Technische Universität Berlin

Computer Science
Fakultät IV
Marchstr. 23
10587 Berlin
https://www.tu.berlin/cv

Master Thesis

Dynamic Surface Reconstruction from Multi-View Videos using Gaussian Splatting

Brianne Oberson

Supervisor:
Decai Chen

Examiners:
Priv.-Doz. Dr.-Ing. Oliver Schreer
Prof. Dr.-Ing. Olaf Hellwich

Submitted on April 28th, 2025

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 28.04.2025

. . . . . .

Brianne Oberson

Abstract

Dynamic 3D reconstruction focuses on creating detailed, time-varying 3D representations of objects and scenes that undergo changes such as motion or deformation. Unlike static 3D reconstruction, which only deals with fixed objects, dynamic reconstruction must capture both the geometry of objects and how they evolve over time. This capability is crucial for a range of applications, including motion capture, augmented reality (AR), and virtual reality (VR), where high-quality, real-time modeling of moving objects is essential.

In recent years, techniques like 3D Gaussian Splatting (3DGS) have significantly advanced static 3D view synthesis by leveraging Gaussian representations to efficiently model complex 3D scenes. These methods have led to major improvements in both fidelity and computational efficiency, and have been successfully extended to static surface reconstruction. There has also been attempts to extend 3DGS to dynamic view synthesis, and while these methods perform well at generating realistic and temporally coherent reconstructions for visualization purposes, they tend to be inadequate when integrated into modern graphics engines, where detailed and temporally coherent surface meshes are important for applications such as geometry editing, physics-based simulations, animation, and texture mapping. However, to date, there are no existing methods that effectively achieve dynamic surface reconstruction from multi-view videos using gaussian splatting, highlighting a significant gap in the current research landscape.

This master's thesis explores state-of-the-art extensions of 3DGS to both dynamic view synthesis and 3D surface reconstruction. Among existing approaches, two methods were selected as foundations for this work: Gaussian Surfels [Dai et al., 2024] for its quality in static surface reconstruction and Spacetime Gaussians [Li et al., 2024] for its performance in dynamic view synthesis. The proposed approach integrates key concepts from both methods to create an effective solution for dynamic 3D surface reconstruction, addressing the unique challenges posed by motion and deformation over time while leveraging the spatial information provided by multi-view videos.

Zusammenfassung

Die dynamische 3D-Rekonstruktion konzentriert sich auf die Erstellung detaillierter, zeitlich veränderlicher 3D-Darstellungen von Objekten und Szenen, in denen Veränderungen wie Bewegung oder Verformung geschehen. Im Gegensatz zur statischen 3D-Rekonstruktion, die sich nur mit festen Objekten befasst, muss die dynamische Rekonstruktion sowohl die Geometrie der Objekte als auch deren zeitliche Entwicklung erfassen. Diese Fähigkeit ist entscheidend für eine Reihe von Anwendungen, darunter Motion Capture, Augmented Reality (AR) und Virtual Reality (VR), bei denen hochwertige Echtzeitmodellierung bewegter Objekte unerlässlich ist.

In den letzten Jahren haben Techniken wie 3D Gaussian Splatting (3DGS) die statische 3D-Ansichtssynthese durch die Nutzung von Gauß'schen Darstellungen zur effizienten Modellierung komplexer 3D-Szenen erheblich vorangebracht. Diese Methoden haben zu wesentlichen Verbesserungen sowohl in der Genauigkeit als auch in der Recheneffizienz geführt und wurden erfolgreich auf die statische Oberflächenrekonstruktion erweitert. Es gab auch Versuche, 3DGS auf die dynamische Ansichtssynthese auszuweiten, und während diese Methoden bei der Erzeugung realistischer und zeitlich kohärenter Rekonstruktionen für Visualisierungszwecke gut funktionieren, sind sie tendenziell unzureichend, wenn sie in moderne Grafik-Engines integriert werden, wo detaillierte und zeitlich kohärente Oberflächennetze wichtig für Anwendungen wie Geometriebearbeitung, physikbasierte Simulationen, Animation und Texturmapping sind. Bislang gibt es jedoch keine existierenden Methoden, die eine effektive dynamische Oberflächenrekonstruktion aus Multi-View-Videos unter Verwendung von Gaussian Splatting erreichen, was eine bedeutende Lücke in der aktuellen Forschungslandschaft darstellt.

Diese Masterarbeit untersucht modernste Erweiterungen von 3DGS sowohl für die dynamische Ansichtssynthese als auch für die 3D-Oberflächenrekonstruktion. Unter den bestehenden Ansätzen wurden zwei Methoden als Grundlage für diese Arbeit ausgewählt: Gaussian Surfels [Dai et al., 2024] für seine Qualität in der statischen Oberflächenrekonstruktion und Spacetime Gaussians [Li et al., 2024] für seine Leistung in der dynamischen Ansichtssynthese. Der vorgeschlagene Ansatz integriert Schlüsselkonzepte aus beiden Methoden, um eine effektive Lösung für die dynamische 3D-Oberflächenrekonstruktion zu schaffen, die die einzigartigen Herausforderungen bewältigt, die durch Bewegung und Verformung im Laufe der Zeit entstehen, während die räumlichen Informationen aus Multi-View-Videos genutzt werden.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Dynamic 3D reconstruction is a rapidly advancing field focused on generating detailed, time-varying representations of objects and scenes that undergo transformations such as motion or deformation. Unlike static 3D reconstruction, which is limited to modeling fixed objects or environments, dynamic reconstruction captures both the geometry of objects and their temporal evolution. This dual capability enables the creation of highly detailed and accurate representations of moving or deforming entities, making it a critical component in numerous emerging technologies.

The applications of dynamic 3D reconstruction are far-reaching. It plays a key role in immersive experiences such as augmented reality (AR) and virtual reality (VR), where accurate, real-time tracking of moving objects is crucial. Beyond these, it also finds use in fields like video game development, cinema, and cultural heritage. Whether it's creating lifelike characters for interactive games, enhancing special effects in films, or preserving the movements of historical artifacts, dynamic 3D reconstruction is shaping how we represent and interact with the physical world in digital form.

The growing need for accurate and efficient dynamic 3D reconstruction has driven the development of sophisticated algorithms and techniques, with artificial intelligence (AI) playing an important role in recent years. This thesis explores the current state of dynamic 3D reconstruction, highlighting the challenges, recent innovations, and building on these findings, it aims to develop a novel method to further advance the field.

1.2 Problem Formulation

The field of 3D reconstruction has experienced significant improvements in recent years, driven in large part by the increasing accessibility and advancements in machine learning techniques. Among these developments, the 3D Gaussian Splatting (3DGS) framework [Kerbl et al., 2023] has emerged as a groundbreaking contribution. Initially designed to enable free-viewpoint rendering of static scenes, 3DGS has since been extended to encompass the generation of detailed surface meshes for static scenes and the rendering of free-viewpoint perspectives for dynamic scenes.

Despite these advancements, there remains a significant gap in the development of methods for dynamic surface reconstruction. While dynamic view synthesis and static surface

reconstruction have been addressed individually, no existing approach effectively combines these capabilities to reconstruct dynamic surfaces using Gaussian Splatting. This limitation presents a critical research opportunity at the intersection of these rapidly evolving techniques.

While dynamic view-synthesis methods offer impressive visual quality and temporal coherence for rendering purposes, these approaches primarily optimize for visual appearance rather than geometric accuracy. Modern graphics engines require detailed and temporally coherent surface meshes to support a wide range of applications beyond mere visualization. These include geometry editing, physics-based simulations, collision detection, and texture mapping, all of which depend on accurate geometric representations rather than view-dependent rendering optimizations. The existing methods for dynamic view synthesis using Gaussian representations fail to provide the necessary geometric stability and surface continuity required for these applications.

This thesis aims to address this gap by exploring current methods for dynamic view synthesis and static surface reconstruction within the context of 3D Gaussian Splatting. By analyzing and integrating the most effective techniques from both areas, the goal is to develop a novel framework for dynamic surface reconstruction, advancing the state of the art in 3D reconstruction.

1.3 Objectives

The primary objective of this thesis is to develop an efficient and accurate method for reconstructing the surface mesh of a dynamic object from multi-view videos using the 3D Gaussian Splatting (3DGS) technique. This research aims to analyze whether adapting the 3DGS approach to dynamic surface reconstruction can provide advantages over current state-of-the-art methods in terms of reconstruction quality and training time efficiency.

The research focuses on datasets captured in professional studio environments, where subjects are positioned in the center of the scene with cameras arranged in a dome-like structure. This controlled setup provides significant advantages: precise pose control, calibrated lighting, and synchronized camera systems ensure high-quality input data while minimizing artifacts and inconsistencies. Such conditions are particularly beneficial for the target applications in virtual reality experiences and interactive digital content, where accuracy and realism are paramount. Furthermore, this approach enables the research to focus on advancing the core reconstruction methodology rather than dealing with the additional complexity of uncontrolled environmental factors. This strategic choice allows for the development of more sophisticated and efficient algorithms while maintaining the high quality standards required for professional VR and digital media applications.

A critical component of this research will be to rigorously evaluate the performance

of the proposed method against established baseline approaches. The evaluation will employ two complementary assessment strategies. First, quantitative metrics for rendering quality (such as PSNR, SSIM, and LPIPS) will be used to objectively compare the visual fidelity of the reconstructions, as ground truth meshes are not available for direct comparison. Second, a qualitative analysis will be conducted to assess the reconstructed mesh surfaces, examining properties such as smoothness, detail preservation, temporal consistency and artifact presence. This dual evaluation approach will provide a comprehensive understanding of both the perceptual quality and the geometric accuracy of the reconstructions. Additionally, computational efficiency during the training process will be measured to determine whether the 3DGS approach offers practical advantages in terms of resource utilization and time requirements. Together, these evaluations will provide valuable insights into the practical advantages and limitations of the 3DGS approach within the dynamic surface reconstruction domain and inform future research directions in this rapidly evolving field.

1.4 Contributions

This thesis makes the following key contributions to the field of dynamic surface reconstruction:

1. A comprehensive review and analysis of state-of-the-art methods in 3D reconstruction, with particular emphasis on 3D Gaussian Splatting (3DGS) techniques.

2. Development of a novel Gaussian Splatting-based dynamic surface reconstruction method that integrates two approaches identified during the literature review—combining the geometric precision of an advanced static surface reconstruction technique with the temporal handling capabilities of a cutting-edge dynamic view-synthesis model to create an effective solution for processing multi-view video data.

3. Investigation of multiple enhancement strategies aimed at improving geometry quality, reducing model size, and enforcing temporal consistency—providing valuable documentation of both successful approaches and unsuccessful attempts that inform future research directions.

4. Extensive comparative evaluation of the proposed method against established baseline approaches using public datasets, providing both quantitative benchmarks through rendering quality metrics and qualitative assessments of surface characteristics, offering a multifaceted analysis of the method's performance in realistic application scenarios.

1.5 Structure

Below is a brief outline of each chapter in the thesis.

Chapter 2 presents a literature review of 3D reconstruction techniques, from traditional Multi-View Stereo approaches to modern machine learning methods. After establishing key terminology, it examines the evolution of the field with particular focus on Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) as state-of-the-art approaches.

Chapter 3 details the methodological framework of this thesis. Starting with the fundamentals of 3DGS, it presents an approach that extends the base model by modifying its core representation to incorporate surface and temporal parameters, along with additional loss functions derived from contemporary methods in the field.

Chapter 4 presents the experimental validation of the proposed method. After introducing the multi-view video datasets of dynamic human performances, the chapter provides a comprehensive evaluation through qualitative and quantitative comparisons. The chapter concludes with an examination of several experimental modifications aimed at improving mesh quality, training efficiency, and memory usage. While these investigations yielded valuable insights into the method's behavior, empirical results demonstrated that the vanilla implementation achieved the most effective performance across the tested scenarios.

Chapter 5 concludes the thesis by analyzing the strengths and limitations of the proposed approach, discussing potential improvements, and summarizing the key contributions. It reflects on the research findings and suggests directions for future work in dynamic scene reconstruction.

2 Related Works

This chapter provides an overview of key concepts, technologies, and methodologies relevant to the field of 3D reconstruction. The discussion begins in section 2.1, which clarifies two essential terms frequently used throughout this literature review and crucial for differentiating between various approaches to 3D reconstruction. Section 2.2 delves into the state-of-the-art methods in 3D reconstruction, starting with a brief discussion of traditional techniques, such as Multi-View Stereo (MVS), detailed in section 2.2.1. These early methods laid the groundwork for subsequent advancements in the field. The focus then shifts to modern approaches based on Neural Radiance Fields (NeRF) in section 2.2.2, which have significantly reshaped research directions in 3D reconstruction. Section 2.2.3 examines both static and dynamic point based representations, namely 3D Gaussian Splatting (3DGS) and its derivatives, a pioneering technique that represents the current state of the art in this domain. The chapter concludes with Section 2.3, which synthesizes the key findings and identifies critical research gaps in the existing literature, establishing the foundation for this thesis's research objectives.

2.1 View Synthesis vs. Surface Reconstruction

In the field of 3D reconstruction, two distinct methodological approaches have emerged to address the challenge of representing three-dimensional objects. The first approach, View Synthesis, focuses on generating novel viewpoints of an object directly through image synthesis, allowing viewers to observe the object from perspectives not present in the original dataset. The second approach, Surface Reconstruction, aims to create an explicit geometric representation of the object, capturing its physical form in a way that enables further manipulation and analysis.

When discussing surface geometry, what is effectively needed is a triangle mesh of the surface. Triangle meshes provide the explicit connectivity information and topological structure essential for advanced geometric operations. Unlike implicit or point-based representations, triangle meshes offer well-defined surface normals, boundaries, and connectivity relationships that enable reliable spatial reasoning. This explicit representation allows for precise control over surface properties, facilitates efficient collision detection algorithms, and provides a framework for consistent texture parameterization—capabilities that are fundamental requirements for most practical applications in computer graphics, computational geometry, and physical simulation.

While recent research has gravitated toward view synthesis due to its computational

efficiency and reduced storage requirements, surface reconstruction remains crucial for many practical applications. Modern graphics engines and virtual 3D environments predominantly rely on triangle meshes for object manipulation and integration. This technical requirement makes surface reconstruction essential for applications that demand detailed, interactive 3D models.

The distinction between these approaches is fundamental to understanding their respective strengths and applications. View synthesis excels at generating visual representations quickly and efficiently, while surface reconstruction provides the geometric foundation necessary for complex object manipulation and integration with existing 3D graphics pipelines.

2.2 Overview of 3D Reconstruction Techniques

This section presents a chronological progression of 3D reconstruction methods, tracing their evolution from traditional approaches to cutting-edge techniques. The methods fall into three main categories: Multi-View Stereo (MVS), Neural Implicit Representations, and Point-Based Representations.

2.2.1 Multi-View Stereo

Classical multi-view stereo (MVS) techniques were the dominant approach for 3D reconstruction before the rise of neural rendering methods. These classical approaches can be broadly categorized into depth-based methods [Barnes et al., 2023], [Furukawa and Ponce, 2010], [Galliani et al., 2015], [Schonberger and Frahm, 2016], which reconstruct point clouds by identifying correspondences across images, and voxel-based methods [Bonet, 1999], [Broadhurst et al., 2001], [Seitz and Dyer, 1997], [Sinha and Pollefeys, 2005], which estimate occupancy in a 3D grid using photometric consistency. While depth-based methods struggle with correspondence matching in textureless regions, voxel-based approaches are limited by memory constraints and resolution. The field later evolved with the introduction of learning-based MVS approaches like MVSNet [Yao et al., 2018], which uses deep neural networks for end-to-end depth estimation. The resulting point clouds or voxel grids are typically converted into meshes using techniques like Screened Poisson reconstruction [Kazhdan and Hoppe, 2013] or Marching Cubes [Lorensen and Cline, 1987].

