foundation for subsequent 3DGS training and editing in our system.

2.2 Text-Driven 3D Editing Methods

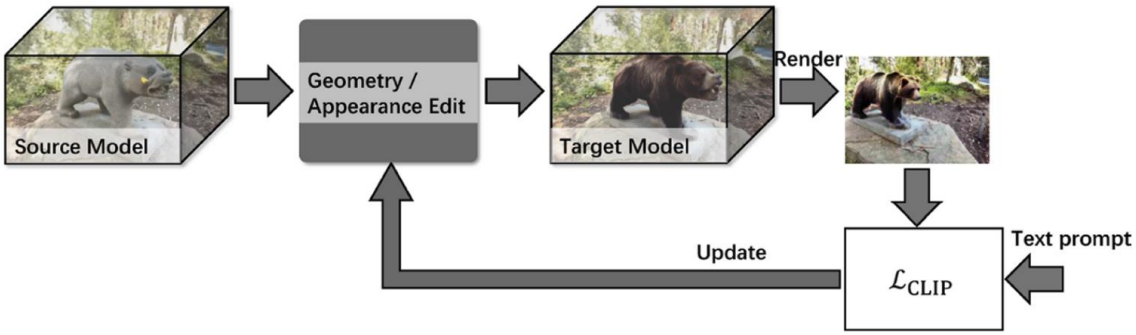
In this section, we discuss text-driven 3d editing methods. A primary challenge in text-driven 3D editing is to modify specific regions of a 3D scene based on natural language prompts, while preserving unedited areas. To address this, existing methods are broadly categorized into two strategies:

- **Direct 3D Editing**: Text prompts are used to directly modify 3D representations (e.g., meshes or NeRFs) by optimizing their geometry, texture, or volumetric content using differentiable rendering and pre-trained text-image models.
- **2D-3D Lifting**: Edits are first performed on individual 2D views using text-conditioned diffusion models or image editors. These edited views are then consolidated back into a coherent 3D representation.

As illustrated in Fig. 2.4, the key difference lies in where the editing operation takes place—either directly in 3D space, or indirectly via 2D projections.



(a) The 2D-3D lifting-based 3D editing with reconstruction loss.



(b) Direct 3D editing using CLIP loss.

Figure 2.4: Overview of 2D-3D lifting-based 3D editing and direct 3d editing reprinted from [34].

2.2.1 Direct 3D Editing

Direct 3D editing enables modifications to the geometry or texture of 3D representations based on target text prompts, by leveraging pre-trained text-vision models such as CLIP [43]. This approach typically employs a differentiable rendering pipeline that connects the 3D model to the supervision signal from these models.

Prior work by Fan et al. [17] introduced a pipeline that generates rendered images of a target 3D model and optimizes them to match the text prompt using CLIP similarity. For instance, X-Mesh [36] allows editing of both geometry and texture of a mesh representation.

More recent methods are inspired by score distillation sampling (SDS) [42], which enables text-guided 3D optimization using gradients from pre-trained diffusion models. For example, Latent-NeRF [37] trains a NeRF in the latent space of a diffusion model for editing, which is then mapped back to the original NeRF representation in RGB space for high-quality rendering.

However, direct 3D editing methods often inherit inherent limitations from their 2D generative model foundations, such as difficulties in generating objects at specific orientations. Moreover, without leveraging priors from 2D text-based editing tasks, these methods lack precise spatial control. In particular, they offer limited ability to restrict edits to user-specified regions and often lack structural constraints, making it challenging to ensure that unreferenced parts of the scene remain consistent after editing [34].

Direct 3D Editing

- **X-Mesh** [36]: Edits both geometry and texture of a mesh using text prompts through differentiable rendering and CLIP supervision. However, the method relies on mesh inputs and may suffer from limited flexibility when dealing with sparse or noisy input views, as it depends on high-quality mesh reconstructions that are difficult to obtain in such conditions, and lacks compatibility with alternative 3D representations such as point clouds or implicit fields.

- **Progressive3D** [11]: Applies progressive text-guided edits directly in NeRF space, allowing sequential control over editing steps. Nonetheless, it inherits NeRF's slow optimization and may struggle with view inconsistency in complex edits, as localized modifications can lead to geometric and appearance mismatches across viewpoints due to NeRF's implicit volumetric representation and lack of explicit surface constraints.

- **NeRF-Art** [55]: Converts mesh inputs into NeRF for direct CLIP-guided editing of appearance and structure. Yet, the reliance on mesh-to-NeRF conversion can limit edit quality if the initial mesh lacks detail or accuracy.

- **LENeRF** [25]: Enhances local editability by using a triplane representation within NeRF, optimized directly from multi-view images. However, the method still faces NeRF's computational inefficiencies and lacks robust structural constraints during editing, as it does not enforce explicit surface boundaries or semantic consistency, which may lead to unrealistic deformations or view-dependent artifacts.

- **CLIP-NeRF** [54]: Aligns NeRF-rendered views with text prompts using CLIP similarity for direct editing of neural radiance fields. This approach is vulnerable to the Janus problem and often results in ambiguous object orientations.

- **3D Paintbrush** [13]: Enables intuitive editing of mesh parts by associating textual prompts with segmentation masks. While user-friendly, it is limited to mesh-based pipelines and cannot be directly extended to implicit representations like NeRF or 3DGS.

2.2.2 2D-3D Lifting 3D Editing

Compared with direct 3D editing, 2D-3D lifting-based methods can benefit from editing priors of text-based image editing to complete 3D editing, and improve the controllability of the 3D editing process. 2D-3D Lifting 3D editing has achieved state-of-the-art results

in novel view synthesis. However, editing a single image at a time lacks 3D awareness, which compromises multi-view consistency during the editing process. When editing results across different views exhibit inconsistencies, 2D-3D lifting methods struggle to achieve proper consolidation in 3D space [7, 59, 61]. The development of effective and consistent editing methods from 2D to 3D remains a fundamental challenge in the field.

Recently, IN2N proposed by Haque et al. [21] uses an image-conditioned diffusion model (InstructPix2Pix proposed by Brooks et al. [4]) to iteratively edit the input images while optimizing the underlying scene, resulting in an optimized 3D scene that respects the edit instruction. However, IN2N suffers from slow convergence and unstable training due to the need for iterative multi-view refinements. GS2GS proposed by Vachha et al. [53] attempts to employ 3DGS to improve editing efficiency while maintaining consistency across views based on IN2N. Nevertheless, it inherits IN2N's iterative update scheme and fails to enforce global consistency across all viewpoints, often resulting in residual artifacts in distant or occluded views. ViCA-NeRF, proposed by Dong et al. [15], utilizes depth maps predicted by NeRF to establish dense correspondences across multiple views. These correspondences act as geometric constraints, guiding the model to maintain structural consistency during the editing process. Yet, the quality of NeRF-predicted depth maps is often suboptimal, especially in textureless or occluded regions, which compromises the reliability of the correspondence-based constraints. GaussCtrl proposed by Wu et al. [61] achieves multi-view consistency by depth-conditioned editing that enforces geometric consistency across multi-view images by leveraging naturally consistent depth maps. Despite its effectiveness, it still relies on pre-rendered depth without refinement, leading to inaccurate guidance in areas with fine structures or motion blur.

We briefly describe the representative text-driven 3D editing methods listed in Table 2.1, categorized by their editing strategies.

2D-3D Lifting

- **Instruct3D-to-3D** [27]: Edits multi-view images using InstructPix2Pix, lifting re-

sults into a DVGO-based 3D representation. However, the method lacks explicit 3D consistency constraints, as the image-space edits are applied independently across views without enforcing geometric coherence or cross-view feature alignment. This often results in mismatched textures or structural artifacts when the edited scene is reconstructed or rendered from novel viewpoints.

- **DreamEditor** [71]: Applies 2D edits and consolidates them into a mesh-based neural field (MNF) for 3D rendering. The mesh dependency limits adaptability to scenes where surface geometry is noisy or incomplete.

- **GSEdit** [41]: Combines 2D diffusion editing with 3D Gaussian Splatting (3DGS) to enable fast and consistent edits. Despite its speed, the editing process does not explicitly enforce depth consistency across views or splats. Since diffusion-based edits operate in image space without tightly coupling to the underlying 3D structure, this can lead to subtle geometric artifacts such as depth discontinuities or spatial misalignments when the splats are reprojected or viewed from novel angles.

- **InstructN2N** [21]: Uses InstructPix2Pix to iteratively edit NeRF-rendered views, refining the scene accordingly. However, iterative editing may accumulate artifacts over successive generations, as errors from earlier render-edit cycles can compound due to the lack of global consistency enforcement. Moreover, the method is sensitive to the quality of the initial NeRF renderings—low-fidelity or noisy views can mislead the diffusion model, resulting in degraded or unstable edits across views.

- **EfficientN2N** [50]: Improves runtime efficiency of N2N-style pipelines via compact NeRF representations. Yet, it still inherits the limitations of 2D editing inconsistency and lacks geometric alignment mechanisms.

- **RMNE** [39]: Exploits reference-view matching to maintain semantic alignment across NeRF edits. However, the reliance on a limited set of manually or heuristically selected reference views may introduce semantic bias, leading to overfitting

on specific viewpoints. As a result, unreferenced or occluded regions may be under-constrained, causing inconsistencies or missing details when the scene is rendered from novel perspectives.

- **DN2N** [18]: Incorporates diffusion models into the NeRF editing loop to better capture instruction-driven edits. Still, the lack of explicit geometric guidance—such as surface normals, depth priors, or correspondence constraints—makes it difficult to enforce consistent geometry across views, often resulting in misaligned textures or structure distortion in complex or occluded regions.

- **FreeEditor** [28]: Supports open-ended text instructions without predefined semantic categories, enabling more flexible and user-driven 3D editing. This flexibility, however, comes at the cost of reduced predictability and structural control during 3D reconstruction, as the absence of constrained semantics or category priors makes it harder to ensure consistent geometry, coherent shape transformations, or adherence to physically plausible structures—especially when interpreting ambiguous or abstract user prompts.

- **ViCA-NeRF** [15]: Uses NeRF-predicted depth maps to enforce geometric consistency across views during editing. Nonetheless, NeRF-predicted depth can still be noisy or low-resolution, limiting the overall precision.

- **LatentEditor** [30]: Edits latent NeRF features using 2D diffusion models, then decodes them back to RGB space. While computationally efficient, it may lose spatial interpretability during latent-to-RGB decoding, as the learned latent features lack direct geometric or semantic correspondence to the scene structure, making it difficult to control or trace how edits affect 3D consistency and visual appearance.

- **GaussianEditor** [57]: Leverages edited 2D views and integrates them into a 3D Gaussian Splatting (3DGS) model for high-quality rendering. However, it lacks strong multi-view consistency constraints during 2D-to-3D integration, as the edits

applied to individual images are not explicitly aligned across views, which can lead to inconsistencies in geometry or appearance when reconstructing complex scenes with occlusions or non-rigid structures.

- **GaussCtrl** [61]: Incorporates depth-conditioned 2D editing with 3DGS to enforce multi-view consistency through cross-view depth guidance. While effective, the method depends heavily on the quality of estimated depth maps—errors in depth prediction can propagate through the editing pipeline, resulting in misaligned details or distorted geometry in regions requiring precise fine-scale edits.

- **VcEdit** [59]: Enhances consistency in 3DGS editing via view-correlated constraints across 2D edits. Despite improved coherence, it does not fully model occlusion-aware geometry during editing.

- **GaussianEditor-SC** [9]: Introduces scene-consistency constraints to improve the fidelity of lifted 3DGS edits. Nevertheless, it assumes well-aligned input views and may struggle under strong motion blur or incomplete views.

Method	Input	Output	Representation	Editing Strategy
X-Mesh [36]	Mesh	Vertex-colored mesh	Mesh	3D Direct
Progressive3D [11]	MV	Novel views	NeRF	3D Direct
NeRF-Art [55]	Mesh	Novel views	NeRF	3D Direct
LENeRF [25]	MV	Novel views	Triplane	3D Direct
CLIP-NeRF [54]	MV	Novel views	NeRF	3D Direct
3D Paintbrush [13]	Mesh	Textured mesh	Mesh	3D Direct
Instruct3D-to-3D [27]	MV	Novel views	DVGO	2D-3D Lifting
DreamEditor [71]	MV	Novel views	MNF	2D-3D Lifting
GSEdit [41]	MV	Textured mesh	3DGS	2D-3D Lifting
InstructN2N [21]	MV	Novel views	NeRF	2D-3D Lifting
EfficientN2N [50]	MV	Novel views	NeRF	2D-3D Lifting
RMNE [39]	MV	Novel views	NeRF	2D-3D Lifting
DN2N [18]	MV	Novel views	NeRF	2D-3D Lifting
FreeEditor [28]	MV	Novel views	NeRF	2D-3D Lifting
ViCA-NeRF [15]	MV	Novel views	NeRF	2D-3D Lifting
LatentEditor [30]	MV	Novel views	NeRF	2D-3D Lifting
GaussianEditor [57]	MV	Novel views	3DGS	2D-3D Lifting
GaussCtrl [61]	MV	Novel views	3DGS	2D-3D Lifting
VcEdit [59]	MV	Novel views	3DGS	2D-3D Lifting
GaussianEditor-SC [9]	MV	Novel views	3DGS	2D-3D Lifting

Table 2.1: "MV" stands for Multi-View Images. "MNF" denotes the mesh-based neural field. "DVGO" denotes Direct Voxel Grid Optimization. "TensoRF" denotes tensorial radiance fields [6].

2.3 Depth Enhancement Methods

Depth information plays a vital role in applications like 3D reconstruction, autonomous systems, and augmented reality, which encompasses several critical tasks that improve the quality and utility of depth maps for downstream 3D applications. These include completion, which addresses missing or invalid depth values due to occlusion or sensor limitations by inferring geometry from neighboring regions or RGB guidance; denoising, which removes random noise—common in low-cost sensors—through smoothing, edge-aware regularization, or learning-based residual correction; and deblurring, which restores sharp depth discontinuities in motion-blurred or temporally integrated maps using temporal cues or learned blur-reversal models. Together, these techniques enhance depth fidelity, structural integrity, and consistency, which are essential for high-quality 3D reconstruction and editing.

In our 3D editing system, enhancing depth maps is necessary to improve geometric awareness and editing consistency across views. However, conventional depth enhancement networks, which are typically trained on large-scale RGB-D datasets, do not meet the real-time or limited-data constraints of our application. Moreover, the lack of ground truth depth supervision in our setting presents an additional challenge. Therefore, it is important to develop lightweight, unsupervised or self-supervised methods that can operate effectively with limited data.

Depth enhancement methods can be broadly categorized into two main approaches: **optimization-based** and **learning-based** methods, each with distinct strengths and limitations.

- **Optimization-based methods** treat depth enhancement as an ill-posed inverse problem, typically incorporating handcrafted priors or regularization terms to stabilize the solution. These methods are often model-driven and do not require large training datasets. For example, Wang et al. [56] proposed an RGB-guided depth

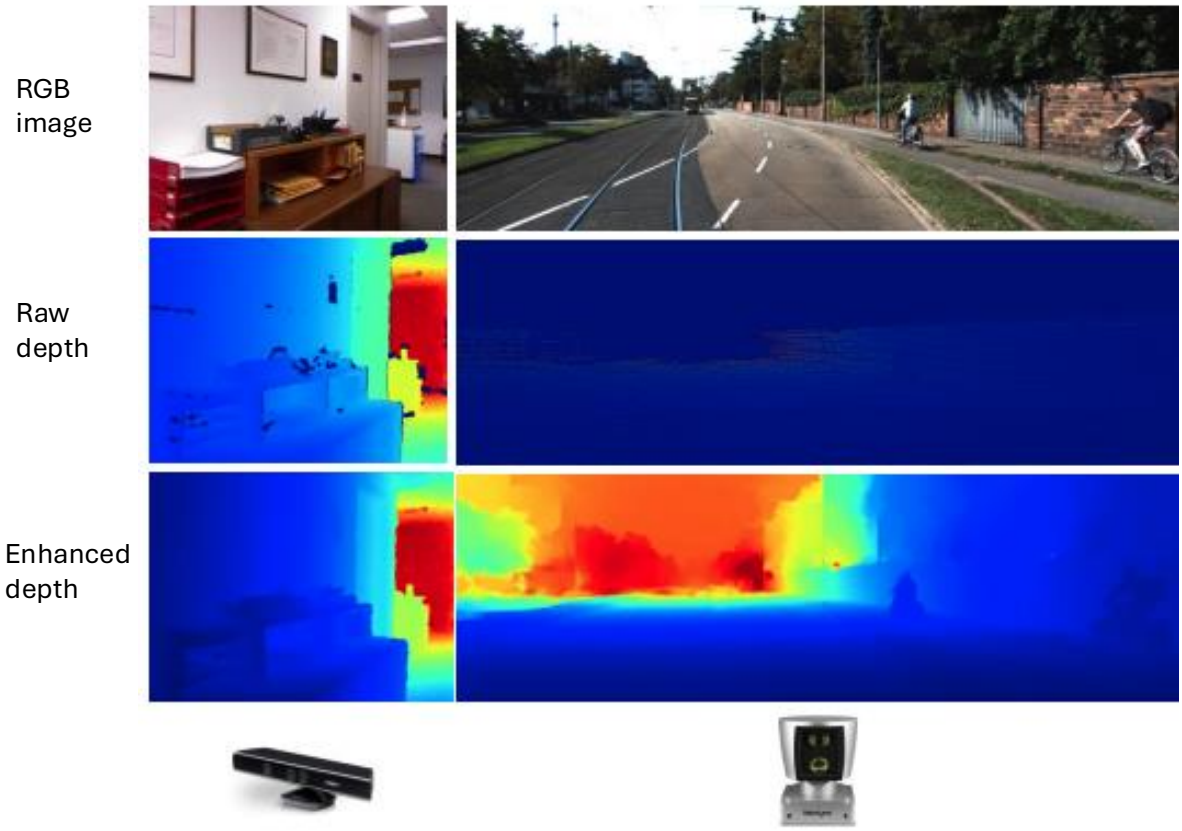


Figure 2.5: Illustration of the sparsity and variability in depth maps captured by different sensors. The middle column shows the raw sparse depth outputs. The left example is from a Kinect sensor in an indoor setting, while the right example is from a LiDAR scan in an outdoor street. This figure highlights the need for robust depth enhancement methods capable of handling diverse sensing conditions and noise patterns. Reprinted from [24]

.

recovery framework that jointly utilizes local and global contextual cues from RGB images to reconstruct high-quality depth maps. While effective under ideal conditions, such approaches often assume accurate alignment between RGB and depth inputs. In real-world multi-view scenarios with occlusions or sensor misalignment, their performance may degrade significantly, limiting their robustness and generalization.

