Towards Geometric Reasoning for Dynamic and Unconstrained 3D Scenes

Joel Julin

CMU-RI-TR-25-72

July 31, 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Dr. László Jeni *(chair)*
Dr. Jun-Yan Zhu
Mosam Dabhi

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my family, friends, and mentors.

Abstract

Dynamic scene reconstruction is fundamental to real-time photorealistic rendering for applications such as virtual reality, telepresence, and robotic perception. Although early methods like Neural Radiance Fields (NeRFs) have achieved impressive results in novel view synthesis, their heavy computational demands severely hinder real-time deployment. This thesis presents four novel methods that collectively advance dynamic 3D scene reconstruction, offering substantial improvements in speed, controllability, visual fidelity, and the robustness to handle challenging, unconstrained environments.

First, we propose **DyLiN**, the first Light Field Network (LFN) framework capable of modeling dynamic scenes with topological variation. DyLiN incorporates a ray-based deformation MLP and hyperspace lifting network, trained via knowledge distillation from dynamic NeRFs, achieving superior visual quality with order-of-magnitude faster inference. Next, we develop **CoGS**, an extension of 3D Gaussian Splatting (3DGS) that enables dynamic, controllable scene modeling without dense supervision. CoGS leverages explicit Gaussian representations alongside novel regularization losses to allow reasonable manipulation of scene elements.

We further present **GS-LK**, a hybrid method integrating Gaussian Splatting with a differentiable Lucas-Kanade pipeline for improved geometric consistency in monocular video. Finally, **Focus4DGS** introduces uncertainty-aware modeling in 4D Gaussian Splatting to enhance robustness in uncontrolled environments. It achieves this by separating high-confidence reconstructions from ambiguous motion regions, effectively finding the subset of dynamic objects that can be reconstructed from those that cannot.

Collectively, these contributions advance the field toward real-time, high-fidelity dynamic scene modeling by addressing core limitations of existing methods. Extensive experiments on synthetic and real datasets demonstrate superior performance in rendering speed, image quality, and scene controllability, making these methods suitable for deployment in interactive and resource-constrained applications.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. László Jeni, for his unwavering mentorship, vision, and support throughout my time at CMU. Over the past three years, Dr. Jeni has played a pivotal role in shaping my academic trajectory, from my first research internship in the Robotics Institute Summer (RISS) program to my transition into the Ph.D. program beginning in Fall 2025. His technical insight, patience, and encouragement continually pushed me to grow as a researcher and a thinker. I feel incredibly fortunate to have had such a thoughtful and dedicated mentor.

I am also deeply grateful to Dr. Simon Lucey, who first introduced me to the field and gave me the opportunity to work with him during my undergraduate studies. That year-long collaboration not only sparked my interest in computer vision and robotics but also laid the foundation for everything that followed. His early guidance and trust were instrumental in putting me on this path.

I would like to thank my thesis committee members, Dr. László Jeni, Dr. Jun-Yan Zhu, and Mosam Dabhi, for their valuable time, insightful feedback, and diverse perspectives. Their input was essential in refining the direction and clarity of this work.

I am also especially thankful to Rachel Burcin and Dr. John Dolan for organizing and leading the RISS program. Participating in RISS was one of the most formative experiences of my career.

This thesis would not have been possible without the contributions of my collaborators. I am especially grateful to Heng Yu, Louise Xie, Zoltán A. Milacski, Ananya Bal, Koichiro Niinuma, and Mose Sakashita whose insights, creativity, and dedication elevated every part of this work. It was a privilege to work with such talented and motivated individuals.

Finally, I am profoundly grateful to my family and friends for their constant encouragement and patience. Their belief in me has been a motivating force throughout this process.

Funding

This work was supported by Fujitsu.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recent advancements in machine vision have significantly improved our ability to reconstruct 3D scenes from 2D observations. A key driver of this progress has been the development of coordinate-based networks, particularly Neural Radiance Fields (NeRFs), which represent scenes implicitly using multi-layer perceptrons (MLPs) to enable high-fidelity novel view synthesis. This is achieved by synthesizing views through volumetric integration along cast rays. While powerful, this reliance on per-ray numerical sampling is computationally intensive and poorly suited for real-time applications.

To overcome this limitation, one line of research has focused on accelerating NeRFs through techniques like voxel pruning and multi-resolution grids. A more fundamental approach, taken by Light Field Networks (LFNs), bypasses the costly integration step entirely by directly regressing a ray's final color with a single MLP evaluation. This yields significantly faster inference but, prior to this work, no method had successfully extended LFNs to dynamic scenes, which require the modeling of time-varying geometry and appearance.

While the LFN paradigm had yet to bridge this gap, research in the NeRF space did push into the dynamic domain. These dynamic NeRF variants typically address the temporal challenge by introducing deformation fields or hyperspace embeddings to represent motion and topology changes. While expressive, these models remain

computationally heavy and are impractical for real-time deployment. Moreover, few methods address topological variation or achieve controllability in dynamic scenes.

In contrast, explicit scene representations like 3D Gaussian Splatting (3DGS) offer a compelling alternative. 3DGS represents scenes as collections of 3D Gaussians projected onto image planes using differentiable rasterization. This explicit formulation enables faster rendering, simpler optimization, and interpretable scene elements. Although early efforts have extended 3DGS to dynamic scenes, they have not fully exploited the representation's explicit structure to address key challenges like fine-grained controllability, geometric robustness under sparse views, or the visual ambiguities inherent in unconstrained environments.

This thesis pushes the boundaries of dynamic 3D scene reconstruction by systematically addressing these challenges. It introduces four novel methods with the collective goal of making dynamic reconstruction not only faster and more controllable, but also more effective at modeling highly dynamic regions from limited camera views and more robust to visual distractors.

1.2 Contributions

To address these challenges, this thesis presents the following four core contributions, each targeting a critical aspect of the overall goal:

- **DyLiN (Dynamic Light Field Network):** A novel extension of LFNs for dynamic scenes. DyLiN models non-rigid deformation and topological variation using a ray-based deformation MLP and a hyperspace lifting network. It is trained via knowledge distillation from pretrained dynamic NeRFs and achieves high image quality with an order-of-magnitude faster rendering time than the teacher models and other dynamic NeRF baselines.
- **CoGS (Controllable Gaussian Splatting):** A dynamic scene reconstruction method that extends 3D Gaussian Splatting to time-varying scenes. CoGS introduces multiple regularization losses to ensure spatial-temporal consistency, and leverages the explicit nature of Gaussians to enable intuitive, fine-grained control of scene elements without requiring labeled control signals or dense supervision.

- **GS-LK (Gaussian Splatting with Lucas-Kanade):** A hybrid method that combines Gaussian Splatting with a differentiable Lucas-Kanade optimization pipeline. GS-LK improves geometric consistency and motion tracking in monocular video scenarios where limited camera movement makes dynamic 3D reconstruction particularly challenging.
- **Focus4DGS (Distractor-Free 4D Gaussian Splatting):** A framework for robust dynamic scene reconstruction that explicitly models uncertainty. Focus4DGS decomposes the scene into a primary set of dynamic Gaussians to represent the core, consistent motion, and auxiliary per-view Gaussians that account for difficult to reconstruct regions. A novel uncertainty-guided loss function drives this decomposition. This results in an effective separation between well-reconstructed geometry and ambiguous regions, enabling reliable dynamic scene modeling even in challenging, unconstrained conditions.

Together, these methods offer scalable, high-fidelity, and controllable 3D dynamic scene reconstruction pipelines suitable for real-time applications.

1.3 Overview

The remainder of this thesis is structured to guide the reader from foundational concepts to our novel contributions in dynamic scene reconstruction.

- **Chapter 2** reviews the Background on foundational rendering and scene representation techniques, providing the necessary context for the contributions that follow.
- **Chapter 3** begins the technical contributions with DyLiN [77], a method that accelerates dynamic scene reconstruction by introducing the first dynamic Light Field Network to achieve significant gains in rendering speed.
- **Chapter 4** then transitions from implicit to explicit representations with CoGS [76], a framework that leverages the structure of 3D Gaussian Splatting to enable direct, intuitive control over dynamic scene elements
- **Chapter 5** tackles the challenge of insufficient geometric constraints by presenting GS-LK [71], a hybrid method that improves motion tracking in monocular

videos with limited camera movement by integrating classical optical flow principles into the 3DGS framework.

- **Chapter 6** addresses the final challenge of real-world capture by introducing Focus4DGS. This framework learns to model uncertainty in order to separate high-confidence geometry from visual distractors, enhancing robustness in unconstrained environments.
- **Chapter 7** concludes the thesis by summarizing our key findings, discussing the limitations of the proposed methods, and offering suggestions for future research.

Chapter 2

Background

2.1 Rendering

The reconstruction of 3D scenes from 2D observations is fundamentally tied to the method used to render an image from a scene representation. Two dominant paradigms to this thesis: volumetric rendering, which synthesizes images by integrating properties through space, and primitive-based rasterization, which projects explicit geometry onto an image plane.

2.1.1 Volumetric Rendering

Volumetric rendering is the foundational technique behind Neural Radiance Fields (NeRFs). In this paradigm, an image is formed by casting a camera ray through each pixel and numerically integrating color and density contributions along that ray. The color for a single pixel, C, is calculated by accumulating the color c_i of samples along the ray, weighted by their density σ along the ray from a near plane t_n to a far plane t_f. This continuous integral is expressed as

$$C(r) = \int_{t_n}^{t_f} T(t)\,\sigma(r(t))\,c(r(t), d)\,dt, \quad \text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))\,ds\right) \tag{2.1}$$

Here, $T(t)$ represents the transmittance from the near plane t_n to point t without being occluded.

In practice, this continuous integral is approximated with:

$$C = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) c_i, \quad \text{with } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \tag{2.2}$$

δ_i is the distance between samples. While producing high fidelity reconstructions of captured scenes, this process is computationally expensive as it requires sampling and evaluating a network at hundreds of points for every single pixel in a given image.

2.1.2 Primitive-based Rendering

In contrast to volumetric rendering, primitive-based rasterization offers a more direct and more efficient approach. This method represents scenes as a collection of explicit geometric primitives. In the case of 3D Gaussian Splatting (3DGS), these are anisotropic 3D Gaussians. To produce an image, these primitives are projected from 3D to 2D using differentiable rasterization. The final color for each pixel is then determined by first sorting the primitives in reverse-depth ordering and blending them together.

Because the primitives are explicitly defined, their exact locations in 3D space are known, eliminating the need for expensive sampling along rays. This leads to significantly faster rendering, making the method especially well-suited for real-time and interactive applications, such as virtual reality (VR).

2.2 Scene Representations

With the fundamental rendering paradigms established, we now introduce the specific scene representation techniques that are central to this thesis. These methods define how a 3D scene's geometry and appearance are captured.

2.2.1 Neural Radiance Fields (NeRFs)

Neural Radiance Fields (NeRFs) [36] have emerged as a cornerstone of 3D scene reconstruction, known for synthesizing high-fidelity novel views from a sparse set of 2D images. NeRFs model the volumetric radiance field of a scene by training a

Multi-Layer Perceptron (MLP) that maps 3D coordinates and viewing directions to density and color. Specifically, this network maps a 3D coordinate x and a 2D viewing direction d to a volume density d and an emitted color c. An image is then rendered using the volumetric rendering technique described in 2.1.1.

While NeRFs offer high fidelity, they are slow due to the need to sample and evaluate hundreds of points per ray, making real-time applications impractical. Subsequent works have extended NeRFs to handle dynamic scenes by introducing deformation fields or hyperspace embeddings to model motions and changes in topology. However, these dynamic NeRF variants remain computationally expensive.

2.2.2 Light Field Networks (LFNs)

To address the computational inefficiency of NeRFs, Light Field Networks (LFNs) were proposed as an alternative that replaces the costly numerical integration process. LFNs are MLPs trained to directly regress the final color of a camera ray from the ray's parameters alone, requiring only a single network evaluation per-pixel. With only a single network evaluation per-ray, this approach is significantly faster than standard NeRF rendering.

The performance of an LFN is highly dependent on how rays are represented. Various parameterizations have been explored in the literature, each affecting generalization and rendering quality. Of particular relevance to this work is R2L [62], which employs a deep residual MLP architecture trained via knowledge distillation from a pretrained NeRF. This method achieves superior image quality at a fraction of the rendering cost, making it a compelling alternative for efficient scene synthesis.