However, both classical and learning-based MVS methods fundamentally rely on photometric consistency across views, making them struggle with challenging scenarios such as texture-less regions, fine geometric details, and view-dependent effects. These limitations motivated the development of approaches like NeRF and 3D Gaussian Splatting, which represent scenes with implicit or explicit neural representations capable of modeling complex view-dependent appearance, as will be discussed in the following sections.

2.2.2 Neural Implicit Representations

In recent years, neural implicit representations have emerged as a powerful paradigm for encoding 3D geometry and appearance, representing scenes through learned continuous functions rather than discrete data structures like meshes or voxels. These approaches leverage neural networks to learn mappings from spatial coordinates to scene properties, enabling compact yet expressive scene representations.

NeRF [Mildenhall et al., 2020] marked a significant advancement in neural implicit representations. NeRF introduced a novel application of Multilayer Perceptrons (MLPs) to represent and enable high-quality view synthesis of complex 3D scenes. NeRF utilizes MLPs to encode the radiance field of a scene, a mathematical representation that encapsulates both its geometry and appearance. Instead of explicitly storing scene information, NeRF represents it implicitly within the neural network's weights. The rendering process works by casting rays from a virtual camera through image pixels into the scene, sampling points along each ray's path. The MLP transforms these sampled points into density and color values, which are combined through alpha blending, a classical volume rendering technique, to synthesize the final image. While this approach enables high-quality novel view synthesis, it has two key limitations: first, its volume density representation lacks explicit surface constraints, making it challenging to extract accurate surface geometry through simple density thresholding, and second, its reliance on a relatively large MLP (8 layers) makes both training and inference computationally intensive, requiring several hours and even days of training time depending on the scene, thus making real-time rendering impractical. These limitations spawned numerous follow-up works focused on both improving surface reconstruction quality and computational efficiency.

UniSurf [Oechsle et al., 2021] demonstrates that by formulating implicit surface models and radiance fields in a unified mathematical framework, a single model can seamlessly support both surface and volume rendering approaches. Instead of using NeRF's continuous density field, UniSurf learns an occupancy field that represents surfaces as binary boundaries between empty and occupied space.

VolSDF [Yariv et al., 2021] and NeuS [Wang et al., 2023a] introduce different approaches to integrate signed distance functions (SDFs) into neural volume rendering frameworks. Both methods represent surfaces as zero-level sets of SDFs, with positive values indicating points outside objects and negative values inside. However, they differ in how they bridge SDFs with volume rendering: VolSDF employs a simple exponential transformation to convert SDF values to density values, while NeuS introduces a theoretically-grounded "S-density" derived from the logistic density distribution. NeuS's formulation ensures unbiased and occlusion-aware rendering by concentrating rendering contributions around the zero-level set while correctly handling multiple surface intersections. Both methods combine the geometric benefits of SDFs with neural volume rendering, though NeuS's specialized density formulation offers improved handling of complex geometries and more precise surface reconstruction. The final meshes in both approaches can be extracted us-

ing Marching Cubes [Lorensen and Cline, 1987].

Instant-NGP [Müller et al., 2022] introduces a breakthrough approach to neural scene representation that fundamentally addresses the computational inefficiency and training time limitations of previous methods like NeRF. Rather than using traditional sinusoidal positional encoding, Instant-NGP employs a novel multi-resolution hash encoding scheme that efficiently maps spatial coordinates to a high-dimensional feature space. This hash-based encoding allows the network to adaptively capture fine details while maintaining a compact representation. By combining this encoding with small a small MLP (3 layers vs 8 for NeRF), Instant-NGP achieves orders of magnitude faster training and inference times compared to traditional NeRF architectures, reducing training times from 1-2 days to a few minutes (5-15 minutes) while preserving high-quality results. The method's versatility extends beyond just radiance fields - it can be applied effectively to various neural graphics primitives including SDFs, occupancy fields, and texture mapping. The hash encoding's ability to efficiently store and access high-frequency information, coupled with its memory-efficient design that grows only log-linearly with resolution, enables real-time applications that were previously impractical with neural rendering approaches. This combination of computational efficiency and rapid convergence, while maintaining high visual fidelity, represented a significant step toward making neural graphics primitives practical for real-world applications.

Instant-NSR [Zhao et al., 2022] and NeuS2 [Wang et al., 2023b] both extend SDF-based neural surface reconstruction to dynamic scenes by combining NeuS's volume rendering framework with hash encoding for efficiency. While Instant-NSR employs a two-stage approach - first reconstructing per-frame surfaces then applying nonrigid deformation for temporal coherence - NeuS2 integrates temporal coherence directly into the learning process through incremental training and a Global Transformation Prediction component. The methods also differ in their technical solutions: Instant-NSR introduces a truncated SDF formulation to address numerical instabilities with hash encoding, while NeuS2 develops lightweight second-order derivative computations optimized for CUDA parallelism. Both approaches achieve efficient surface reconstruction of dynamic scenes, though NeuS2's integrated temporal modeling offers more direct handling of motion and deformation compared to Instant-NSR's post-process approach.

2.2.3 Point-Based Representations

Point-based representations emerged in the early 1990s as an alternative to traditional mesh and volumetric approaches, representing geometry through discrete 3D samples rather than explicit surface connectivity. Westover [Westover, 1991] introduced the foundational 'splatting' algorithm, which derives its name from his mental picture of throwing snowballs at a wall, where each volume sample is projected onto the image plane and spreads its contribution across it, like a snowball flattening on impact. While this

approach demonstrated the potential of simple primitives for rendering complex geometry, its use of isotropic Gaussian kernels resulted in blurred reconstructions. EWA (Elliptical Weighted Average) splatting [Zwicker et al., 2001] significantly improved upon Westover's method by introducing elliptical Gaussian kernels with resampling filters, effectively addressing the blurring and aliasing issues while enabling efficient perspective projection. These early developments laid the groundwork for modern 3D Gaussian Splatting techniques.

2.2.3.1 Static Methods

3D Gaussian Splatting (3DGS) [Kerbl et al., 2023], marked the beginning of a new wave in view synthesis and surface reconstruction techniques. By revisiting the foundational concepts of splatting and elliptical Gaussian kernels, 3DGS combines the computational efficiency of traditional splatting methods with contemporary optimization strategies, establishing a powerful framework for 3D scene representation.
In 3DGS, a scene is modeled as a collection of 3D Gaussians, each characterized by its position, opacity, anisotropic covariance, and spherical harmonic coefficients for view-dependent color. The initial set of 3D Gaussians is derived from sparse point clouds generated by Structure-from-Motion (SfM) techniques [Schonberger and Frahm, 2016]. The optimization process continuously refines Gaussian properties while adaptively controlling point density through dynamic addition and removal of Gaussians. The rendering pipeline employs an efficient tile-based approach: it first computes screen-space bounding boxes for each Gaussian, then sorts them by depth and organizes them into tiles. The final image is generated through volumetric alpha blending, integrating the Gaussians' contributions from front to back. This approach delivers significant improvements in both visual quality and real-time rendering performance compared to NeRF-based methods. However, the unstructured nature of optimized 3D Gaussians makes direct mesh extraction challenging, as these Gaussians typically do not align well with the actual surface geometry of the scene. Several methods have focused specifically on addressing this challenge to enable reliable mesh reconstruction from Gaussian representations.

SuGaR [Guédon and Lepetit, 2023] was the first to introduce an extension to 3DGS that addresses the limitations in scene geometry representation and mesh extraction. The method introduces a novel regularization term that promotes better Gaussian distribution over scene surfaces through a volume density function. To overcome the challenges of 3DGS's densification process, which creates millions of tiny Gaussians for detail preservation, SuGaR develops an efficient algorithm that samples points from depth maps of training viewpoints. This enables rapid Poisson reconstruction of triangle meshes within minutes on a single GPU. The method also includes an optional joint optimization of the mesh and Gaussians through Gaussian splatting rendering, enabling high-quality rendering while supporting traditional mesh editing capabilities. However, despite these innovations, the mesh quality remained inferior to established neural-based surface reconstruction methods.

NeuSG [Chen et al., 2023] and 3DGSR [Lyu et al., 2024] combine NeuS with 3DGS to achieve detailed surface reconstruction, though they employ different integration strategies. NeuSG establishes a two-way interaction where 3D Gaussian Splatting generates dense point clouds to guide the neural implicit surface reconstruction. The method introduces scale and normal regularizers to flatten Gaussians and align them with surface normals. The neural implicit component then uses these regularized point clouds as geometric constraints while providing normal direction guidance back to the Gaussian Splatting component. 3DGSR differentiates itself through a differentiable SDF-to-opacity transformation that directly connects the SDF field to the 3D Gaussians. This enables SDF field optimization through photometric loss, where high-opacity Gaussians guide the SDF output toward zero, effectively pulling them to the surface. Both methods employ Eikonal regularization for proper SDF behavior, but 3DGSR additionally implements specific geometric regularizations, including normal consistency loss and explicit point loss to constrain Gaussian centers to the zero-level set of the SDF. While NeuSG focuses on bidirectional interaction between components, 3DGSR emphasizes direct mathematical linking of the SDF field with Gaussian properties.

Gaussian Opacity Fields (GOF) [Yu et al., 2024] utilize ray-Gaussian intersections to determine opacity contributions. By defining a point's opacity through a unique cross-view minimal opacity calculation, GOF enables direct surface extraction without relying on complex reconstruction techniques like Poisson methods or TSDF fusion. The approach approximates surface normals by calculating intersection planes between rays and Gaussian primitives, allowing for more precise geometry reconstruction. GOF leverages tetrahedra grids and extracts triangle meshes using the Marching Tetrahedra algorithm [Bagley et al., 2016].

GS2Mesh [Wolf et al., 2024] takes a novel approach to surface reconstruction from 3D Gaussian Splatting by leveraging pre-trained stereo matching networks rather than relying on Gaussian properties. Instead of adding geometric constraints during optimization, it generates stereo-calibrated novel views from each training pose by creating a second virtual camera with a horizontal offset. These stereo pairs are fed into a pre-trained stereo network to estimate depth, which is masked to handle occlusions and constrained to a specific range relative to the baseline. Finally, all depth maps are fused using TSDF to create a consistent mesh. By using stereo matching on rendered views, GS2Mesh bypasses the inherent noise in Gaussian positions while benefiting from real-world geometric priors encoded in pre-trained stereo networks.

2DGS [Huang et al., 2024] and Gaussian Surfels (GSurfs) [Dai et al., 2024] both enhance 3DGS by adapting the representation to better align with surface geometry through 2D Gaussian elements, though they implement this concept differently. GSurfs achieves surface alignment by constraining the last scale axis of 3D Gaussians to zero, effectively converting them into surfels, and extracts the final mesh using Poisson reconstruction. In contrast, 2DGS directly defines Gaussians using center positions, 2D scale vectors,

and tangential vectors to determine plane orientation, with the final surface extracted through TSDF fusion of depth maps. The methods also diverge in their intersection computation approaches, with 2DGS employing exact ray-splat intersection calculations while GSurfs opts for Taylor expansion approximation. Each method introduces specific regularization strategies: 2DGS implements depth-normal consistency and depth distortion losses, while GSurfs incorporates monocular normal priors alongside opacity and mask losses.

2.2.3.2 Dynamic Methods

4D-GS [Wu et al., 2024] is a method for real-time dynamic scene rendering that extends 3D Gaussian Splatting to handle dynamic scenes. The approach represents dynamic scenes using a single set of canonical 3D Gaussians combined with a deformation field network that transforms these Gaussians for each timestamp. The deformation field network consists of two key components: a spatial-temporal structure encoder that uses multi-resolution HexPlane voxel grids [Cao and Johnson, 2023] to efficiently encode both spatial and temporal features of nearby Gaussians, and a lightweight multi-head decoder that predicts position, rotation, and scaling deformations for each Gaussian at novel timestamps. Rather than storing separate Gaussians for each frame, 4D-GS maintains just one set of canonical Gaussians and learns to deform them, significantly reducing memory requirements while enabling real-time rendering.

RealTime4DGS [Yang et al., 2024] extends 3DGS to dynamic scenes by introducing a unified 4D Gaussian representation that treats time and space coordinates together. Instead of modeling time separately, it formulates the covariance matrix as a 4D ellipsoid parameterized by scaling and rotation matrices, where the 4D rotation is represented using two quaternions for left and right isotropic rotations. The mean is represented in 4D space-time with coordinates. To handle view-dependent color changes over time, the method introduces 4D spherindrical harmonics (4DSH), which combine traditional spherical harmonics with Fourier series to model temporal appearance variations. The training process employs batch sampling in time to prevent temporal flickering artifacts, and extends the density control mechanism to consider both spatial and temporal gradients. During Gaussian splitting, RealTime4DGS performs joint spatial-temporal position sampling to better handle regions prone to over-reconstruction.

4K4D [Xu et al., 2023] was proposed as an alternative to Gaussian Splatting for real-time view synthesis of dynamic 3D scenes at 4K resolution. Instead of Gaussians, it employs a 4D point cloud representation with hardware rasterization and network precomputation to achieve exceptional rendering speeds. The approach uses a 4D feature grid to regularize point clouds and combines discrete image blending with continuous spherical harmonics for appearance modeling. Its differentiable depth peeling algorithm exploits hardware acceleration while remaining trainable from RGB videos. 4K4D delivers state-of-the-art visual quality with significantly lower storage requirements than alternatives, making it suitable for immersive free-viewpoint applications. However, it

requires substantial training time to achieve high-quality renderings, typically needing 800,000 iterations for a 200-frame sequence, which takes over 24 hours on a single RTX 4090 GPU.

Spacetime Gaussians (STG) [Li et al., 2024] tackles dynamic scene modeling by introducing temporal components to 3DGS. The method represents temporal opacity using a radial basis function centered at a specific timestamp, allowing Gaussians to smoothly appear and disappear over time. Motion is modeled through polynomial trajectories relative to each Gaussian's peak temporal visibility. For view- and time-dependent appearance, each Gaussian stores a compact 9-dimensional feature vector combining base color, view-dependent, and temporal information, which is processed by a lightweight MLP to produce the final RGB colors. The approach uniquely handles sparse regions through guided sampling, where new Gaussians are strategically placed along rays corresponding to high-error image regions, using coarse depth information to constrain sampling ranges. Unlike methods that treat time as an additional dimension, STG keeps the spatial and temporal aspects separate, with time-dependent functions controlling position, rotation, and opacity while maintaining time-independent spatial scaling.

4DRotorGS [Duan et al., 2024] extends 3D Gaussian Splatting to dynamic scenes through a rotor-based 4D Gaussian representation. Its key innovation lies in using an 8-component rotor representation that separates 3D spatial rotation from spatio-temporal rotation, enabling natural handling of both static and dynamic elements. When projecting 4D Gaussians into 3D space, the method incorporates temporal decay for Gaussian visibility control and a motion term for linear movement approximation. The approach is stabilized through two regularization terms: an entropy loss that helps condense points and filter noise, and a 4D consistency loss that enforces motion similarity among spatially and temporally related points in 4D space.

3DGStream [Sun et al., 2024] enables free-viewpoint video streaming by adapting 3D Gaussian Splatting to dynamic scenes through a two-stage pipeline. At its core is the Neural Transformation Cache (NTC), which uses multi-resolution hash encoding and a shallow fully-fused MLP to efficiently model how 3D Gaussians transform between frames. The hash encoding system maps 3D positions to feature vectors through multi-resolution voxel grids, naturally focusing computation on dynamic regions through hash collisions while maintaining O(1) complexity. In the first stage, the NTC transforms existing Gaussians from the previous frame, while the second stage handles emerging objects through an adaptive Gaussian addition strategy that identifies high-gradient regions and strategically spawns new Gaussians there, inheriting properties from nearby existing ones. The system maintains efficiency through quantity control, adding Gaussians where needed while pruning those with low opacity. Instead of storing complete Gaussian sets for each frame, the method only requires the NTC parameters and frame-specific additional Gaussians, enabling efficient streaming without offline processing of complete sequences.