- **Learning-based methods** utilize neural networks trained on RGB-D datasets to directly learn depth refinement from data. These approaches can model complex patterns and nonlinear relationships, leading to high-quality outputs. For instance, SGNet [60] enhances structural precision using a gradient-frequency-aware architecture, while IPPNet [8] proposes an intrinsic phase-preserving design for cross-modality depth enhancement. Despite their impressive accuracy and detail preservation, these methods typically require extensive labeled datasets for supervised training. This dependency limits their scalability and applicability in data-scarce environments, especially for multi-view or real-time applications where paired ground truth is unavailable.

2.4 Motion Deblur Methods

Motion blur is a critical factor that affects the fidelity of 3D reconstruction and editing, especially in dynamic scenes captured under low-light or fast-moving conditions. Technically, motion blur arises from the integration of scene radiance over a finite camera exposure period. Under fast motion or low-light conditions, this integration leads to severe spatial smearing. This degradation is commonly modeled as a spatial convolution process:

$$\mathbf{Y}_b = \mathbf{B} * \mathbf{X}_s + \boldsymbol{\theta}_\mu, \tag{2.3}$$

where $\mathbf{Y}_b$ is the observed blurry image, $\mathbf{X}_s$ is the latent sharp image, $\mathbf{B}$ denotes the blur kernel, and $\boldsymbol{\theta}_\mu$ represents additive noise. To restore sharp images from blurred observa-

tions, a variety of deblurring methods have been proposed. We categorize and compare these methods based on their design philosophy and 3D suitability. Below, we review the representative approaches evaluated in our experiments, highlighting their strengths and limitations in the context of multi-view 3D Gaussian Splatting.

MPRNet [64]: A single-image deblurring network that adopts a multi-stage, progressive refinement framework with hierarchical supervision. It achieves high-quality deblurring by fusing local and global features through residual blocks. *Advantages:* – Strong generalization on a wide range of natural blur patterns. – Effective for moderate blur in single-view scenarios. *Limitations:* – Lacks awareness of 3D structure and inter-view coherence. – Operates independently on each frame, causing inconsistent geometry when applied to multi-view inputs. – Cannot leverage temporal redundancy or scene priors across views.

PVD [48]: A recurrent video deblurring method that progressively refines predictions across temporal frames using memory units. *Advantages:* – Incorporates temporal information through recurrence. – Improves consistency across video frames in 2D scenarios. *Limitations:* – Temporal modeling is weak under large motion. – Like MPRNet, it lacks any 3D scene understanding and cannot enforce view-to-view consistency. – Struggles with depth ambiguity and occlusion when applied to 3DGS data.

SRN-Deblur [52]: A scale-recurrent neural network that performs deblurring in a coarse-to-fine manner with shared weights across multiple scales. *Advantages:* – Robust to different levels of blur intensity due to multi-scale structure. – Efficient and compact compared to other large CNN models. *Limitations:* – Primarily designed for 2D images, not suitable for enforcing 3D spatial consistency. – Cannot handle inter-view variance or temporal misalignment in multi-view inputs. – Tends to oversmooth fine details that are essential for accurate geometry recovery.

DeblurNeRF [35]: A NeRF-based approach that integrates a blur-aware rendering model into the volumetric rendering process. It estimates both scene radiance and blur

parameters jointly from blurred inputs. *Advantages:* – Explicit modeling of blur within the 3D volume improves geometric consistency. – Joint optimization over multi-view inputs benefits sharp view synthesis. *Limitations:* – Computationally intensive due to volumetric rendering. – Requires significant training time and GPU memory. – Struggles with high-frequency appearance restoration and often fails to reconstruct sharp textures.

BAD-NeRF [58]: An advanced NeRF-based method that disentangles motion blur and scene content using a blur-aware volume and learns better priors for deblurring under large camera motion. *Advantages:* – State-of-the-art results in volumetric deblurring with improved motion modeling. – More accurate disentanglement of geometry and blur compared to earlier NeRF-based models. *Limitations:* – High computational cost and long optimization cycles. – Complex architecture makes real-time application impractical. – Limited scalability to dense scenes or long sequences due to memory bottlenecks.

To overcome these limitations, we propose a 3DGS-based deblurring strategy that retains high-quality geometry and appearance restoration while reducing computational overhead. Unlike NeRF-based methods, 3DGS allows direct manipulation of point-based Gaussian primitives, enabling faster rendering and convergence. By integrating a low-pass filtering strategy in the optimization loop, our method effectively suppresses high-frequency noise introduced by motion blur. This design ensures robust reconstruction from blurred views with better efficiency, consistency, and visual quality.

2.5 Summary

This chapter reviewed the foundational technologies relevant to our 3D editing framework. We first introduced diffusion models and their impact on deep generative editing. We then surveyed 3D representations, highlighting 3DGS's advantages in rendering speed and training efficiency. We compared direct 3D editing and 2D-3D lifting strategies, outlining their trade-offs in consistency, controllability, and scalability. Finally, we discussed auxil-

iary tasks such as depth enhancement and motion deblurring, which are critical for robust 3D reconstruction under real-world conditions.

While progress has been made in efficient and controllable 3D editing, challenges remain in ensuring geometric consistency and handling degraded inputs. Our research addresses these limitations by integrating depth-aware diffusion editing and blur-robust 3DGS optimization into a unified framework.

Chapter 3

Methodology

In this chapter, we present a unified methodology for high-quality 3D scene editing and reconstruction based on the 3D Gaussian Splatting (3DGS) representation. As illustrated in Algorithm 3.1, our approach consists of two main components:

- **Efficient 3D Editing with Complementary and Consensus Information:** To address the limitations of conventional 2D-3D lifting frameworks—particularly the lack of view consistency and geometric awareness—we propose a depth-guided editing strategy. The rendered depth maps from 3DGS are refined using a **Complementary Information Mutual Learning Network (CIMLN)**, which fuses features from both depth and RGB views to suppress texture noise and enhance structural fidelity. To further ensure consistent edits across multiple viewpoints, we introduce **Wavelet Consensus Attention (WCA)** to align latent representations across views in both spatial and frequency domains. This combination enables precise, semantically aligned, and multi-view-consistent editing in a non-iterative fashion.

- **Joint 3DGS and Camera Poses Optimization for Motion Blur Images :** To recover high-fidelity 3D scenes from real-world motion-blurred images, we propose a joint optimization framework that refines both the 3DGS representation and camera

poses. The pipeline first estimates a motion trajectory by interpolating between the start and end poses using SE(3) interpolation. To mitigate high-frequency artifacts introduced during blur, we incorporate a **Low-Pass Filtering (LPF)** strategy into the Gaussian convolution process. This regularization smooths overlapping Gaussians while preserving geometric details, enabling accurate 3D reconstruction under severe motion blur conditions.

Together, these two modules form a comprehensive solution that extends 3D Gaussian Splatting to support text-driven 3D editing with high view consistency, and motion-deblurred 3D scene reconstruction with structural stability. The proposed framework eliminates the need for iterative dataset updates and achieves robust performance across both static and dynamic real-world scenarios.

Algorithm 3.1 outlines our approach for efficient 3D editing based on Gaussian Splatting, incorporating both complementary and consensus information. If motion blur is detected, a low-pass filter is first applied to each 3D Gaussian to suppress high-frequency noise, followed by motion trajectory estimation and view sampling along this trajectory to generate sharp intermediate frames. Next, multi-view RGB images and depth maps are rendered from the original 3D scene. The depth maps are refined using CIMLN to ensure spatial consistency across views, yielding $\mathcal{D}^*$. A diffusion-based editing process is then performed. Starting from a noisy latent code z_T initialized from the rendered images, the model iteratively denoises the latent code over T steps. At each step, the model predicts noise conditioned on the text prompt, timestep, and refined depth. To enhance consistency, we apply Wavelet Consensus Attention (WCA) by decomposing latent features into frequency bands, aligning them across views using consensus attention, and reconstructing the aligned latent via inverse wavelet transform. After denoising, the final latent z_0 is decoded via a VAE to produce edited images $\hat{\mathcal{R}}$, which serve as supervision for optimizing the original 3D Gaussian scene $\mathcal{G}$, resulting in the edited scene $\hat{\mathcal{G}}$.

3.1 Preliminaries

To facilitate the understanding of our proposed 3D editing framework, we first introduce the frequently used notations and definitions adopted throughout this paper. Establishing a clear and consistent notation system is crucial for accurately describing the various mathematical operations, tensor representations, and algorithmic procedures involved in our method.

In particular, we differentiate between tensors, matrices, and vectors using distinct typographic conventions to avoid ambiguity and enhance clarity. Specifically, we use: Calligraphy letters (*e.g.*, $\mathcal{A}$, $\mathcal{G}$, $\mathcal{D}$) to denote high-dimensional tensors or complex data structures such as 3D Gaussian sets, multi-view image collections, and volumetric representations. Bold uppercase letters (*e.g.*, $\mathbf{A}$, $\mathbf{R}$, $\mathbf{T}$) to represent matrices, such as camera transformation matrices, wavelet decomposition matrices, or feature projections. Bold lowercase letters (*e.g.*, $\mathbf{a}$, $\mathbf{x}$, $\mathbf{t}$) to denote vectors, including spatial coordinates, translation vectors, and feature embeddings. Regular lowercase letters (*e.g.*, a, x, t) for scalar values, indices, or time steps.

To promote consistency, we summarize the key notations used throughout the paper in Table 3.1, providing their corresponding descriptions and dimensions. This table serves as a quick reference guide for readers, helping to disambiguate symbols and ensure a coherent understanding of the mathematical formulations introduced in subsequent sections.

Moreover, the definitions outlined here are essential for understanding the various components of our framework, including: the parameterization of 3D Gaussian splats, the modeling of camera motion through SE(3) transformations, the depth refinement process in our 3D editing system. By unifying these concepts under a shared notation system, we ensure that the formulation of our algorithms and the analysis of experimental results remain clear, precise, and accessible. This notation convention is consistently followed throughout the entire thesis, covering theoretical derivations, algorithmic descriptions, and experimental evaluations.

Table 3.1: Primary notations and corresponding information.

Notation	Meaning
$\mathcal{A}$,$\mathbf{A}$,a	Tensor, Matrix, Vector
n	Number of samples
V	Number of views
$\|\cdot\|_1$	l_1-norm (sum of absolute values)
$\|\cdot\|_2$	l_2-norm (Euclidean norm)
$\mathbf{SE}(\mathbf{3})$	Special Euclidean Group in 3D (rotation + translation)
$\mathbf{SO}(\mathbf{3})$	Special Orthogonal Group in 3D (rotation group)
SiLU	Sigmoid Linear Unit activation function
LN	Layer Normalization
Conv1d	1D Convolution (temporal or sequence modeling)
Conv2d	2D Convolution (spatial feature extraction)
WT	Wavelet Transform
IWT	Inverse Wavelet Transform
∇_x, ∇_y	Gradient operators along x and y axes
$\mathcal{G}$	3D Gaussian Splatting representation
$\mathbf{T}$	Camera pose in $\mathbf{SE}(\mathbf{3})$
$\mathbf{T}_{\text{start}}, \mathbf{T}_{\text{end}}$	Camera poses at exposure start and end
p	Text prompt for editing
$\mathcal{D}$	Depth maps
$\mathcal{R}$	Rendered RGB images
$\hat{R}$	Edited RGB images
z_t	Latent code at timestep t in diffusion
e^t	Predicted noise at timestep t

3.2 Efficient 3D Editing based on Gaussian Splatting with Complementary and Consensus Information

To overcome these limitations, our method eliminates the need for iterative dataset updates, significantly reducing computational costs while improving multi-view consistency. Instead of relying on iterative refinement, our approach incorporates depth information at the image editing stage to ensure consistency across different viewpoints.

Since depth maps are rendered from a consistent 3D Gaussian Splatting (3DGS) representation, they serve as a reliable structural reference during image modification. By leveraging depth-aware editing, our method enhances spatial coherence, ensuring that modifications applied to one view remain consistent when projected into other viewpoints. This depth-guided approach preserves geometric integrity, minimizes inconsistencies, and produces more stable and visually accurate edits across the entire dataset.

We achieve these through two key components: (1) CIMLN for precise depth enhancement, and (2) WCA mechanism for latent code alignment in ControlNet as shown in Fig. 3.1.

3.2.1 Reconstruction-Based 2D-3D Lifting Using 3DGS

We adopt the **3D Gaussian Splatting (3DGS)** representation for efficient 3D scene reconstruction, modeled as a mixture of Gaussians:

$$\mathcal{G} = \{(\sigma_i, \mu_i, \Sigma_i, c_i)\}_{i=1}^{N}, \tag{3.1}$$

where $\sigma_i \geq 0$ denotes the opacity, $\mu_i \in \mathbb{R}^3$ is the mean position, $\Sigma_i \in \mathbb{R}^{3\times3}$ is the covariance matrix controlling the spatial extent and orientation, c_i represents the color of each Gaussian, and N is the total number of Gaussian ellipsoids. Although 3DGS provides a computationally efficient and differentiable representation for real-time rendering,

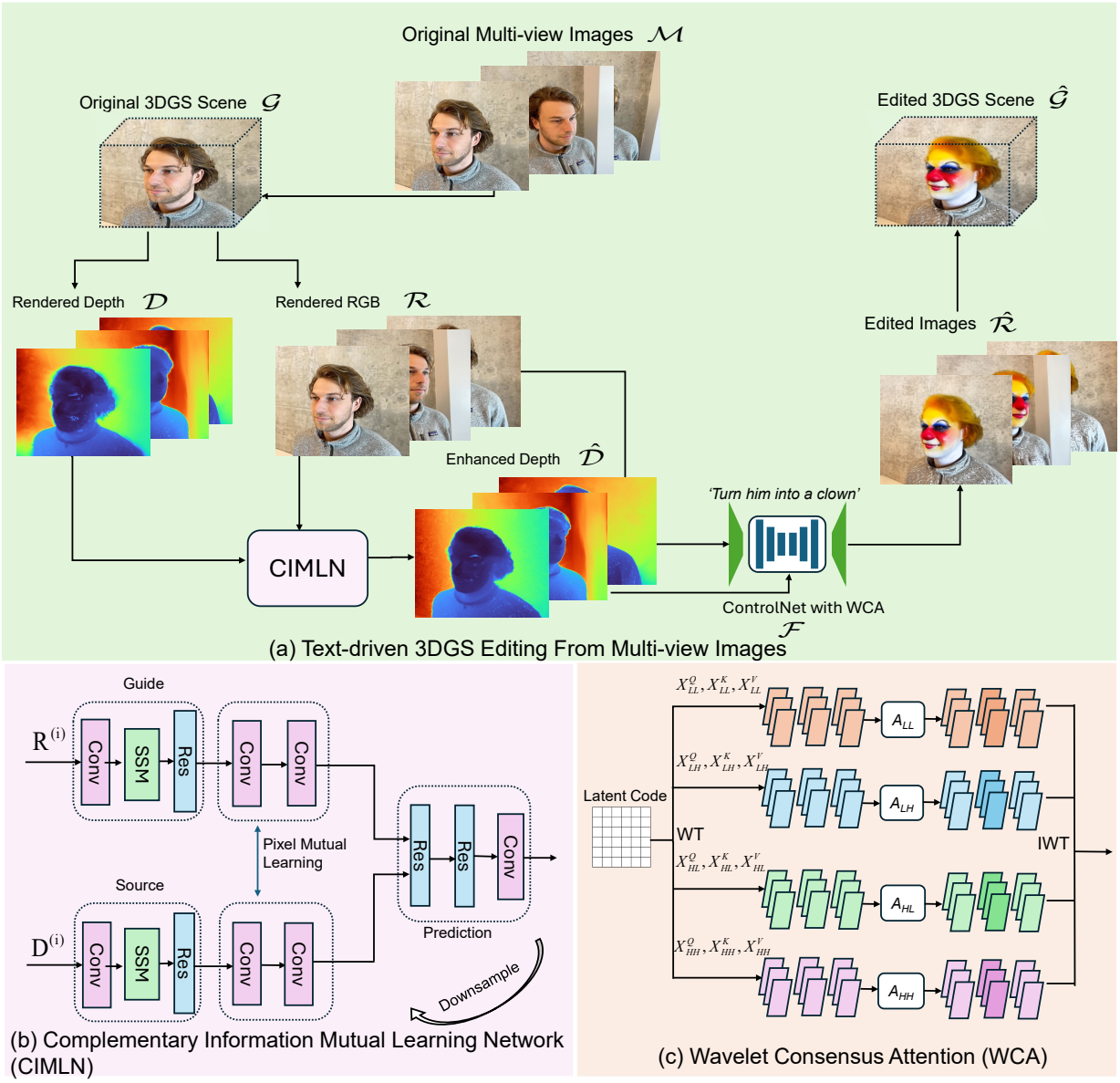


Figure 3.1: Overview of the proposed 3D Gaussian splatting editing system. (a) Multi-view images $\mathcal{M}$ are used to train a 3DGS model, from which the rendered depth and RGB images can be obtained from a certain viewpoint. (b) Rendered depth maps and RGB images are processed through Source and Guide branches respectively. Through pixel mutual learning and downsampling, the system enables self-supervised training. (c) The framework replaces ControlNet's image self-attention with WCA to better align images with reference views.

directly modifying these parameters often leads to noisy or unstable results, due to the entanglement of geometry and appearance attributes within the Gaussians [47]. This poses challenges for fine-grained editing tasks.