2.2.3 3D Gaussian Splatting

More recently, 3D Gaussian Splatting (3DGS) [21] was introduced as a highly efficient explicit scene representation technique. Unlike the implicit, network-based representations of NeRFs and LFNs, 3DGS models a scene using a large collection of 3D Gaussians. Each Gaussian is explicitly defined by a full 3D covariance matrix $\mathbf{\Sigma}$, position (mean) μ, opacity α, and color represented via spherical harmonics (SH). To render these Gaussians, projecting them from 3D to 2D Gaussians, we follow the

procedure outlined in [80] to obtain the view space covariance matrix $\mathbf{\Sigma}'$:

$$\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^T\mathbf{J}^T, \tag{2.3}$$

where $\mathbf{W}$ is the view transform and $\mathbf{J}$ is the Jacobian of the affine approximation of the projective transformation.

Since the physical meaning of a covariance matrix is only valid if it is positive semi-definite, it cannot be easily optimized to best represent a scene's radiance field [21]. However, we can obscure this complexity by employing a parameterization that inherently maintains the positive semi-definiteness of the matrix. The covariance matrix Σ can be decomposed into intuitive and optimizable components that correspond to an ellipsoid's scaling and orientation with rotation matrix $\mathbf{R}$ and scaling matrix $\mathbf{S}$:

$$\mathbf{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T. \tag{2.4}$$

and optimize $\mathbf{R}$, $\mathbf{S}$ instead of $\mathbf{\Sigma}$. After projection, the now 2D Gaussians are sorted from front-to-back and blended together using an alpha-blending rule similar to the volumetric rendering equation in 2.1.1 to produce the final pixel color. During optimization, the number of 3D Gaussians is adaptively controlled to best represent the scene. For a more comprehensive outline of this procedure, we kindly ask the readers to refer to [21].

2.3 Modeling Scene Dynamics

Representing a scene that changes over time introduces the significant challenge of modeling motion, deformation, and potential changes in topology. A dominant paradigm for tackling this challenge, applied to both implicit and explicit representations, is the use of a canonical space mapped to a specific time via a deformation field.

- The **canonical space** serves as a static, time-independent 3D template of the scene. It encodes the underlying geometry and appearance, acting as a consistent reference frame throughout the temporal evolution of the scene.
- The **deformation field**, commonly implemented as an MLP and also referred

to as a forward warp field, defines the motion. In the context of this thesis, this field primarily functions as a forward warp, which deforms the geometry from the static canonical space to its dynamic position in the world at a given time t. This allows a single, learned canonical model to represent the scene across its entire evolution.

This canonical-to-deformed formulation is central to many dynamic scene modeling methods. It underpins dynamic NeRF variants such as D-NeRF [46], as well as explicit approaches like DeformableGS [73] and 4DGaussians [70], which extend 3DGS to dynamic environments.

Chapter 3

Making Light Field Networks Dynamic

Addressing the limitations of existing methods for dynamic scene reconstruction, this thesis first explores extending Light Field Networks (LFNs) to dynamic settings. While LFNs offer significant advantages in inference speed and computational efficiency, adapting them to handle time-varying geometry and complex topological changes remains an open challenge. To bridge this gap, this chapter introduces DyLiN, a novel framework that enhances LFNs with deformation fields, hyperspace embeddings, and knowledge distillation from dynamic NeRF models. DyLiN significantly accelerates dynamic scene reconstruction and achieves high visual fidelity without the computational overhead inherent in previous approaches.

3.1 Introduction

Machine vision has made tremendous progress with respect to reasoning about 3D structure using 2D observations. Much of this progress can be attributed to the emergence of coordinate networks [9, 34, 41], such as Neural Radiance Fields (NeRF) [36] and its variants [2, 33, 37, 65]. They provide an object agnostic representation for 3D scenes and can be used for high-fidelity synthesis for unseen views. While NeRFs mainly focus on static scenes, a series of works [13, 42, 46, 56] extend the idea to dynamic cases via additional components that map the observed deformations to a

canonical space, supporting moving and shape-evolving objects. It was further shown that by lifting this canonical space to higher dimensions the method can handle changes in scene topology as well [43].

However, the applicability of NeRF models is considerably limited by their computational complexities. From each pixel, one typically casts a ray from that pixel, and numerically integrates the radiance and color densities computed by a Multi-Layer Perceptron (MLP) across the ray, approximating the pixel color. Specifically, the numerical integration involves sampling hundreds of points across the ray, and evaluating the MLP at all of those locations.

Several works have been proposed for speeding up static NeRFs. These include employing a compact 3D representation structure [12, 30, 75], breaking up the MLP into multiple smaller networks [47, 48], leveraging depth information [10, 38], and using fewer sampling points [29, 38, 72]. Yet, these methods still rely on integration and suffer from sampling many points, making them prohibitively slow for real-time applications. Recently, Light Field Networks (LFNs) [53] proposed replacing integration with a direct ray-to-color regressor, trained using the same sparse set of images, requiring only a single forward pass. R2L [62] extended LFNs to use a very deep residual architecture, trained by distillation from a NeRF teacher model to avoid overfitting. In contrast to static NeRF acceleration, speeding up dynamic NeRFs is a much less discussed problem in the literature. This is potentially due to the much increased difficulty of the task, as one also has to deal with the high variability of motion. In this direction, [11, 63] greatly reduce the training time by using well-designed data structures, but their solutions still rely on integration. LFNs are clearly better suited for acceleration, yet, to the best of our knowledge, no works have attempted extending LFNs to the dynamic scenario.

To address this gap, this chapter introduces DyLiN, a framework that extends Light Field Networks to model dynamic scenes with non-rigid deformations and topological changes. Our method incorporates a deformation field and a hyperspace representation, distilling knowledge from a pretrained dynamic NeRF teacher model to learn the complex dynamics. We will demonstrate through empirical experiments on synthetic and real datasets that DyLiN achieves superior image quality and an order-of-magnitude faster rendering speed when compared to its NeRF teacher and the state-of-the-art TiNeuVox method. The chapter will also present detailed ablation

studies to validate the effectiveness of each component in the DyLiN architecture.

3.2 Related Works

Dynamic NeRFs. Extending NeRFs to dynamic (deformable) domains has sparked considerable research interest [13, 42, 43, 46, 56]. Among these works, the ones that most closely resemble ours are D-NeRF [46] and HyperNeRF [43]. D-NeRF uses a translational deformation field with temporal positional encoding. HyperNeRF introduces a hyperspace representation, allowing topological variations to be effectively captured. Our work expands upon these works, as we propose DyLiN, a similar method for LFNs. We use the above dynamic NeRFs as pretrained teacher models for DyLiN, achieving better fidelity with orders of magnitude shorter rendering times.

Accelerated NeRFs. The high computational complexity of NeRFs has motivated several follow-up works on speeding up the numerical integration process. The following first set of works are restricted to static scenarios. NSVF [30] represents the scene with a set of voxel-bounded MLPs organized in a sparse voxel octree, allowing voxels without relevant content to be skipped. KiloNeRF [48] divides the scene into a grid and trains a tiny MLP network for each cell within the grid, saving on pointwise evaluations. AutoInt [29] reduces the number of point samples for each ray using learned partial integrals. In contrast to the above procedures, speeding up dynamic NeRFs is much less discussed in the literature, as there are only 2 papers published on this subject. Wang *et al.* [63] proposed a method based on Fourier plenoctrees for real-time dynamic rendering, however, the technique requires an expensive rigid scene capturing setup.

TiNeuVox [11] reduces training time by augmenting the MLP with time-aware voxel features and a tiny deformation network, while using a multi-distance interpolation method to model temporal variations. Interestingly, all of the aforementioned methods suffer from sampling hundreds of points during numerical integration, and none of them support changes in topology, whereas our proposed DyLiN excels from both perspectives.

Knowledge Distillation. The process of training a student model with synthetic data generated by a teacher model is called Knowledge Distillation (KD) [4], and it has been widely used in the vision and language domains [6, 24, 61, 64] as a form of data augmentation. Like R2L [62], we also use KD for training, however, our teacher and student models are both dynamic and more complex than their R2L counterparts.

3.3 Methods

In this section, we present our solution for extending LFNs to the dynamic scenario. We propose DyLiN, supporting dynamic deformations and hyperspace representations via two respective MLPs. We use KD to train DyLiN with synthetic data generated by a pretrained dynamic NeRF teacher model.

3.3.1 Network Architecture

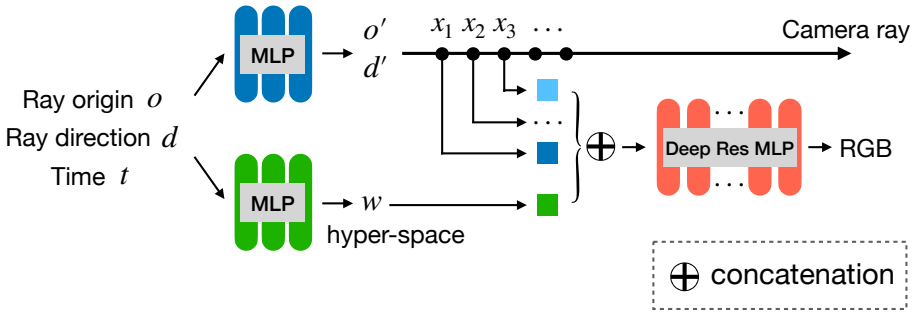


Figure 3.1: Schematic diagram of our proposed DyLiN architecture. We take a ray $r = (o, d)$ and time t as input. We deform r into $r' = (o', d')$, and sample few points x_k, $k = 1, \ldots, K$ along r' to encode it (blue). In parallel, we also lift r and t to the hyperspace code w (green), and concatenate it with each x_k. We use the concatenation to regress the RGB color of r at t directly (red).

Our overall DyLiN architecture G_ϕ is summarized in fig. 3.1. It processes rays instead of the widely adopted 3D point inputs as follows.

Specifically, our deformation MLP T_ω maps an input ray $r = (o, d)$ to canonical space ray $r' = (o', d')$:

$$(o', d') = T_\omega(o, d, t). \tag{3.1}$$

Unlike the pointwise deformation MLP proposed in Nerfies [42], which bends rays by offsetting their points independently, our MLP outputs rays explicitly, hence no ray bending occurs. Furthermore, after obtaining r', we encode it by sampling and concatenating K points along it.

Our hyperspace MLP H_ψ is similar to T_ω, except it outputs a hyperspace representation w:

$$w = H_\psi(o, d, t). \tag{3.2}$$

In contrast to HyperNeRF [43], which predicts a hyperspace code w for each 3D point, we use rays and compute a single w for each ray.

Both MLPs further take the index t as input to encode temporal deformations.

Once the K points and w are obtained, we concatenate them and feed the result into our LFN R_π, which is a deep residual color MLP regressor. Overall, we can collect the model parameters as $\phi = [\omega, \psi, \pi]$.

Note that without our two MLPs T_ω and H_ψ, our DyLiN falls back to the vanilla LFN.

3.3.2 Training Procedure

Our training procedure is composed of 3 phases.

First, we pretrain a dynamic NeRF model F_θ (e.g., D-NeRF [46] or HyperNeRF [43]) by randomly sampling time t and input ray r, and minimizing the Mean Squared Error (MSE) against the corresponding RGB color of monocular target video I:

$$\min_\theta \mathbb{E}_{t,r=(o,d)} \left[\|F_\theta(o, d, t) - I(o, d, t)\|_2^2 \right]. \tag{3.3}$$

Recall, that F_θ is slow, as it performs numerical integration across the ray $r = (o, d)$.

Second, we employ the newly obtained F_{θ^*} as the teacher model for our DyLiN student model G_ϕ via KD. Specifically, we minimize the MSE loss against the respective pseudo ground truth ray color generated by F_{θ^*} across S ray samples:

$$\min_\phi \mathbb{E}_{t,r=(o,d)} \left[\|G_\phi(o, d, t) - F_{\theta^*}(o, d, t)\|_2^2 \right], \tag{3.4}$$

yielding $G_{\tilde{\phi}}$. Note how this is considerably different from R2L [62], which uses a

static LFN that is distilled from a static NeRF.

Finally, we initialize our student model G_ϕ with parameters $\tilde{\phi}$ and fine-tune it using the original real video data:

$$\min_{\phi,\, \phi_0=\tilde{\phi}} \mathbb{E}_{t,r=(o,d)} \left[\|G_\phi(o,d,t) - I(o,d,t)\|_2^2 \right], \tag{3.5}$$

obtaining ϕ^*.

3.4 Experimental Setup

3.4.1 Datasets

To test our hypotheses, we performed experiments on three types of dynamic scenes: synthetic, real and real controllable.

Synthetic Scenes. We utilized the synthetic 360° dynamic dataset introduced by [46], which contains 8 animated objects with complicated geometry and realistic non-Lambertian materials. Each dynamic scene consists of 50 to 200 training images and 20 testing images. We used 400×400 image resolution. We applied D-NeRF [46] as our teacher model with the publicly available pretrained weights.