MaGS (Mesh-adsorbed Gaussian Splatting) [Ma et al., 2024], presents a unified framework for 3D reconstruction and simulation from monocular video. Its key innovation lies in enabling 3D Gaussians to "roam" near mesh surfaces instead of being rigidly fixed, allowing for a flexible yet structured representation. The framework incorporates three neural networks: the first extracts pose information from mesh vertices, the second learns motion priors for mesh deformation, and the third models the relative displacement between the mesh and the Gaussians. These MLPs, along with the mesh and Gaussians, are jointly optimized. Additionally, the framework supports deformation and physics-based simulation, facilitating both reconstruction and realistic simulation.

Vidu4d [Wang et al., 2024] reconstructs 3D surfaces from generated videos. It introduces an extended version of GSurfs to dynamic scenes by adding motion capabilities trhough non-rigid warping fields. The method works in two stages: first initializing warping fields, then optimizing the Dynamic Gaussian Surfels. The key innovation is how it handles motion - instead of creating new Gaussians for each frame, it warps existing ones through learned transformation functions. It ensures accurate surface alignment using normal consistency during warping, and it refines rotation and scaling parameters through a dual branch structure to reduce flickering artifacts. When combined with the Vidu video generation model [Bao et al., 2024], the system can create high-quality 4D content from text prompts.

DG-Mesh [Liu et al., 2024] presents a framework for reconstructing dynamic 3D meshes with correspondence from monocular inputs. The method maintains a set of canonical 3D Gaussians that are transformed across different timesteps using deformation networks. For mesh reconstruction, the method converts oriented Gaussians to an indicator function using a differentiable Poisson solver, then applies differentiable Marching Cubes to generate watertight meshes. A key innovation is the "Gaussian-Mesh Anchoring" technique that creates one-to-one correspondence between mesh faces and Gaussians by merging or creating Gaussians based on mesh face proximity. The method also employs "Cycle-Consistent Deformation" with forward and backward networks to maintain point correspondence across time. Training combines multiple objectives including rendering losses from both mesh rasterization and Gaussian splatting, along with anchoring and cycle-consistency losses.

MAGS [Guo et al., 2024] and GaussianFlow [Gao et al., 2024] propose using optical flow supervision for improving dynamic reconstruction. While MAGS establishes correspondence through "soft selection" of nearest Gaussians around each pixel location, GaussianFlow derives an analytical solution by preserving the Mahalanobis distance of pixels to deforming Gaussians. Both methods then compare their computed screen-space motion with optical flow priors from pre-trained networks. The key insight in both approaches is establishing a differentiable connection between 3D Gaussian motion and 2D pixel velocities, allowing direct supervision of temporal dynamics without relying solely on

photometric losses.

2.3 Summary and Research Gaps

To summarize, the field of 3D reconstruction has evolved through several distinct paradigms. Classical MVS methods initially dominated the field, using either depth-based approaches for point cloud reconstruction through correspondence matching, or voxel-based methods for occupancy estimation through photometric consistency. While these methods established fundamental principles for 3D reconstruction, they faced significant challenges with textureless regions, fine geometric details, and view-dependent effects. The introduction of learning-based MVS approaches like MVSNet attempted to address these limitations through deep neural networks, but still struggled with the fundamental constraints of correspondence-based reconstruction.

The evolution of neural surface reconstruction began with NeRF-based approaches. While NeRF itself struggled with surface extraction due to its volume density representation, subsequent methods like UniSurf, VolSDF, and NeuS introduced various techniques to better represent surfaces. These approaches progressively improved surface quality by incorporating occupancy fields, signed distance functions, and specialized density formulations. Instant-NGP addressed the computational efficiency challenges through hash encoding, significantly reducing training times while maintaining quality. Building upon NeuS's surface reconstruction capabilities, both Instant-NSR and NeuS2 adapted the SDF-based formulation with hash encoding to handle dynamic scenes. While Instant-NSR employed a two-stage approach, first reconstructing surfaces per frame and then applying nonrigid deformation for temporal coherence, NeuS2 incorporated temporal coherence directly into the optimization through incremental training and global transformation prediction.

Several approaches have emerged to extract high-quality meshes from 3D Gaussian Splatting. SuGaR introduces surface-aligned regularization and efficient level set sampling for Poisson reconstruction. NeuSG and 3DGSR combine neural implicit functions with Gaussian Splatting, using complementary constraints to improve surface alignment. More recent methods like GOF, GS2Mesh, 2DGS, and Gaussian Surfels take different approaches: GOF utilizes ray-Gaussian intersections, GS2Mesh leverages pre-trained stereo networks, while both 2DGS and GSurfs constrain Gaussians to lie on surfaces through flat 2D representations.

Current dynamic scene reconstruction methods can be categorized by their key technical approaches. Several methods extend 3DGS by incorporating temporal components: 4D-GS and 3DGStream use deformation fields to transform canonical Gaussians, while RealTime4DGS and 4DRotorGS treat time as an additional dimension in their representations. Others, like Spacetime Gaussians, model temporal behavior through opacity functions and polynomial trajectories. 4K4D takes a different approach by using a point

cloud representation built on a 4D feature grid, combining hardware rasterization with a hybrid appearance model to achieve real-time rendering speeds while maintaining high visual fidelity, though at the cost of longer training times.

While extensive research has been conducted independently in the domains of surface reconstruction and dynamic view synthesis, the intersection of these fields − dynamic surface reconstruction − remains relatively unexplored. Recent approaches such as MaGS, Vidu4D, and DG-Mesh have made notable contributions to dynamic surface reconstruction using Gaussian Splatting, yet their applications are predominantly confined to monocular inputs and specialized use cases. MaGS focuses on physics-based simulation and mesh deformation, while Vidu4D specializes in text-to-4D content generation. DG-Mesh, while achieving mesh consistency through Gaussian-Mesh anchoring and cycle-consistent deformation, similarly operates within the constraints of monocular input scenarios.

In terms of multi-view approaches, NeuS2 and Instant-NSR are the primary methods that align with this thesis's objectives, specifically addressing dynamic surface reconstruction from multi-view images. However, as they are fundamentally based on neural implicit functions, there exists a significant research gap in implementing dynamic surface reconstruction from multi-view input within the 3D Gaussian Splatting paradigm.

The core objective of this thesis is to advance the field by developing a novel approach for reconstructing dynamic three-dimensional surfaces from multi-view video inputs. This approach aims to achieve two critical benchmarks: generating a sequence of high-quality meshes that accurately represent the underlying geometry at each time-frame, and maintaining computational efficiency through reduced training times. This work will specifically explore how the 3D Gaussian Splatting framework can be adapted and extended to accomplish these goals, addressing a crucial gap in the current research landscape.

3 Methodology

3.1 Preliminaries

This section presents the fundamental concepts of 3D Gaussian Splatting, which forms the theoretical foundation for the methodology developed in this thesis. The following subsections provide a detailed examination of the mathematical principles and key components that underpin this approach.

3.1.1 3D Gaussian Splatting

A 3D gaussian i is characterized by its position x_i, covariance matrix Σ_i, opacity o_i and spherical harmonics (SH) coefficients h_i. The covariance matrix defines the shape and spread of the Gaussian, while the opacity controls its transparency. The SH coefficients allow for efficient representation of the color across different viewing angles. The transparency α_i of a gaussian (which will in this thesis also be referred to as the "alpha-value") at any spatial position x can be computed as:

$$\alpha_i(x) = o_i \exp\left(-\frac{1}{2}(x - x_i)^\top \Sigma_i^{-1}(x - x_i)\right) \tag{3.1}$$

A key contribution of 3DGS is the representation of the covariance matrix in a form that is conducive to optimization. The covariance matrix Σ_i must be positive semi-definite to maintain a valid physical interpretation. However, Stochastic Gradient Descent (SGD) optimization does not inherently guarantee this property. To ensure the covariance remains physically valid during optimization, it is parameterized as the product of a rotation matrix R_i and a scaling matrix S_i.

$$\Sigma_i = R_i S_i S_i^\top R_i^\top, \tag{3.2}$$

where R_i is parametrized by a quaternion q_i, representing the orientation of the gaussian and S_i is a diagonal matrix parametrized by a 3D vector $\mathbf{s}_i$, representing the gaussian's scale along each axis.

Analogous to NeRF, the rendering process uses α-blending to generate an image from the set of Gaussians. For each pixel, a ray is cast from the camera through the scene. The ray intersects a sequence of Gaussians, ordered by depth along its path. The final accumulated color C at each pixel is computed as a weighted sum of the contributions from these N ordered Gaussians, where each contribution is influenced by the Gaussian's transparency α_i at that intersection, transmittance T_i, and color c_i, where the color c_i

is computed from the Gaussian's SH coefficients h_i. The expression for the accumulated color is given by:

$$C = \sum_{i=1}^{N} T_i \alpha_i c_i \tag{3.3}$$

Transmittance refers to the fraction of light that passes through all preceding Gaussians along the ray without being absorbed. Mathematically, it represents the accumulated transparency along a viewing ray after considering all Gaussians that the ray has passed through and is given by:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_i) \tag{3.4}$$

The position, opacity, covariance matrix, and SH coefficients of each Gaussian are optimized iteratively by rendering the scene at views for which a ground truth image exists and comparing the output to the ground truth views. This optimization is performed using Stochastic Gradient Descent on the following loss function:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{D-SSIM}} \tag{3.5}$$

Here, $\mathcal{L}_1$ represents the standard L1 loss, which computes the absolute differences between the rendered image and the ground truth at each pixel, promoting pixel-wise accuracy. The term $\mathcal{L}_{\text{D-SSIM}}$ corresponds to the Structural Dissimilarity Index Measure [Baker et al., 2023]. The hyperparameter ($\lambda \in [0, 1]$) controls the balance between these two loss terms, allowing for adjustment of the relative importance of pixel-wise accuracy versus structural similarity in the optimization process.

While SGD is effective for adjusting existing Gaussians, it struggles in regions that lack points or have an excess of them. The authors of 3DGS found that both under-reconstruction and over-reconstruction result in large positional gradients. Following this observation, adaptive densification is introduced to control the number of Gaussians based on a threshold. Every 100 iterations, when positional changes exceed the threshold, the system densifies the representation in two ways:

- In areas with insufficient geometry (under-reconstruction), small Gaussians are cloned and displaced along the direction of the positional gradient.
- In regions with high variance (over-reconstruction), large Gaussians are split into two smaller ones, with their positions sampled from the original Gaussian's probability distribution.

Figure 3.1 illustrates this adaptive densification process. To further regulate the total number of Gaussians, the system selectively removes those that have become nearly transparent (with transparency value α_i below a minimum threshold ϵ_α) or those that have grown excessively large in the 3D scene.

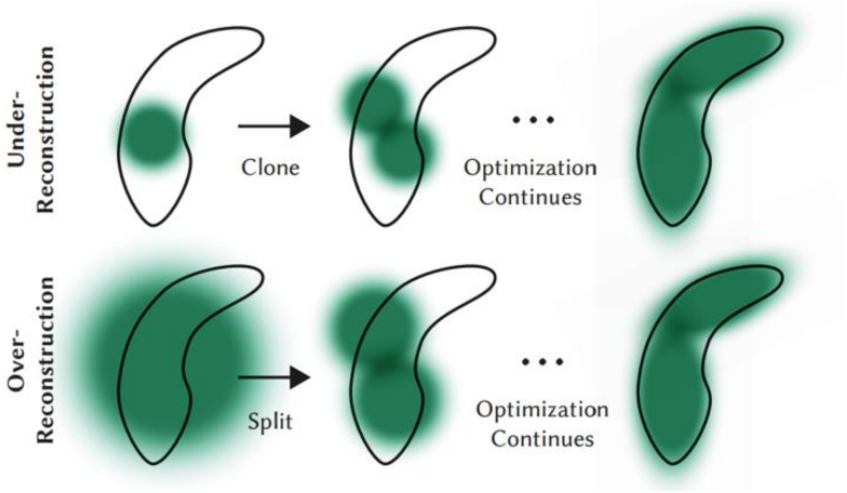

Figure 3.1: Visualization of the densification process: small gaussians are cloned (top), while large gaussians are split into smaller ones (bottom). Figure from [Kerbl et al., 2023].

A particularly notable contribution of 3DGS is its rasterization pipeline, which serves as the cornerstone of the method's remarkable speed, enabling real-time performance. This efficiency is achieved through a well-designed tile-based approach for fast and differentiable rendering. The process starts by projecting 3D Gaussians into 2D screen space, dividing the image plane into 16×16 pixel tiles. To enhance performance, the method employs view frustum culling* and leverages confidence intervals to identify Gaussians overlapping with each tile. The system then sorts the Gaussians by depth and tile ID using GPU-based radix sort [Merrill and Grimshaw, 2011], enabling efficient α-blending without requiring per-pixel sorting. This approach significantly reduces computational overhead compared to traditional NeRF rendering, as it eliminates the need for repeated ray-primitive intersection calculations and MLP evaluations. The method maintains memory efficiency by reusing forward pass data during gradient computation in the backward pass, making it suitable for scenes of arbitrary complexity without requiring hyperparameter tuning.

3.2 Proposed Method

This thesis proposes the method Spacetime Surfels (STS), which leverages two state-of-the-art approaches: Gaussian Surfels (GSurfs) [Dai et al., 2024] for surface representation and Spacetime Gaussians (STG) [Li et al., 2024] for dynamic view-synthesis. The selection of these methods was motivated by their demonstrated superiority in benchmark evaluations and substantial community endorsement, as evidenced by their repository

*View frustum culling is a common technique in 3D graphics that improves rendering efficiency by excluding objects outside the visible area, i.e. outside the camera's field of view

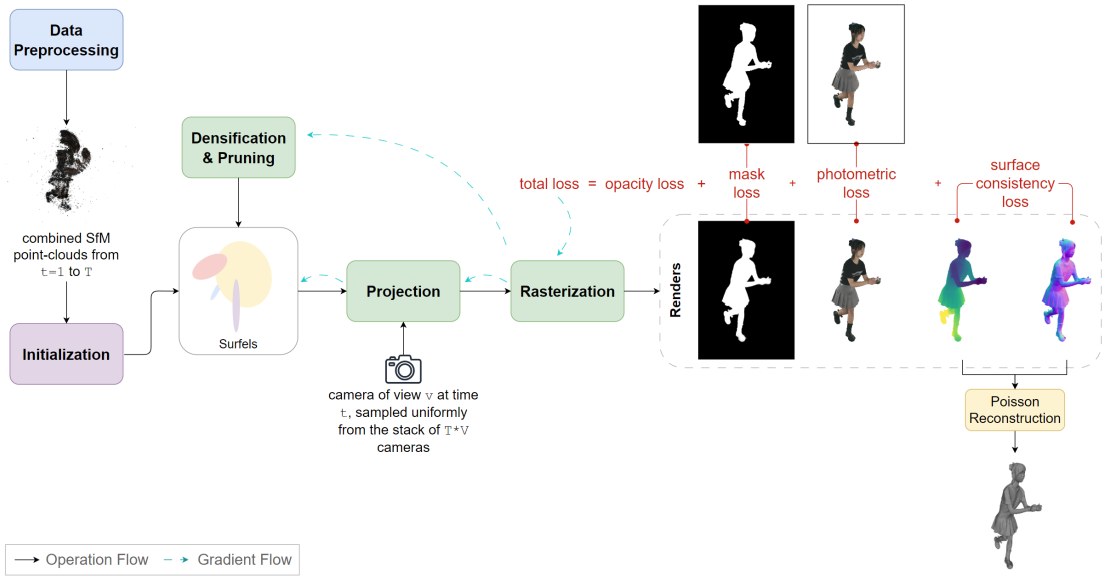

Figure 3.2: Pipeline architecture of Spacetime Surfels. The system processes multi-view video input through surfel-based optimization with multiple loss terms, followed by final mesh reconstruction. Solid black arrows indicate operation flow, while dashed cyan arrows show gradient propagation paths.

star counts in open-source platforms. The inherent compatibility between these methods, combined with their respective strengths, provides a solid foundation for the proposed approach.