As illustrated in Fig. 3.1, we first render RGB images $\mathcal{R}$ and their corresponding depth maps $\mathcal{D}$ from the optimized 3DGS model. These rendered outputs serve as conditional inputs to ControlNet, enabling depth-aware image editing guided by external text prompts. However, the depth maps derived from the 3DGS representation typically lack high-frequency details and suffer from limited accuracy due to the inherent smoothness of Gaussian-based geometry. This results in inconsistent or artifact-prone editing when passed directly to ControlNet.

To address this issue, we propose CIMLN for depth map refinement. CIMLN takes the initial depth maps $\mathcal{D}$ and iteratively enhances them through multi-scale feature aggregation, yielding refined depth maps with sharper boundaries and improved structural consistency. These refined depth maps are critical for ensuring spatial alignment during conditional generation.

Once the enhanced depth maps $\mathcal{D}$ are obtained, we feed them into ControlNet F along with the rendered RGB images and a user-specified text instruction p. During the editing process, DDIM inversion [22] is applied to reverse the diffusion process. Specifically, given an edited image $\hat{R}$, we invert it into its corresponding latent code z_0 and recover the intermediate noisy latent code z_t at timestep t. The noise prediction at step t is computed as:

$$e^t = F(z_t; t, p, \mathcal{D}), \tag{3.2}$$

and the latent update is:

$$z_{t+1} = \sqrt{\alpha_{t+1}} z_t - \frac{\sqrt{1-\alpha_t} \cdot e^t}{\sqrt{\alpha_t}} + \sqrt{1-\alpha_{t+1}} e^t, \tag{3.3}$$

where α_t is the noise schedule controlling the trade-off between signal and noise at each timestep.

To further enhance the temporal and spatial coherence of the diffusion process, we introduce WCA, which aligns latent features across different steps in both spatial and frequency domains. This mitigates feature drift and stabilizes the denoising trajectory. After applying WCA, the updated noise $\hat{e}^t$ is used to refine the denoising step:

$$z_{t-1} = \sqrt{\alpha_{t-1}} z_t - \frac{\sqrt{1-\alpha_t} \cdot \hat{e}^t}{\sqrt{\alpha_t}} + \sqrt{1-\alpha_{t-1}} \hat{e}^t. \tag{3.4}$$

Once we obtain the final latent representation z_0, it is decoded into the edited image $\hat{R}$ through a Variational Autoencoder (VAE) [31] decoder. Specifically, the VAE consists of an encoder–decoder architecture that models the distribution of natural images within a learned latent space. The decoder transforms the low-dimensional latent code z_0 back into the high-dimensional RGB image space through a learned generative process:

$$\hat{R} = \text{Decoder}(z_0). \tag{3.5}$$

Here, $\hat{R}$ denotes the edited image. The VAE is trained to accurately reconstruct images while enforcing a Gaussian prior over the latent space. This regularization encourages smooth interpolation and ensures that small changes in z_0 lead to semantically meaningful variations in the output image. This property is crucial for controlled image editing within diffusion models.

Finally, the edited images $\hat{R}$ generated by ControlNet are used to supervise the update of the original 3DGS model. By optimizing the parameters $\mathcal{G}$ to minimize the discrepancy between the rendered images from the edited 3DGS and $\hat{R}$, we achieve 3D-aware editing that faithfully propagates 2D modifications back into the 3D representation, enabling consistent rendering from novel viewpoints.

ControlNet

Before detailing our proposed modules, we first introduce ControlNet, the foundational conditioning mechanism upon which our architecture is built. ControlNet serves as a powerful framework for injecting auxiliary information—such as depth maps or edge cues—into

diffusion-based models without disrupting their pre-trained weights. As our method adapts and extends ControlNet to enable depth-guided and view-consistent 3D editing, understanding its core principles is critical to contextualizing the modifications we introduce. Specifically, we modify ControlNet's internal structure to incorporate our WCA and depth refinement pipeline, making it well-suited for 3D Gaussian Splatting scenarios.

In particular, we modify ControlNet by replacing its internal self-attention with our proposed WCA, and adapt it to the 3DGS editing task using dual-branch mutual learning. To establish this foundation, we briefly describe ControlNet's mechanism below.

ControlNet injects additional conditions into the blocks of a neural network as shown in Fig. 3.2. Here,the term *network block* to refer to a set of neural layers that are commonly put together to form a single unit of a neural network, e.g., ResNet block, Conv-BN-ReLU block, multi-head attention block, transformer block, etc.

Suppose $F(\cdot;\Theta)$ is such a trained neural block, with parameters Θ, that transforms an input feature map x into another feature map y as:

$$y = F(x;\Theta). \tag{3.6}$$

In our setting, x and y are usually 2D feature maps, i.e., $x \in \mathbb{R}^{h\times w\times c}$, where h, w, and c denote the height, width, and number of channels in the feature map, respectively (Figure 2a).

As shown in Fig. 3.2, The U-Net architecture of Stable Diffusion is augmented with a ControlNet, which is connected to both the encoder blocks and the middle block of the network. The locked (gray) blocks represent the original architecture of Stable Diffusion V1.5 (and V2.1, as both versions share the same U-Net design). To integrate the ControlNet, additional trainable components (shown as blue blocks) and zero convolution layers (shown as white blocks) are introduced. These components enable the ControlNet to guide the diffusion process while preserving the pre-trained parameters of the original Stable Diffusion model.

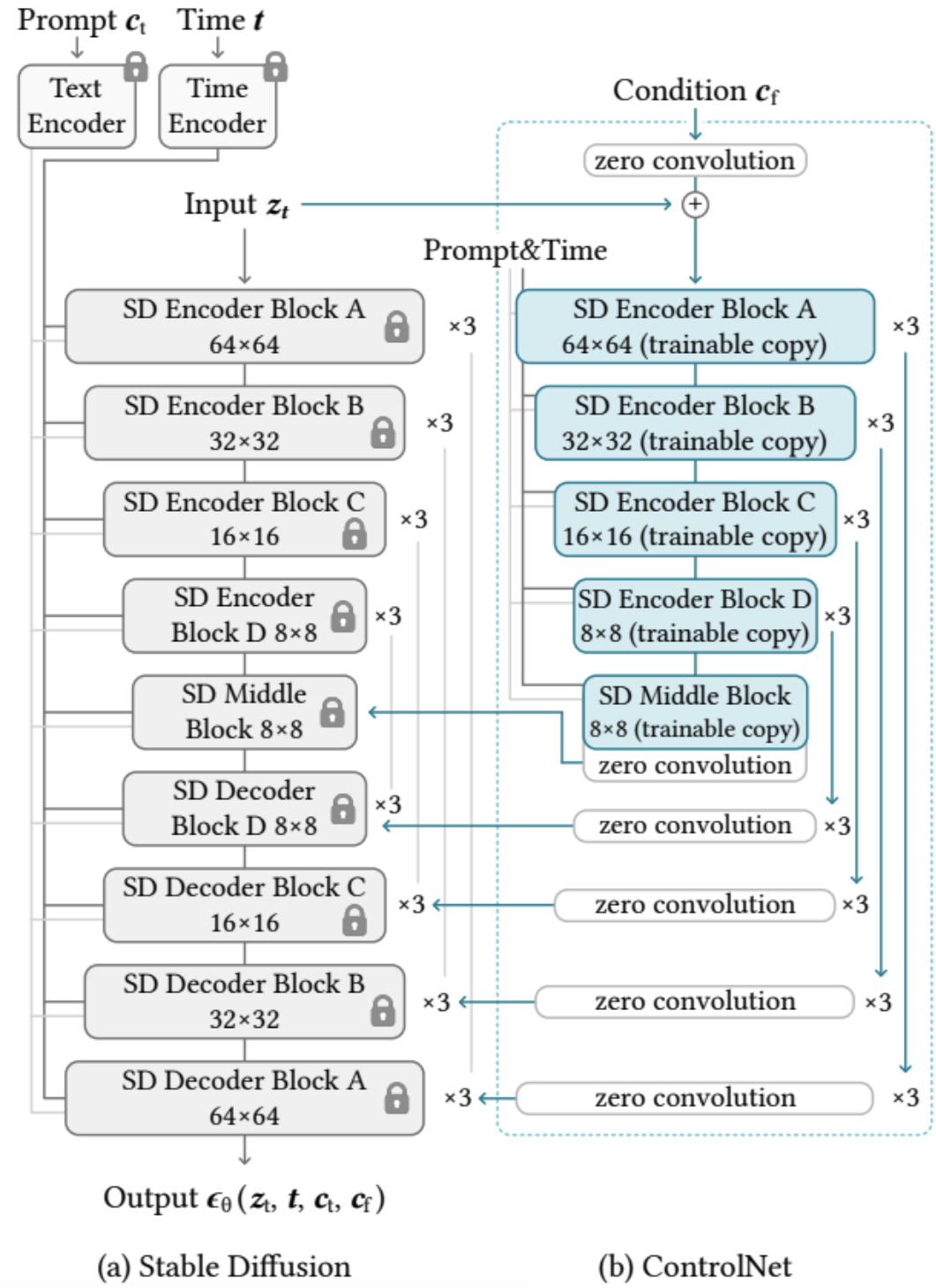


Figure 3.2: Overview of Stable Diffusion and ControlNet reprinted from [65].

The trainable copy is connected to the locked model via zero convolution layers, denoted as $Z(\cdot;\cdot)$. Specifically, $Z(\cdot;\cdot)$ represents a 1×1 convolution layer with both weights and biases initialized to zeros. To construct a ControlNet, we use two instances of zero convolutions with parameters Θ_{z1} and Θ_{z2}, respectively. The complete ControlNet then computes:

$$y = F(x;\Theta) + Z(F_c(x, c;\Theta_c);\Theta_{z1}) + Z(x;\Theta_{z2}), \tag{3.7}$$

where $F(x;\Theta)$ represents the locked model, and $F_c(x, c;\Theta_c)$ represents the trainable copy with conditioning input c. The zero convolution layers ensure that the additional pathways initially do not contribute to the output, allowing gradual fine-tuning without disrupting the pre-trained model.

Dataset Update

It is crucial to update the multi-view image datasets to ensure consistency in 3D reconstruction. Existing approaches such as IN2N [21] attempt to refine the 3D scene iteratively until convergence. However, this method lacks an explicit guarantee for maintaining consistency across multi-view images, leading to several challenges. Specifically, IN2N suffers from instability, slow processing speeds, and significant visual artifacts, particularly in complex scenes where viewpoint consistency is essential.

More recent methods follow a similar iterative refinement approach but introduces a different strategy to propagate edits across views. They select a subset of reference images from the dataset, applies modifications using InstructPix2Pix, and then propagates these edits to the remaining dataset by blending the projected reference images. While this blending process helps distribute modifications across views, it does not fully resolve the multi-view consistency issue and often introduces blurry artifacts, which degrade the quality of the reconstructed 3D model.

To address the limitations of prior approaches, our method avoids iterative dataset updates, which significantly reduces computational overhead. Unlike methods such as GaussCtrl that perform multiple rounds of geometry and appearance refinement, our approach integrates depth information directly into the image editing process. This encourages cross-view consistency early on and avoids redundant processing across iterations. Although our method demonstrates faster convergence in practice, we qualify this by noting that convergence refers specifically to the reduced number of editing steps and optimization passes needed to achieve consistent multi-view outputs. In our experiments, we observe that our pipeline converges in fewer editing steps compared to GaussCtrl, while maintaining or improving visual quality and coherence across views.

Since depth maps are rendered from a consistent 3D Gaussian Splatting (3DGS) representation, they serve as a reliable structural reference during image modification. By leveraging depth-aware editing, our method enhances spatial coherence, ensuring that modifications applied to one view remain consistent when projected into other viewpoints. This depth-guided approach preserves geometric integrity, minimizes inconsistencies, and produces more stable and visually accurate edits across the entire dataset. By integrating depth information directly into the editing process, our method achieves higher-quality reconstructions, avoids iterative inefficiencies, and mitigates common artifacts observed in prior approaches.

3.2.2 Complementary Information Mutual Learning Network

To estimate depth from the 3DGS representation, 3D Gaussians are projected into the camera space as 2D Gaussians. These 2D Gaussians are globally sorted by depth per pixel, and the depth estimates $\hat{\mathbf{D}}$ are computed through the discrete volume rendering approximation:

$$\hat{\mathbf{D}} = \sum_{i \in \mathbf{N}} d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{3.8}$$

where d_i denotes the depth of the i-th Gaussian in view space, and α_i denotes its opacity. While this method approximates per-pixel depth effectively, its accuracy is limited in regions with overlapping Gaussians, where the blending of multiple Gaussians introduces ambiguity. Moreover, the detailed textures rendered from the Gaussians often introduce undesirable high-frequency noise into the depth map, leading to artifacts and reduced precision.

To address these issues and enhance the depth maps rendered by the 3DGS model, we propose the CIMLN, which is designed to extract sharp structural information from the depth image while suppressing texture details from the color image. By learning complementary features from both the rendered depth and RGB images, CIMLN produces enhanced depth maps with sharper edges, improved geometric consistency, and higher spatial resolution.

As illustrated in Fig. 3.1 (b), the rendered depth maps $\mathcal{D}$ and rendered RGB images $\mathcal{R}$ are sent as inputs to the *source branch* and *guide branch*, respectively. Given the inputs $\mathbf{I}_s \in \mathbb{R}^{H\times W\times 1}$ (depth map) and $\mathbf{I}_g \in \mathbb{R}^{H\times W\times 3}$ (RGB image), CIMLN leverages a simplified State Space Model (SSM) [20,69] within each branch to capture global context and extract robust features, as shown in the following formulation:

$$
\begin{aligned}
\mathbf{F}_t &= \text{Conv2d}(\mathbf{F}),\\
\mathbf{X} &= \text{SiLU}(\text{Conv1d}(\mathbf{F}_t)),\\
\mathbf{Y} &= \text{SiLU}(\text{Conv1d}(\mathbf{F}_t)),\\
\mathbf{X}_t &= \text{LN}(\text{SSM}(\mathbf{X})),\\
\mathbf{X}_{\text{out}} &= \text{Linear}(\mathbf{X}_t \odot \mathbf{Y}),
\end{aligned}
\tag{3.9}
$$

where $\mathbf{F}$ denotes the input feature map, `Conv2d` and `Conv1d` are convolutional layers, SiLU is the Sigmoid Linear Unit activation function, LN denotes Layer Normalization, and $\odot$ represents element-wise multiplication. SSM effectively captures long-range dependencies across spatial dimensions, providing a global receptive field that complements the local feature extraction of traditional convolutional layers. Related proof of ssm can be found in Appendix A.

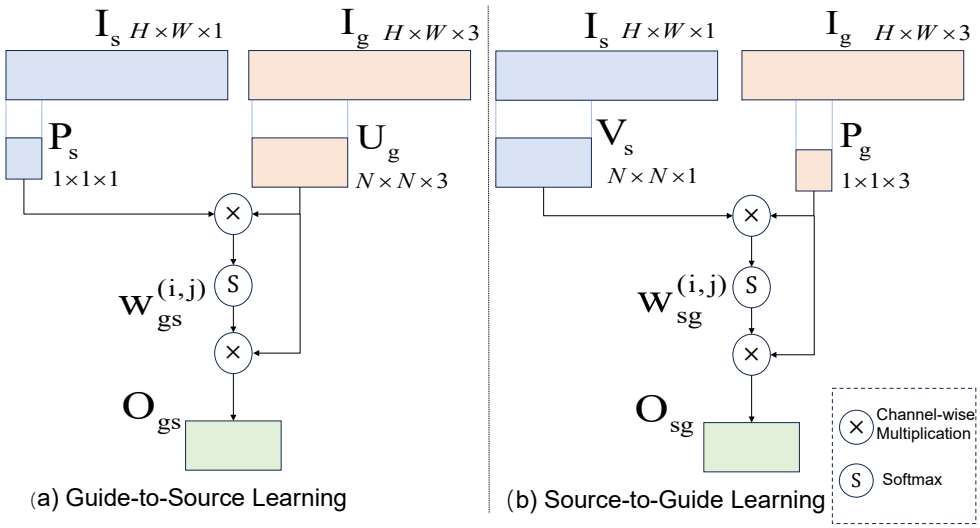


Figure 3.3: Detailed design of Pixel Mutual Learning.

Compared to conventional CNN-based architectures, which are limited by local receptive fields, SSM enhances the model's ability to capture holistic structural cues and maintain global consistency across the depth map. By jointly optimizing the source and guide branches through mutual learning, CIMLN effectively filters out texture noise from the RGB image while reinforcing structural edges from the depth map, ultimately producing a high-quality, refined depth map suitable for downstream editing tasks.