Real Scenes. We collected real dynamic data from 2 sources. First, we utilized 5 topologically varying scenes provided by [43] (Broom, 3D Printer, Chicken, Americano and Banana), which were captured by a rig encompassing a pole with two Google Pixel 3 phones rigidly attached roughly 16 cm apart. Second, we collected human facial videos using an iPhone 13 Pro camera. We rendered both sets at 960×540 image resolution. We pretrained a HyperNeRF [43] teacher model from scratch for each scene.

3.4.2 Settings

Throughout our experiments, we use the settings listed below, many of which follow [62].

In order to retain efficiency, we define T_ω and H_ψ to be small MLPs, with T_ω

consisting of 7 layers of 128 units with $r' \in \mathbb{R}^6$, and H_ψ having 6 layers of 64 units with $w \in \mathbb{R}^8$. Then, we use $K = 16$ sampled points to represent rays, where sampling is done randomly during training and evenly spaced during inference.

Contrary to T_ω and H_ψ, our LFN R_π is a very deep residual color MLP regressor, containing 88 layers with 256 units per layer, in order to have enough capacity to learn the video generation process.

We generate rays within Eqs. 3.3 to 3.5 by sampling ray origins $o = (x_o, y_o, z_o)$ and normalized directions $d = (x_d, y_d, z_d)$ randomly from the uniform distribution U as follows:

$$x_o \sim U(x_o^{min}, x_o^{max}), \quad x_d \sim U(x_d^{min}, x_d^{max}), \tag{3.6}$$

$$y_o \sim U(y_o^{min}, y_o^{max}), \quad y_d \sim U(y_d^{min}, y_d^{max}), \tag{3.7}$$

$$z_o \sim U(z_o^{min}, z_o^{max}), \quad z_d \sim U(z_d^{min}, z_d^{max}), \tag{3.8}$$

where the min, max bounds of the 6 intervals are inferred from the original training video. In addition to uniform sampling, we also apply the hard example mining strategy suggested in [62] to focus on fine-grained details.

We used $S = 10\,000$ training samples during KD in (3.4). Subsequently, we also randomly sample time step t uniformly from the unit interval: $t \sim U(0, 1)$.

During training, we used Adam [22] with learning rate 5×10^{-4} and batch size 4096. We performed all experiments on single NVIDIA A100 GPUs.

3.4.3 Baseline Models

For testing our methods, we compared quality and speed against several baseline models, including NeRF [36], NV [31], NSFF [28], Nerfies [42], HyperNeRF [43], two variants of TiNeuVox [11], DirectVoxGo [55], Plenoxels [12], T-NeRF and D-NeRF [46], as well as CoNeRF [20].

In addition, we performed an ablation study by comparing against 2 simplified versions of our DyLiN architecture. First, we omitted both of our deformation and hyperspace MLPs and simply concatenated the time step t to the sampled ray points (essentially resulting in a dynamic R2L). This method is illustrated in fig. 3.2a. Second, we employed a pointwise deformation MLP (5 layers of 256 units) inspired by [46],

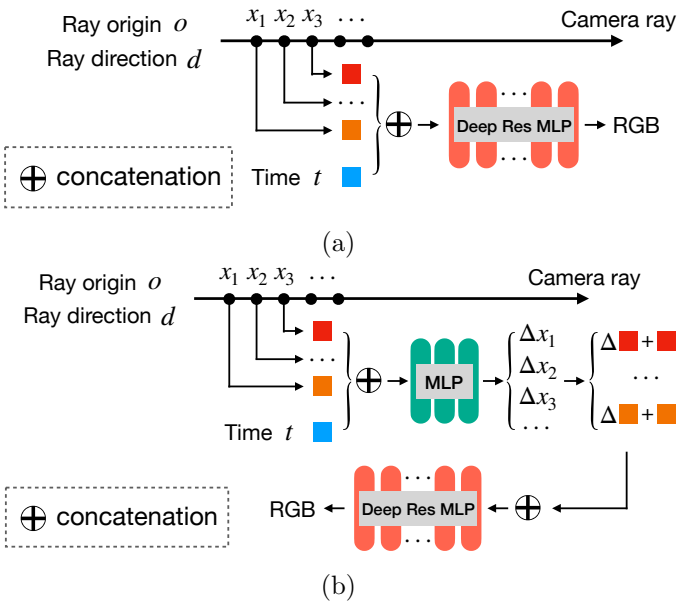


Figure 3.2: Our two ablated baseline models, omiting components of our DyLiN. (a) Without our two proposed MLPs. (b) Pointwise deformation MLP only, predicting offsets jointly.

which deforms points along a ray by predicting their offsets jointly, i.e., it can bend rays. This is contrast to our DyLiN, which deforms rays explicitly without bending and also applies a hyperspace MLP. This scheme is depicted in fig. 3.2b. In both baselines, the deep residual color MLP regressors were kept intact. Next, we also tested the effects of our fine-tuning procedure from (3.5) by training all of our models both with and without it. Lastly, we assessed the dependences on the number of sampled points along rays K and on the number of training samples S during KD in (3.4).

3.4.4 Evaluation Metrics

For quantitatively evaluating the quality of generated images, we calculated the Peak Signal-to-Noise Ratio (PSNR) [18] in decibels (dB), the Structural Similarity Index (SSIM) [40, 68], the Multi-Scale SSIM (MS-SSIM) [67] and the Learned Perceptual Image Patch Similarity (LPIPS) [78] metrics. Intuitively, PSNR is a pixelwise score, while SSIM and MS-SSIM also take pixel correlations and multiple scales into account, respectively, yet all of these tend to favor blurred images. LPIPS compares deep

neural representations of images and is much closer to human perception, promoting semantically better and sharper images.

Furthermore, for testing space and time complexity, we computed the storage size of parameters in megabytes (MB) and measured the wall-clock time in milliseconds (ms) while rendering the synthetic Lego scene with each model.

3.5 Results

3.5.1 Quantitative Results

table 3.1 and table 3.2 contain our quantitative results for reconstruction quality on synthetic and real dynamic scenes, accordingly. We found that among prior works, TiNeuVox-B performed the best on synthetic scenes with respect to each metric. On real scenes, however, NSFF took the lead. Despite having strong metrics, NSFF is qualitatively poor and slow. Surprisingly, during ablation, even our most basic model (DyLiN without the two MLPs from fig. 3.2a) could generate perceptually better looking images than TiNeuVox-B, thanks to the increased training dataset size via KD. Incorporating the MLPs T_ω and H_ψ into the model each improved results slightly. Interestingly, fine-tuning on real data as in (3.5) gave a substantial boost. In addition, our relative PSNR improvement over the teacher model (table 3.1=+1.93 dB, up to +3.16 dB per scene; table 3.2=+2.7 dB, up to +13.14 dB) is better than that of R2L [62] (+1.4 dB, up to +2.8 dB).

table 3.3 shows quantitative results for space and time complexity on the synthetic Lego scene. We found that there is a trade-off between the two metrics, as prior works are typically optimized for just one of those. In contrast, all of our proposed DyLiN variants settle at the golden mean between the two extremes. When compared to the strongest baseline TiNeuVox-B, our method requires 3 times as much storage but is nearly 2 orders of magnitude faster. Plenoxels and NV, the only methods that require less computation than ours, perform much worse in quality.

Table 3.1: Quantitative results on synthetic dynamic scenes. Notations: MLP = Multi-Layer Perceptron, PD = pointwise deformation, FT = fine-tuning. We utilized D-NeRF as the teacher model for our DyLiNs. Best results in **bold**.

Method	PSNR↑	SSIM↑	LPIPS↓
NeRF [36]	19.00	0.8700	0.1825
DirectVoxGo [55]	18.61	0.8538	0.1688
Plenoxels [12]	20.24	0.8688	0.1600
T-NeRF [46]	29.51	0.9513	0.0788
D-NeRF [46]	30.50	0.9525	0.0663
TiNeuVox-S [11]	30.75	0.9550	0.0663
TiNeuVox-B [11]	**32.67**	0.9725	0.0425
DyLiN, w/o two MLPs, w/o FT (ours)	31.16	0.9931	0.0281
DyLiN, w/o two MLPs (ours)	32.07	0.9937	0.0196
DyLiN, PD MLP only, w/o FT (ours)	31.26	0.9932	0.0279
DyLiN, PD MLP only (ours)	31.24	0.9940	0.0189
DyLiN, w/o FT (ours)	31.37	0.9933	0.0275
DyLiN (ours)	32.43	**0.9943**	**0.0184**

3.5.2 Qualitative Results

fig. 3.3 and fig. 3.4 depict qualitative results for reconstruction quality on synthetic and real dynamic scenes, respectively. Both show that our full DyLiN model generated the sharpest, most detailed images, as it was able to capture cloth wrinkles (fig. 3.3j) and the eye of the chicken (fig. 3.4e). The competing methods tended to oversmooth these features. We also ablated the effect of omitting fine-tuning (fig. 3.3i, fig. 3.4d), and results declined considerably.

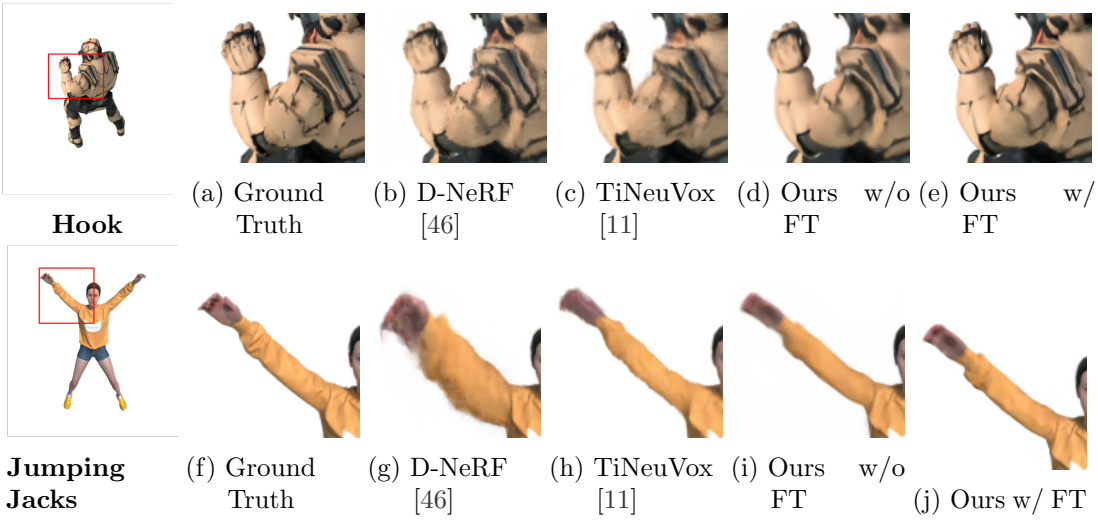
For the sake of completeness, fig. 3.5 illustrates qualitative ablation results for our model components on real dynamic scenes. We found that sequentially adding our two proposed MLPs T_ω and H_ψ improves the reconstruction, e.g., the gum between the teeth (fig. 3.5e) and the fingers (fig. 3.5j) become more and more apparent. Without the MLPs, these parts were heavily blurred (fig. 3.5c, fig. 3.5h).

3.6 Conclusion

We proposed an architecture for extending LFNs to dynamic scenes. Specifically, we introduced DyLiN, which models ray deformations without bending and lifts whole

Table 3.2: Quantitative results on real dynamic scenes. Notations: MLP = Multi-Layer Perceptron, PD = pointwise deformation, FT = fine-tuning. We used HyperNeRF as the teacher model for DyLiN. Best results in **bold**.

Method	**PSNR↑**	**MS-SSIM↑**
NeRF [36]	20.1	0.745
NV [31]	16.9	0.571
NSFF [28]	**26.3**	**0.916**
Nerfies [42]	22.2	0.803
HyperNeRF [43]	22.4	0.814
TiNeuVox-S [11]	23.4	0.813
TiNeuVox-B [11]	24.3	0.837
DyLiN, w/o two MLPs, w/o FT (ours)	23.8	0.882
DyLiN, w/o two MLPs (ours)	24.2	0.894
DyLiN, PD MLP only, w/o FT (ours)	23.9	0.885
DyLiN, PD MLP only (ours)	24.6	0.903
DyLiN, w/o FT (ours)	24.0	0.886
DyLiN (ours)	25.1	0.910



(a) Ground Truth (b) D-NeRF [46] (c) TiNeuVox [11] (d) Ours w/o FT (e) Ours w/ FT

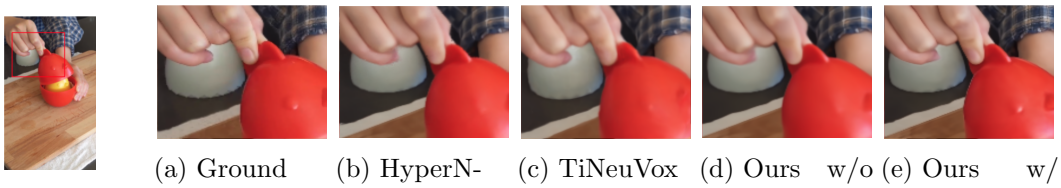
(f) Ground Truth (g) D-NeRF [46] (h) TiNeuVox [11] (i) Ours w/o FT (j) Ours w/ FT

Figure 3.3: Qualitative results on synthetic dynamic scenes. We compare our DyLiN (Ours-1, Ours-2) with the ground truth, the D-NeRF teacher model and TiNeuVox. Ours-1 and Ours-2 were trained without and with fine-tuning on the original data, respectively.

rays into a hyperspace. We trained this technique via knowledge distillation from various dynamic NeRF teacher models. We found that DyLiN produces state-of-the-art quality even without ray bending, while nearly being 2 orders of magnitude faster

Table 3.3: Quantitative results for space and time complexity on the synthetic Lego scene. Notations: MLP = Multi-Layer Perceptron, PD = pointwise deformation, FT = fine-tuning.