Figure 3.2 presents the complete pipeline of the proposed approach. The system begins with a preprocessing stage, which will be explained in more detail in Section 3.2.5.2. This step converts multi-view video frames into a consolidated point cloud obtained using Structure-from-Motion (SfM), providing the foundation for subsequent pipeline stages. Each point in the point cloud is then initialized and processed into a surfel. These surfels undergo gradient-driven densification and pruning, followed by projection according to sampled camera views and rasterization. The optimization process is guided by four key loss components: opacity loss, mask loss, photometric loss, and surface consistency loss. Once optimization completes, the depth and normal maps from the optimized surfels are fused to generate the final 3D mesh using Screened Poisson surface reconstruction [Kazhdan and Hoppe, 2013].

The following sections detail the key components of the proposed method. Section 3.2.1 introduces the fundamental properties of Spacetime Surfels and their mathematical formulation, including temporal opacity modeling and polynomial motion and rotation representation. Section 3.2.2 describes the rendering approach, explaining how both color and geometric attributes are computed through alpha-blending. Section 3.2.3 presents

the optimization framework, detailing the various loss terms that guide the learning process. Finally, Section 3.2.4 outlines the mesh extraction process, while Section 3.2.5 provides comprehensive implementation details, covering aspects from initialization and sampling strategies to parameter choices and optimization procedures.

3.2.1 Spacetime Surfels

Spacetime Surfels serve as fundamental elements for representing dynamic 3D scenes. Each Spacetime Surfel can be understood as a time-varying probabilistic primitive that captures both spatial and temporal characteristics of the scene.

3.2.1.1 Properties

A Spacetime Surfel i is characterized by the following properties:

- Position: $x_i(t) \in \mathbf{R}^3$ at time t, representing the center of the surfel in 3D space
- Rotation: $q_i(t) \in \mathbf{S}^{3\dagger}$ at time t, representing orientation as a unit quaternion
- Opacity: $o_i(t) \in [0, 1]$, controlling the surfel's visibility
- Scale: $s_i \in \mathbf{R}^3$, defining the spatial extent
- Spherical Harmonics (SH) coefficients: $h_i \in \mathbf{R}^{16}$, for view-dependent color, where 16 is the number of spherical harmonics coefficients for degree 3

Following GSurfs, the scale vector is constrained along the third axis: $s_i = [s_{x_i}, s_{y_i}, 0]$, yielding a flat surfel representation. The orientation quaternion $q_i(t)$ maps to a rotation matrix through $R_i(t) = R(q_i(t)) \in SO(3)^{\ddagger}$, while the scale vector is expressed as a diagonal matrix $S_i = \text{diag}(s_i) \in \mathbf{R}^{3\times3}$. These components combine to form the positive semi-definite covariance matrix:

$$\Sigma_i(t) = R_i(t) S_i S_i^\top R_i^\top(t) \in \mathbf{R}^{3\times3} \tag{3.6}$$

This construction ensures the surfel maintains a locally flat geometry while allowing arbitrary orientations in 3D space and time. The transparency α_i of a surfel i at any position x and time t can then be computed as:

$$\alpha_i(x, t) = o_i(t) \exp\left\{-\frac{1}{2}(x - x_i(t))^T \Sigma_i(t)^{-1}(x - x_i(t))\right\} \tag{3.7}$$

[†] $\mathbf{S}^3$ is the unit 3-sphere consisting of all unit quaternions with norm 1

[‡] $SO(3)$ is the special orthogonal group in three dimensions representing all 3D rotations with determinant 1

3.2.1.2 Temporal Opacity Model

Following STG, the time-varying opacity factor $o_i(t)$ is modeled using a temporal radial basis function (tRBF):

$$o_i(t) = \sigma_i \exp\left(-\beta_i |t - \mu_i|^2\right) \tag{3.8}$$

where:

- $\sigma_i \in [0, 1]$ represents the base opacity, controlling the maximum visibility of the surfel
- $\beta_i \in \mathbf{R}^+$ defines the temporal scale parameter, inversely related to the surfel's temporal duration - smaller values result in longer visibility periods
- $\mu_i \in \mathbf{R}$ specifies the temporal center, marking the time at which the surfel reaches its peak visibility σ_i

This formulation enables precise control over the temporal evolution of each surfel's visibility. The exponential decay governed by β_i creates a smooth transition between visibility states, while the temporal center μ_i determines when the surfel is most prominent in the scene. The visibility of the surfel effectively diminishes as $|t - \mu_i|$ increases, with the rate of decay controlled by β_i. This allows surfels to naturally appear and disappear over time, facilitating the representation of emerging and disappearing elements in a dynamic scene.

3.2.1.3 Polynomial Motion and Rotation Model

The spatiotemporal evolution of position and rotation is modeled through polynomial functions centered at each surfel's temporal peak μ_i. For the position,

$$x_i(t) = \sum_{k=0}^{D_p} m_{i,k}(t - \mu_i)^k \tag{3.9}$$

where:

- $m_{i,k} \in \mathbf{R}^3$ represents the position polynomial coefficients
- D_p denotes the polynomial degree for position

Similarly, the rotation is parameterized using real-valued quaternions:

$$\hat{q}_i(t) = \sum_{k=0}^{D_q} r_{i,k}(t - \mu_i)^k \tag{3.10}$$

which is then normalized to ensure a valid unit quaternion:

$$q_i(t) = \frac{\hat{q}_i(t)}{\|\hat{q}_i(t)\|} \tag{3.11}$$

where:

- $r_{i,k} \in \mathbf{R}^4$ represents the quaternion polynomial coefficients
- D_q denotes the polynomial degree for rotation

This representation follows the design choices of STG, adopting polynomial degrees of $D_p = 3$ for position and $D_q = 1$ for rotation. This choice was motivated by STG's analysis showing that cubic polynomials for position provide the optimal balance between motion complexity and computational cost, while linear interpolation sufficiently captures rotational dynamics. The implementation maintains a time-independent scale matrix S_i, as STG demonstrated that allowing temporal variations in scale produced no meaningful improvements in rendering quality.

It is important to note that the polynomial functions used to model dynamic motion and rotation cannot adequately capture complex object movements over extended periods. This representation does not model the surfel's trajectory across the entire sequence as though it were precisely adhering to the object's surface. While the surfels are present throughout the entire sequence, each surfel becomes visible only for a specific time-range, typically spanning from 2-3 to 8-10 frames. Through optimization, surfels automatically restrict their influence to shorter time intervals where they can most accurately approximate local surface dynamics.

3.2.2 Rendering

The rendering approach follows the same method as the original 3DGS technique reviewed in Section 3.1. Specifically, the color at pixel p and timestep t is computed by alpha-blending the color of the surfels that the ray intersects:

$$C_p(t) = \sum_{i=1}^{N} T_i(t)\alpha_i(t)c_i \tag{3.12}$$

where:

- N is the number of surfels that intersect with the ray shooting from pixel p, ordered by increasing depth (front-to-back)
- $\alpha_i(t)$ is the alpha value (transparency) of the surfel at the point where the ray from pixel p intersects it, computed according to Eq. 3.7
- c_i is the color of the surfel, computed from the spherical harmonics coefficients h_i
- $T_i(t) = \prod_{j=1}^{i-1}(1 - \alpha_j(t))$ is the transmittance, representing accumulated transparency

For simplicity, the spherical harmonics coefficients are kept time-independent, as surfels have short temporal duration and their color is assumed to remain relatively constant throughout their lifespan.

Depth and normal maps can be rendered similarly to color maps using α-blending, which will be used for the surface loss introduced in Section 3.2.3.4. In the following equations, the timestep t is omitted for clarity but is implied for all parameters.

For normal map rendering, the normal n_i of surfel i is given by the third column of its rotation matrix, representing the axis along which the surfel's scale is set to zero: $n_i = R_i[:, 2]$, where [.] follows Python array notation. The predicted normal map $\tilde{N}$ is computed through α-blending:

$$\tilde{N} = \frac{1}{1 - T_{n+1}} \sum_{i=0}^{n} T_i \alpha_i n_i \tag{3.13}$$

For depth map rendering, depth $d_i(u)$ needs to be computed for each surfel i at each pixel u. This represents the intersection of the ray cast through pixel u with surfel i. Following GSurfs, this computation can be simplified using a first-order Taylor expansion:

$$d_i(u) = d_{x_i} + (W n_i)^T J_{p_r}^{-1} (u - u_i) \tag{3.14}$$

where:

- d_{x_i} is the depth of the center of surfel i
- u_i is the 2D projection of the surfel center onto the image plane
- n_i is the normal vector of surfel i in world space
- W transforms the normal from world space to camera space
- $J_{p_r}^{-1}$ is the Jacobian of inverse mapping a pixel in the image space onto the surfel's tangent plane as in [Zwicker et al., 2001]
- $(u - u_i)$ is the offset from the projected surfel center in image coordinates

This formula expresses how the depth varies as one moves away from the surfel's center in image space. The rendered depth map $\tilde{D}$ is then computed through α-blending the depths of all surfels along the ray:

$$\tilde{D} = \frac{1}{1 - T_{n+1}} \sum_{i=0}^{n} T_i \alpha_i d_i \tag{3.15}$$

For both normal and depth maps, the term $\frac{1}{1-T_{n+1}}$ normalizes the blending weights $T_i \alpha_i$ to ensure they sum to 1. This weighting ensures that surfels are weighted according to their visibility - surfels that are more occluded (with lower accumulated transmittance) contribute less to the final depth and normal values, while more visible surfels have a stronger influence on the result.

Parameter	Description
μ_i	Center of the temporal Radial Basis Function (tRBF)
β_i	Scale parameter of the tRBF, controlling temporal extent
σ_i	Base opacity of the surfel
$r_{i,k}$	Rotation parameters governing surfel orientation
$m_{i,k}$	Motion parameters describing surfel trajectory
s_i	Scale factor determining surfel size
h_i	Spherical harmonics coefficients for color representation

Table 3.1: Trainable parameters optimized for each surfel i during the optimization process.

3.2.3 Optimization

The proposed method optimizes parameters for each surfel i through gradient descent using the Adam optimizer [Kingma and Ba, 2017]. These parameters, summarized in Table 3.1, define the spatial, temporal, and appearance characteristics of the surfels. To optimize these parameters, the proposed method minimizes a composite loss function:

$$\mathcal{L} = \mathcal{L}_p + \lambda_o \mathcal{L}_o + \lambda_m \mathcal{L}_m + \lambda_s \mathcal{L}_s \tag{3.16}$$

where $\mathcal{L}_p$ represents the photometric loss, $\mathcal{L}_o$ the opacity loss, $\mathcal{L}_m$ the mask loss, and $\mathcal{L}_s$ the surface loss. The parameters λ_o, λ_m, and λ_s are weighting coefficients that balance the contribution of each loss term to the overall optimization objective. Each of these loss terms are explained in detail in the following sections.

3.2.3.1 Photometric Loss

The photometric loss $\mathcal{L}_p$ follows the formulation from the original 3DGS approach:

$$\mathcal{L}_p = (1 - \lambda_p)\mathcal{L}_1 + \lambda_p \mathcal{L}_{\text{D-SSIM}} \tag{3.17}$$

where $\mathcal{L}_1$ represents the mean absolute difference between the rendered images and the ground truth images, $\mathcal{L}_{\text{D-SSIM}}$ is the Dissimilarity Structural Similarity Index Measure and λ_p is a weighting factor balancing the contributions of the $\mathcal{L}_1$ and $\mathcal{L}_{\text{D-SSIM}}$ terms.

3.2.3.2 Opacity Loss

The opacity loss $\mathcal{L}_o$ plays a crucial role in achieving sharp surface boundaries by enforcing a binary state for each surfel, either as part of the object's surface or empty space:

$$\mathcal{L}_o = \exp(-(\sigma - 0.5)^2/\gamma) \tag{3.18}$$

where σ represents the base opacity value of each Gaussian surfel and γ (set to 0.05) controls the sharpness of the bimodal distribution. This formulation pushes opacity values toward either 0 or 1, allowing subsequent pruning of transparent surfels ($\sigma \approx 0$) while maintaining those on the object's surface ($\sigma \approx 1$). The exponential term ensures smooth gradients during optimization while preserving this binary behavior, preventing the accumulation of semi-transparent surfels that could degrade surface quality.

3.2.3.3 Mask Loss

Since foreground masks are available from the input data, an additional mask loss $\mathcal{L}_m$ is incorporated to enhance reconstruction quality:

$$\mathcal{L}_m = \text{BCE}(\sum_{i=0}^{n} T_i \alpha_i, M) \tag{3.19}$$

where BCE represents binary cross-entropy loss, $T_i\alpha_i$ denotes the accumulated alpha values from Gaussian splatting, i.e. the rendered opacity, and M is the ground truth foreground mask.

3.2.3.4 Surface Loss

Following GSurfs, a surface loss $\mathcal{L}_s$ is introduced to ensure alignment between the rendered depth and the rendered normal map:

$$\mathcal{L}_s = 1 - \tilde{N} \cdot \boldsymbol{N}(\boldsymbol{V}(\tilde{D})), \tag{3.20}$$

where $\tilde{N}$ and $\tilde{D}$ are the rendered normal and depth maps derived in section 3.2.2, $\boldsymbol{V}$ converts each pixel and its associated depth $\tilde{D}$ into a 3D point, and $\boldsymbol{N}$ computes the normal vector of this point from its neighboring points using the cross product. This loss essentially checks how well the rendered normal aligns with the normal computed from the rendered depth. This term is crucial for maintaining geometric consistency.

The original GSurfs introduces a normal supervision loss that leverages monocularly predicted normals from methods like OmniData [Eftekhar et al., 2021]. However, this additional supervision was deliberately excluded in the Spacetime Surfels implementation, as experimental results revealed that it provides minimal quality improvement while significantly increasing memory requirements due to the need to load normal maps for every view at every timestep alongside the ground truth images and masks. This design choice allows the method to maintain computational efficiency while achieving high-quality results through the careful balance of the remaining loss terms.

3.2.4 Meshing

Following the approach introduced in GSurfs, the conversion of trained surfels into a triangulated mesh employs a two-step process. First, normal and depth maps are rendered

from multiple viewpoints around the scene and fused together. These fused maps are then processed using Screened Poisson Surface Reconstruction [Kazhdan and Hoppe, 2013], with a tree depth parameter of 9, to generate a coherent mesh representation.

A notable challenge addressed in GSurfs involves surfels extending beyond their actual surface boundaries. In such cases, these surfels incorrectly contribute to background depth calculations, resulting in background surfaces appearing closer to the camera than their true position, which results in incorrect depth maps and in turn introduces noise in the final mesh. To address this issue, GSurfs implements *Volumetric Cutting*, a method that identifies and removes problematic areas prior to meshing. The approach constructs a 512^3 voxel grid within the surfel points' bounding box. Through traversal of all grid voxels, the weighted opacity of surfels is accumulated, and voxels with accumulated opacity below a threshold ($\lambda = 1$) are pruned, along with any 3D points contained within them. This same volumetric cutting technique is adopted in the present method and adapted to the dynamic representation to ensure clean mesh reconstruction.