As shown in Fig. 3.3, the guide-to-source and source-to-guide branches exchange features through a Pixel Mutual Learning module, which aligns pixel-level information between RGB and depth. In Fig. 3.3 (a), we first extract a depth $\mathbf{P}_s$ from source image and an RGB $\mathbf{U}_g$ from guide image, and then perform matrix multiplication for them. A weight filter is employed to evaluate the correlation value between a pixel in source p_s and each pixel in guide $\mathbf{U}_{g(i,j)}$:

$$\begin{aligned} \mathbf{w}_{gs}^{(i,j)} &= \text{Softmax}\left(\left(\mathbf{U}_{g_{(i,j)}}\right)^{\top} p_s\right), \\ \mathbf{O}_{gs} &= \mathbf{U}_{g_{(i,j)}}, \mathbf{w}_{gs}^{(i,j)}. \end{aligned} \tag{3.10}$$

The source-to-guide pixel mutual learning follows the same process. Lastly, the convolution layers and residual blocks are introduced for the final prediction to obtain the output image.

Output Image is reconstructed as $\mathbf{I}_{out}$ for self-supervised training. To ensure high-quality and structurally consistent outputs, we design the overall training loss as a combination of pixel-level fidelity and boundary-aware enhancement. The total loss is defined as:

$$\mathcal{L}_{total} = \lambda \mathcal{L}_{L1} + \gamma \mathcal{L}_{ba}, \tag{3.11}$$

where λ and γ are weighting coefficients that balance the contributions of each term.

Self-Supervised Cycle-Consistent Learning.

In our setting, acquiring ground-truth high-quality depth maps is often infeasible, especially for real-world multi-view datasets. To enable self-supervised training under such conditions, it is essential to enforce structural consistency between the model's output and the original input without relying on paired supervision.

To this end, we adopt the principle of cycle-consistent learning, a widely used strategy in deep learning that encourages reversible transformations between two domains. Specifically, given a data pair (A, B), the model learns a forward mapping from A to B and a backward mapping from B to A, ensuring that translating $A \to B \to A$ approximates the original A. This principle has been effectively applied in dense semantic alignment [70] and image super-resolution [16] for preserving structure in unpaired settings.

In our framework, we apply cycle consistency to the depth enhancement module by introducing a loss that enforces alignment between the rendered depth and the rendered RGB view. The enhanced result is constrained to match the input, thus preserving spatial fidelity across resolutions.

This design allows our model to be trained in a fully self-supervised fashion, making it compatible with unannotated multi-view datasets and promoting reliable geometric consistency across views.

Importantly, this combined loss enables a fully self-supervised learning paradigm, eliminating the need for explicit ground-truth labels. Instead, the framework capitalizes on

internal consistency through recursive supervision, leveraging previously generated outputs ($\mathbf{I}_{m-1}^{out}$) as pseudo-targets to iteratively refine its predictions. This approach ensures that the network learns to enhance details and improve fidelity without external annotations.

Additionally, the framework incorporates structural cues extracted from the input $\mathbf{I}^h$, reinforcing spatial coherence and preserving fine-grained textures throughout the learning process. By integrating the reconstruction loss $\mathcal{L}_{L1}$, which enforces overall intensity fidelity, and the boundary-aware loss $\mathcal{L}_{ba}$, which prioritizes edge sharpness and structural consistency, the model achieves a balance between global accuracy and local precision.

Pixel-wise Reconstruction Loss ($\mathcal{L}_{L1}$).

The first component of the loss is the pixel-level L1 loss, which penalizes absolute differences between the output image $\mathbf{I}_{out}$ and the ground-truth image $\mathbf{I}$:

$$\mathcal{L}_{L1} = \|\mathbf{I}_{out} - \mathbf{I}\|_1. \tag{3.12}$$

This term encourages global accuracy in pixel intensities, promoting overall structural correctness and color consistency. However, L1 loss alone often leads to blurry results, especially around object boundaries, since it equally penalizes all pixel differences without emphasizing local edge information.

Boundary-Aware Loss ($\mathcal{L}_{ba}$).

To specifically enhance edge sharpness and preserve high-frequency details such as object boundaries, we introduce a boundary-aware loss. This loss operates on the gradient domain, enforcing consistency between the gradients of the high-quality reference image $\mathbf{I}^h$ and the intermediate output from the previous stage $\mathbf{I}_{m-1}^{out}$. Formally, it is defined as:

$$\mathcal{L}_{ba} = \left(\nabla_x \mathbf{I}^h - \nabla_x \mathbf{I}_{m-1}^{out}\right) \odot \left(\nabla_y \mathbf{I}^h - \nabla_y \mathbf{I}_{m-1}^{out}\right), \tag{3.13}$$

where:

- ∇_x and ∇_y denote the first-order derivatives (gradients) along the horizontal and vertical directions, respectively.
- $\mathbf{I}^h$ represents the high-quality reference or estimated sharp image, serving as the edge guidance.
- $\mathbf{I}_{m-1}^{out}$ is the model's output at the previous iteration or stage, providing a recursive supervision mechanism.
- $\odot$ indicates element-wise multiplication, which amplifies the loss where gradient differences occur simultaneously along both axes, emphasizing corner and boundary regions.

The boundary-aware loss enhances the restoration of fine structures by explicitly minimizing the gradient discrepancies between the reference and predicted outputs. Since image gradients are sensitive to edges, this loss prioritizes accurate recovery of boundary transitions and prevents oversmoothing effects commonly seen in pixel-based losses. By operating in the gradient domain, $\mathcal{L}_{ba}$ ensures that the restored images retain sharp, well-defined contours and detailed textures, which are critical in high-fidelity 3D scene reconstructions.

3.2.3 Wavelet Consensus Attention-Based Latent Code Alignment

Text-driven image editing with diffusion models has achieved remarkable success in the 2D domain. These models generate high-quality images by iteratively denoising a latent variable initialized from Gaussian noise. In the context of 3D scene editing, however, direct application of diffusion models to multi-view renderings leads to view-dependent inconsistencies, such as color shifts, texture misalignments, or geometric distortions across different viewpoints. This is because each view is edited independently, with no mechanism to enforce coherence in the latent space.

To resolve this, we introduce the Wavelet Consensus Attention (WCA) module, which aligns the latent representations across different views at every denoising step of the diffusion process. Before detailing WCA, we first explain the underlying principles of diffusion-based generative modeling and its inversion.

Diffusion models generate images by simulating a Markovian forward process that gradually adds Gaussian noise to the original data over T steps:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}, \mathbf{x}_{t-1}, \beta_t\mathbf{I}\right), \tag{3.14}$$

where β_t is a small noise variance added at timestep t.

By composing these steps, we can express the noisy latent at step t as a closed-form function of the clean data $\mathbf{x}_0$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}, \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{3.15}$$

where $\bar{\alpha}_t = \prod i = 1^t(1-\beta_i)$.

The goal of the reverse process is to learn a denoising model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ that recovers clean data step-by-step. In practice, we train a neural network to predict the noise $\boldsymbol{\epsilon}_\theta$ at each step by minimizing the following loss:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)|^2\right]. \tag{3.16}$$

During inference, the denoising process is iteratively applied:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}, \tag{3.17}$$

where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ and σ_t controls the noise scale at step t.

In our 3D editing system, the diffusion model operates in the latent space of a pretrained variational autoencoder, and is conditioned on the input image $\mathcal{R}$, text prompt p, and depth map $\mathcal{D}$ using ControlNet.

When rendering a 3D scene from multiple viewpoints and editing each view with a diffusion model, each view has its own latent z_t^i at timestep t. If the editing is done

independently, these latents may diverge, resulting in: inconsistent object appearance across views; broken geometry when converting back to 3D; failure to preserve semantic or photometric coherence. This motivates the need for cross-view alignment of latents during the diffusion process.

To address this, we propose Wavelet Consensus Attention (WCA)—a cross-view attention mechanism that aligns latent features across views and across frequency bands at each diffusion step.

Given a latent z_t^i from view i, we first apply a Wavelet Transform (WT):

$$\mathrm{WT}(z_t^i) = z_{LL}^i, z_{LH}^i, z_{HL}^i, z_{HH}^i, \tag{3.18}$$

where LL captures low-frequency global structures, and LH, HL, HH capture high-frequency details.

Next, for each frequency band $C \in \{LL, LH, HL, HH\}$, we compute **cross-view attention** as follows:

$$\mathbf{A}_C = \mathrm{Softmax}\left(\frac{\mathbf{Q}_C \mathbf{K}_C^\top}{\sqrt{d}}\right)\mathbf{V}_C, \tag{3.19}$$

where $\mathbf{Q}_C, \mathbf{K}_C, \mathbf{V}_C$ are linear projections of the wavelet-transformed features from different views. This attention mechanism captures both intra-view and inter-view dependencies across views. To balance **view-specific detail** with **cross-view consistency**, we blend self-attention with consensus attention using a weighted fusion:

$$\mathrm{Attn}_i = \lambda \cdot \mathbf{A}_{i,i} + (1-\lambda) \cdot \frac{1}{N_r}\sum_{j=1}^{N_r} \mathbf{A}_{i,j}, \tag{3.20}$$

where $\mathbf{A}_{i,i}$ represents self-attention within view i, and the second term averages attention from all other reference views. The weighting factor $\lambda \in [0, 1]$ controls the trade-off between preserving individual view characteristics and enforcing cross-view consistency.

Finally, the aligned wavelet components are merged back into the latent domain via the Inverse Wavelet Transform (IWT):

$$\hat{z}_t^i = \mathrm{IWT}(z_{LL}^{\mathrm{attn}}, z_{LH}^{\mathrm{attn}}, z_{HL}^{\mathrm{attn}}, z_{HH}^{\mathrm{attn}}). \tag{3.21}$$

After applying WCA to refine the latent code at step t, we proceed with the standard DDIM update:

$$z_{t-1}^{i} = \sqrt{\alpha_{t-1}}\hat{z}_{t}^{i} - \frac{\sqrt{1-\alpha_t}\cdot\hat{e}^{t}}{\sqrt{\alpha_t}} + \sqrt{1-\alpha_{t-1}}\cdot\hat{e}^{t}. \tag{3.22}$$

Here, the refined noise $\hat{e}^t$ is also computed from the aligned latent using the conditioned diffusion model.

WCA provides the following key advantages: Frequency-aware alignment: Separates and harmonizes global style (low-frequency) and fine detail (high-frequency). Cross-view coherence: Prevents inconsistency and drift across different viewpoints. Efficient and scalable: Operates on wavelet subbands, reducing computational cost compared to full-resolution attention.

In summary, WCA enhances the denoising process of diffusion models by enforcing latent consistency across views in both spatial and frequency domains. By embedding WCA into the latent update loop, we ensure that the 3D scene edits remain semantically meaningful, geometrically consistent, and photorealistically coherent across all views.

3.3 Joint 3DGS and Camera Poses Optimization for Motion Blur Images

The overall pipeline of our method is presented in Fig. 3.4. Given that the exposure duration τ is typically short in practical scenarios, we assume the camera undergoes smooth and continuous motion, which can be well-approximated using a linear motion model in the Lie group $\mathbf{SE(3)}$. This assumption allows us to interpolate camera poses across the exposure period without introducing complex motion dynamics, while still capturing the essential geometric transformations that cause motion blur.

A camera pose $\mathbf{T} \in \mathbf{SE(3)}$ is commonly expressed as a 4×4 homogeneous transforma-

tion matrix:

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}, \quad \mathbf{R} \in \mathbf{SO(3)}, \quad \mathbf{t} \in \mathbb{R}^3, \tag{3.23}$$

where $\mathbf{R}$ denotes a 3×3 rotation matrix that describes the orientation of the camera, and $\mathbf{t}$ is a 3-dimensional translation vector that specifies the position of the camera in 3D space. In our framework, we explicitly model two key poses of the camera during exposure: the initial pose $\mathbf{T}_{\text{start}} \in \mathbf{SE(3)}$, which corresponds to the camera's state at the beginning of the exposure time, and the final pose $\mathbf{T}_{\text{end}} \in \mathbf{SE(3)}$, which defines the camera's state at the end of the exposure. These two endpoints fully describe the camera's motion trajectory under the linear motion assumption, and serve as anchors for interpolating intermediate virtual camera poses.

To simulate the continuous motion of the camera throughout the exposure, we interpolate between $\mathbf{T}_{\text{start}}$ and $\mathbf{T}_{\text{end}}$ within $\mathbf{SE(3)}$ using linear interpolation in the Lie algebra space. This is accomplished by computing the relative motion between the two poses via logarithmic mapping, scaling it proportionally to the elapsed time, and mapping it back through the exponential map. The virtual camera pose at any time $t \in [0, \tau]$ during the exposure is therefore defined as:

$$\mathbf{T_t} = \mathbf{T}_{\text{start}} \cdot \exp\left(\frac{t}{\tau} \cdot \log\left({\mathbf{T}_{\text{start}}}^{-1} \cdot \mathbf{T}_{\text{end}}\right)\right), \tag{3.24}$$

where τ represents the total exposure time, and $\log(\cdot)$ and $\exp(\cdot)$ denote the logarithmic and exponential maps on $\mathbf{SE(3)}$, respectively. This formulation enables smooth interpolation between the start and end poses, ensuring that the camera motion trajectory is continuous and geometrically valid across the entire exposure interval.

For practical implementation, we discretize the continuous exposure time into n uniformly sampled steps to generate n virtual sharp images along the motion path. The time ratio $\frac{t}{\tau}$ for the i-th sample can be expressed as $\frac{i}{n-1}$, where $i = 0, 1, ..., n-1$. Correspondingly, the interpolated pose $\mathbf{T_i}$ at the i-th time step is computed following the same interpolation strategy. This discrete sampling process enables us to efficiently approximate

the continuous integration of sharp images during exposure with a finite sum of rendered frames at interpolated poses.

A key advantage of this formulation is that the interpolated poses $\mathbf{T_i}$ are fully differentiable with respect to both $\mathbf{T}_{\text{start}}$ and $\mathbf{T}_{\text{end}}$. This property is critical for gradient-based optimization, allowing the camera motion parameters to be jointly learned with other model components through backpropagation. As a result, our method not only simulates motion blur through realistic camera trajectories but also enables end-to-end optimization of camera motion, depth estimation, and image restoration within a unified framework. This seamless integration of differentiable pose interpolation and motion blur modeling provides strong support for downstream tasks requiring temporally consistent multi-view supervision and blur-aware image synthesis.

The process of fitting a 3D Gaussian Splatting (3DGS) model shares fundamental similarities with the optimization of a Gaussian Mixture Model (GMM) [26]. Both models aim to approximate a complex data distribution by learning a set of Gaussian components, where each component encodes localized spatial and appearance information. In the context of 3DGS, the scene is represented as a collection of N 3D Gaussian ellipsoids $\{G_i\}_{i=1}^{N}$, each defined by its mean μ_i, covariance matrix Σ_i, opacity σ_i, and color c_i. During the rendering stage, each 3D Gaussian G_i is projected onto the image plane, resulting in a corresponding 2D Gaussian G_i' used for rasterization, as formalized in Eq. 3.25.

$$G_i' = \mathcal{P}(G_i) = \mathcal{N}(\pi(\mu_i), J_i \Sigma_i J_i^\top), \tag{3.25}$$

However, a key challenge in the rendering process arises when the projected 2D Gaussians become extremely small—smaller than the area of a single pixel. Such under-sampling leads to aliasing artifacts, including flickering and instability during viewpoint changes. To address this issue, Jung et al. [26] propose an anti-aliasing strategy that ensures the spatial footprint of each 2D Gaussian covers at least one pixel. This is achieved by regularizing the projected covariance matrix, specifically by adding a small constant ϵ to its diagonal

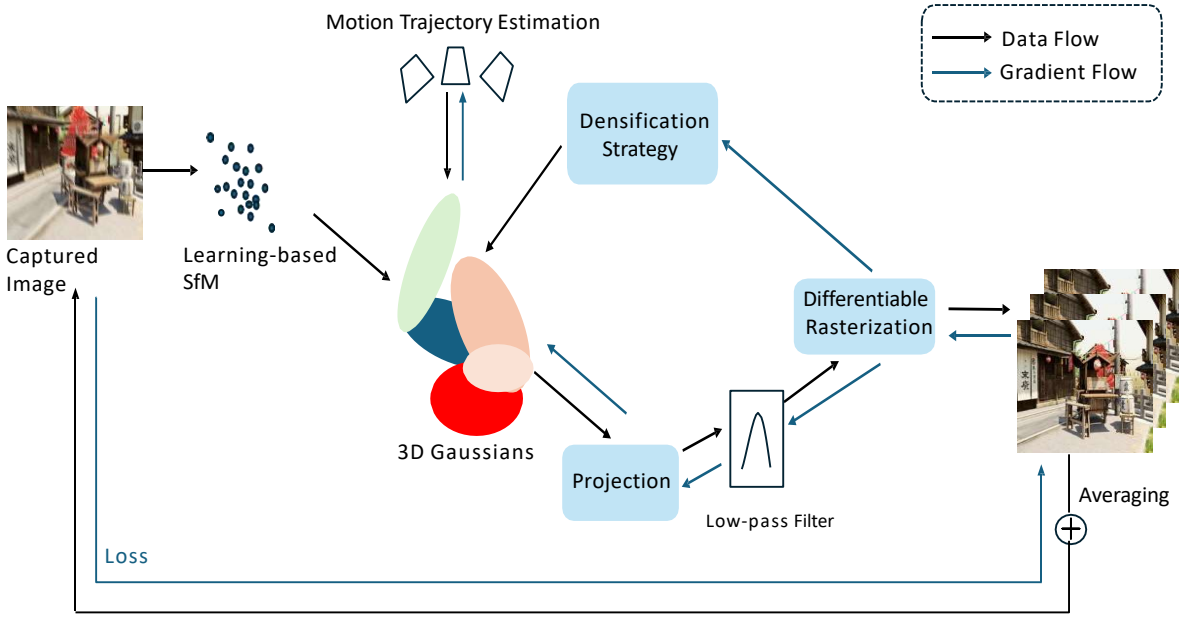


Figure 3.4: Overview of our proposed method. The process begins with the initialization protocol, where the motion blur images are input into a learning-based SfM [46] module to generate rough estimates of the point cloud and camera poses, which are the initial parameters for the 3D Gaussians. Then we perform joint optimization of the Gaussians and their corresponding camera poses. The optimization process minimizes the L1 loss and a SSIM term between outputs and real-captured images.

elements, effectively inflating the Gaussian's scale to meet a minimum threshold. This simple yet effective solution helps maintain visual stability without significantly compromising rendering precision.