Method	**Storage (MB)**	**Wall-clock time (ms)**
NeRF [36]	5.00	2950.0
DirectVoxGo [55]	205.00	1090.0
Plenoxels [12]	717.00	50.0
NV [31]	439.00	74.9
D-NeRF [46]	4.00	8150.0
NSFF [28]	14.17	5450.0
HyperNeRF [43]	15.36	2900.0
TiNeuVox-S [11]	23.70	3280.0
TiNeuVox-B [11]	23.70	6920.0
DyLiN, w/o two MLPs, w/o FT (ours)	68.04	115.4
DyLiN, w/o two MLPs (ours)	68.04	115.4
DyLiN, PD MLP only, w/o FT (ours)	72.60	115.7
DyLiN, PD MLP only (ours)	72.60	115.7
DyLiN, w/o FT (ours)	70.11	116.0
DyLiN (ours)	70.11	116.0

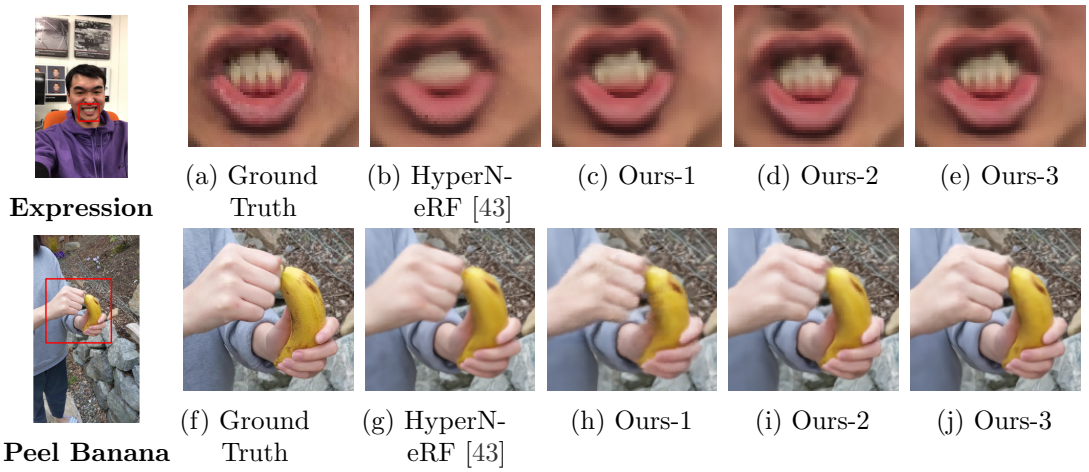


Chicken (a) Ground Truth (b) HyperNeRF [43] (c) TiNeuVox [11] (d) Ours w/o FT (e) Ours w/ FT

Figure 3.4: Qualitative results on a real dynamic scene. We compare our DyLiN (Ours-1, Ours-2) with the ground truth, the HyperNeRF teacher model and TiNeuVox. Ours-1 and Ours-2 were trained without and with fine-tuning on the original data, respectively.

than their strongest baselines.

Our method does not come without limitations, however. Most importantly, it focuses on speeding up inference, as it requires pretrained teacher models, which can be expensive to obtain. In some experiments, our solutions were outperformed in terms of the PSNR score. Using the winners as teacher models could improve performance. Additionally, distillation from multiple teacher models or joint training of the teacher and student models are also yet to be explored. Moreover, we currently



(a) Ground Truth (b) HyperNeRF [43] (c) Ours-1 (d) Ours-2 (e) Ours-3

(f) Ground Truth (g) HyperNeRF [43] (h) Ours-1 (i) Ours-2 (j) Ours-3

Figure 3.5: Qualitative results for ablation on real dynamic scenes. We compare our DyLiN (Ours-1, Ours-2, Ours-3) with the ground truth and the HyperNeRF teacher model. Ours-1 was trained without our two MLPs. Ours-2 was trained with pointwise deformation MLP only. Ours-3 is our full model with both of our proposed two MLPs.

represent rays implicitly by sampling K points along them, but increasing this number can lead to overfitting. An explicit ray representation may be more effective. Finally, voxelizing and quantizing our models could improve efficiency.

Our results are encouraging steps towards achieving real-time volumetric rendering and animation, and we hope that our work will contribute to the progress in these areas.

Chapter 4

Controllable Gaussian Splatting

Although DyLiN provides fast and high-quality dynamic scene reconstruction, many practical applications require more than visual fidelity alone. They demand direct, intuitive, and fine-grained control over dynamic elements. Such capabilities are particularly essential for interactive systems, animation, and scene editing, where implicit representations, such as DyLiN, offer limited flexibility. Building upon the advantages of explicit scene representations, this chapter introduces Controllable Gaussian Splatting (CoGS), a framework leveraging 3D Gaussian Splatting. By explicitly representing scenes through 3D Gaussians, CoGS enables straightforward and precise manipulation of dynamic elements without relying on dense supervision or pre-labeled control signals.

4.1 Introduction

Recent methods such as Neural Radiance Fields (NeRF) [36] have dramatically advanced our ability to reconstruct static 3D scenes, yet extending these methods effectively to dynamic scenes remains challenging. Implicit representations, such as those used by NeRF, complicate direct manipulation of scene elements, particularly in dynamic settings, as they require intricate mechanisms to represent time-varying geometries.

In contrast, explicit scene representations such as 3D Gaussian Splatting (3DGS) [21] and its variants significantly simplify the manipulation of scene elements by

directly modeling geometry and appearance using explicit 3D Gaussians. While Chapter 2 introduced the foundational aspects of Gaussian Splatting, here we explicitly extend this representation to dynamic scenes and introduce intuitive control mechanisms that enable direct, fine-grained manipulation without dense supervision or pre-labeled control signals.

We propose a novel framework that adapts GS for dynamic environments captured by a monocular camera, integrating control mechanisms that allow for intuitive and straightforward manipulation of scene elements. This development addresses the limitations of prior methods in terms of computational complexity and challenges in scene manipulation due to their implicit representation. By leveraging the explicit 3D Gaussian representations and combining them with advanced control techniques, our method opens new avenues for real-time, high-fidelity scene rendering and manipulation, particularly relevant in fields such as virtual reality, augmented reality, and interactive media.

4.2 Related Works

This work builds upon prior efforts in both dynamic and controllable scene synthesis.

The challenge of creating editable or re-animated dynamic scenes has primarily been explored within the NeRF framework. Methods like CoNeRF [20] have shown impressive results by introducing manually labeled control signals into a hyperspace framework to manipulate the rendered scene. However, a significant limitation of these approaches is their reliance on pre-computed or labeled control signals and masks. This requirement stems from the implicit nature of the neural radiance field and restricts their practical application.

Our work instead leverages the explicit nature of 3D Gaussian Splatting (3DGS), as discussed in Chapter 2. While concurrent research has successfully extended 3DGS to dynamic scenarios by adding networks to model motion , these approaches often follow paradigms from dynamic NeRFs and do not fully exploit the explicit structure of the Gaussians for scene manipulation. This thesis introduces a method for dynamic and controllable GS that takes full advantage of the explicit representation to overcome the supervision requirements of previous controllable methods.

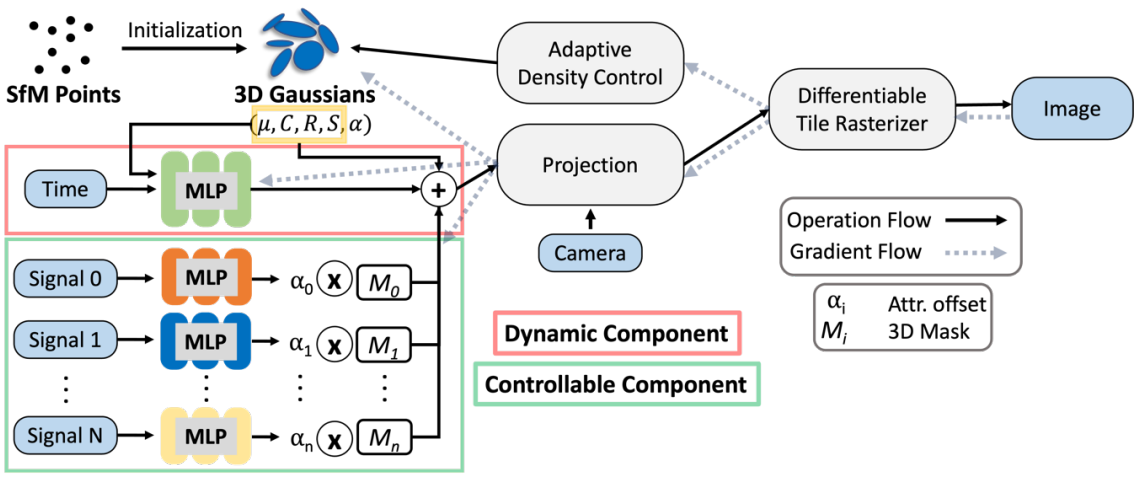


Figure 4.1: CoGS Overview. CoGS consists of two parts: Dynamic GS and Controllable GS. For Dynamic GS, an offset is learned for (μ, C, R, S) by separate MLPs (only one shown in figure). To extend to controllable scenarios, signals extracted from the dynamic model are used to obtain attribute offsets, which are then masked to affect the desired control region.

4.3 Methods

In order to realize controllable GS, it is essential to first establish a GS framework capable of modeling dynamic scenes. This chapter is dedicated to unfolding this process in two distinct phases: initially, we introduce the concept and methodology of dynamic GS. Subsequently, we build upon this foundation to evolve these methods into a controllable framework, thereby enhancing their adaptability and applicability in dynamic scene modeling.

4.3.1 Dynamic Gaussian Splatting

To represent dynamic scenes and enable fine-scale attribute control, we begin with the differentiable Gaussian rasterization pipeline proposed by Kerbl et al. [21], which we previously detailed in Chapter 2.

Our approach bridges the gap to dynamic scenarios by learning independent deformation networks for each parameter. Additionally, we introduce multiple losses to maintain geometric consistency across time. The pipeline overview is presented in Fig. 4.1, with the dynamic component highlighted in red.

We initialize a set of 3D Gaussians from a Structure from Motion (SfM) [52] point-cloud (or randomly selecting N points within the scene box), each defined by the same parameters as in Chapter 2. For the first 3000 iterations, our focus is exclusively on learning the static elements within the scene. This deliberate emphasis on stabilizing the static portions proves to be crucial for achieving high performance on the dynamic reconstruction. Establishing a robust static foundation lays the groundwork for a more accurate reconstruction of dynamic elements. During this phase, the deformation network (green MLP) does not update any parameters. Instead, these 3D Gaussians adhere to the same differentiable rasterization pipeline as covered in Chapter 2.

In the subsequent phases, the deformation network is employed to update each parameter, tailoring them to the dynamic scene. Although not explicitly depicted in Fig. 4.1, we learn j separate networks, one for each parameter. For ($\mu_\mathbf{i}$, $\mathbf{C}_i$, $\mathbf{R_i}$, $\mathbf{S_i}$), we have a network N_j such that:

$$N_j(\mu_\mathbf{i}, t) = (\Delta\mu_\mathbf{i}, \Delta\mathbf{C}_i, \Delta\mathbf{R_i}, \Delta\mathbf{S_i}), \tag{4.1}$$

where t is the current time step. Different from [73], we also learn an offset for color to account for any changes that may occur over time (e.g. shadows and reflections). The outputs from these networks are then added to the corresponding parameters, and the differentiable rasterization pipeline proceeds.