Adapting this meshing procedure to the dynamic representation requires filtering surfels at the specific timestamp of interest. As outlined at the end of section 3.2.1.3, surfels remain mostly transparent throughout their lifespan and are only visible for approximately 2-10 frames. Therefore, during the meshing process, only those surfels that actively contribute to the specific timestamp being reconstructed must be considered. This filtering is accomplished by evaluating their opacity at time t according to the tRBF (Equ. 3.8), where surfels are considered "active" or visible when their opacity exceeds a threshold value. For this thesis, a default threshold of 0 was selected, meaning only surfels with temporal opacity strictly greater than 0 are considered when creating the mesh.

3.2.5 Implementation Details

3.2.5.1 Sampling

The camera sampling strategy differs from STG's approach to ensure comprehensive coverage. While STG implements random sampling across all views and timestamps, this method risks overlooking certain camera positions or timesteps. To guarantee uniform sampling frequency, Spacetime Surfels' implementation employs a stack-based sampling mechanism. The process begins with a stack containing all cameras from every view and timestep in the sequence. During each iteration, a camera is randomly selected from the stack and removed once used. When the stack is depleted, it is replenished with all space-time camera positions, and the random sampling process continues. This systematic approach maintains balanced view sampling while preserving the benefits of randomization.

3.2.5.2 Initialization

The initialization of surfel parameters begins with the generation of sparse point clouds from the multi-view image sequence, as illustrated in Figure 3.3. This process utilizes a SfM technique, more specifically COLMAP [Schonberger and Frahm, 2016], to process the multi-view images at each temporal frame. The algorithm generates separate point clouds for each timestep in the sequence, which are subsequently merged into a unified point cloud. Each point in this consolidated cloud serves as an initialization point for a Gaussian surfel primitive, with its spatial coordinates, RGB values and timestep directly mapped to the Gaussian's initial position, color parameters and tRBF center respectively.

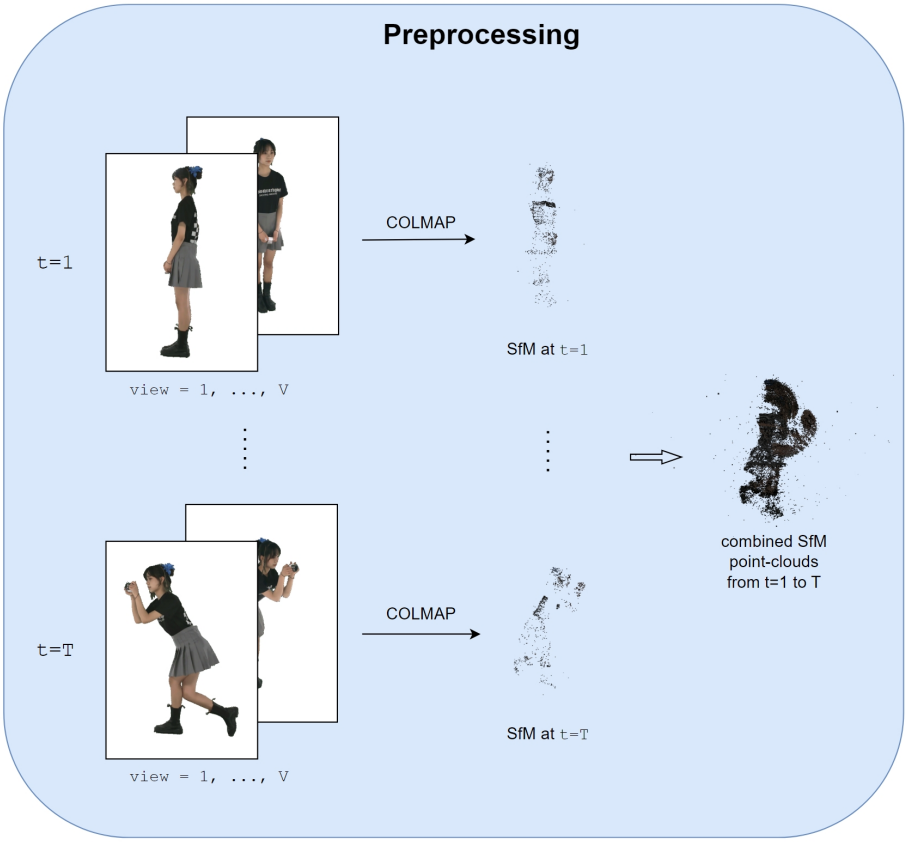

Figure 3.3: Preprocessing pipeline overview: The system processes multi-view footage in a sequential manner. At each timestep $\mathtt{t} \in [1, \mathtt{T}]$, where $\mathtt{T}$ represents the total number of timesteps, all available camera views $\mathtt{1, \ldots, V}$ (where $\mathtt{V}$ is the number of views) are processed through COLMAP to generate a sparse Structure-from-Motion (SfM) point-cloud representation. These individual timestep point-clouds are subsequently merged into a unified point-cloud structure encompassing the entire temporal sequence from $\mathtt{t=1}$ to $\mathtt{T}$.

Note that instead of running COLMAP's full structure-from-motion pipeline, the known camera poses from the dataset are directly provided to COLMAP to compute the sparse reconstructions more precisely at each temporal frame.

3.2.5.3 Training Strategy

Due to the random sampling strategy used during training, all frames must be simultaneously available in memory to enable arbitrary view and timestamp selection. This memory requirement makes it impractical to load all frames of the sequence at once. Instead, the training processes the data in sequential rounds of 50 frames each, striking a balance between having sufficient frames for effective training while maintaining feasible memory consumption. While longer sequences could theoretically be accommodated by storing images and masks on CPU memory and transferring them to GPU as needed, this approach would significantly increase training time due to the frequent data transfer overhead between CPU and GPU memory. Therefore, following the established method STG, the sequential processing approach offers the best compromise between memory efficiency and training speed.

3.2.5.4 Parametrization

The temporal opacity $o_i(t)$ is defined as $o_i(t) = \sigma_i \exp\left(-\beta_i |t - \mu_i|^2\right)$, where a reparameterization trick is used such that the tRBF output remains bounded between 0 and 1. Through the parameterization $\beta_i = e^{-2\gamma_i}$ where γ_i is a learnable parameter, the exponential term can be rewritten as $\exp\left(-e^{-2\gamma_i}(t - \mu_i)^2\right)$. This parameterization, adopted from STG, ensures that the temporal modulation of the base opacity σ_i is properly bounded.

3.2.5.5 Pruning & Densification

The densification process operates on fixed intervals for pruning and densification ($N = 100$ iterations), following the approach established in 3DGS with an additional pruning criterion from GSurfs. This process begins at iteration 200 to allow initial surfel positioning and continues until iteration 15000, ensuring sufficient time for both surface refinement and subsequent optimization of the remaining surfels. During each pruning interval, Gaussian surfels are evaluated against four criteria and removed if any of the following conditions are met:

1. The base opacity falls below a threshold ($\sigma < 0.1$)
2. The maximum scale exceeds 50% of the scene's extent
3. The product of minimum and maximum scale falls below a threshold ($s_{\min} \cdot s_{\max} < 10^{-8}E^2$, where E is the scene's extent), which helps control variance in scale along each axis

4. The surfel has received no gradient updates, indicating it hasn't been visible in any views

While the first three criteria are inherited from 3DGS, the final criterion was introduced by GSurfs to eliminate noisy points that remain invisible from all camera views. The visibility condition also uses a more stringent threshold than original 3DGS, considering a Gaussian visible only when its projected size (radii) on the image plane exceeds 1 pixel, rather than 0.

During densification intervals, surfels whose positional gradient magnitude exceeds a set threshold undergo densification through either cloning or splitting. Small surfels, defined as those with maximum scale less than 1% of the scene's extent, are cloned by creating a duplicate displaced along the positional gradient direction. Larger surfels undergo binary splitting, where the original surfel is replaced by two new surfels scaled by a factor of 1.6, following the 3DGS approach. The positions of the new surfels are initialized by using the original 3D Gaussian as a PDF for sampling. To control surfel population growth, the opacity values σ_i are reset to zero every $N = 3000$ iterations. This periodic reset maintains an optimal surfel count through natural selection: essential surfels quickly recover their opacity through the optimization process, while unnecessary surfels are removed during pruning due to persistent low opacity.

3.2.5.6 Hyperparameters

The method requires different parameter settings to handle varying levels of geometric detail across datasets. Please refer to Section 4.1 for an introduction to each dataset mentioned here. For the NHR dataset, which contains moderate surface detail, a higher gradient threshold (3e-4) limits densification to avoid introducing noise through excessive gaussian primitives, while stronger shape regularization ($\lambda_s = 1.0$) helps maintain surface consistency. The DNA dataset, however, contains significantly finer geometric detail, necessitating more aggressive parameter settings. A lower gradient threshold (1e-4) enables denser surfel placement to capture these fine details, while reduced shape regularization ($\lambda_s = 0.3$) prevents over-smoothing of the intricate surface features. The surface loss weight is gradually increased during training, starting at $0.01\lambda_s$ and linearly increasing to $0.11\lambda_s$ over the first half of training. This allows the model to first establish rough geometry before enforcing stricter surface consistency.

Due to the high resolution of DNA images and GPU memory constraints, the DNA dataset is processed at half resolution, which results in image dimensions comparable to the NHR dataset. Table 3.2 provides a complete overview of the hyperparameters used in this implementation, focusing on parameters that differ from the baseline methods GSurfs and STG, as well as those specific to the temporal implementation. Parameters that remain identical to the baselines are omitted for brevity.

Category	Parameter	Value
Training	Sequential Frame Count	50 frames
	GT Image Resolution Factor	NHR: 1.0 / DNA: 0.5
	Total Iterations	30,000
	Batch Size	1
	Motion Learning Rate	35
	TRBF Scale Learning Rate	0.06
Densification	Densification Start	Iteration 200
	Densification End	Iteration 15,000
	Opacity Reset Interval	Every 3,000 iterations
	Densification Interval	Every 100 iterations
	Gradient Threshold	NHR: 3e-4 / DNA: 1e-4
Loss Weights	λ_o (Opacity)	0.1
	λ_m (Mask)	0.1
	λ_s (Surface)	NHR: 1.0 / DNA: 0.3
Model Parameters	Position Polynomial Degree (D_p)	3
	Rotation Polynomial Degree (D_q)	1
	TRBF Scale (γ_i) Initial Value	-2
	Initial Opacity	0.1
Meshing	Poisson Depth	9

Table 3.2: Summary of hyperparameters used in the implementation. Parameters are grouped by their function in the system. Only parameters that differ from the baselines or are specific to the temporal implementation are shown. Parameters not listed here are identical to those used in the baseline methods GSurfs and STG.

4 Experiments

This chapter presents a comprehensive experimental evaluation of the Spacetime Surfels method for dynamic scene reconstruction. In Section 4.1, two complementary multi-view video datasets are presented, featuring diverse human performances ranging from controlled movements to challenging sequences with rapid motion and complex garments. In Section 4.3, the method is evaluated against state-of-the-art baselines in both surface reconstruction and view synthesis through quantitative metrics and qualitative assessment. Finally, in Section 4.4, several attempted architectural improvements are examined, providing insights into the robustness of the base implementation and the challenges of optimizing dynamic scene representations.

4.1 Datasets

This thesis focuses on reconstructing dynamic scenes from multi-view video sequences. For thorough evaluation of the proposed method, two publicly available datasets were selected, each providing high-quality synchronized multi-view captures of human performances. These datasets, which are described in detail in this section, offer diverse scenarios and challenging test cases for dynamic scene reconstruction.

4.1.1 Diverse Neural Actor Rendering

The Diverse Neural Actor (DNA) Rendering dataset [Cheng et al., 2023] is a large-scale collection of multi-view human performances, captured using a professional camera system. The dataset includes diverse daily actions and special performances from multiple subjects in various clothing combinations, along with comprehensive annotations including 2D and 3D body keypoints, foreground masks, and fitted SMPL-X models (although the latter are not needed for the method proposed in this thesis). The camera poses and intrinsics are also provided.

For the experiments in this thesis, a subset of five scenes from the dataset was utilized. Each scene consists of 150 frames captured by 60 synchronized cameras. The original images, sized 2048×2448 pixels, required undistortion to conform to the pinhole camera model. This process introduced distortion rings at the image boundaries, necessitating cropping to 1228×1762 pixels around the subject to eliminate these artifacts and ensure stable training.

An example frame for each of five scenes can be found in Figure 4.1. Among these scenes, two present relatively straightforward cases for reconstruction: 0012_11 and 0013_09.

These scenes feature slow, controlled movements where the subject transitions gradually between positions. In contrast, scenes 0008_01, 0013_01, and 0013_03 present significant technical challenges. The traditional attire in 0008_01 creates complex occlusion patterns as its multiple layers of fabric move and overlap, constantly hiding and revealing different parts of the garment. This scene is further complicated by a thin pendant attached to the fan, which exhibits rapid motion throughout the sequence. Scenes 0013_01 and 0013_03 introduce additional complexity through rapid movements and drastic topological changes. These scenes create demanding test conditions that evaluate the method's ability to handle both dramatic pose changes and fast motion patterns.

Figure 4.1: Example ground truth frames from the DNA dataset with their corresponding segmentation masks applied, showing representative poses from the five different sequences.

For evaluation purposes, a test set was held out from the training process, consisting of the complete frame sequences from cameras 11, 25, 37 and 57. These specific views were selected to ensure comprehensive coverage of different angles around the subject.

4.1.2 Neural Human Rendering

The Neural Human Rendering (NHR) dataset [Wu et al., 2020] consists of multi-view sequences captured in a dome-like structure, providing 360° views of human performances. The camera setup uses a mix of different resolution cameras: some cameras capture at 1024×768 resolution while others record at 1224×1024. The dataset provides camera poses, intrinsic parameters, and foreground masks for each frame.

For the experiments in this thesis, a subset of four scenes from the dataset was utilized. An example frame for each of the scenes can be found in Figure 4.2. Three of these scenes (sport_1, sport_2, and sport_3) present relatively straightforward cases for reconstruction, featuring controlled exercise routines such as lunges and squats. These scenes were recorded using a setup of 57 synchronized cameras and are characterized by slow, delib-

erate movements with minimal self-occlusions, making them ideal for evaluation. The basketball scene, captured with an enhanced setup of 73 synchronized cameras, presents more significant technical challenges. The rapid movements of both the subject and the basketball create complex motion patterns that test the method's temporal consistency. Additionally, this scene's ground truth masks exhibit lower precision compared to the other scenes, introducing an extra layer of complexity to the reconstruction process.

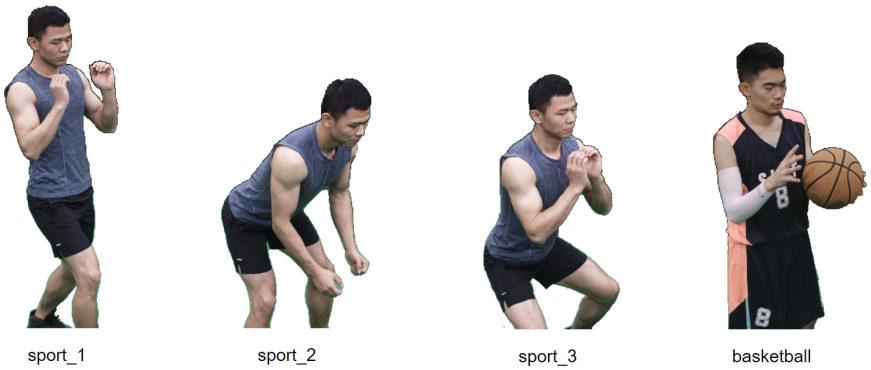

Figure 4.2: Example ground truth frames from the NHR dataset with their corresponding segmentation masks applied, showing representative poses from the four different sequences.