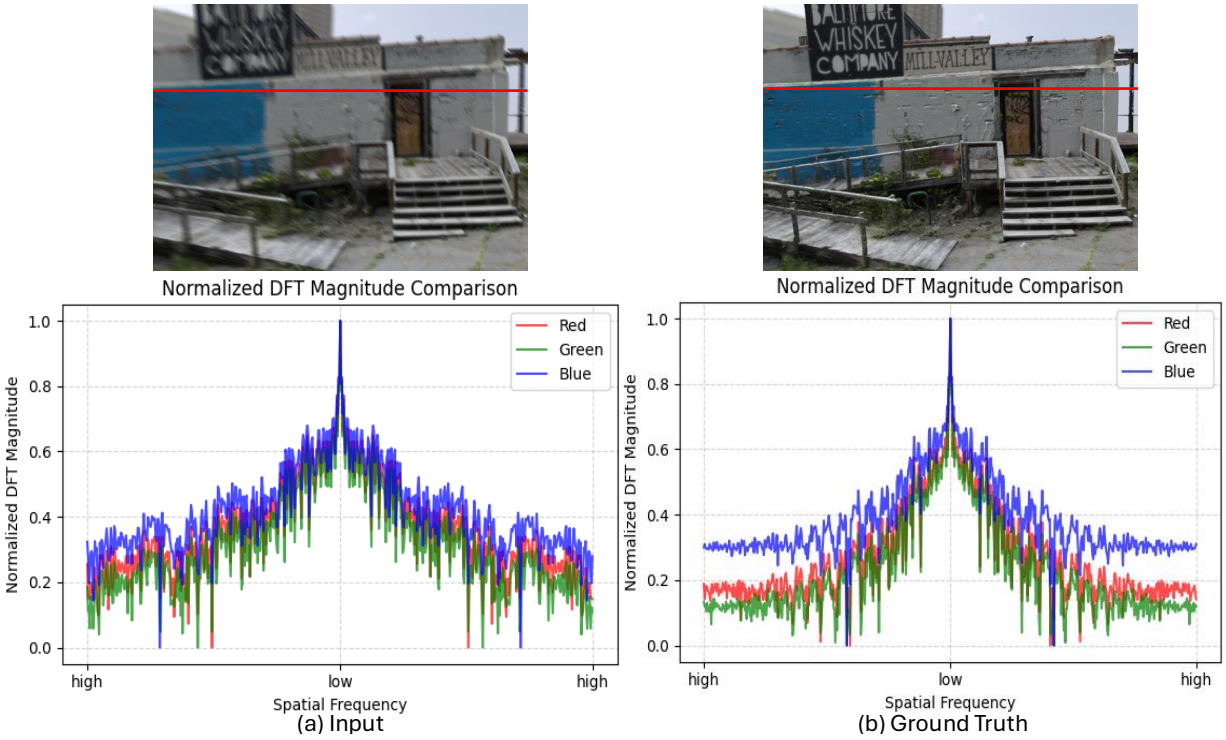


Figure 3.5: Analysis of Factory Motion Blur Images in Frequency for Each Channel. The right image shows the GT image, and the left image is the motion blur image. We randomly sample a horizontal line from the image marked in red. The graphs visualize the magnitude of the frequency components of images in R, G, B channels. We observe that the main difference exists in the high frequency, which will be overfitted during 3DGS training [26] and result in artifacts.

In addition to the pixel-scale problem, another important artifact source in 3DGS arises from the interaction between overlapping Gaussians. As illustrated in Fig. 3.5, when optimizing the parameters of Gaussians concentrated within one ellipsoid (or local region), high-frequency discrepancies can emerge in neighboring regions or overlapping ellipsoids. This is particularly problematic when the contributions of nearby Gaussians interfere destructively, leading to ringing artifacts, sharp discontinuities, or unnatural color transitions. These high-frequency artifacts are amplified when using high-resolution textures or when the camera moves rapidly, causing abrupt changes in visibility across Gaussians.

To mitigate these artifacts, we adopt a low-pass filtering strategy that smooths the 3D Gaussian components before rasterization. Rather than directly rendering the raw Gaussians, we convolve each 3D Gaussian G_i with a predefined isotropic low-pass Gaussian filter G_{F}. This operation effectively diffuses the energy of each Gaussian over a slightly larger region, suppressing high-frequency components and stabilizing the overall appearance of the rendered scene. The filtered 3D Gaussian is given by:

$$G_i'(x) = (G_i * G_{\mathrm{F}})(x) = \sqrt{\frac{|\Sigma_i|}{|\Sigma_i + \gamma I|}} \exp\left(-\frac{1}{2}(x-\mu_i)^\top (\Sigma_i + \gamma I)^{-1}(x-\mu_i)\right), \quad (3.26)$$

where γ is a hyperparameter controlling the strength of the smoothing effect, I denotes the identity matrix, and G_{F} is the low-pass Gaussian filter applied in 3D space. The convolution modifies the covariance matrix of the original Gaussian from Σ_i to $\Sigma_i + \gamma I$, effectively expanding its spread isotropically while preserving its center μ_i.

This filtering step offers several advantages. First, it guarantees that the effective support of each Gaussian remains compatible with the sampling interval of the rasterization pipeline, preventing excessive blurring while eliminating aliasing-prone high-frequency details. Second, the smooth transition between neighboring Gaussians reduces the likelihood of discontinuities and sharp edges, resulting in more photorealistic and temporally stable renderings. Third, since the convolution operation is differentiable with respect to the original Gaussian parameters, it integrates seamlessly into the optimization process, allowing the filter strength γ to be treated as a learnable or tunable parameter.

By integrating this low-pass filtering technique into the 3DGS framework, we achieve a balance between sharpness and smoothness in the rendered images. The filter effectively regularizes the spatial frequency content of the Gaussian field, providing robustness to both local noise and global inconsistencies introduced during training or real-time viewpoint transitions. In practice, selecting an appropriate value for γ is crucial; excessively large values can oversmooth fine details, whereas small values may fail to suppress artifacts. Therefore, γ is typically chosen empirically based on the resolution of the dataset, the density of Gaussians, and the desired rendering quality.

Algorithm 3.1 Efficient 3D Editing based on Gaussian Splatting with Complementary and Consensus Information

Input: 3D Gaussian scene $\mathcal{G}$, text prompt p, number of editing steps T, exposure time τ (if motion blur exists)

Output: Edited 3D Gaussian scene $\hat{\mathcal{G}}$

if motion blur exists **then**
    for each Gaussian $G_i \in \mathcal{G}$ **do**
        Apply low-pass filter: $G_i \leftarrow G_i * G_{\mathrm{F}}$
    end for
    Estimate motion trajectory $\mathbf{T}(t)$, $t \in [0, \tau]$
    Sample n sharp views along $\mathbf{T}(t)$
end if
Render multi-view RGB images $\mathcal{R}$ and depth maps $\mathcal{D}$ from $\mathcal{G}$
Refine depth maps with CIMLN:
$\mathcal{D}^* \leftarrow \mathrm{CIMLN}(\mathcal{R}, \mathcal{D})$
Initialize latent code z_T from $\mathcal{R}$ and noise
for $t = T$ to 1 **do**
    Predict noise: $e^t = \mathcal{F}(z_t; t, p, \mathcal{D}^*)$
    Apply Wavelet Consensus Attention (WCA):
    Decompose z_t via wavelet transform
    Align features across views with consensus attention
    Reconstruct aligned latent via inverse wavelet transform
    Update latent code z_{t-1} via DDIM step with aligned noise
end for
Decode z_0 via VAE to obtain edited images $\hat{\mathcal{R}}$
Optimize $\mathcal{G}$ using $\hat{\mathcal{R}}$ as supervision to obtain $\hat{\mathcal{G}}$
return $\hat{\mathcal{G}}$

Chapter 4

Experiment

4.1 Experiment Setup

For consistent evaluation, we adopted the camera path extraction strategy proposed in NeRFStudio [51], which provides accurate camera pose estimations and trajectory smoothing across datasets. This standardization ensures that rendered viewpoints, depth maps, and scene coverage remain comparable across different methods, thereby eliminating potential biases caused by inconsistent camera configurations.

4.1.1 Metrics

To provide a comprehensive assessment of our editing framework, we adopted multiple evaluation metrics capturing fidelity, perceptual quality, and semantic consistency between text prompts and visual outputs. The following metrics were employed, along with their formal definitions:

- **CLIPdir** [43]: Measures semantic alignment between edits and text prompts via directional similarity in CLIP embedding space:

$$\text{CLIPdir} = \frac{\langle \mathbf{E}_{\text{edit}} - \mathbf{E}_{\text{orig}}, \mathbf{E}_{\text{text}} \rangle}{\|\mathbf{E}_{\text{edit}} - \mathbf{E}_{\text{orig}}\| \cdot \|\mathbf{E}_{\text{text}}\|}. \tag{4.1}$$

- **PSNR**: Evaluates pixel-level fidelity between generated image I and ground truth I_{gt}:

$$\text{PSNR} = 10 \log_{10} \left(\frac{L^2}{\text{MSE}} \right), \quad \text{MSE} = \frac{1}{WH} \sum (I - I_{\text{gt}})^2. \tag{4.2}$$

- **RMSE**: Measures geometric accuracy between predicted depth D and ground truth D_{gt}:

$$\text{RMSE} = \sqrt{\frac{1}{WH} \sum (D - D_{\text{gt}})^2}. \tag{4.3}$$

- **LPIPS**: Computes perceptual similarity using deep features from a pre-trained network Φ:

$$\text{LPIPS} = \sum_{l} \frac{1}{H_l W_l} \sum \|\Phi_l(I) - \Phi_l(I_{\text{gt}})\|_2^2. \tag{4.4}$$

Through the combination of these complementary metrics, we provide a holistic evaluation of our method across geometric accuracy, pixel-level fidelity, semantic consistency, and perceptual quality. This multi-dimensional evaluation not only verifies the correctness of our 3D editing process but also demonstrates its robustness across different data domains, ensuring that our edits remain faithful to textual prompts while preserving high-quality visual and structural integrity in the rendered outputs.

4.1.2 Efficient 3D Editing based on Gaussian Splatting with Complementary and Consensus Information

Datasets Following the methodology of IN2N [21], we conducted extensive experiments on several widely used datasets to validate the effectiveness and generalizability of our approach. The selected datasets include "Garden" "Dinosaur" in Mip-NeRF [3], which offers high-quality anti-aliased neural radiance fields suitable for evaluating geometric and photometric fidelity; "Bear" and "Horse" inBlendedMVS [63], a large-scale multi-view stereo dataset designed for dense 3D reconstruction in real-world environments; and "Face" "Fangzhou" in NeRF-Art [55], a collection of stylized and artistic NeRF-rendered

scenes, which provides a challenging testbed for evaluating the robustness of editing in non-photorealistic settings. Together, these datasets cover a diverse range of scenes, motion patterns, lighting conditions, and rendering complexities, allowing us to systematically assess the performance of our method across both synthetic and real-world scenarios.

Figure 4.1: Examples of the unedited multi-view images (Bear, Dinosaur, Face, Fangzhou, Garden, Horse, and Stump) of the experimental datasets following IN2N [21].

Methods To comprehensively evaluate the performance of our proposed framework, we compare it with several state-of-the-art reconstruction-based 3D editing methods, including

both NeRF-based and Gaussian-based approaches. The compared methods are summarized as follows:

- IN2N [21]: A NeRF-based editing framework that directly integrates text guidance into the 3D scene optimization process. IN2N jointly optimizes the radiance field to match the text prompt while preserving multi-view consistency.
- GS2GS [53]: A Gaussian Splatting-based editing pipeline that transfers edits between different Gaussian scenes by leveraging shared structural information to maintain consistency across edits.
- ViCA-NeRF [15]: A viewpoint-consistent NeRF editing framework that enhances cross-view alignment using cross-attention mechanisms to maintain semantic coherence across views.
- GaussCtrl [61]: A Gaussian Splatting editing method that applies ControlNet-based guidance to manipulate 3D Gaussian scenes with user-defined prompts, achieving high-quality edits with relatively fast inference.

These methods represent the current leading approaches for 3D scene editing under various paradigms, allowing us to thoroughly benchmark our method in terms of accuracy, perceptual quality, semantic alignment, and computational efficiency.

All experiments were executed on an NVIDIA GeForce RTX 4090 GPU equipped with 24GB of VRAM, utilizing the PyTorch framework for implementation. This hardware configuration allowed us to train and evaluate large-scale 3D Gaussian Splatting models with high efficiency, benefiting from CUDA-accelerated tensor operations, automatic differentiation, and seamless integration with neural rendering pipelines. The codes of our 3D editing system can be found at https://github.com/sinjinchang/CCGSE.

4.1.3 3D Gaussian Splatting from Motion Blur Images via Low-pass Filter

Datasets To further evaluate the effectiveness of our proposed method in handling motion blur within 3D scenes, we conducted comprehensive experiments on the ExBluRF dataset [32]. The ExBluRF dataset is synthetically generated using Blender [1] and is specifically designed for benchmarking deblurring and 3D scene reconstruction under significant camera motion blur. Each scene within the dataset contains a set of 29 motion-blurred training images, which simulate realistic camera trajectories during exposure, and a set of 5 sharp test images, which are captured without motion blur and serve as ground-truth references for evaluation. This dataset provides a controlled yet challenging environment for assessing both 2D deblurring quality and 3D consistency across multiple views.

Methods For fair and thorough comparison, we benchmarked our method against both traditional single-image deblurring approaches and recent neural radiance field (NeRF)-based deblurring techniques. Specifically, the compared methods include:

- MPRNet [64]: A multi-stage progressive image restoration network that operates on single blurry images using a hierarchical encoder-decoder design.
- PVD [48]: A recurrent framework for progressive video deblurring that processes sequences but can be adapted to single-frame scenarios.
- SRN-Deblur [52]: A scale-recurrent network that restores sharp images through a coarse-to-fine deblurring pipeline, widely adopted as a baseline in single-image deblurring tasks.
- DeblurNeRF [35]: A NeRF-based method that jointly models scene radiance and motion blur to reconstruct sharp novel views from blurred multi-view observations.
- BADNeRF [58]: A state-of-the-art deblurring radiance field model that explicitly disentangles motion blur and scene appearance for improved sharp view synthesis.

For all methods, we follow the same experimental protocol to ensure a fair comparison. Specifically, the motion-blurred training images from each scene in ExBluRF are used as inputs to train or fine-tune the respective models. For single-image deblurring methods (MPRNet, PVD, SRN-Deblur), each blurred training image is independently processed to generate a sharp prediction, without exploiting multi-view consistency. For NeRF-based methods (DeblurNeRF, BADNeRF), the blurred training images are used jointly to optimize the corresponding radiance field or Gaussian field representations, allowing the models to leverage cross-view information and camera pose supervision during reconstruction.

At test time, we render sharp novel views from the optimized 3D representations and compare them to the ground-truth sharp test images provided in ExBluRF. Evaluation is performed on the sharp test views per scene to ensure consistency across methods. To comprehensively assess the performance of each method, we report the same suite of metrics described in the previous section, including PSNR, RMSE, LPIPS, and CLIPdir, thereby capturing fidelity, perceptual quality, reconstruction accuracy, and semantic alignment with textual prompts where applicable. For reconstruction-based methods, we evaluate the consistency between edited 2D views and the corresponding renderings from the modified 3D scene by computing image-to-image similarity. This is quantified using the Peak Signal-to-Noise Ratio (PSNR), which measures the fidelity of the rendered views relative to the edited inputs.

All experiments are conducted under identical hardware and software settings, utilizing an NVIDIA RTX 4090 GPU and the PyTorch framework. To ensure reproducibility, hyperparameters for each baseline are set according to their respective official implementations or recommended configurations, and the same random seeds are applied across runs. This rigorous experimental design guarantees a fair and transparent comparison, allowing us to isolate the contributions of our complementary and consensus-based Gaussian Splatting framework in mitigating motion blur and enhancing 3D reconstruction quality. The codes of our 3D editing system can be found at https://github.com/sinjinchang/LPFGS.

Figure 4.2: Visual comparison of text-driven 3D editing across different scenes using IN2N [21], GS2GS [53], and our method.

4.2 Experimental Results of Our 3D Editing Sytem

4.2.1 Qualitative Results

We conducted extensive qualitative evaluations to compare the visual editing effects of our method against state-of-the-art techniques, including IN2N [21], GS2GS [53], Gauss-Ctrl [61], and ViCA-NeRF [15]. As shown in Fig. 4.3, the proposed method consistently produces high-quality images of the bear with superior multi-view consistency. Other methods, such as GS2GS and VICA-NeRF, struggle to produce multi-view consistent results after editing.