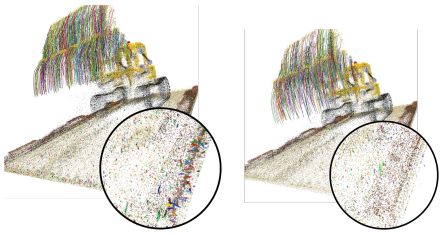
Learning offsets alone results in a method that is unaware of consistent trajectories and accurate movement. Thus, we employ our multiple regularization losses to further constrain this difficult problem.

For each time step, the mean of the normalized predicted position offsets ($\Delta\mu$) is computed to ensure their consistency with one another. Specifically, we use this to localize position offsets.

$$\mathcal{L}^{\text{norm}} = \frac{1}{N}\sum_{i=1}^{N} \|\Delta\mu_\mathbf{i}\|. \tag{4.2}$$

As shown in Fig. 4.2, the trajectories of static portions of the scene tend to stabilize with the addition of this loss.

After 15000 iterations, we enforce the remainder of our losses. Specifically, a local difference loss, denoted as $\mathcal{L}^{\text{diff}}$, is used to ensure the movement, or trajectory, for



(a) (b)

Figure 4.2: Lego synthetic scene visualized as a pointcloud of colored Gaussian centers. The smaller and fewer colored lines indicate less change in position over time. Adding $\mathcal{L}^{\text{norm}}$ stabilizes the static Gaussian's positions. (a) Without $\mathcal{L}^{\text{norm}}$. (b) With $\mathcal{L}^{\text{norm}}$.

each Gaussian is consistent with its neighbors over time. This loss is formulated as follows:

$$\mathcal{L}_{i,j}^{\text{diff}} = \left| \|\mu_{i,j,t} - \mu_{\mathbf{i,t}}\| - \|\mu_{\mathbf{i,j,t-1}} - \mu_{\mathbf{i,t-1}}\| \right|. \tag{4.3}$$

Here, $\mu_{i,j,t}$ represents the position of a nearest neighbor Gaussian j to Gaussian i at time t. Similarly, $\mu_{i,t}$ is the position of Gaussian i at time t, and analogous notation is used for the time step $t-1$.

The overall local difference loss is then defined as the average over all Gaussians i and their k-nearest neighbours:

$$\mathcal{L}^{\text{diff}} = \frac{1}{k|G|} \sum_{i \in G} \sum_{j \in \text{knn}_{i;k}} \mathcal{L}_{i,j}^{\text{diff}}. \tag{4.4}$$

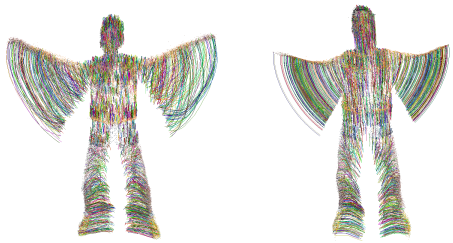
Demonstrated in Fig. 4.3a, this loss yields a much more consistent dynamic representation when compared to without it as shown in 4.3b.

The next two loss functions are directly taken from [32], for more in depth details, please refer to their paper. Each of these losses are assigned a weight through an unnormalized Gaussian weighting factor:

$$w_{i,j} = \exp\left(-\lambda_w \|\mu_{j,0} - \mu_{i,0}\|_2^2\right) \tag{4.5}$$

The distance between each Gaussian's position is computed at the first time step, and then it is fixed for the remaining part of the sequence. In doing this, each of the following losses are explicitly locally enforced.

Using this weighting scheme, a local-rigidity loss is employed, denoted as $\mathcal{L}^{\text{rigid}}$,



(a) (b)

Figure 4.3: Jumping Jack synthetic scene visualized as a pointcloud of colored Gaussian centers. The smaller and fewer colored lines indicate less change in position over time. Adding $\mathcal{L}^{\text{diff}}$ stabilizes the 3D Gaussian's trajectories. (a) Without $\mathcal{L}^{\text{diff}}$. (b) With $\mathcal{L}^{\text{diff}}$.

defined as follows:

$$\mathcal{L}_{i,j}^{\text{rigid}} = w_{i,j}||(\mu_{j,t-1} - \mu_{i,t-1}) - \mathbf{R}_{i,t-1}\mathbf{R}_{i,t}^{-1}(\mu_{j,t} - \mu_{i,t}||_2, \tag{4.6}$$

$$\mathcal{L}^{\text{rigid}} = \frac{1}{k|G|}\sum_{i\in G}\sum_{j\in \text{knn}_{i;k}} \mathcal{L}_{i,j}^{\text{rigid}}. \tag{4.7}$$

This loss ensures that for each Gaussian i, neighboring Gaussians j should move in a manner consistent with the rigid body transform of the coordinate system over time.

Additionally, we incorporate a rotational loss $\mathcal{L}^{\text{rot}}$ to maintain consistency in rotations among nearby Gaussians across different time steps. This is expressed as:

$$\mathcal{L}^{\text{rot}} = \frac{1}{k|G|}\sum_{i\in G}\sum_{j\in \text{knn}_{i;k}} w_{i,j}\|\hat{q}_{j,t}\hat{q}_{j,t-1}^{-1} - \hat{q}_{i,t}\hat{q}_{i,t-1}^{-1}\|_2, \tag{4.8}$$

where $\hat{q}$ is the normalized quaternion rotation of each Gaussian. The same k-nearest neighbors are used, as in the preceding losses.

Each of the described losses is critical to success at dynamic scene reconstruction, as there exist multiple facets that require precise constraints.

4.3.2 Controllable Gaussian Splatting

Having established the framework for dynamic GS, we can now extend it to accommodate controllable scenarios as shown in Fig. 4.1. This extension is facilitated by its

explicit, Gaussian-based representation. The comprehensive pipeline of our approach comprises four key steps:

1. **Building a Dynamic GS Model:** As previously discussed, this foundational step establishes the groundwork for subsequent extensions.
2. **3D Mask Generation:** This step involves translating two-dimensional mask data into a three-dimensional context, bridging the gap between simple representations and complex spatial models. The 2D mask is either annotated manually quite easily or automatically inferred by existing methods.
3. **Control Signal Extraction:** A pivotal phase where control signals are identified and extracted manually or automatically from explict Gaussian sets, serving as the primary drivers for scene manipulation.
4. **Control Signal Re-Alignment:** The final phase, which entails adjusting and aligning the control signals to ensure their seamless integration and responsiveness within the dynamic model.

In the following sections, we will explore the details of the last three steps, elucidating their roles in enhancing the overall efficacy and controllability of our dynamic GS.

3D Mask Generation

To delineate the controllable set of Gaussians, we introduce a mask vector $m_i \in \mathbb{R}^L$ for each Gaussian, where L denotes the number of attributes to be controlled. The straightforward approach of selecting these in 3D introduces two major challenges: the complexity of manually labeling Gaussian positions for each attribute and the difficulty in achieving an exact fine-grained boundary for the 3D point set, potentially leading to control artifacts.

Addressing these challenges, we propose an effective method to obtain the mask vector m. We start by acquiring K 2D masks for the 2D frames, where K can vary from all frames (for scenarios with available automatic mask generation methods like face recognition [77], as illustrated in Fig. 4.6d) to a single frame (which can be manually labeled, as demonstrated in Figs. 4.6a & 4.6b & 4.6c). After the 2D mask acquisition, we perform a 2D-to-3D mask projection. A practical method involves associating the 3D point with the corresponding 2D pixel, utilizing depth maps and

camera poses as in [32]. However, this method falls short of attaining the fine-grained boundary of the controllable part due to the splatting process 2.2.3.

Therefore, we suggest a learning process to obtain the mask vector for each Gaussian. We allocate a learnable mask tensor $m_i \in \mathbb{R}^L$ to each Gaussian and implement a softmax operation to normalize the sum of the tensor to 1, aiming for a categorical distribution. For rendering the 2D mask M from the 3D m_i, we use the same GS equation, referred to as Eq. 2.2, employing the same point μ, rotation matrix R, and scaling matrix S, except that we set the color for each Gaussian as a constant (1), and take the mask tensor as opacity. This rendered 2D mask is supervised using the ground-truth mask. Importantly, rather than enforcing an exact match between the rendered and ground-truth masks for each control area, we focus on ensuring that the rendered mask has no impact (is black) on other control areas as in Eq. 4.9, significantly reducing artifacts at the boundaries. M_i is the rendering 2D mask for the ith attribute and M_i^{gt} is the corresponding ground truth. Here we want the rendering mask M_i to ideally be black on other controllable areas so as to make no effect on these parts.

$$\mathcal{L}^{\text{mask}} = \sum_{i=1}^{L} \|(1 - M_i) - \sum_{j=1, j\neq i}^{L} M_j^{gt}\| \cdot \sum_{j=1, j\neq i}^{L} M_j^{gt}. \tag{4.9}$$

In this step, we maintain all other learnable weights and tensors, except for the mask tensor, as fixed.

Control Signal Extraction

Our method uniquely eliminates the need for pre-computed control signals, significantly expanding its range of applications. This is accomplished by unsupervised learning of the control signal directly from the Gaussians. The first step involves selecting a set of Gaussians, denoted as G, which represents movement within the control part, as indicated by the previously learned mask m. This set G can be either manually selected in 3D or automatically based on movement trajectories, such as by choosing the set of points $\mathbf{p}$ with the largest movement distance. The size of this Gaussian set G can be as minimal as a single Gaussian.

Utilizing the explicit representation of GS, we calculate the centroid $\mathbf{c}$ of the points in G and trace its movement trajectory. We employ a simple linear model for

trajectory analysis, although more complex models are feasible. Principal Component Analysis (PCA) is applied to determine the primary movement direction, denoted as $\mathbf{d}$. The positions of the Gaussian (means) μ are then projected onto this direction $\mathbf{d}$ at each timestep t, as shown in Eq. 4.10:

$$\mathbf{proj_d}(\mu_t) = \frac{(\mu_t - \mathbf{c}) \cdot \mathbf{d}}{\|\mathbf{d}\|}. \tag{4.10}$$

This projection enables us to define the start and end points, $\mathbf{s}$ and $\mathbf{e}$, along the movement direction. Subsequently, the distances of all points from the start point $\mathbf{s}$ are normalized to a range between 0 and 1, resulting in our control signal σ, as expressed in Eq. 4.11:

$$\sigma(\mu_t) = \frac{\mathbf{proj_d}(\mu_t) - \mathbf{proj_d}(\mathbf{s})}{\mathbf{proj_d}(\mathbf{e}) - \mathbf{proj_d}(\mathbf{s})}. \tag{4.11}$$

This process culminates in the control signal σ, enabling dynamic scene manipulation.

Control Signal Re-Alignment

After obtaining the control signal σ, the next crucial step is its integration into the network to facilitate manipulation using these signals. This is accomplished by developing a unique network N_i^c for each control signal, designed to output the corresponding offset Δ for each Gaussian attribute, as determined in the dynamic modeling stage. Let μ_i, $\mathbf{C}_i$, $\mathbf{R}_i$, and $\mathbf{S}_i$ denote the mean, rotation, and scaling of each Gaussian, respectively. The control network N_i^c modifies these attributes in response to the control signal:

$$N_i^c(\sigma) = (\Delta\mu_i, \Delta\mathbf{C}_i, \Delta\mathbf{R_i}, \Delta\mathbf{S_i}) \tag{4.12}$$

In this phase, the focus is solely on training these control signal networks N_i, while keeping all other learnable parameters Θ fixed.

Upon achieving reliable estimates for the attribute offsets $\Delta\mu_i, \Delta\mathbf{C}_i, \Delta\mathbf{R_i}, \Delta\mathbf{S_i}$ of each Gaussian, we move towards the end-to-end fine-tuning of all learnable parameters. This all-encompassing fine-tuning is crucial for completing our controllable GS model.

The final model (represented by f) is formulated as:

$$f(\Theta; \mu_i + \Delta\mu_i, \mathbf{C}_i + \Delta\mathbf{C}_i, \mathbf{R_i} + \Delta\mathbf{R_i}, \mathbf{S_i} + \Delta\mathbf{S_i}) \tag{4.13}$$

This final step guarantees precise and effective control over dynamic scene renderings.

4.4 Experiments

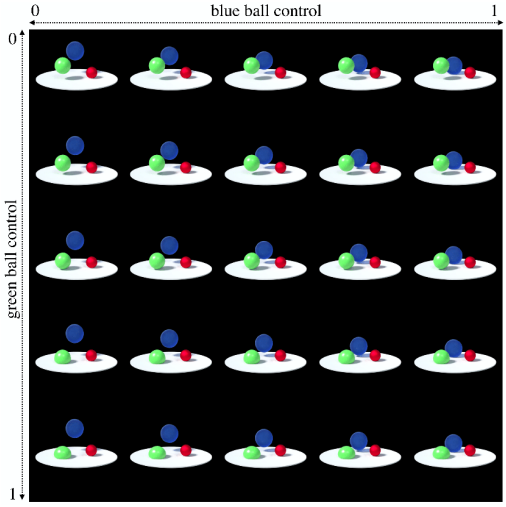


Figure 4.4: Control blue ball and green ball separately.