Similarly to DNA, a test set was carefully selected from these scenes, comprising complete frame sequences from cameras 18, 28, 37, and 46. These specific viewpoints were strategically chosen to ensure thorough evaluation from diverse angles around the subject, enabling comprehensive assessment of reconstruction quality from multiple perspectives.

4.2 Metrics

The quantitative evaluation of dynamic scene reconstruction methods relies on several complementary metrics that assess different aspects of output quality. Peak Signal-to-Noise Ratio (PSNR) measures the pixel-level accuracy of reconstructed images compared to ground truth, with higher values indicating better reconstruction quality. The Structural Similarity Index (SSIM) evaluates the preservation of structural information and perceptual quality, operating on a scale from 0 to 1 where 1 represents perfect structural similarity. To capture more subtle aspects of visual quality that align with human perception, the evaluation also employs the Learned Perceptual Image Patch Similarity (LPIPS) metric, where lower values indicate better perceptual similarity to the ground truth. For comprehensive evaluation, these metrics are computed and averaged across all testing views, frames, and scenes within each dataset, providing a robust assessment of each method's performance.

It is important to note that these image-based metrics primarily evaluate the view synthesis capabilities of the methods, measuring how well they can reproduce ground truth camera viewpoints. For geometry evaluation, specifically the similarity between the output mesh and its ground truth, the Chamfer Distance metric is typically employed. However, no publicly available synthetic dynamic dataset was found on which to evaluate this metric. Due to the absence of ground truth geometry in the datasets, direct quantitative evaluation of the reconstructed mesh quality is not possible. To address this limitation, the evaluation includes detailed visual comparisons of mesh renders from multiple viewpoints, allowing qualitative assessment of geometric reconstruction quality between the proposed method and baselines. Additionally, temporal consistency or "jitter" is evaluated qualitatively since no suitable metric exists to measure this aspect. While the results of these dynamic comparisons are presented and discussed in this thesis, the video demonstrations themselves can only be properly viewed during the defense presentation due to the limitations of the printed format.

Beyond image quality metrics, the evaluation considers computational efficiency through measurements of training duration. These performance metrics provide crucial insights into the practical applicability of different reconstruction approaches in real-world scenarios. Together, this comprehensive set of metrics enables thorough comparison between methods and highlights their respective strengths and limitations across both quality and computational dimensions.

4.3 Evaluation

This section presents a comprehensive evaluation of STS against relevant baseline methods. Based on the analysis of related work, NeuS2 serves as the primary baseline for comparison due to its shared objective of dynamic surface reconstruction. Additionally, while STS was not specifically designed for dynamic view synthesis, comparisons with STG and 4K4D are included to provide broader context for its performance capabilities. Table 4.1 presents quantitative comparisons across both datasets using the metrics described in Section 4.2. All experiments were conducted on a single NVIDIA RTX 6000 Ada Generation GPU utilizing CUDA 11.8, Python 3.10.14, and PyTorch 2.3.1+cu118.

Following the training strategy established by STG, the training process for STG and STS was structured to handle sequences of 150 frames by dividing them into three consecutive segments of 50 frames each (frames 0-49, 50-99, and 100-150). This segmentation approach addresses both dataset constraints and memory limitations while maintaining processing efficiency. The training times reported in Table 4.1 represent the aggregate duration across all three segments, providing an accurate measurement of the total computational time required to process the complete sequence. This standardized approach allows for consistent performance evaluation and direct comparison between methods.

4.3.1 View-Synthesis Evaluation

The experimental results summarized in Table 4.1 demonstrate that the proposed STS method achieves an exceptional balance of quality and efficiency across datasets. On the NHR dataset, STS delivers near-benchmark performance (PSNR: 33.01) comparable to 4K4D (PSNR: 33.65) and NeuS2 (PSNR: 33.04) while requiring only 26.5 minutes of training time compared to 4K4D's 34.7 hours and NeuS2's 1.2 hours. STS significantly outperforms the STG method on this dataset (PSNR: 33.01 vs 28.03) while maintaining similar efficiency (26.5 minutes vs 23.5 minutes). On the more challenging DNA dataset, STS achieves competitive performance (PSNR: 31.22) superior to NeuS2 (PSNR: 30.15) but with dramatically improved efficiency (32.0 minutes vs 3.3 hours). While 4K4D sets the quality benchmark on both datasets, its extensive training requirements (35+ hours) make it impractical for many applications. The STG method maintains efficiency advantages but shows substantially lower performance than STS across datasets, particularly on DNA (PSNR: 24.00 vs 31.22). These results validate that the adaptation of STG to better represent surfaces through STS was effective not only in extracting surface meshes but also in significantly improving view-synthesis quality, establishing STS as a superior method for efficient high-quality neural surface reconstruction.

Dataset	Method	PSNR↑	SSIM↑	LPIPS↓	Training Time
NHR	4K4D	33.65	0.972	0.039	34.7 h
	STG	28.03	0.949	0.075	23.5 min
	NeuS2	33.04	0.973	0.038	1.2 h
	STS (ours)	33.01	0.967	0.062	26.5 min
DNA	4K4D	34.52	0.985	0.025	35.2 h
	STG	24.00	0.909	0.108	25.2 min
	NeuS2	30.15	0.971	0.063	3.3 h
	STS (ours)	31.22	0.965	0.072	32.0 min

Table 4.1: Quantitative comparison across DNA and NHR datasets. Methods above the dotted line are view synthesis approaches, while those below are surface reconstruction methods. Evaluation metrics include PSNR, SSIM (↑ higher is better) and LPIPS (↓ lower is better). Cell highlighting indicates relative performance: red = best, orange = second best, yellow = third best for each metric within each dataset.

A notable advantage of STS is its consistent training efficiency across datasets with varying complexity. While NeuS2 shows a significant disparity in training times between the DNA dataset (3.3 hours) and NHR dataset (1.2 hours), STS maintains relatively stable training times of approximately 30 minutes regardless of scene complexity. This discrepancy in NeuS2's performance can be attributed to its sequential transformation approach. NeuS2 processes temporal information by predicting the transformation between consecutive timesteps and accumulating these transformations to map each timestep back to the canonical space (ie. first frame). Experimental observations revealed that NeuS2's

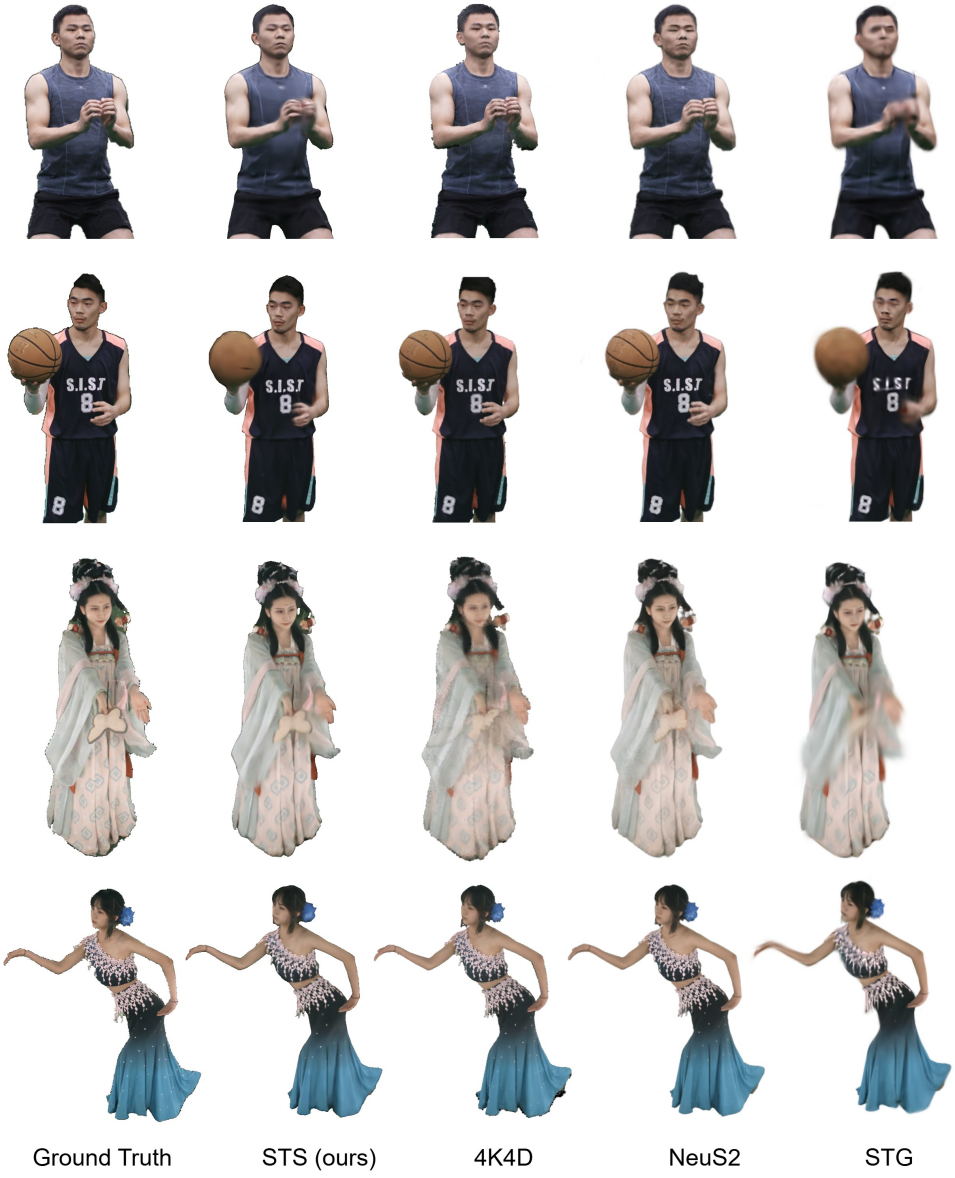

Figure 4.3: Qualitative comparison of the proposed STS method against existing dynamic view-synthesis techniques (4K4D, NeuS2, and STG), showing performance on various human subjects relative to ground truth images.

per-timestep training time increased progressively throughout the sequence for the DNA dataset, while remaining constant for NHR. This suggests that the accumulation of larger transformations in DNA scenes, which feature more dramatic movements, leads to computational overhead that compounds with each subsequent timestep. In contrast, STS's direct spatiotemporal representation avoids this accumulation of transformations, resulting in more predictable and consistent training times across diverse datasets.

The qualitative results in Figure 4.3 reinforce the quantitative findings. Visual comparisons across different human subjects demonstrate that the proposed STS method shows significant improvement over STG while maintaining comparable visual quality to NeuS2 and 4K4D for most scenes. This improvement is particularly evident in the third row showing the traditional costume subject, where STS preserves fine details in flowing fabrics and the handheld fan with greater fidelity than STG and 4K4D, and achieves results comparable to NeuS2. Similarly, for the dancer in the blue gradient dress in the fourth row, STS captures the intricate costume details with remarkable accuracy and renders the hand gestures with much less blur than STG, better preserving the dancer's pose and form. For the athlete scene in row 1, STS better captures facial details and hand movements compared to STG, though still not achieving the same level of hand detail fidelity as 4K4D and NeuS2.

Notably, the STS method does face challenges with certain high-frequency details, as evidenced in the basketball scene in the second row. Both NeuS2 and 4K4D significantly outperform STS in rendering the distinctive black lines on the basketball, which STS fails to recover accurately. This limitation, which is also visible with STG, can be attributed to the rapid rotation of the ball, causing the black lines to move at high velocities across frames. This example highlights an inherent tradeoff in the approach, where the computational efficiency gains come at the cost of some fine detail reconstruction. Nevertheless, STS's limitations in these specific cases are outweighed by its substantial computational efficiency and strong overall performance in scenes with complex fabrics and accessories. These results demonstrate that STS effectively bridges the performance gap between the lower-quality but efficient STG approach and the high-quality but computationally intensive 4K4D method, offering a practical solution that balances quality and efficiency for dynamic scene reconstruction.

4.3.2 Surface Reconstruction Evaluation

The qualitative mesh comparisons between STS and NeuS2 reveal interesting insights that complement the above findings. The close-up insets in Figure 4.4 particularly highlight STS's superior performance in capturing fine geometric details, especially evident in the precise reconstruction of hand poses and finger articulation across different sequences. In the DNA dataset sequences (rows 1-3 in Figure 4.4), STS demonstrates notably better reconstruction of intricate features, as shown in the detailed hand-prop interactions during the dance performances with the basket (third row) and the accurate finger positioning during phone photography poses (second row). This geometric advan-

Figure 4.4: Qualitative comparison between ground truth images and 3D reconstructions using NeuS2 and our proposed STS method, with detail views highlighting improved fidelity in challenging areas.

tage aligns with the PSNR metrics, where STS demonstrated superior performance over NeuS2. This suggests that STS not only achieves better overall image reconstruction but also excels in preserving the detailed geometric structure of the scenes.

In the NHR dataset comparisons (rows 4-5 in Figure 4.4), the differences between STS and NeuS2 are less pronounced but still favorable for STS. The close-ups of the basketball player scene (fourth row) show that STS produces better facial reconstruction, whereas NeuS2 exhibits some facial deformation that affects the subject's appearance. STS also generates slightly smoother surface reconstructions throughout the NHR dataset while maintaining important details. This improved performance on the NHR dataset is consistent with the quantitative metrics, where STS achieved similar PSNR (33.01 vs 33.04) and SSIM (0.967 vs 0.973) values compared to NeuS2, while requiring only about a third of the training time (26.5 minutes vs 74.5 minutes). These results demonstrate that STS offers both computational efficiency and better reconstruction quality for human performances, with particular advantages in preserving facial features and surface details.

4.3.3 Temporal Consistency Analysis

While the previous sections evaluated view synthesis quality and surface reconstruction fidelity, it is also important to address the temporal consistency of the reconstructed meshes across the entire sequence. Since there is no established quantitative method for measuring mesh jitter across frames, the evaluation relies on qualitative visual comparison. Video outputs of mesh sequences from both STS and NeuS2 were rendered and placed side by side for direct visual assessment. This comparative approach allows for a clear demonstration of the different temporal artifacts produced by each method.

As mentioned in Section 4.3, the training strategy for STS follows STG by dividing the 150-frame sequences into three consecutive segments of 50 frames each. This segmentation approach, while necessary to address memory limitations and processing efficiency, introduces a notable limitation in temporal continuity. Visual inspection of the reconstructed sequences reveals noticeable discontinuities or "jumps" at the boundaries between training segments, specifically between frames 49-50 and 99-100. These transition artifacts are an inherent consequence of independently optimizing each 50-frame segment without explicit constraints to ensure smooth transitions between segments. While the mesh topology and appearance remain consistent within each segment, the lack of cross-segment consistency constraints results in visible discontinuities when the complete sequence is rendered as a video.

In contrast, NeuS2 employs an incremental training approach that eliminates these specific segment boundary jumps. However, this approach introduces its own consistency issues, resulting in frame-by-frame jitter throughout the sequence. The mesh consistency is actually worse with NeuS2, producing more jittery results than STS. Despite NeuS2's canonical space approach attempting to map all frames back to a common reference frame, it fails to deliver the expected consistency benefits. Meanwhile, STS maintains

good temporal consistency within each 50-frame segment. The direct spatiotemporal representation effectively captures smooth transitions between consecutive frames within a segment, preserving the overall dynamic nature of the performance while avoiding the continuous jitter present in NeuS2's results.