For example, when applying the text prompt "Give him a mustache" to the Fangzhou scene, as shown in Fig. 4.2 and Fig. 4.4, IN2N [21] and ViCA-NeRF [15] generated

Figure 4.3: Visual comparison of prompt 'Turn it into a polar bear'. Our method maintains multi-view consistency for rendered images.

noisy and inconsistent outputs. GaussCtrl yielded better results but introduced significant style changes, potentially conflicting with user expectations. The supplementary materials contain an extensive collection of alternative viewpoints and angles of these 3D scenes. Our method effectively preserved the intended transformation while maintaining background consistency.

Furthermore, we present additional multi-view images rendered from edited 3DGS using our method, as shown in Fig. 4.6 and Fig. 4.7. These images exhibit exceptional multi-view consistency across different viewpoints, further validating the effectiveness of our proposed modules.

Figure 4.4: Visual comparison of text-driven 3D editing across different scenes using Vica-NeRF [15], GaussCtrl [61], and our method.

4.2.2 Quantitative Results

The quantitative performance of different methods is summarized in Table 4.1 and Table 4.2. Table 4.1 reports the average performance across six diverse scenes, capturing a broad range of geometry, textures, and motion patterns. Our method consistently outperformed competing approaches in terms of PSNR, RMSE, and LPIPS, reflecting its superior capability in both pixel-wise reconstruction accuracy and perceptual quality preservation. In particular, our method achieved a notable improvement of 1.29 dB in PSNR over the strong baseline GaussCtrl, indicating a significant reduction in reconstruction error. This performance gain is attributed to our use of complementary information from RGB and depth modalities, where depth refinement enhances geometric awareness while RGB tex-

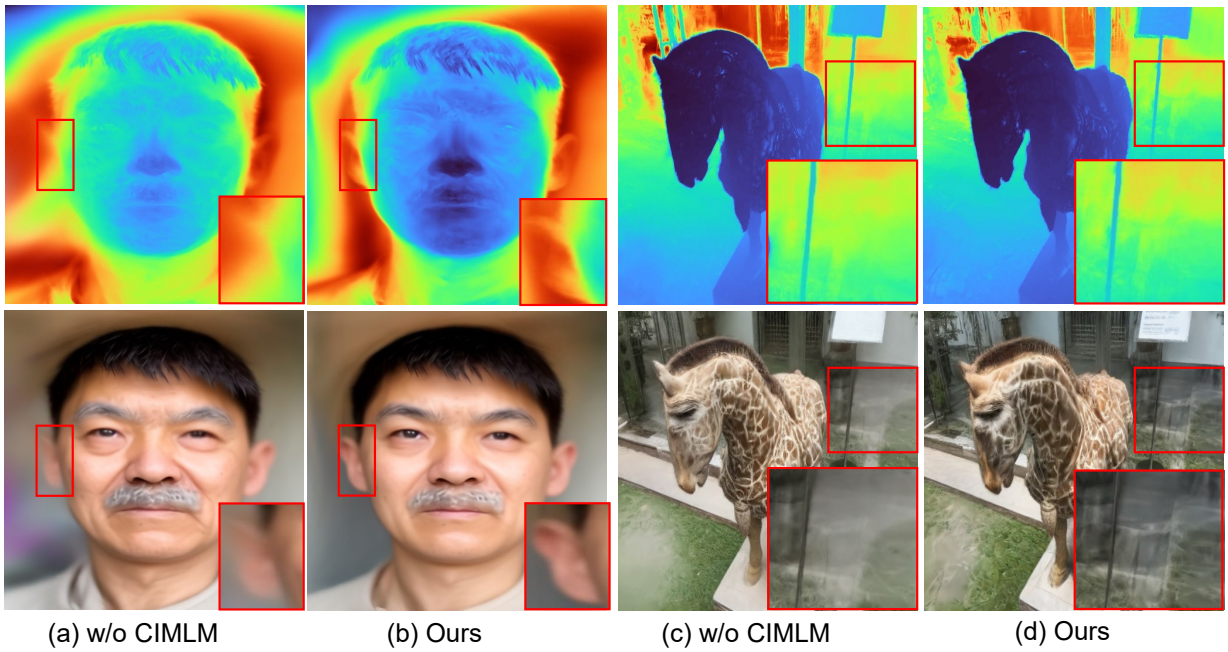


Figure 4.5: Ablation study on the effect of the proposed CIMLN module.

tures preserve photorealism during the diffusion process. By tightly integrating these modalities into the editing pipeline, our method produces sharper, more consistent outputs and avoids common degradation patterns such as over-smoothing, texture misalignment, and geometric distortions that are frequently observed in competing techniques.

Table 4.2 presents the CLIPdir scores, which measure the semantic alignment between the provided text prompts and the resulting edited images. Our method consistently achieved the highest CLIPdir scores across most datasets, demonstrating its strong capacity to translate textual instructions into accurate and meaningful visual modifications. This result highlights the effectiveness of our Wavelet Consensus Attention mechanism, which reinforces cross-view consistency while respecting prompt-driven semantic guidance, ensuring that edits are not only visually plausible but also semantically faithful. In contrast, other methods often failed to localize or interpret the prompt correctly across all views, leading to incoherent or partial transformations.

In addition to accuracy and semantic alignment, we also assessed the overall editing

Figure 4.6: Visual comparison of our 3D editing method's multi-view consistency in *Dinosaur* scenes. Multi-view consistency is well preserved.

Figure 4.7: Visual comparison of our 3D editing method's multi-view consistency in *Stump* scenes. Color information is consistent across views.

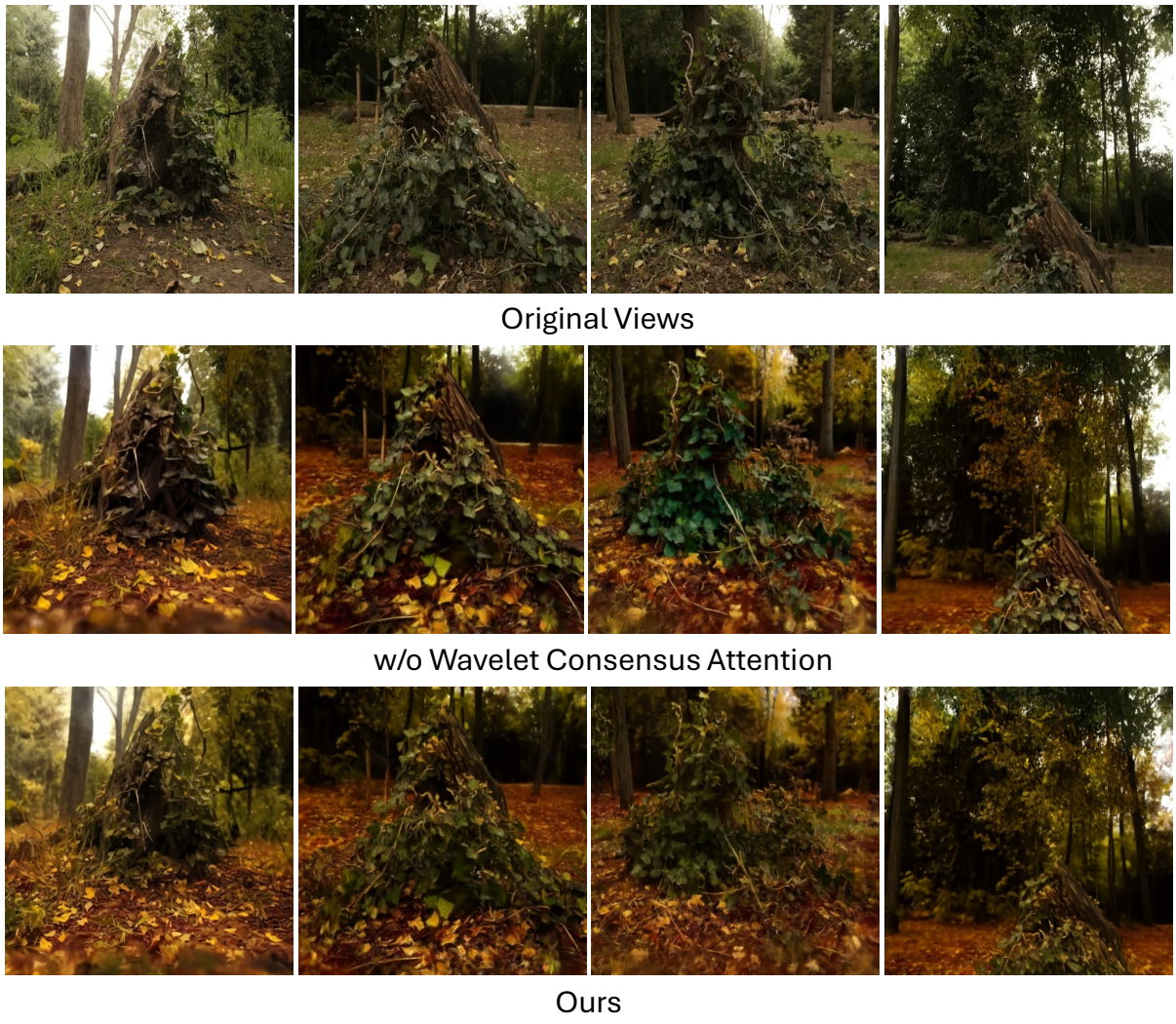


Figure 4.8: Ablation study on the effect of the proposed WCA module.

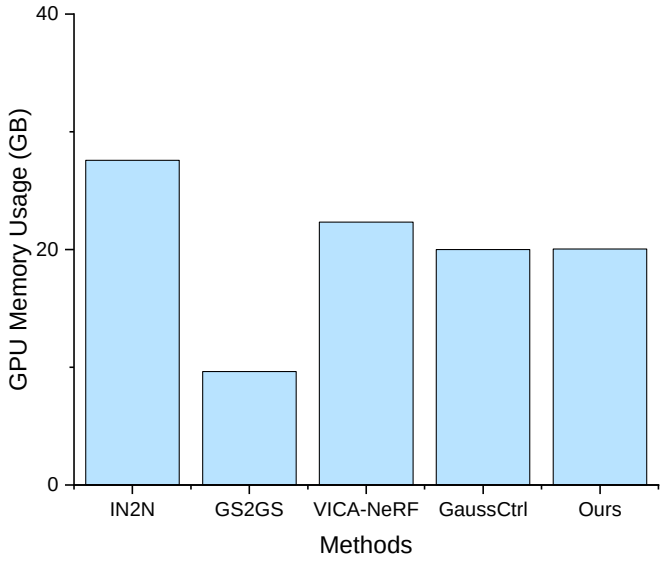


Figure 4.9: Average GPU memory usage across different 2D-3D lifting 3D editing methods.

efficiency to evaluate the practical usability of each method, particularly for real-time or interactive applications. As shown in Table 4.1, NeRF-based methods such as IN2N and ViCA-NeRF exhibited substantially higher computational costs due to their reliance on dense volumetric representations and iterative optimization procedures. These methods typically require multiple cycles of scene refinement, per-view rendering, and prompt-guided updates, which cumulatively lead to long processing times, often making them impractical for time-sensitive tasks. Our method provided a balance between speed and performance due to the efficiency of the 3D Gaussian Splatting representation. One-time dataset editing also contributed to faster convergence compared to iterative update methods like IN2N, GS2GS, and VICA-NeRF.

In contrast, our approach demonstrates a favorable balance between editing speed and output quality. Benefiting from the inherent efficiency of the 3D Gaussian Splatting representation, which directly models scene content as a set of point-based Gaussian primitives, our method enables fast rendering and rapid edit propagation without the need for ex-

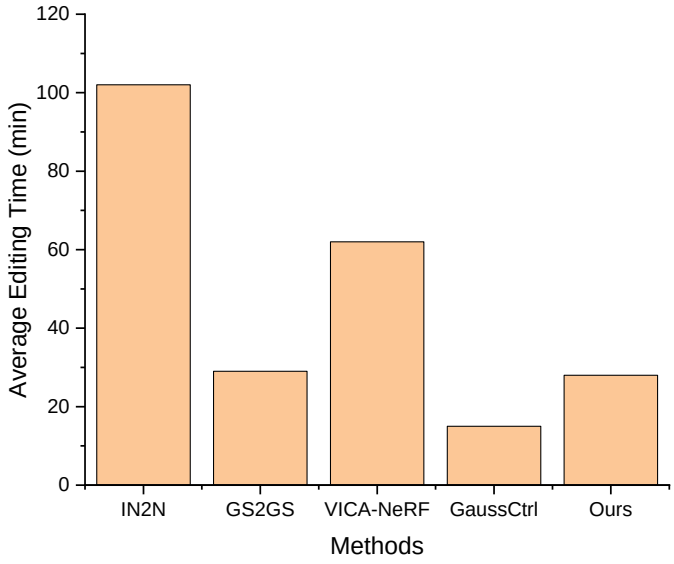


Figure 4.10: Average editing across different 2D-3D lifting 3D editing methods.

haustive global scene recomputation. Moreover, by employing a one-time dataset editing strategy, our method avoids the iterative optimization loops seen in methods like IN2N, GS2GS, and ViCA-NeRF, leading to significantly faster convergence and reduced resource consumption. As illustrated in Fig. 4.9, our method demonstrates superior GPU efficiency, consuming significantly less memory than NeRF-based approaches such as IN2N and Vica-NeRF. This reduced memory footprint makes our method a more practical and efficient solution for 3D editing tasks. This computational advantage is particularly valuable in applications requiring frequent or interactive edits, such as AR/VR environments, content creation workflows, and real-time scene adaptation.

Overall, these quantitative results confirm that our method not only excels in delivering high-fidelity and semantically accurate edits but also achieves this with greater computational efficiency, making it a compelling solution for practical, real-world 3D scene editing scenarios.

Table 4.1: Quantitative comparison of reconstruction-based methods. The best results are in bold.

Methods	Venue	PSNR ↑	RMSE ↓	LPIPS ↓	Time(min) ↓
IN2N [21]	CVPR 2023	42.35	0.492	0.319	∼82
GS2GS [53]	arXiv 2024	40.68	0.559	0.402	∼29
ViCA-NeRF [15]	NeurIPS 2023	48.21	0.215	0.182	∼61
GaussCtrl [61]	ECCV 2024	50.53	0.121	0.205	**∼15**
Ours	-	**51.82**	**0.118**	**0.145**	∼28

4.2.3 Ablation Study

To better understand the contribution of each key component in our framework, we conducted comprehensive ablation studies focusing on two critical modules: the **Complementary Information Mutual Learning Network (CIMLN)** and the **Wavelet Consensus Attention (WCA)**. These modules are designed to address common challenges in 3D Gaussian Splatting-based editing, such as noisy depth estimation and inconsistent appearance across multi-view renderings. By selectively disabling these modules and comparing the results, we are able to isolate their respective impacts on the final editing quality.

Effect of CIMLN. Depth quality plays a fundamental role in ensuring the spatial consistency and structural accuracy of text-driven 3D edits. To evaluate the effectiveness of our proposed CIMLN, we conducted ablation experiments on two representative scenes: the *Face* scene with the prompt *"Make him older"* and the *Stone Horse* scene with the prompt *"Turn it into a giraffe"*. As shown in Fig. 4.5, the inclusion of CIMLN leads to visibly improved depth maps, characterized by sharper object boundaries and significant noise reduction. Without CIMLN, the depth maps exhibit irregularities and blurred transitions around edges, which propagate into the diffusion editing process, resulting in distorted geometries and inconsistent textures in the generated RGB images. This often manifests

Table 4.2: CLIP_{dir} Comparison of Text-to-Image Consistency. The best results are in bold.

Scene	CLIP Text-Image Directional Similarity ↑				
	IN2N	GS2GS	GaussCtrl	ViCA-NeRF	Ours
Bear Statue	0.1019	0.1165	0.1388	0.1104	**0.1428**
Dinosaur	0.1466	0.1490	0.1584	0.0723	**0.1654**
Garden	**0.3027**	0.1663	0.2891	0.2903	0.2911
Stone Horse	0.1654	0.1947	0.2268	0.1926	**0.2317**
Fangzhou	0.1598	0.2032	0.1887	0.1809	**0.2153**
Face	0.1332	0.1357	0.1503	0.1119	**0.1648**

as misplaced aging features in the *Face* scene or incomplete shape transformations in the *Stone Horse* scene.

By contrast, when CIMLN is applied, the network successfully leverages complementary cues from the RGB and depth inputs, filtering out irrelevant texture noise and enhancing geometric features critical to the edit. The improved depth guidance stabilizes the conditioning of ControlNet during diffusion-based generation, enabling precise localization of semantic modifications while preserving background consistency. These results confirm that high-quality, noise-free depth maps are essential for complex structural edits and that CIMLN serves as a robust depth enhancement module, leading to superior editing accuracy and visual fidelity.

Effect of WCA. Multi-view consistency is crucial for 3D scene editing, as incoherent edits across different perspectives can break the realism and structural integrity of the reconstructed scene. To assess the impact of our proposed Wavelet Consensus Attention (WCA) module, we performed ablations on the *Stump* scene with the prompt *"Make it in autumn"*. As depicted in Fig. 4.8, disabling WCA leads to noticeable inconsistencies in leaf coloration across views. Without cross-view alignment, the diffusion model applies the au-

tumnal color transformation unevenly, causing color shifts, patchy foliage, and inconsistent lighting effects when navigating around the scene. This visual instability severely degrades the multi-view experience, making the edit appear unnatural and scene coherence difficult to maintain.

In contrast, when WCA is incorporated, the model effectively aligns the latent representations of each view in both spatial and frequency domains. By applying wavelet-based decomposition, WCA captures low-frequency global appearance attributes, such as overall color tones and lighting conditions, while preserving high-frequency local details like leaf textures and branch edges. This alignment ensures that the autumn transformation is applied uniformly across all viewpoints, yielding a cohesive, seasonally consistent forest scene with warm, harmonious color grading. The ablation demonstrates that WCA is instrumental in enforcing temporal and spatial coherence during multi-view diffusion editing, eliminating view-dependent artifacts and supporting robust 3D editing under free-view navigation.