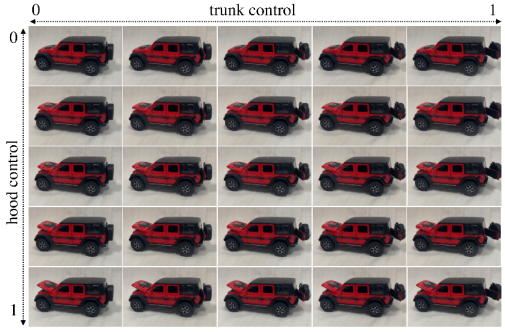
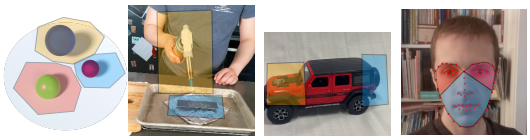


Figure 4.5: Opening the hood and the trunk of the toy car.

In this section, we present the experiments conducted to demonstrate the effectiveness of our method in dynamic and controllable scenarios.



(a) (b) (c) (d)

Figure 4.6: 2D mask labeling. (a)-(c) Manually labeling a single frame. (d) Automatic labeling using facial key-point detection.

Table 4.1: Quantitative results on synthetic dynamic scenes. We color code the results as best (red), second best (orange), and third best (yellow).

Method	PSNR↑	SSIM↑	LPIPS↓ (100x)
NeRF [36]	18.98	0.870	18.25
DirectVoxGo [55]	18.64	0.853	16.88
Plenoxels [12]	20.24	0.868	16.00
T-NeRF [46]	29.50	0.951	7.88
D-NeRF [46]	30.44	0.952	6.63
TiNeuVox-S [11]	30.75	0.955	6.63
TiNeuVox-B [11]	32.67	0.972	4.25
3D GS [21]	23.07	0.928	8.22
Ours	37.90	0.983	1.74
Ours, w/o $\mathcal{L}^{\text{norm}}$	37.41	0.984	1.70
Ours, w/o $\mathcal{L}^{\text{diff}}$	37.68	0.982	1.65
Ours, w/o $\mathcal{L}^{\text{rigid}}$	37.75	0.981	1.71

4.4.1 Datasets

To evaluate our dynamic model, experiments were conducted on two categories of dynamic scenes: synthetic and real. Additionally, the performance of our controllable model was assessed on a synthetic scene, real face scene, real dynamic scene, and a self-captured toy car scene.

Synthetic Scenes. We employed the 360° dynamic synthetic dataset introduced by [46], comprising 8 animated objects with complex geometries and non-Lambertian materials. Each scene in this dataset includes 50 to 200 training images and 20 test images, all at an 800×800 resolution.

Real Scenes. Four topologically diverse scenes from [43] (torchocolate, cut-lemon,

Table 4.2: Quantitative results on real dynamic scenes. We color code each row as best, second best, and third best.

Method	PSNR↑	SSIM↑	LPIPS↓ (100x)
NeRF[36]	22.3	0.807	43.3
NV[31]	26.3	0.910	20.9
NSFF[28]	25.7	0.881	24.8
Nerfies[42]	29.3	0.948	17.6
HyperNeRF[43]	29.8	0.954	17.2
TiNeuVox-S[11]	23.6	0.690	54.5
TiNeuVox-B[11]	28.0	0.752	45.1
3D GS [21]	22.1	0.724	43.8
Ours	29.6	0.950	17.1
Ours, w/o $\mathcal{L}^{\text{norm}}$	29.1	0.905	20.1
Ours, w/o $\mathcal{L}^{\text{diff}}$	29.8	0.912	19.8
Ours, w/o $\mathcal{L}^{\text{rigid}}$	29.4	0.920	21.3

chickchicken, and hand) were used. These scenes were captured using a rig consisting of two Google Pixel 3 phones mounted approximately 16cm apart on a pole.

Real Face Scene. For controllable model testing, we utilized a real face scene from [20] (involving actions like closing/opening the eyes/mouth). This scene was captured with either a Google Pixel 3a or an Apple iPhone 13 Pro. Unlike CoNeRF, which requires pre-defined control signals and masks, our method achieves comparable controllability without such prerequisites.

Toy Car Scene. We also ran experiments on two self-captured Toy Car scenes. The capture process involved manually opening the regions of control (doors, hood, trunk, etc.) and recording one video per transition. Once the transitions were completed, we stitched the individual videos together to create a single cohesive dynamic scene. This scene was captured with an Apple iPhone 13 Pro.

4.4.2 Implementation Details

For the training of the dynamic component, we adopted the differential Gaussian rasterization technique from 3DGS [21], and implemented the additional network components using PyTorch [44]. The Gaussians were initialized either using SfM

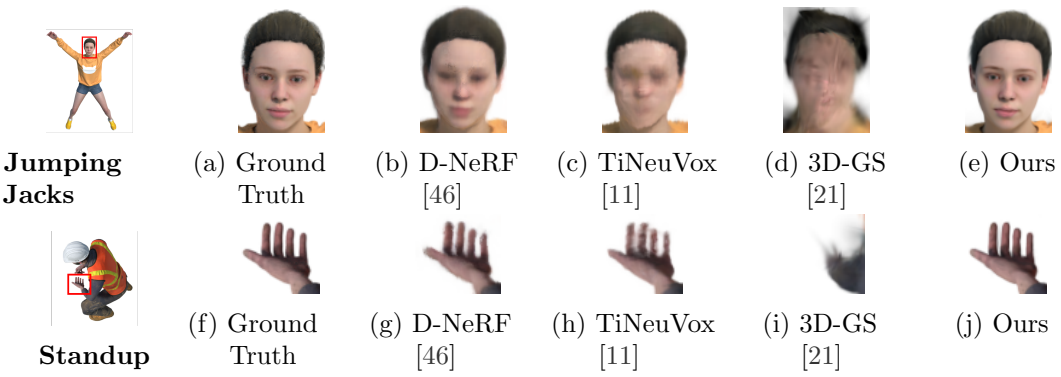


Figure 4.7: Qualitative results on synthetic dynamic scenes. We compare our Dynamic 3D-GS method (Ours) with the ground truth, D-NeRF, TiNeuVox, and the static 3D-GS method.

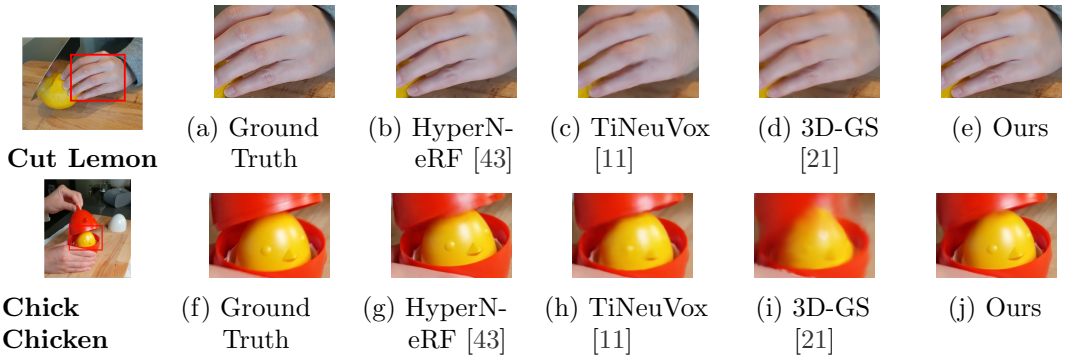


Figure 4.8: Qualitative results on real dynamic scenes. We compare our Dynamic 3D-GS method (Ours) with the ground truth, HyperNeRF, TiNeuVox, and the static 3D-GS method. For Cut Lemon, our method models the knuckles on the hand better than others. As for Chick Chicken, we reconstruct more fine details (red edges).

results or by randomly selecting N points (where $N = 10k$ in our experiments) within the scene box. The initial phase of 3k iterations does not involve learning any deformation field; this phase is akin to the training process of 3D GS, aiding in the convergence of the learning process. Following this, we jointly train the 3D Gaussian attributes and the deformation network for a total of 50k iterations. The learning rate for each Gaussian attribute is kept consistent with that used in 3D GS, as detailed in [21], while the learning rate for the deformation network is set to exponentially decay from $1e-3$ to $1e-6$.

In the controllable part, the learning rate is set to 1 for the initial 1k iterations

during the 3D Mask Generation phase. During the Control Signal Re-Alignment phase, we apply an exponential decay of the learning rate from $1e-2$ to $1e-4$ over 5k iterations, specifically for training the control signal networks. For the final end-to-end finetuning, the learning rate is set to $1e-6$, and the process is run for an additional 5k iterations. Optimization throughout these processes is performed using the Adam optimizer [22] with a β value range of $(0.9, 0.999)$. All experiments were conducted using single 80GB NVIDIA A100 GPUs.

4.4.3 Results

We show the qualitative and quantitative results in this section to demonstrate the effectiveness of our method. We use Peak Signal-to-Noise Ratio (PSNR) [18] in decibels (dB), the Structural Similarity Index (SSIM) [40, 68] and the Learned Perceptual Image Patch Similarity (LPIPS) [78] as evaluation metrics. All detailed results for each scene can be found in the supplementary material.

We first show our dynamic GS modeling part. We compare our method with existing works using dynamic synthetic scenes from [46] on the novel view synthesis task. We report the quantitative results in Table 4.1 and we can see that our method can achieve much better performance than existing methods. The qualitative results are shown in Fig. 4.8 and we can see that our method has better face and hand details. For the real scenes from [43], we run interpolation experiments as in [43] instead of the novel view synthesis task because of the rendering pose problem mentioned in [73]. We show our results in Table 4.2 and Fig. 4.8. We can see that our method can capture better details on complex real dynamic scenes. We also perform ablation experiments on the regularizations we utilize as shown in Table 4.1 & 4.2. To better illustrate the role of the regularizations, we also visualize the trajectories in Fig. 4.3.

For the controllable GS, we use four datasets (bouncingball, torchocolate, face and car) and first fit a dynamic GS on them. Then we obtain the labels as shown in Fig. 4.6. For the eye scene, we manually select the point set as the control points, and for other scenes, we get the point sets automatically from the point movement as mentioned before. We also visualize the control results to demonstrate our method's performance.

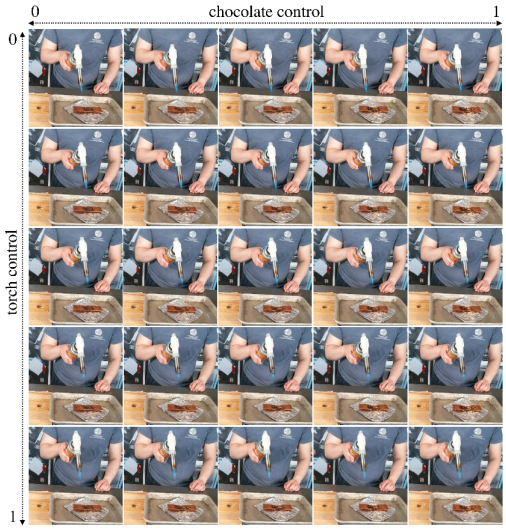
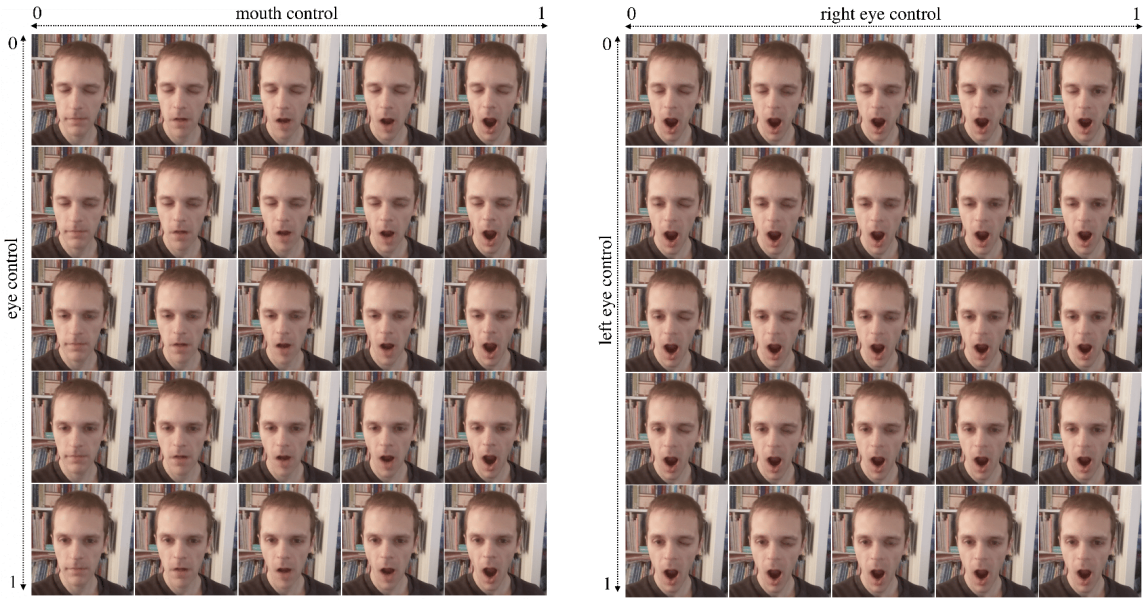


Figure 4.9: Controlling the blowtorch and the melting chocolate separately.