4.4 Attempted Model Improvements and Insights

Several experimental approaches were investigated during the course of this thesis to try and improve the vanilla Spacetime Surfels implementation detailed in section 3.2.1, focusing on mesh quality, training efficiency, and memory optimization. The experiments explored adaptive frame sampling based on scene velocity, optimization of surfel temporal duration and optical flow supervision for motion accuracy. While each modification addressed specific limitations identified in the base implementation, particularly for scenes with complex motion, empirical results demonstrated that these modifications offered minimal benefits relative to their computational and quality trade-offs.

The following subsections document these experimental approaches for the sake of completeness, detailing their methodology and implementation while analyzing their limitations. These investigations, though ultimately not incorporated into the final implementation, provide valuable insights into the robustness of the base architecture and the challenges of optimizing dynamic scene representation. Furthermore, as these approaches constituted a substantial component of the research effort during this thesis, their documentation honors the comprehensive investigative process and may offer instructive perspectives for future work in this domain.

4.4.1 Velocity Based Frame Sampling

The vanilla implementation of Spacetime Surfels demonstrated limitations in scenes with rapid movements, particularly evident in the DNA dataset. This is exemplified in scene 0013_03, where a woman performs quick hand gestures, picking up and throwing imaginary objects into a basket—resulting in hand reconstructions lacking fine details such as distinct fingers. Similar limitations appeared in scene 0008_01, where the pendant of a butterfly-shaped fan oscillates rapidly. These reconstruction challenges extended to the NHR dataset's basketball scene, where the model struggled to accurately capture the black lines on the rapidly rotating ball. This challenge is not unique to Spacetime Surfels but is also observed in STG, the dynamic view synthesis method upon which Spacetime Surfels builds. To address this limitation, an improved camera sampling strategy was proposed. Rather than employing uniform random sampling across frames as explained in Section 3.2.5.1, the adaptive sampling approach prioritizes frames containing rapid movement. The detection of such frames is accomplished by computing the temporal derivative of surfel positions to obtain their velocities. By calculating the maximum velocity of all relevant surfels within each frame, it becomes possible to identify frames exhibiting significant motion, even if localized to specific regions of the scene.

A crucial implementation detail involves the careful selection of "active" surfels, as mentioned in Section 3.2.4, those that are visible at a specific timestep. A surfel is considered active when its opacity exceeds a predetermined threshold. This filtering step ensures that the velocity calculations are based only on surfels that meaningfully contribute to the scene representation at that moment, avoiding potential bias from transparent surfels.

This adaptive sampling strategy activates after 10,000 iterations of initial training, allowing surfels to first establish stable positions and trajectories through uniform sampling. This timing preserves 5,000 iterations where densification occurs concurrently with the adaptive sampling strategy, ensuring that rapidly moving areas can still be adequately densified. After basic scene geometry converges, sampling alternates between two phases. First, frames above the 90th percentile of velocity measurements are identified and their corresponding camera views are maintained in a stack for sampling. The size of this high-velocity stack is determined by the number of frames above the 90th percentile multiplied by the number of views per timestep. Once this stack is exhausted, the system switches to uniform sampling from all camera views for a period of one-third the number of high-velocity iterations. This cycling between focused sampling of high-velocity frames and briefer periods of uniform sampling should provide better coverage of rapid movements by allocating more updates to the corresponding surfels, while maintaining periodic updates across all surfels.

In practice, this approach yielded no measurable improvements. Evaluation metrics remained equivalent to those of vanilla STS, with no visually significant enhancements observed in the output mesh sequence. This failure may stem from a flawed initial assumption. The assumption that fast movements are challenging to reconstruct proved valid only for localized rapid motion. When examining cases with localized high-velocity movements, such as the rapid hand gestures in DNA scene 0013_03 and the oscillating pendant in DNA scene 0008_01, the model exhibited clear limitations. However, in scenarios where the subject moved rapidly but uniformly, the model demonstrated adequate performance. Hence, although the method successfully increased emphasis on frames containing rapid motion overall, it lacked the granular precision needed for optimal results. The findings suggest that while modifying frame sampling can increase focus on frames with rapid movement, this approach alone cannot achieve the necessary precision to target specific fast-moving surfels within those frames. A more effective solution would require fundamental changes to the surfel processing pipeline itself, rather than merely adjusting the frame sampling strategy.

4.4.2 Temporal Duration Control in Surfels

An optimization approach explored reducing memory requirements by extending surfels' temporal duration. The idea being that if surfels last longer, fewer surfels are necessary to represent the scene, thereby decreasing the overall model size. The temporal extent of a surfel is determined by comparing its tRBF to a standard Gaussian distribution. Given

the temporal opacity function $o_i(t) = \sigma_i \exp\left(-\beta_i |t - \mu_i|^2\right)$ and the standard Gaussian form $f(x) = A \exp\left(-\frac{(x-\mu)^2}{2\text{std}^2}\right)$, the temporal standard deviation can be derived through term matching. Since $\beta_i = \frac{1}{2\text{std}^2}$, and using the reparametrization $\beta_i = e^{-2\gamma_i}$ from section 3.2.5.4, this yields $\frac{1}{2\text{std}^2} = e^{-2\gamma_i}$. The resulting temporal standard deviation is $\frac{e^{\gamma_i}}{\sqrt{2}}$. The approximate duration in frames can be computed as twice the standard deviation divided by the time per frame. Given that the sequence duration, denoted by D, is normalized to the interval [0,1], the time per frame is $\frac{1}{D}$, hence:

$$\text{duration in frames} = 2 * D * \frac{e^{\gamma_i}}{\sqrt{2}} \tag{4.1}$$

Conversely, to determine the required γ_i for a desired number of frames:

$$\gamma_i = \ln\left(\frac{\#\text{frames} * \sqrt{2}}{2 * D}\right) \tag{4.2}$$

This relationship suggested that encouraging larger γ_i values would lead to longer-persisting surfels, potentially reducing the total number needed to represent the dynamic scene. This motivated adding a "tRBF scale loss" $\mathcal{L}_{\text{tRBFs}}$ to the total loss, to push the surfels to have larger γ_i values:

$$\mathcal{L}_{\text{tRBFs}} = \text{softplus}\left(\text{target} - \text{clip}(\overline{\gamma}, -1.5)\right) \tag{4.3}$$

where $\overline{\gamma}$ is the mean of all surfel tRBF scale parameters. For surfels to have reasonable temporal coherence, a minimum duration of 5 frames was chosen in a 50-frame sequence, requiring $\gamma_i >= \ln\left(\frac{5\sqrt{2}}{2*50}\right) = -2.64$. Setting the target to -2.64 ensures γ_is below this threshold receive a large loss, pushing them higher. The clipping of γ_i to -1.5 prevents surfels from spanning more than approximately 15 frames, maintaining a balance between temporal coherence and localization.

However, empirical results demonstrate that this approach was ineffective for model size reduction. As shown in Table 4.2, artificially extending surfel lifespans through the tRBF scale loss not only degrades reconstruction quality but also increases the model size. This counterintuitive increase in model size can be attributed to compensation behavior: when surfels are forced to persist longer than optimal for a given motion, additional surfels must be created to correct for the resulting temporal blur and maintain reconstruction quality. Analysis of the vanilla model reveals that surfel duration actually naturally adapts to scene complexity: in scenes with complex motion like DNA 0013_01, surfels optimize to shorter lifespans: the average duration of a surfel is $\overline{\gamma} = -3.67$ which translate to ~ 1.8 frames. While in scenes with simpler motion like NHR sport_1, surfels maintain longer durations: $\overline{\gamma} = -2.12$, so surfels last ~ 8.45 frames on average. These results demonstrate that the original optimization process effectively balances temporal coherence with reconstruction fidelity by adapting surfel parameters to the inherent complexity of the dynamic scene, with the values for γ_i representing an optimal trade-off between temporal extent and reconstruction accuracy.

	Model Size (KB)		PSNR		Size	Quality
Dataset	Vanilla	with $\mathcal{L}_{\text{tRBFs}}$	Vanilla	with $\mathcal{L}_{\text{tRBFs}}$	Red. (%)	Loss (%)
DNA	292.56	611.90	31.22	30.54	+109.15	-2.18
NHR	102.24	116.28	33.01	32.68	+13.73	-1.00

Table 4.2: Impact of Enforced Longer Surfel Lifespans on Model Size and Quality. Results are averaged across all scenes within each dataset

4.4.3 Optical Flow Supervision

Optical flow supervision was investigated to improve motion parameter accuracy, surface reconstruction quality, and temporal consistency. The method adapts the Gaussian flow computation from MAGS [Ma et al., 2024] to Spacetime Surfels, combining it with pseudo ground truth flows from RAFT [Teed and Deng, 2020].

Before training, the pseudo ground truth optical flows are predicted using RAFT, which is done as a preprocessing step to ensure all flows are available during training. To generate these flows, the preprocessing script iterates through frame pairs at time t and $t+1$, feeding each pair to RAFT for prediction. The pairs are processed in both increasing time order (to obtain forward flow $\mathbf{F}_{\text{GT}}^{\text{fwd}}$) and decreasing time order (to obtain backward flow $\mathbf{F}_{\text{GT}}^{\text{bwd}}$), creating bidirectional flow predictions for each frame transition. Using the forward and backward flow, a consistency mask can be computed between the two by measuring the cycle consistency error. This is done by warping each flow according to the other and calculating how closely they cancel each other out when combined. The function creates masks by checking if the error is below a threshold that's proportional to the flow magnitudes plus a small constant. These masks identify regions where the bidirectional flows agree with each other, indicating reliable motion estimation. The forward and backward masks are combined using a logical AND operation, and the background segmentation mask is also applied to exclude background pixels from consideration. This final combined mask, called the "valid flow mask", ensures that only reliable flow estimates from foreground objects are considered in the flow loss computation.

To compute the flow prediction of the model, the flow of each surfel is calculated by evaluating its polynomial motion function at times t and $t+1$ to obtain their 3D positions. These positions are projected onto the image plane using the camera parameters, yielding 2D pixel coordinates $\mathbf{p}$ of surfel i: $\mathbf{p}_i(t)$ and $\mathbf{p}_i(t+1)$. The per-surfel flows are computed as the difference in pixel positions, where forward and backward flows represent opposite displacements: $\mathbf{f}_i^{\text{fwd}} = \mathbf{p}_i(t+1) - \mathbf{p}_i(t)$ and $\mathbf{f}_i^{\text{bwd}} = -\mathbf{f}_i^{\text{fwd}}$. To get the full flow at each time frame, these individual surfel flows must be combined, intuitively this can be done using the same rendering strategy as for color or normals using alpha-blending as was detailed in Section 3.2.2. However, to address the optimization instability observed by MAGS authors, instead of using all surfels, only the $K = 20$ closest surfels to the

Dataset	Method	PSNR↑	SSIM↑	LPIPS↓	Training Time
DNA	vanilla	31.60	0.969	0.063	1.04 h
	w/ flow loss	31.18	0.965	0.071	3.59 h
NHR	vanilla	33.37	0.970	0.053	1.08 h
	w/ flow loss	33.21	0.969	0.055	3.33 h

Table 4.3: Quantitative comparison between vanilla STS and STS with optical flow supervision across DNA and NHR datasets. Evaluation metrics include PSNR, SSIM (↑ higher is better) and LPIPS (↓ lower is better).

image plane are considered. According to the authors, this modification helps prevent optimization collapse caused by rendering 3D primitives to 2D flow fields and reduces noise from extended Gaussian influence.

The rendered flow fields are compared against pseudo ground truth forward and backward flows $\mathbf{F}_{\text{GT}}^{\text{fwd}}$ and $\mathbf{F}_{\text{GT}}^{\text{bwd}}$ generated by RAFT using weighted L1 losses:

$$\mathcal{L}_{\text{fwd}} = \sum_{p \in \mathcal{V}} \sum_{i \in \mathcal{K}p} w_i |\mathbf{f}_i^{\text{fwd}} - \mathbf{F}_{\text{GT}}^{\text{fwd}}(p)| \tag{4.4}$$

$$\mathcal{L}_{\text{bwd}} = \sum_{p \in \mathcal{V}} \sum_{i \in \mathcal{K}p} w_i |\mathbf{f}_i^{\text{bwd}} - \mathbf{F}_{\text{GT}}^{\text{bwd}}(p)| \tag{4.5}$$

where $\mathcal{V}$ denotes foreground pixels where forward and backward ground truth flows are consistent, i.e. the "valid flow mask" explained above, indicating reliable flow estimates, $\mathcal{K}p$ represents the set of K nearest surfels for pixel p and the weights $w_i = \frac{T_i \alpha_i}{\sum_{j \in \mathcal{K}p} T_j \alpha_j}$ are the normalized surfel contributions. The total flow loss combines forward and backward components:

$$\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{fwd}} + \mathcal{L}_{\text{bwd}} \tag{4.6}$$

This flow loss is incorporated into the total loss function with a weighting parameter $\lambda_f = 0.0001$:

$$\mathcal{L} = \mathcal{L}_p + \lambda_o \mathcal{L}_o + \lambda_m \mathcal{L}_m + \lambda_s \mathcal{L}_s + \lambda_f \mathcal{L}_f \tag{4.7}$$

The implementation encountered memory constraints with the original configuration. The necessity to load predicted optical flows in conjunction with the RGB images and masks exceeded the available GPU memory capacity for the standard 50-frame sequences. To address this limitation, the training procedure was modified to process 5 distinct sequences of 30 frames each per scene, rather than 3 sequences of 50 frames. To maintain experimental validity, vanilla STS was also retrained using 5 sequences of 30 frames, thereby ensuring a fair comparison between the flow loss variant and the vanilla version. The standard 30,000 total iterations were maintained across both approaches, which explains the extended the training duration for vanilla STS to approximately 1 hour per scene, in contrast to the 30 minutes reported in Table 4.1.

Experimental results demonstrated that incorporating the flow loss did not improve PSNR, SSIM nor LPIPS metrics across the evaluated scenes, but actually degraded performance, as can be seen in Table 4.3. Additionally, this approach imposed significant computational costs. The training time increased substantially from ∼1 hour to ∼3.5 hours per scene. This increase stemmed from both the additional computational overhead of loading predicted optical flows and the increased complexity of the backpropagation step when incorporating flow-based supervision. The resulting meshes exhibited inferior quality compared to the vanilla implementation, with observable noise artifacts and reduced temporal consistency. Further investigation revealed a direct correlation between flow loss weight and performance degradation: the smaller the weight, the closer the PSNR was to the vanilla results. This suggests a fundamental conflict between the flow-based temporal consistency objective and per-frame reconstruction quality. Notably, despite being explicitly designed to improve inter-frame consistency, the flow loss paradoxically resulted in worse temporal coherence, indicating that naive application of optical flow supervision may disrupt the model's ability to learn stable geometric representations over time.

Given the combination of worse PSNR results, degraded mesh quality, reduced temporal consistency, and significantly increased training times, there was no justification for including the flow loss in the final model. The experimental evidence clearly demonstrated that this approach not only failed to deliver its intended benefits but actively undermined overall performance while substantially increasing computational overhead.

4.4.4 Summary of Experiments

The experimental modifications explored in this section, while theoretically promising, demonstrated limited practical value compared to the vanilla Spacetime Surfels implementation. The velocity-based frame sampling approach failed to deliver meaningful improvements in reconstruction quality, even for scenes containing rapid motion. The attempted surfel lifespan extension unexpectedly increased model size while degrading reconstruction quality, as the model compensated for forced temporal persistence by creating additional surfels. The optical flow supervision degraded PSNR metrics and mesh quality while requiring prohibitively long training times (3.5 hours versus 1 hour per scene), making it counterproductive for the final implementation.