Through these ablation studies, we validate the necessity of both CIMLN and WCA in achieving high-quality 3D edits. CIMLN enhances geometric reliability through depth refinement, ensuring accurate structure-aware modifications, while WCA enforces cross-view appearance consistency, preserving global visual coherence across all perspectives. Together, these modules address two of the most pressing challenges in 3D scene editing—spatial accuracy and multi-view stability—establishing our method as a comprehensive solution for realistic, prompt-driven 3D editing tasks.

4.3 Experimental Results of Our Motion Deblur Module

To comprehensively assess the effectiveness of our method in handling motion blur in 3D scene reconstruction and editing, we conducted experiments on the ExBluRF dataset [32],

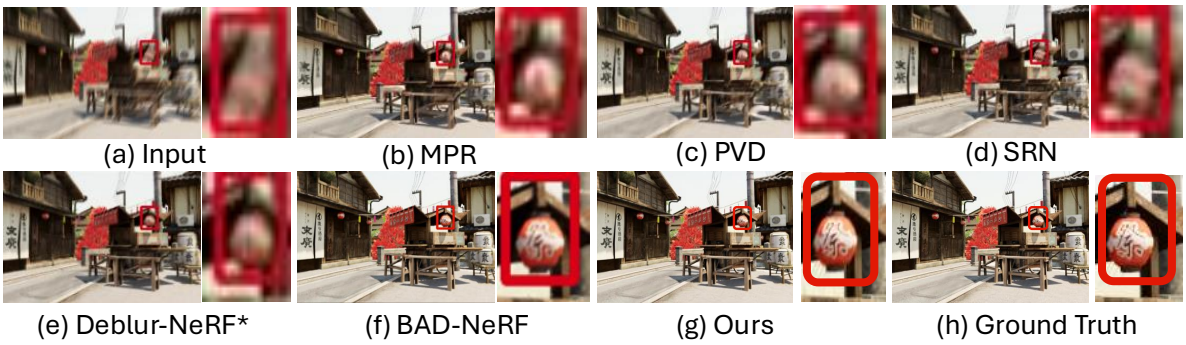


Figure 4.11: Qualitative results of different methods with Wine scene.

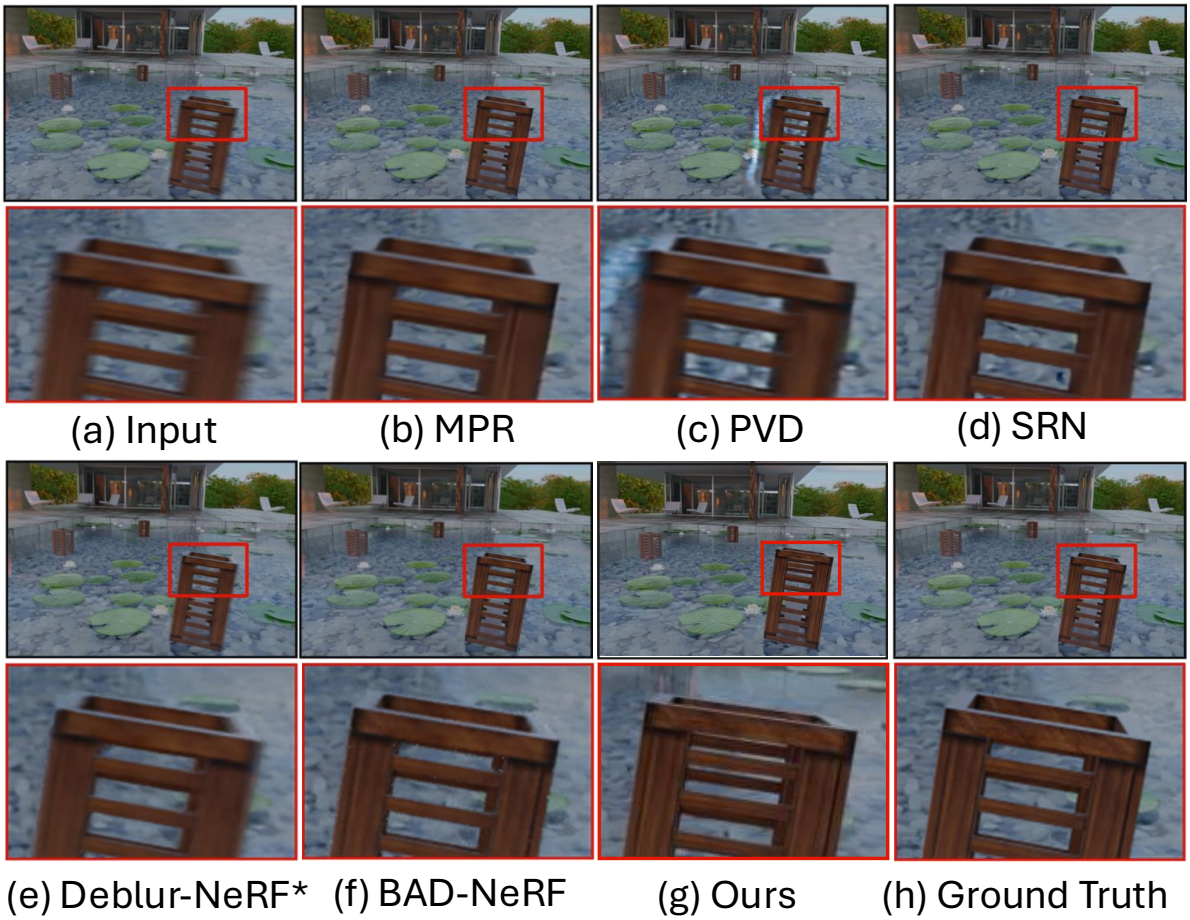


Figure 4.12: Qualitative results of different methods with Pool scene.

	Cozy2room			Factory			Pool		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Park	23.82	0.7221	0.2020	21.02	0.5090	0.4193	27.98	0.7258	0.2305
MPR	29.90	0.8862	0.0915	25.07	0.6994	0.2409	33.28	0.8938	0.1290
PVD	28.06	0.8443	0.1315	24.57	0.6877	0.3150	30.38	0.8393	0.1977
SRNDeblur	29.47	0.8759	0.0950	26.54	0.7604	0.2404	32.94	0.8847	0.1045
DeblurNeRF	25.96	0.7979	0.1024	23.21	0.6487	0.2618	31.21	0.8518	0.1382
DeblurNeRF*	30.26	0.8933	0.0791	26.40	0.7991	0.2191	32.30	0.8755	0.1345
BAD-NeRF	32.15	0.9170	**0.0547**	32.08	0.9105	0.1218	33.36	0.8912	0.0802
Ours	**34.42**	**0.9557**	0.0610	**32.36**	**0.9273**	**0.1171**	**37.13**	**0.9595**	**0.0344**

Table 4.3: Quantitative comparison of different methods for Cozy2room, Factory, and Pool datasets. The best results are in bold.

which is synthetically generated using Blender [1]. ExBluRF is specifically designed for benchmarking deblurring performance in scenarios involving camera motion. Each scene consists of 29 motion-blurred training images and 5 sharp test images that serve as ground truth for evaluation. The dataset contains diverse scenes with varying textures, geometries, and motion trajectories, providing a challenging and realistic testbed for 3D deblurring methods.

We compare our method against a range of state-of-the-art baselines, including both single-image deblurring techniques and NeRF-based multi-view deblurring methods. The compared baselines include:

- MPRNet [64]: A progressive image restoration model that operates on single blurred images using a multi-stage architecture.
- PVD [48]: A recurrent video deblurring framework capable of handling temporal

	Tanabata			Wine			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Park	17.91	0.4637	0.4030	19.96	0.5610	0.3222	22.14	0.5963	0.3154
MPR	22.60	0.7203	0.2507	26.24	0.8356	0.1762	27.42	0.8071	0.1777
PVD	22.54	0.6872	0.3351	24.44	0.7746	0.2600	26.00	0.7666	0.2519
SRNDeblur	23.20	0.7274	0.2438	25.36	0.8119	0.1618	27.50	0.8121	0.1691
DeblurNeRF	22.46	0.6946	0.2455	24.94	0.7923	0.1766	25.56	0.7571	0.1849
DeblurNeRF*	24.56	0.7749	0.2166	26.24	0.8254	0.1671	27.95	0.8336	0.1633
BAD-NeRF	27.88	0.8642	0.1179	29.25	0.8892	0.0833	30.94	0.8946	0.0916
Ours	**30.28**	**0.9316**	**0.0780**	**32.55**	**0.9503**	**0.0374**	**33.35**	**0.9448**	**0.0656**

Table 4.4: Quantitative comparison of different methods for Tanabata, Wine, and Average datasets. The best results are in bold.

coherence, adapted here for single-frame restoration.

- SRN-Deblur [52]: A widely-used scale-recurrent architecture for single-image deblurring using a coarse-to-fine strategy.
- DeblurNeRF [35]: A NeRF-based approach that jointly models motion blur and scene radiance, enabling sharp novel view synthesis from blurred multi-view inputs.
- BADNeRF [58]: A recent NeRF-based model that disentangles motion blur and appearance features to enhance deblurring performance in radiance fields.

4.3.1 Qualitative Results.

As shown in Fig. 4.11, we observe clear differences between single-image and multi-view methods. Single-image deblurring methods such as MPRNet, PVD, and SRN-Deblur strug-

gle to effectively remove motion blur in complex 3D scenes. These methods tend to produce over-smoothed results with residual blur, failing to restore fine structures and sharp object boundaries. This is expected, as they lack multi-view geometric consistency and cannot leverage cross-view correlations, which are essential in handling large 6-DoF camera motions common in ExBluRF scenes.

NeRF-based approaches, including DeblurNeRF and BADNeRF, exhibit superior deblurring capabilities by jointly modeling scene geometry and appearance from multiple views. These methods better restore structural details and reduce motion artifacts. However, they often suffer from inconsistent high-frequency details and may introduce ringing artifacts due to limitations in radiance field representations under severe blur.

In contrast, our method achieves the best qualitative performance, generating sharp, high-fidelity reconstructions with clean edges and minimal artifacts. The advantages stem from the inherent efficiency of the 3D Gaussian Splatting (3DGS) representation, which allows explicit control over the spatial extent of each Gaussian. Moreover, the incorporation of a low-pass filtering mechanism into the Gaussian convolution stage helps to suppress high-frequency noise and aliasing artifacts during Gaussian expansion, ensuring smooth transitions and visually coherent deblurring results across views.

4.3.2 Quantitative Results

The quantitative evaluation is summarized in Table 4.3 and Table 4.4, where we report performance on PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index Measure), and LPIPS (Learned Perceptual Image Patch Similarity). Across most scenes and metrics, our method ranks the highest, demonstrating its robustness and generalization across diverse motion blur patterns.

In particular, our method surpasses the second-best baseline, BADNeRF, by a substantial margin of 3.77 dB in PSNR, confirming its superior reconstruction fidelity. The gains in SSIM further reflect improvements in structural coherence, while lower LPIPS

scores indicate that our outputs are perceptually closer to the sharp ground truth. Single-image deblurring methods (Park, MPR, PVD, and SRNDeblur) perform significantly worse across all datasets. These methods lack explicit 3D scene modeling and multi-view constraints, which leads to inconsistent restorations in complex scenes with large motions and occlusions. For example, MPR achieves only 29.90 dB PSNR on the Cozy2room dataset, while our method improves this by 4.52 dB. While NeRF-based approaches (DeblurNeRF, DeblurNeRF*, and BAD-NeRF) leverage multi-view consistency, they still suffer from limitations in preserving high-frequency details and global consistency under severe motion blur. Among them, BAD-NeRF performs best, achieving 32.15 dB PSNR on *Cozy2room*, yet our method surpasses it by a clear margin of 2.27 dB. Additionally, our LPIPS score of 0.0344 on the *Pool* dataset demonstrates that our method produces perceptually more faithful results compared to BAD-NeRF's 0.0802. These results collectively demonstrate that our method not only excels in reducing pixel-wise errors but also preserves high-level perceptual quality and geometric consistency.

The strong performance of our approach can be attributed to several key design choices. First, the use of 3DGS provides a flexible and efficient point-based scene representation that handles motion blur through spatially-aware Gaussian modeling. Second, the low-pass filtering applied during the Gaussian convolution stage effectively attenuates artifacts caused by rapid motion and dense overlaps of Gaussians, which are common in heavily blurred regions. Finally, our pipeline naturally benefits from the complementary use of depth information and multi-view constraints, which stabilize reconstruction and ensure temporal consistency, outperforming both single-frame and volumetric NeRF-based baselines.

In summary, the experimental results on ExBluRF validate the capability of our method to deliver high-quality deblurred 3D scenes with superior accuracy, perceptual quality, and computational efficiency.

	Cozy2room			Factory			Pool		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
w/o LPF	33.68	0.9521	**0.0258**	31.88	0.9270	**0.0952**	36.95	0.9434	0.0425
Ours	**34.42**	**0.9557**	0.0610	**32.36**	**0.9273**	0.1171	**37.13**	**0.9595**	**0.0344**

Table 4.5: Ablation Study on Effects of Low-pass Filter for Cozy2room, Factory, and Pool datasets. The best results are in bold.

	Tanabata			Wine			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
w/o LPF	29.12	0.9281	0.1464	29.25	0.9392	0.0833	32.17	0.9380	0.0787
Ours	**30.28**	**0.9316**	**0.0780**	**32.55**	**0.9503**	**0.0374**	**33.35**	**0.9448**	**0.0656**

Table 4.6: Ablation Study on Effects of Low-pass Filter for Cozy2room, Factory, and Pool datasets. The best results are in bold.

4.3.3 Ablation Study

To evaluate the effectiveness of the proposed low-pass filter (LPF) in handling motion blur and stabilizing 3D Gaussian Splatting representations, we conducted ablation experiments on six diverse datasets: *Cozy2room*, *Factory*, *Pool*, *Tanabata*, *Wine*, and the overall *Average* across all scenes. The quantitative results are presented in Table 4.5 and Table 4.6 compare the performance of our full method (with LPF) against its variant without the low-pass filter (w/o LPF). We report standard evaluation metrics, including PSNR (higher is better), SSIM (higher is better), and LPIPS (lower is better).

As shown in Table 4.3, on the *Cozy2room*, *Factory*, and *Pool* datasets, our method achieves superior performance in most metrics. Specifically, for *Cozy2room*, the LPF-

equipped model improves PSNR from 33.68 dB to 34.42 dB and SSIM from 0.9521 to 0.9557, indicating better reconstruction accuracy and structural similarity. Although the LPIPS score slightly increases from 0.0258 to 0.0610, the improvement in PSNR and SSIM demonstrates that LPF helps enhance global image quality while maintaining high structural fidelity. Similarly, in the *Pool* dataset, our method improves PSNR and SSIM while achieving a better LPIPS score (reduced from 0.0425 to 0.0344), demonstrating improved perceptual quality.

Table 4.4 presents results on the *Tanabata*, *Wine*, and *Average* performance across all datasets. Our method consistently outperforms the baseline w/o LPF variant in all metrics. In the challenging *Tanabata* scene, LPF improves PSNR by 1.16 dB and reduces LPIPS from 0.1464 to 0.0780, reflecting significant perceptual enhancement. For *Wine*, the improvement is even more pronounced, with PSNR increasing from 29.25 dB to 32.55 dB and LPIPS nearly halving from 0.0833 to 0.0374. Overall, our method achieves an average PSNR of 33.35 dB and LPIPS of 0.0656, surpassing the w/o LPF variant (32.17 dB PSNR and 0.0787 LPIPS), confirming the generalization and robustness of LPF across diverse scenes.

These results demonstrate that the low-pass filter plays a crucial role in suppressing high-frequency artifacts introduced during Gaussian expansion under motion blur conditions. By regularizing the covariance of Gaussians and mitigating aliasing effects, LPF ensures smoother blending of Gaussians and improves both the numerical and perceptual quality of rendered images. In particular, LPF is especially beneficial in scenes with strong motion blur or fine structural details, where high-frequency noise can severely degrade reconstruction quality. Therefore, incorporating LPF into our 3DGS pipeline is essential for achieving high-fidelity, artifact-free 3D scene editing under motion blur scenarios.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this work, we presented a novel and efficient framework for 3D scene editing based on the 3D Gaussian Splatting (3DGS) representation, with a particular focus on tackling motion blur, depth refinement, and multi-view consistency in text-driven 3D editing scenarios. Our approach systematically addresses several fundamental challenges in 3D-aware editing pipelines by integrating three key innovations: depth enhancement, cross-view latent alignment, and motion-blur-aware 3D reconstruction.

First, we eliminated the Iterative Dataset Update (IDU) strategy used in previous 2D-3D lifting-based editing methods, significantly reducing computational complexity. To maintain multi-view consistency, we leveraged multi-view consistent depth maps rendered from 3DGS, which serve as geometric priors for multi-view image editing. To further refine depth quality, we introduced the **Complementary Information Mutual Learning Network (CIMLN)**, designed for robust depth map enhancement. CIMLN effectively integrates complementary cues from RGB images and their corresponding depth maps, suppressing texture noise while enhancing geometric edges. This refinement produces sharper, more reliable depth estimates, which are crucial for providing accurate geometric guidance

during diffusion-based editing. By stabilizing the editing process and preserving structural integrity, CIMLN ensures precise and consistent 3D scene modifications.

Second, we designed the **Wavelet Consensus Attention (WCA)** mechanism to enforce cross-view consistency during multi-view editing. By aligning latent codes across spatial and frequency domains through wavelet decomposition, WCA ensures that edits are coherently propagated across all viewpoints. This prevents view-dependent artifacts, such as color drift or structural inconsistency, and guarantees globally consistent scene modifications under arbitrary camera trajectories.