4.5 Conclusion

We presented Controllable Gaussian Splatting named CoGS, a novel method for dynamic scene manipulation. It overcomes the limitations of NeRFs and similar neural methods by using an explicit representation that enables real-time, controllable manipulation of dynamic scenes. Our approach, validated through extensive experiments, shows superior performance in visual fidelity and manipulation capabilities compared to existing techniques. The explicit nature of CoGS not only enhances efficiency in rendering but also simplifies scene element manipulation. It has the potential to democratize 3D deformable content creation using commodity hardware, making it more accessible and feasible for a broader range of users and applications.

Our method is not without limitations. CoGS faces challenges with shiny or intricately lit objects, common in GS pipelines. Dynamic modeling may struggle with non-rigid deformation and large-scale movements in monocular settings. Limitations may also arise from controllable signal extraction and re-alignment, with the current PCA method potentially struggling with highly complex movements. Addressing these limitations will be the focus of future work.



(a) (b)

Figure 4.10: Fine-grained facial control. (a) shows independent control of eyes and mouth. (b) demonstrates independent control of each eye.

Chapter 5

Gaussian Splatting Lucas-Kanade

CoGS significantly advances control over dynamic scenes, yet its performance, along with other explicit methods, often deteriorates in settings with static or near-static camera movement. In these cases, insufficient parallax creates inherent geometric ambiguities. Existing methods struggle to accurately reconstruct and track scene geometry under these conditions. Addressing this critical limitation, this chapter introduces Gaussian Splatting Lucas-Kanade (GS-LK), a hybrid approach integrating Gaussian Splatting with classical optical flow techniques inspired by the Lucas-Kanade algorithm. GS-LK explicitly enforces geometric consistency and robust motion tracking, even under challenging monocular or low-parallax conditions.

5.1 Introduction

The task of reconstructing dynamic 3D scenes from monocular video presents a significant challenge, particularly when constrained by limited camera movement. This limitation results in a lack of parallax and epipolar constraints, which are crucial for accurately estimating scene structures [14, 52]. Implicit neural representation methods like NeRF have shown promise in mitigating this issue by modeling dynamic scenes using an implicit radiance field [8, 28, 36]. Explicit approaches like Gaussian splatting [21], while more efficient and often improve rendering quality, face additional hurdles. Gaussian splatting relies on an explicit representation of the scene using volumetric Gaussian functions. This necessitates a high degree of accuracy in geometric

understanding to ensure proper scene depiction. Consequently, despite advancements in dynamic Gaussian splatting techniques, the requirement for diverse viewing angles to effectively constrain the placement of these Gaussians remains a limiting factor. This inherent limitation restricts their effectiveness for capturing scenes from static viewpoints or dealing with objects exhibiting rapid and complex movements.

Recent Gaussian splatting frameworks for modeling dynamic scenes commonly utilize a canonical Gaussian space as their foundation. This space is typically initialized using Structure from Motion (SfM) [35] point clouds, with which each Gaussian is assigned attributes that describe its position, orientation, and lighting-dependent color. This canonical space acts as a reference point from which deformations are applied to represent the dynamic scene at different time steps. The process of deforming the canonical space to depict the scene at a particular time step is achieved through a warp field, often referred to as a forward warp field due to its function of warping the canonical representation forward to a specific point in time. This warp field is typically modeled using learnable Multi-Layer Perceptrons (MLPs) or offset values, as seen in frameworks like DeformableGS [73] and Dynamic-GS [32]. These approaches have laid the groundwork for dynamic Gaussian scene modeling.

Structural cues such as optical flow and monocular depth have been explored to regulate the temporal transitions of Gaussians. However, imposing structural constraints on a generic forward warp field, as seen in DeformableGS [73], is challenging. Although many studies leverage depth and flow priors, they often fail to regularize motion through the warp field, resulting in sub-optimal trajectories and insufficient motion regularization. Additionally, employing time integration on an auxiliary network to predict the velocity field can introduce computational overhead and lack theoretical constraints. In contrast to this data-driven approach, we demonstrate that it is possible to derive velocity fields directly from a generic warp field by adapting the Lucas-Kanade method with a small motion constraint for Gaussian Splatting. This allows for effective time integration for structural comparisons. By regulating Gaussian motions through analytical velocity fields, we address the limitations of data-driven structural regularizations, enabling continuous learning of the warp field in a time domain without the need for additional learnable parameters.

In real-world captures, our approach outperforms state-of-the-art methods, including other dynamic Gaussian techniques. We demonstrate that our warp field

regularization accommodates deformations in complex scenes with minimal camera movement, achieving results that are competitive with NeRF frameworks and enabling Gaussian splatting to model highly dynamic scenes from static cameras. To summarize, our approach offers two key advantages:

Reduced Bias: Methods relying solely on data-driven learning of warp fields can be biased towards the visible time steps, neglecting the overall movement of Gaussians. Our method, applicable to forward warp field techniques, ensures accurate Gaussian trajectories throughout the sequence.

Improved Tractability: The derived analytical solution for warp field velocities provides greater tractability compared to directly supervising the flow field as the difference in estimated deformations between consecutive frames.

5.2 Related Works

Neural Radiance Fields (NeRFs). NeRFs [36] have demonstrated exceptional capabilities in synthesizing novel views of static scenes. Their approach uses a fully connected deep network to represent a scene's radiance and density. Since its introduction, NeRF has been extended to dynamic scenarios by several works [13, 42, 43, 46, 56, 77]. Some of these methods leverage rigid body motion fields [42, 43] while others incorporate translational deformation fields with temporal positional encoding [46, 56]. Despite the advantages of NeRF and its dynamic extensions, they often impose high computational demands for both training and rendering.

Gaussian Splatting. The computational complexity of Neural Radiance Fields (NeRF) has driven the development of alternative 3D scene representation methods. 3D Gaussian Splatting (3DGS) [21] has emerged as a promising solution, leveraging 3D Gaussians to model scenes and offering significant advantages over NeRF in terms of rendering speed and training efficiency.

While initially focused on static scenes, recent works [70, 73] have extended 3DGS to the dynamic domain. These approaches introduce a forward warp field that maps canonical Gaussians to their corresponding spacetime locations, enabling the representation of dynamic scene content. DeformableGS [73] employs an MLP to learn positional, rotational, and scaling offsets for each Gaussian, creating an over-parameterized warp field that captures complex spacetime relationships. 4DGaussians

[70] further refine this approach by leveraging hexplane encoding [5, 11] to connect adjacent Gaussians. DynamicGS [32] incrementally deforms Gaussians along the tracked time frame. Despite these advancements, existing dynamic 3DGS methods struggle with highly dynamic scenes and near-static camera viewpoints. To overcome these limitations, we propose a novel approach that grounds the warp field directly on approximated scene flow.

Flow supervision. Optical flow supervision has been widely adopted for novel view synthesis and 3D reconstruction. Dynamic NeRFs [8, 25, 27, 28, 56, 60] have explored implicit scene flow representation with an Invertible Neural Network, semi-explicit representation from time integrating a learned velocity field, and explicit analytical derivation from the deformable warp field. Flow-sup NeRF [60] pioneered the use of analytical flow supervision in NeRF, laying the foundation for more precise and physically grounded modeling of dynamic scenes.

Several Gaussian Splatting works have begun exploring flow supervision. Motion-aware GS [17] applies flow supervision on the cross-dimensionally matched Gaussians from adjacent frames, without explicitly accounting for the flow contributions of Gaussians to each queried pixel. Gao et al. [15] on avatar rendering apply flow supervision for adjacent frames after rendering the optical flow map with α-blending [59]. However, while these approaches introduce data-driven flow supervision, they do not effectively regularize the motion through the warp field, leading to suboptimal trajectories and insufficient motion regularization. This limitation makes them suitable only for enforcing short-term geometric consistency, while their performance deteriorates for out-of-distribution time steps [1, 26]. In the context of dynamic Gaussian Splatting, we propose to regularize the deformable warp field through analytical time integration to ensure consistent geometric relationships across time steps.

5.3 Preliminary

Dynamic Gaussian Splatting. Our method builds upon the Dynamic 3D Gaussian Splatting framework, which extends the static optimization pipeline detailed in Chapter 2 to model time-varying scenes. The central paradigm is the use of a canonical space, a static, time-independent 3D template of the scene.

To model motion over time, we introduce a warp field $\mathcal{F}$, which transforms canonical Gaussians $\mathcal{G}$ into their deformed states at time t. The offset in Gaussian parameters due to this transformation is expressed as:

$$\delta\mathcal{G} = \mathcal{F}_\theta(\mathcal{G}, t). \tag{5.1}$$

Typically, these deformed geometries are mapped through a neural network [32, 70, 73], with optional color mapping applied as part of the transformation process. Following deformation, a novel-view image $\hat{I}$ is rendered using differential rasterization with a view matrix V and target time t. While MLP-based deformation mapping often suffers from overfitting to training views, leading to degraded novel-view reconstruction, we address this by enforcing analytical constraints on the warp field. This ensures Gaussian motions adhere to the expected scene flow field, mitigating the issues of overfitting and SfM-initialized point cloud limitations.

Warp field expression with twist increments. Warp fields, previously defined for point cloud registration tasks [1], can be used to model rigid transformations between sets of Gaussian means. Let $\boldsymbol{\mu}_\tau \in \mathbb{R}^{N_1\times 3}$ and $\boldsymbol{\mu}_s \in \mathbb{R}^{N_2\times 3}$ represent the target and source Gaussian means, respectively, where N_1 and N_2 denote the number of Gaussians in each set. The warp field, which defines the rigid transformation aligning the source Gaussian set to the target set in the special Euclidean group $SE(3)$, can be expressed as $\mathcal{W}(\boldsymbol{\xi}) = \exp\left(\sum_{\mu=1}^{6} \boldsymbol{\xi}_\mu P_\mu\right)$.

Here, $\boldsymbol{\xi} \in \mathbb{R}^6$ are the twist parameters, and P are the generator matrices of the group. The warp field is optimized by minimizing the difference between the transformed source Gaussian means and the target means:

$$\underset{\boldsymbol{\xi}}{\operatorname{argmin}} \left\|\phi(\mathcal{W}(\boldsymbol{\xi}) \cdot \boldsymbol{\mu}_s) - \phi(\boldsymbol{\mu}_\tau)\right\|_2^2, \tag{5.2}$$

where $\phi : \mathbb{R}^{N\times 3} \to \mathbb{R}^K$ is an encoding function that either explicitly extracts geometric features such as edges and normals or implicitly encodes the Gaussian means into feature vectors. The notation $(\cdot)$ represents the application of the warp field transformation.

This optimization can be solved iteratively given the twist parameter Jacobian matrix J, denoting the changes in the warp field concerning twist parameters, that

can be further decomposed into a product of warp field gradient and the encoding functional gradient as:

$$J = \frac{(\mathcal{W}^{-1}(\boldsymbol{\xi}) \cdot \boldsymbol{\mu}_\tau)}{\partial \boldsymbol{\xi}^T} \cdot \frac{\partial \phi(\mathcal{W}^{-1}(\boldsymbol{\xi}) \cdot \boldsymbol{\mu}_\tau)}{\partial (\mathcal{W}^{-1}(\boldsymbol{\xi}) \cdot \boldsymbol{\mu}_\tau)^T}. \tag{5.3}$$

The optimization process iteratively learns the twist increment $\Delta\boldsymbol{\xi}$ that best aligns the encoded source Gaussian means with the target Gaussian means.