These findings collectively indicate that the vanilla implementation of Spacetime Surfels already maintains an optimal balance of performance characteristics. The proposed modifications, while addressing specific theoretical limitations, introduced additional complexity without delivering proportional benefits. This thorough experimental investigation ultimately validates the robustness and efficiency of the original architecture in handling dynamic scene representation.

5 Discussion and Conclusion

5.1 Strengths

The proposed Spacetime Surfels (STS) method demonstrates several notable strengths in addressing the challenge of dynamic surface reconstruction. First, the method achieves a compelling balance between computational efficiency and reconstruction quality. The experimental results show that STS can process complex dynamic scenes in approximately 30 minutes, while maintaining view-synthesis quality comparable to state-of-the-art methods that require several hours of training time. This efficiency stems from the method's direct spatiotemporal representation, which avoids the computational overhead associated with sequential transformation approaches used by methods like NeuS2. In other words, STS achieves similar rendering quality as other leading methods while being significantly faster.

A second key strength lies in STS's ability to capture fine geometric details, particularly evident in the reconstruction of intricate features like hand poses and finger articulation. The qualitative comparisons presented in Chapter 4 demonstrate that STS consistently produces cleaner, more detailed surface reconstructions compared to NeuS2. This improved geometric accuracy is achieved despite some methods showing marginally better PSNR metrics.

Additionally, while STS still exhibits some degree of inter-frame jittering when rendering the reconstructed mesh over time, this temporal instability is notably reduced compared to NeuS2. The rendered mesh geometry in STS maintains better consistency across consecutive frames, resulting in more coherent dynamic reconstructions. This improvement in temporal stability, though not completely eliminating jittering, produces more visually pleasing animations with fewer distracting artifacts than those generated by NeuS2.

5.2 Limitations

The method also exhibits several limitations that warrant discussion. The most significant challenge involves the handling of rapid localized motion, a common problem across reconstruction techniques that is particularly evident in scenes from the DNA dataset containing fast hand movements or oscillating accessories. While the base implementation performs adequately for uniform motion patterns, it struggles —as do many existing methods— to maintain reconstruction quality in areas experiencing sudden or extreme motion. This limitation persists despite attempts to address it through velocity-based

frame sampling, suggesting that the challenge lies deeper in the representation's fundamental structure.

Another limitation concerns the temporal consistency of meshes across frames. The current implementation processes 50 frames at once, introducing noticeable discontinuities at segment boundaries and significantly impacting the temporal coherence of the reconstructed meshes. This is directly related to memory constraints and scaling behavior. While processing the entire sequence simultaneously would address the boundary discontinuity issue, it presents substantial computational challenges. One potential workaround involves storing RGB images and masks on the CPU rather than the GPU, which could theoretically allow processing more than 50 frames concurrently. However, this approach introduces significant training time overhead due to frequent CPU-to-GPU data transfers. Moreover, this solution fails to fully address the fundamental scaling problem—as sequence length increases, the number of required surfels grows proportionally, eventually exceeding available GPU memory regardless of where image data is stored.

While not a primary focus of this research, it's worth noting that the method in its current implementation is not designed for online processing scenarios. The approach requires multiple optimization iterations across temporal chunks of frames, making on-the-fly reconstruction challenging. This characteristic, though not detracting from the method's core contributions, does limit potential applications in contexts requiring processing of new content in real-time, such as real-time motion capture or interactive environments.

5.3 Potential Improvements and Future Work

A critical direction for future research lies in developing sophisticated strategies for handling rapid localized motion within dynamic scenes. The current challenge stems from the inherent global optimization nature of 3D Gaussian Splatting, where the gradient descent step updates all surfels simultaneously during each iteration. Future work could explore mechanisms for identifying regions of rapid motion and selectively intensifying the optimization process for affected surfels while maintaining the method's computational efficiency. This would require developing novel detection algorithms for motion-intensive regions and modifying the optimization framework to support variable update rates across different spatial regions, all while preserving the method's end-to-end differentiability. Such an approach could significantly improve reconstruction quality in areas of rapid motion without compromising the method's overall performance characteristics. However, the technical challenges of implementing local optimization strategies within the current framework remain significant, as successfully integrating local updates without disrupting the global optimization process is far from straightforward and would require fundamental changes to the optimization framework.

A particularly promising avenue for improvement lies in the initialization strategy for consecutive frame segments. Currently, when processing sequences longer than 50 frames,

each new segment starts with a fresh initialization, leading to potential discontinuities. Future work could explore leveraging the trained results from the previous 50-frame segment to initialize the next segment's optimization. This approach would likely improve temporal consistency across segment boundaries and potentially reduce training time by providing a better starting point for optimization. The challenge lies in determining which surfels to carry forward and how to adjust their properties to accommodate the new temporal window while maintaining stable reconstruction quality.

While STS demonstrates significantly improved temporal consistency compared to methods like NeuS2, there remains room for enhancement in this area. The current implementation processes 50 frames at once, which can introduce some discontinuities at segment boundaries. Even within these segments, minor temporal jitter can occasionally be observed, though substantially less than in comparable methods. This stems from the independent processing of each frame using Screened Poisson reconstruction, which does not fully incorporate temporal information. Future work could explore ways to further improve temporal coherence for applications requiring perfectly smooth temporal evolution.

5.4 Summary

This thesis presented Spacetime Surfels (STS), a novel approach for dynamic surface reconstruction that synergizes two state-of-the-art methods: Gaussian Surfels for high-quality surface representation and Spacetime Gaussians for dynamic view synthesis. The method represents dynamic 3D scenes through time-varying surfel primitives, each characterized by polynomial motion trajectories, quaternion-based rotation, and temporal opacity functions. These surfels are initialized from multi-view Structure-from-Motion point clouds and optimized through gradient descent using a composite loss function that balances photometric accuracy, opacity consistency (encouraging binary opacity values), mask alignment, and surface coherence. The system processes multi-view video input in chunks of 50 frames, randomly sampling frames within each chunk for optimization. The surfels undergo gradient-driven densification and pruning to adapt to scene complexity. The optimization framework carefully balances detailed geometric reconstruction with temporal consistency, employing different hyperparameter settings to handle varying levels of surface detail across datasets. Once optimization completes, the depth and normal maps from the optimized surfels are fused through Poisson surface reconstruction to generate a sequence of 3D meshes, yielding a representation of dynamic scenes that achieves competitive reconstruction quality with significantly faster training times compared to existing approaches and improved temporal coherence.

The evaluation component of this thesis assessed the proposed methods against baseline approaches, examining rendering quality, surface mesh fidelity, temporal consistency, and training efficiency. Results demonstrate that the approach substantially outperforms STG in terms of view synthesis, the method from which the dynamic representation was

largely derived. This confirms that integrating this dynamic representation with flat surfels for better surface representation enhances rendering quality. Additionally, STS significantly exceeds 4K4D in terms of training efficiency, while maintaining comparable metrics for view synthesis. Lastly, when holistically evaluating all performance metrics (rendering quality, mesh fidelity, training efficiency, and temporal consistency) it can be conclusively asserted that STS surpasses its primary benchmark counterpart, NeuS2, demonstrating that the 3D Gaussian Splatting paradigm offers superior performance over NeRF-based approaches for surface reconstruction. While the method achieves significant success overall, it still presents considerable opportunities for further refinement and advancement.

Throughout the development process, several alternative optimization strategies were explored. Experiments and investigation of various optimization approaches, including velocity-based sampling, temporal duration control and optical flow supervision, have provided valuable insights into the challenges of dynamic scene representation. While these experimental modifications did not yield significant improvements over the base implementation, they have helped identify promising directions for future research.

5.5 Conclusion

The work presented in this thesis directly addresses the critical gap between dynamic view synthesis and surface reconstruction within the 3D Gaussian Splatting framework. While Neural Radiance Fields (NeRF) had previously demonstrated capabilities in creating dynamic surface meshes, this research establishes that Gaussian Splatting approaches can not only achieve comparable results but potentially offer superior performance in terms of efficiency and quality. By integrating techniques from Gaussian Surfels and Spacetime Gaussians, this work enables the efficient reconstruction of temporally consistent surface meshes of dynamic subjects, providing results that can be directly integrated into modern graphics engines. The novel framework developed here successfully bridges the divide between visual quality and geometric fidelity in dynamic environments, representing a meaningful contribution to the field of 3D reconstruction. As the demand for realistic interactive experiences continues to grow in AR/VR and computer graphics applications, the approach established in this thesis provides a foundation for future work that requires both visual realism and geometric precision in dynamic settings, with the additional benefit of the computational efficiency inherent to Gaussian representations.

Bibliography

[Bagley et al., 2016] Bagley, B., Sastry, S., and Whitaker, R. (2016). A marching-tetrahedra algorithm for feature-preserving meshing of piecewise-smooth implicit surfaces. *Procedia Engineering*, 163:162–174.

[Baker et al., 2023] Baker, A. H., Pinard, A., and Hammerling, D. M. (2023). Dssim: a structural similarity index for floating-point data.

[Bao et al., 2024] Bao, F., Xiang, C., Yue, G., He, G., Zhu, H., Zheng, K., Zhao, M., Liu, S., Wang, Y., and Zhu, J. (2024). Vidu: a highly consistent, dynamic and skilled text-to-video generator with diffusion models.

[Barnes et al., 2023] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. (2023). *PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing*, pages 619–629.

[Bonet, 1999] Bonet, J. S. D. (1999). Poxels: Probabilistic voxelized volume reconstruction.

[Broadhurst et al., 2001] Broadhurst, A., Drummond, T., and Cipolla, R. (2001). A probabilistic framework for space carving. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 388–393 vol.1.

[Cao and Johnson, 2023] Cao, A. and Johnson, J. (2023). Hexplane: A fast representation for dynamic scenes.

[Chen et al., 2023] Chen, H., Li, C., and Lee, G. H. (2023). Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance.

[Cheng et al., 2023] Cheng, W., Chen, R., Yin, W., Fan, S., Chen, K., He, H., Luo, H., Cai, Z., Wang, J., Gao, Y., Yu, Z., Lin, Z., Ren, D., Yang, L., Liu, Z., Loy, C. C., Qian, C., Wu, W., Lin, D., Dai, B., and Lin, K.-Y. (2023). Dna-rendering: A diverse neural actor repository for high-fidelity human-centric rendering.

[Dai et al., 2024] Dai, P., Xu, J., Xie, W., Liu, X., Wang, H., and Xu, W. (2024). High-quality surface reconstruction using gaussian surfels.

[Duan et al., 2024] Duan, Y., Wei, F., Dai, Q., He, Y., Chen, W., and Chen, B. (2024). 4d-rotor gaussian splatting: Towards efficient novel view synthesis for dynamic scenes.

[Eftekhar et al., 2021] Eftekhar, A., Sax, A., Bachmann, R., Malik, J., and Zamir, A. (2021). Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans.

[Furukawa and Ponce, 2010] Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32:1362–76.

[Galliani et al., 2015] Galliani, S., Lasinger, K., and Schindler, K. (2015). Massively parallel multiview stereopsis by surface normal diffusion. pages 873–881.

[Gao et al., 2024] Gao, Q., Xu, Q., Cao, Z., Mildenhall, B., Ma, W., Chen, L., Tang, D., and Neumann, U. (2024). Gaussianflow: Splatting gaussian dynamics for 4d content creation.

[Guo et al., 2024] Guo, Z., Zhou, W., Li, L., Wang, M., and Li, H. (2024). Motion-aware 3d gaussian splatting for efficient dynamic scene reconstruction.

[Guédon and Lepetit, 2023] Guédon, A. and Lepetit, V. (2023). Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering.

[Huang et al., 2024] Huang, B., Yu, Z., Chen, A., Geiger, A., and Gao, S. (2024). 2d gaussian splatting for geometrically accurate radiance fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, SIGGRAPH '24, page 1–11. ACM.

[Kazhdan and Hoppe, 2013] Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3).

[Kerbl et al., 2023] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Li et al., 2024] Li, Z., Chen, Z., Li, Z., and Xu, Y. (2024). Spacetime gaussian feature splatting for real-time dynamic view synthesis.

[Liu et al., 2024] Liu, I., Su, H., and Wang, X. (2024). Dynamic gaussians mesh: Consistent mesh reconstruction from monocular videos.

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169.

[Lyu et al., 2024] Lyu, X., Sun, Y.-T., Huang, Y.-H., Wu, X., Yang, Z., Chen, Y., Pang, J., and Qi, X. (2024). 3dgsr: Implicit surface reconstruction with 3d gaussian splatting.

[Ma et al., 2024] Ma, S., Luo, Y., Yang, W., and Yang, Y. (2024). Mags: Reconstructing and simulating dynamic 3d objects with mesh-adsorbed gaussian splatting.

[Merrill and Grimshaw, 2011] Merrill, D. and Grimshaw, A. (2011). High performance and scalable radix sorting: a case study of implementing dynamic parallelism for gpu computing. *Parallel Processing Letters*, 21:245–272.

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis.

[Müller et al., 2022] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15.

[Oechsle et al., 2021] Oechsle, M., Peng, S., and Geiger, A. (2021). Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction.

[Schonberger and Frahm, 2016] Schonberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Seitz and Dyer, 1997] Seitz, S. and Dyer, C. (1997). Photorealistic scene reconstruction by voxel coloring. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1067–1073.

[Sinha and Pollefeys, 2005] Sinha, S. and Pollefeys, M. (2005). Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. volume 1, pages 349–356.

[Sun et al., 2024] Sun, J., Jiao, H., Li, G., Zhang, Z., Zhao, L., and Xing, W. (2024). 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos.

[Teed and Deng, 2020] Teed, Z. and Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow.

[Wang et al., 2023a] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2023a). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction.

[Wang et al., 2023b] Wang, Y., Han, Q., Habermann, M., Daniilidis, K., Theobalt, C., and Liu, L. (2023b). Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction.

[Wang et al., 2024] Wang, Y., Wang, X., Chen, Z., Wang, Z., Sun, F., and Zhu, J. (2024). Vidu4d: Single generated video to high-fidelity 4d reconstruction with dynamic gaussian surfels.

[Westover, 1991] Westover, L. A. (1991). *Splatting - A parallel, feed-forward volume rendering algorithm.* PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC.

[Wolf et al., 2024] Wolf, Y., Bracha, A., and Kimmel, R. (2024). Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views.

[Wu et al., 2024] Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., and Wang, X. (2024). 4d gaussian splatting for real-time dynamic scene rendering.

[Wu et al., 2020] Wu, M., Wang, Y., Hu, Q., and Yu, J. (2020). Multi-view neural human rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1682–1691.

[Xu et al., 2023] Xu, Z., Peng, S., Lin, H., He, G., Sun, J., Shen, Y., Bao, H., and Zhou, X. (2023). 4k4d: Real-time 4d view synthesis at 4k resolution.

[Yang et al., 2024] Yang, Z., Yang, H., Pan, Z., and Zhang, L. (2024). Real-time photo-realistic dynamic scene representation and rendering with 4d gaussian splatting.

[Yao et al., 2018] Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. (2018). Mvsnet: Depth inference for unstructured multi-view stereo.

[Yariv et al., 2021] Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021). Volume rendering of neural implicit surfaces.

[Yu et al., 2024] Yu, Z., Sattler, T., and Geiger, A. (2024). Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes.

[Zhao et al., 2022] Zhao, F., Jiang, Y., Yao, K., Zhang, J., Wang, L., Dai, H., Zhong, Y., Zhang, Y., Wu, M., Xu, L., and Yu, J. (2022). Human performance modeling and rendering via neural animated mesh.

[Zwicker et al., 2001] Zwicker, M., Pfister, H., van Baar, J., and Gross, M. (2001). Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538.