Third, we introduced a dedicated module for **3D Gaussian Splatting from motion-blurred images** via a **low-pass filtering strategy**. By convolving 3D Gaussians with isotropic low-pass filters prior to rasterization, we effectively suppress high-frequency noise and aliasing artifacts induced by severe motion blur. This filtering mechanism not only improves the stability of Gaussian representations under motion but also ensures smooth blending of overlapping Gaussians, leading to higher-quality 3D reconstructions from blurred multi-view inputs.

Through extensive experiments on diverse datasets such as ExBluRF, Mip-NeRF, BlendedMVS, and NeRF-Art, we validated the superiority of our method over existing baselines. Quantitative results showed consistent improvements in PSNR, SSIM, LPIPS, and CLIPdir, while qualitative visualizations confirmed that our framework produces high-fidelity, semantically aligned, and temporally stable 3D edits. Furthermore, the inherent efficiency of the 3DGS representation, combined with our one-time editing strategy, delivers fast convergence and low computational overhead, making our approach highly suitable for practical applications in fields like augmented reality (AR), virtual reality (VR), digital content creation, and interactive design.

Looking ahead, we plan to extend our framework to support more interactive and controllable editing paradigms. Beyond text-driven modifications, we aim to explore multimodal editing inputs such as sketches, gestures, and region-specific constraints, enabling

users to intuitively specify both local and global modifications with fine-grained control. Further research directions include extending our method to dynamic scenes with temporally consistent editing, supporting continuous scene updates in video sequences, and developing adaptive Gaussian representations that adjust dynamically to scene complexity. Through these efforts, we aim to build a general-purpose, real-time, and user-friendly 3D editing platform capable of handling complex scene modifications with minimal latency and maximum creative freedom.

5.2 Future Work

While the proposed methods have demonstrated promising results in enhancing the quality, consistency, and efficiency of 3D Gaussian Splatting (3DGS)-based scene editing and reconstruction, several directions remain for future exploration.

Dynamic Scenes: Although our text-driven editing framework improves multi-view consistency and depth-aware editing, extending its capabilities to handle dynamic scenes with moving objects and temporal coherence remains an open challenge. Integrating motion information and temporal constraints into the editing pipeline could enable robust, frame-consistent 3D editing in video sequences or time-varying environments.

Multi-Modal Guidance: Our method currently relies on high-quality text prompts and well-aligned multi-view data. In future work, we aim to explore multi-modal guidance, such as combining textual descriptions with sketch inputs, user-provided masks, or spatial constraints, to provide more controllable and intuitive editing experiences.

Motion-Blurred Reconstruction: While our approach effectively mitigates blur and refines camera trajectories, its performance may degrade under extreme blur or severe pose inaccuracies. Future research could investigate self-supervised pretraining strategies or adaptive blur modeling to further improve robustness in such cases.

Object-Level Editing: Current methods primarily focus on scene-level modifications,

but object-level editing remains an important challenge. Enabling precise manipulation of individual objects, such as repositioning, relighting, or replacing them based on textual or user-specified constraints, would greatly enhance flexibility. Future work could incorporate segmentation-based object disentanglement and neural rendering techniques to facilitate fine-grained object-level control.

Scalability: An important consideration for deploying 3DGS-based editing in large-scale or outdoor environments is efficient memory management, hierarchical scene representations, and hybrid techniques that combine Gaussians with other neural representations (e.g., meshes or point clouds). These strategies could support editing of complex, expansive scenes while maintaining real-time performance.

Through these future directions, we hope to push the boundaries of 3D scene editing towards greater flexibility, robustness, and real-world applicability.

References

[1] *Online Community. Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022.

[7] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. In *European Conference on Computer Vision*. Springer, 2024.

[8] Xuanhong Chen, Hang Wang, Jialiang Chen, Kairui Feng, Jinfan Liu, Xiaohang Wang, Weimin Zhang, and Bingbing Ni. Intrinsic phase-preserving networks for depth super resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1210–1218, 2024.

[9] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024.

[10] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023.

[11] Xinhua Cheng, Tianyu Yang, Jianan Wang, Yu Li, Lei Zhang, Jian Zhang, and Li Yuan. Progressive3d: Progressively local editing for text-to-3d content creation with complex semantic prompts. In *The Twelfth International Conference on Learning Representations*, 2024.

[12] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.

[13] Dale Decatur, Itai Lang, Kfir Aberman, and Rana Hanocka. 3d paintbrush: Local stylization of 3d shapes with cascaded score distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4473–4483, 2024.

[14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Proceedings of Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[15] Jiahua Dong and Yu-Xiong Wang. Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. volume 36, 2024.

[16] Xiaoyu Dong, Naoto Yokoya, Longguang Wang, and Tatsumi Uezato. Learning mutual modulation for self-supervised cross-modal super-resolution. In *European Conference on Computer Vision*, pages 1–18. Springer, 2022.

[17] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified implicit neural stylization. In *European Conference on Computer Vision*, pages 636–654. Springer, 2022.

[18] Shuangkang Fang, Yufeng Wang, Yi Yang, Yi-Hsuan Tsai, Wenrui Ding, Shuchang Zhou, and Ming-Hsuan Yang. Editing 3d scenes via text prompts without retraining. *arXiv e-prints*, pages arXiv–2309, 2023.

[19] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[21] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19740–19750, 2023.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Proceedings of Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[23] Hao-Yu Hsu, Zhi-Hao Lin, Albert Zhai, Hongchi Xia, and Shenlong Wang. Autovfx: Physically realistic video editing from natural language instructions. *arXiv preprint arXiv:2411.02394*, 2024.

[24] Junjie Hu, Chenyu Bao, Mete Ozay, Chenyou Fan, Qing Gao, Honghai Liu, and Tin Lun Lam. Deep depth completion from extremely sparse data: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):8244–8264, 2022.

[25] Junha Hyung, Sungwon Hwang, Daejin Kim, Hyunji Lee, and Jaegul Choo. Local 3d editing via 3d distillation of clip knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12674–12684, 2023.

[26] Jaewoo Jung, Jisang Han, Honggyu An, Jiwon Kang, Seonghoon Park, and Seungryong Kim. Relaxing accurate initialization constraint for 3d gaussian splatting. *arXiv preprint arXiv:2403.09413*, 2024.

[27] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*, 2023.

[28] Nazmul Karim, Hasan Iqbal, Umar Khalid, Chen Chen, and Jing Hua. Free-editor: Zero-shot text-driven 3d scene editing. In *European Conference on Computer Vision*, pages 436–453. Springer, 2025.

[29] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139–1, 2023.

[30] Umar Khalid, Hasan Iqbal, Nazmul Karim, Muhammad Tayyab, Jing Hua, and Chen Chen. Latenteditor: Text driven local editing of 3d scenes. In *European Conference on Computer Vision*, pages 364–380. Springer, 2025.

[31] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[32] Dongwoo Lee, Jeongtaek Oh, Jaesung Rim, Sunghyun Cho, and Kyoung Mu Lee. Exblurf: Efficient radiance fields for extreme motion blurred images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17639–17648, 2023.

[33] Sizhe Li, Yiming Qin, Minghang Zheng, Xin Jin, and Yang Liu. Diff-bgm: A diffusion model for video background music generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27348–27357, 2024.

[34] Lihua Lu, Ruyang Li, Xiaohui Zhang, Hui Wei, Guoguang Du, and Binqiang Wang. Advances in text-guided 3d editing: A survey. *Artifical Intelligence Review*, 57(12):1–61, 2024.

[35] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-nerf: Neural radiance fields from blurry images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12861–12870, 2022.

[36] Yiwei Ma, Xiaoqing Zhang, Xiaoshuai Sun, Jiayi Ji, Haowei Wang, Guannan Jiang, Weilin Zhuang, and Rongrong Ji. X-mesh: Towards fast and accurate text-driven 3d stylization via dynamic textual guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2749–2760, 2023.

[37] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12663–12673, 2023.

[38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[39] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G Derpanis, and Igor Gilitschenski. Reference-guided controllable inpainting of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17815–17825, 2023.

[40] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022.

[41] Francesco Palandra, Andrea Sanchietti, Daniele Baieri, and Emanuele Rodolà. Gsedit: Efficient text-guided editing of 3d objects via gaussian splatting. *arXiv preprint arXiv:2403.05154*, 2024.

[42] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[44] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[46] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[47] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 430–440, 2023.

[48] Hyeongseok Son, Junyong Lee, Jonghyeop Lee, Sunghyun Cho, and Seungyong Lee. Recurrent video deblurring with blur-invariant motion estimation and pixel volumes. *ACM Transactions on Graphics (TOG)*, 40(5):1–18, 2021.

[49] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[50] Liangchen Song, Liangliang Cao, Jiatao Gu, Yifan Jiang, Junsong Yuan, and Hao Tang. Efficient-nerf2nerf: Streamlining text-driven 3d editing with multiview correspondence-enhanced diffusion models. *arXiv preprint arXiv:2312.08563*, 2023.

[51] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023.

[52] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8174–8182, 2018.

[53] Cyrus Vachha and Ayaan Haque. Instruct-gs2gs: Editing 3d gaussian splats with instructions, 2024. https://instruct-gs2gs.github.io/.

[54] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022.

[55] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Trans. Vis. Comput. Graph.*, 2023.

[56] Haotian Wang, Meng Yang, Ce Zhu, and Nanning Zheng. Rgb-guided depth map recovery by two-stage coarse-to-fine dense crf models. *IEEE Transactions on Image Processing*, 32:1315–1328, 2023.

[57] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20902–20911, 2024.

[58] Peng Wang, Lingzhe Zhao, Ruijie Ma, and Peidong Liu. Bad-nerf: Bundle adjusted deblur neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4170–4179, 2023.

[59] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *European Conference on Computer Vision*, pages 404–420. Springer, 2025.

[60] Zhengxue Wang, Zhiqiang Yan, and Jian Yang. Sgnet: Structure guided network via gradient-frequency awareness for depth map super-resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 5823–5831, 2024.

[61] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. In *European Conference on Computer Vision*. Springer, 2024.

[62] Junkai Yan, Yipeng Gao, Qize Yang, Xihan Wei, Xuansong Xie, Ancong Wu, and Wei-Shi Zheng. Dreamview: Injecting view-specific text guidance into text-to-3d generation. In *European Conference on Computer Vision*, pages 358–374. Springer, 2025.

[63] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1790–1799, 2020.

[64] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14821–14831, 2021.

[65] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[66] Qihang Zhang, Yinghao Xu, Chaoyang Wang, Hsin-Ying Lee, Gordon Wetzstein, Bolei Zhou, and Ceyuan Yang. 3ditscene: Editing any scene via language-guided disentangled gaussian splatting. *arXiv preprint arXiv:2405.18424*, 2024.

[67] Xuanqi Zhang, Jieun Lee, Chris Joslin, and Wonsook Lee. Advancing 3d gaussian splatting editing with complementary and consensus information. *arXiv preprint arXiv:2503.11601*, 2025.

[68] Xuanqi Zhang, Wonsook Lee, and Chris Joslin. 3d gaussian splatting and camera poses joint optimization for motion blur images via low-pass filter.

[69] Xuanqi Zhang, Haijin Zeng, Jinwang Pan, Qiangqiang Shen, and Yongyong Chen. Llemamba: Low-light enhancement via relighting-guided mamba with deep unfolding network. *arXiv preprint arXiv:2406.01028*, 2024.

[70] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016.

[71] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023.

[72] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.

APPENDICES

A Proof of State Space Model

To rigorously establish the properties of Mamba's Selective State Space Model (SSM), we provide a detailed mathematical derivation below. This proof elucidates the model's ability to handle long-range dependencies while maintaining computational efficiency.

1. Continuous-Discrete Time Conversion

The SSM originates from linear time-invariant (LTI) systems described by continuous-time differential equations:

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t) \tag{1}$$
$$y(t) = \mathbf{C}^\top h(t)$$

Discretization using zero-order hold (ZOH) with step size Δ_t yields:

$$\overline{\mathbf{A}}_t = e^{\Delta_t \mathbf{A}_t} \tag{2}$$
$$\overline{\mathbf{B}}_t = (\Delta_t \mathbf{A}_t)^{-1}(e^{\Delta_t \mathbf{A}_t} - \mathbf{I})\Delta_t \mathbf{B}_t \tag{3}$$

The discrete SSM becomes:

$$h_t = \overline{\mathbf{A}}_t h_{t-1} + \overline{\mathbf{B}}_t x_t \tag{4}$$
$$y_t = \mathbf{C}_t^\top h_t \tag{5}$$

2. Selective Parameter Projection

Mamba's innovation lies in making parameters input-dependent through learned projections:

$$[\Delta_t, \mathbf{B}_t, \mathbf{C}_t] = \text{Linear}(x_t) \in \mathbb{R}^{1\times 2N+1} \tag{6}$$

$$\mathbf{A}_t = -\exp(\text{Linear}(x_t)) \in \mathbb{R}^{N\times N} \quad \text{(ensuring stability)} \tag{7}$$

where N is the state dimension. This allows dynamic adaptation to input characteristics through:

$$\overline{\mathbf{A}}_t = e^{\Delta_t \mathbf{A}_t} = \text{diag}(e^{\Delta_t a_1}, ..., e^{\Delta_t a_N}) \tag{8}$$

3. Convolutional Representation

Unrolling the recurrence reveals an equivalent convolutional form:

$$h_t = \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \overline{\mathbf{A}}_i \right) \overline{\mathbf{B}}_k x_k \tag{9}$$

$$y_t = \mathbf{C}_t^\top \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \overline{\mathbf{A}}_i \right) \overline{\mathbf{B}}_k x_k \tag{10}$$

This can be expressed as a causal convolution:

$$y = \mathcal{K} * x \quad \text{where} \quad \mathcal{K}_t = \mathbf{C}_t^\top \left(\prod_{i=1}^{t} \overline{\mathbf{A}}_i \right) \overline{\mathbf{B}}_0 \tag{11}$$

4. Computational Complexity Analysis

The model achieves $\mathcal{O}(L)$ complexity through:

Recurrent Mode:

$$\text{FLOPs} = L(3N^2 + 2N) \propto \mathcal{O}(L) \tag{12}$$

Convolutional Mode: Using Fast Fourier Transform (FFT):

$$\mathcal{O}(L \log L) \quad \text{but requires} \quad \mathcal{O}(LN) \quad \text{memory} \tag{13}$$

5. Stability Guarantees

The exponential parameterization ensures eigenvalues of $\overline{\mathbf{A}}_t$ lie within the unit circle:

$$|\lambda(\overline{\mathbf{A}}_t)| = |e^{\Delta_t a_i}| = e^{\Delta_t \text{Re}(a_i)} < 1 \quad \text{when} \quad \text{Re}(a_i) < 0 \tag{14}$$

This guarantees asymptotic stability through the Lyapunov condition:

$$\exists P \succ 0 : \overline{\mathbf{A}}_t^\top P \overline{\mathbf{A}}_t - P \prec 0 \tag{15}$$

6. Selective Information Filtering

The input-dependent parameters implement differentiable gating:

$$\text{Retention Rate:} \quad \gamma_t = \sigma(\mathbf{W}_\gamma x_t) \tag{16}$$

$$\text{Forgetting Rate:} \quad \phi_t = \sigma(\mathbf{W}_\phi x_t) \tag{17}$$

These modulate the state update through:

$$h_t = \gamma_t \odot (\overline{\mathbf{A}}_t h_{t-1}) + \phi_t \odot (\overline{\mathbf{B}}_t x_t) \tag{18}$$

7. Parallel Scan Implementation

The recurrence can be parallelized using associative scans:

$$h_{1:L} = \text{scan}((\overline{\mathbf{A}}_{1:L}, \overline{\mathbf{B}}_{1:L} x_{1:L})) \tag{19}$$

Implemented through a binary tree reduction with:

$$(h_i, h_j) \mapsto h_j + \overline{\mathbf{A}}_j h_i \tag{20}$$

This achieves $\mathcal{O}(\log L)$ depth parallelism while maintaining numerical stability through exponentiated $\overline{\mathbf{A}}$ matrices.

8. Expressivity Analysis

The SSM can approximate any linear time-varying system through:

$$y_t = \mathbf{C}_t^\top \left[\prod_{k=1}^{t} \overline{\mathbf{A}}_k \right] h_0 + \sum_{k=1}^{t} \mathbf{C}_t^\top \left[\prod_{i=k+1}^{t} \overline{\mathbf{A}}_i \right] \overline{\mathbf{B}}_k x_k \tag{21}$$

The product term enables modeling exponentially decaying memory with rates:

$$\mu_k = \prod_{i=k}^{t} \overline{\mathbf{A}}_i = \exp\left(\sum_{i=k}^{t} \Delta_i \mathbf{A}_i \right) \tag{22}$$

This mathematical formulation demonstrates that Mamba's Selective SSM achieves:

- Linear time complexity through parallel scans
- Long-range dependency modeling via stable recurrence
- Input-adaptive filtering through parameter projection
- Formal stability guarantees via exponential parameterization

The combination of these properties establishes Mamba as both theoretically sound and computationally efficient for sequence modeling tasks.

B Additional Images

Figure A1: Multi-view images rendered from the edited 3D Gaussian Splatting (3DGS) model using the text prompt: "Turn it into a giraffe."

Figure A2: Multi-view images rendered from the edited 3D Gaussian Splatting (3DGS) model using the text prompt: "Turn it into a tiger."

Figure A3: Multi-view images rendered from the edited 3D Gaussian Splatting (3DGS) model using the text prompt: "Give him a pair of glasses."