5.4 Gaussian Splatting warp field regularization by scene flow

We investigate the problem of fitting deformable Gaussian Splatting from a monocular video sequence. To initialize the process, we employ structure-from-motion methods to set the canonical Gaussians and camera parameters. For a given set of 3D Gaussians $\mathcal{G}$ at time t, the warp field deforms these Gaussians from their canonical positions to their dynamic locations at time t. Our goal is to find the optimal network parameters θ for the forward warp field $\mathcal{W}_{c\rightarrow}(\mathcal{G}, t)$. This optimization aims to ensure that both the derived scene flow $\hat{\mathcal{T}}_{t\rightarrow t+1}$ and the rasterized image $\hat{\mathcal{I}}_t$ closely match the expected flow field $\mathcal{T}_{t\rightarrow t+1}$ and the reference image $\mathcal{I}_t$:

$$\mathcal{L} = \mathcal{L}_{color} + \mathcal{L}_{motion} = |\hat{\mathcal{I}}_t - \mathcal{I}_t| + \left\| \hat{\mathcal{T}}_{t\rightarrow t+1} - \mathcal{T}_{t\rightarrow t+1} \right\|. \tag{5.4}$$

The scene flow component can be further divided into optical flow and depth estimations, corresponding to in-plane and out-of-plane motions. Consider a Gaussian mean $\boldsymbol{\mu} = \boldsymbol{\mu}(t)$ moving in the scene. Its instantaneous scene flow is $\frac{d\boldsymbol{\mu}}{dt}$. If the corresponding pixel location on the image plane is $\boldsymbol{u}_i = \boldsymbol{u}_i(t)$, the optical flow can be expressed as the projection of the scene flow to the image plane, with $\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{u}_i}$ as the instantaneous camera projective relationship:

$$\frac{d\boldsymbol{u}_i}{dt} = \frac{d\boldsymbol{u}_i}{d\boldsymbol{\mu}} \frac{d\boldsymbol{\mu}}{dt}. \tag{5.5}$$

Deriving an approximation to the scene flow needs the expression reversed. We can represent the relationship between $\boldsymbol{\mu}$ and $\boldsymbol{u}_i$ with dependencies on time as

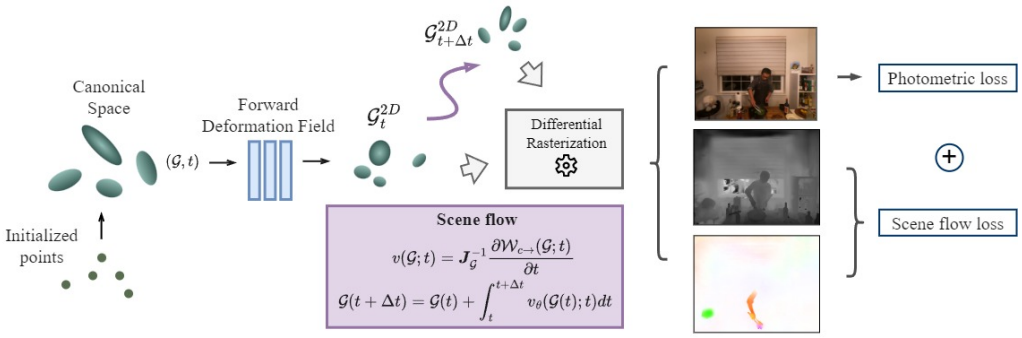


Figure 5.1: **Analytical scene flow from warp field.** With canonical Gaussians $\mathcal{G}_c$, we transform them forward in time to $\mathcal{G}_t$, then perform time integration from warp field velocities $v(\mathcal{G};t)$ to derive $\mathcal{G}_{t+\Delta t}$. The Gaussian offsets $\mathcal{G}_{t+\Delta t} - \mathcal{G}_t$ are compared to reference scene flow.

$\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{u}_i(t);t)$ [58]. By differentiating this mapping with respect to time, we obtain:

$$\frac{d\boldsymbol{\mu}}{dt} = \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{u}_i}\frac{d\boldsymbol{u}_i}{dt} + \frac{d\boldsymbol{\mu}}{dt}\Big|_{\boldsymbol{u}_i}. \tag{5.6}$$

This equation describes the motion of a point in 3D space while decomposing the motion of a Gaussian in the world into two components. The in-plane component $\frac{d\boldsymbol{u}_i}{dt}$ represents the instantaneous optical flow, while the out-of-plane term reflects the motion of $\boldsymbol{\mu}$ along the ray corresponding to $\boldsymbol{u}_i$. This decoupling allows us to regularize the optical flow field and the depth field separately from the estimated scene flow.

The scene flow regularization can thus be re-expressed as a combination of optical flow loss applied over the interval of $(t \to \Delta t)$ and depth loss of Gaussians at $(t + \Delta t)$, where the next step Gaussians are analytically derived:

$$\mathcal{L}_{motion}(t) = \left\| \hat{\mathcal{T}}_{t\to t+1} - \mathcal{T}_{t\to t+1} \right\| = \alpha \mathcal{L}_{flow} + \beta \mathcal{L}_{depth}. \tag{5.7}$$

Decoupling the motions alleviates the constraints on the supervision dataset and adds flexibility to the optimization process. A visualization of the regularization pipeline is provided in Fig. 5.1.

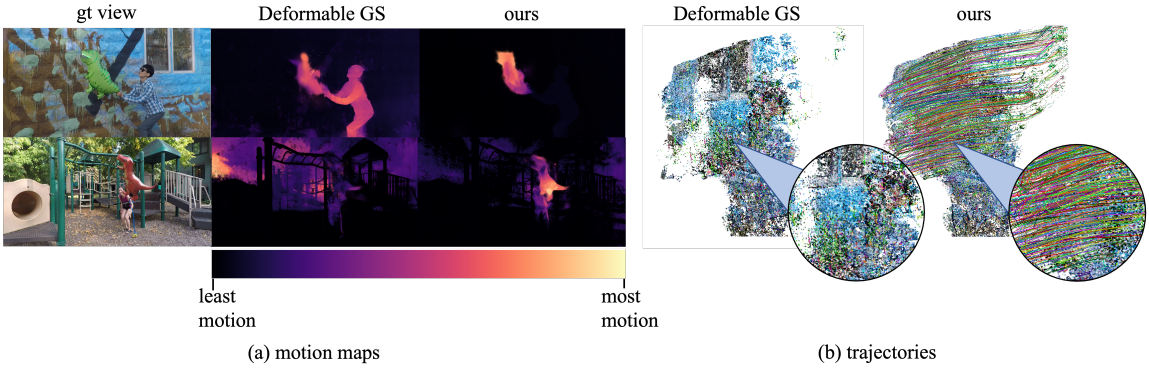


Figure 5.2: (a) Visualization of Gaussians' travel distance $||\boldsymbol{\mu} - \boldsymbol{\mu}_o||_2$. In both of the scenes, the humans are stationary with dinosaur balloons moving around. Our result correctly identifies the dynamic regions, whereas the baseline model forms motions in the background and on supposedly stationary humans to compensate for photometric correctness. (b) 3D visualization of the motion trajectories. Our result shows clean trajectories from the waving balloon.

5.4.1 Forward warp field velocity

To address the challenge of transforming canonical Gaussians $\mathcal{G}$ over time, our goal is to derive the velocity field from the forward warp field. Specifically, we seek to identify a warp field $\mathcal{W}$ that advances the canonical Gaussians in time, with an underlying velocity field $v(\mathcal{G}; t)$ that quantifies the changes in Gaussian parameters within the world coordinate system at a specific time t. This derivation is conceptually similar to that of twist Jacobians [1, 26] and the analytical warp regularization in NeRF [60], effectively capturing continuous deformation over time.

In the context of Gaussian Splatting, the motion of Gaussians between two adjacent frames is bijective because all Gaussians are transformed from the same canonical space, creating a one-to-one correspondence among the Gaussians. Consequently, we can compute the velocity field directly from the forward warp field $\mathcal{W}_{c\rightarrow}(\mathcal{G}, t)$. To analytically derive the velocity field $v(\mathcal{G}; t)$, we must represent the gradient of the warp field in relation to time and warp parameters. Assuming that the Gaussian motions are small, we can utilize a modified Lucas-Kanade expression. By applying a Taylor series expansion to the warp field, we decompose the velocity into two essential components: the feature gradient of the warp field with respect to the corresponding Gaussian parameter and the analytical warp Jacobian [1, 60]. This relationship is

expressed as:

$$v(\mathcal{G};t) = \frac{d\mathcal{W}_{c\to}(\mathcal{G},t)}{dt}\Big|_{(\mathcal{G},t)} = J_{\mathcal{G}}^{-1}\frac{\partial \mathcal{W}_{c\to}(\mathcal{G};t)}{\partial t}. \tag{5.8}$$

In this equation, the term $\frac{\partial \mathcal{W}_{c\to}(\mathcal{G};t)}{\partial t}$ indicates how each output Gaussian parameter varies with respect to the input canonical Gaussian parameters. Meanwhile, $J_{\mathcal{G}}$ describes how modifications in the warp field output influence the transformation of the Gaussians. In practical scenarios, the warp Jacobian can face numerical instabilities when $det(J_{\mathcal{G}}) < \epsilon$, especially if the canonical Gaussian scene has not yet stabilized. Discarding these unstable motions can negatively impact subsequent densification and pruning processes. To mitigate these instabilities, we substitute the inverse Jacobian $J_{\mathcal{G}}^{-1}$ with the Moore–Penrose pseudoinverse $J_{\mathcal{G}}^{+}$.

The derived expression is valid for any Gaussian parameter input into the deformation network, provided the input and output dimensions are consistent. However, given that the scaling terms and spherical harmonics of the Gaussians show minimal changes between adjacent frames, we choose to omit these from the calculations for improved efficiency. Summarizing, we compute the velocity fields corresponding to the displacement and rotation of Gaussians as follows:

$$v(\boldsymbol{\mu};t) = J_{\boldsymbol{\mu}}^{+}\frac{\partial \mathcal{W}_{c\to}(\boldsymbol{\mu};t)}{\partial t}; \quad v(\boldsymbol{q};t) = J_{\boldsymbol{q}}^{+}\frac{\partial \mathcal{W}_{c\to}(\boldsymbol{q};t)}{\partial t}, \tag{5.9}$$

where $v(\boldsymbol{\mu};t) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ and $v(\boldsymbol{q};t) : \mathbb{R}^4 \times \mathbb{R} \to \mathbb{R}^4$ describe the positional and rotational velocity of a Gaussian in the world coordinate, respectively.

5.4.2 Scene field from time integration

Time integration on the velocity field. Given the velocity field from equation 5.9, and 3D Gaussians $\mathcal{G}(t)$ observed at time step t, we can apply time integration using a Runge-Kutta [3] numerical solver on the velocity field to obtain the offset parameters for Gaussians at time $(t + \Delta t)$ Specifically, we employ Monte Carlo integration with a sampling size of 10 points over each domain, which provides an efficient and well-balanced alternative to deterministic approaches. The integration is formulated as:

$$\mathcal{G}(t+\Delta t) = \mathcal{G}(t) + \int_{t}^{t+\Delta t} v(\mathcal{G}(t);t)dt. \tag{5.10}$$

We then train the warp field network, identical to the one outlined in [73], to minimize the difference between the analytically derived scene flow, and the ground truth scene flow decomposed into optical flow and depth describing the in-plane and out-of-plane Gaussian motions.

Optical flow and depth rendering from Gaussians. For each Gaussian at time t, we compute each of the corresponding positions and rotations at $(t + \Delta t)$. For rendering the optical flow and depth map, we consider the full freedom of each Gaussian motion.

For rendering the optical flow map $\hat{\mathcal{O}}_{t\to t+1}$, we want to calculate the motion's influence on each pixel $\boldsymbol{u_i}$ from all Gaussians in the world coordinate. In the original 3D Gaussian Splatting, a pixel's color is the weighted sum of the projected 2D Gaussian's radiance contribution. Analogous to this formulation, the optical flow can be rendered by taking the weighted sum of the 2D Gaussians' contribution to pixel shifts, as derived in previous literature on flow regularization [15, 17]:

$$\hat{\mathcal{O}}_{t\to t+1} = \Sigma_{i\in N} o_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) = \sum_{i=1} w_i [\Sigma_{i,t_2}\Sigma_{i,t_1}^{-1}(\boldsymbol{u}_{t_1} - \boldsymbol{\mu}_{i,t_1}) + \boldsymbol{\mu}_{i,t_2} - \boldsymbol{u}_{t_1}], \quad (5.11)$$

where $\mathcal{O}_i$ denotes the optical flow of Gaussian $\mathcal{G}_i$ over the time interval, and $w_i = \frac{T_i\alpha_i}{\Sigma_i T_i \alpha_i}$ denotes the weighing factor of each Gaussian from α-blending.

Similarly, the frame's z-depth estimates $\hat{D}_{t+\Delta t}$ can be rendered from the discrete volume rendering approximation accounting for the per-Gaussian contributions:

$$\hat{D}_{t+\Delta t} = \Sigma_{i\in N} d_{i,t_2} \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \quad (5.12)$$

where d_{i,t_2} denotes the z-depth value of analytically derived Gaussian $\mathcal{G}_i$ from the viewing space at $(t + \Delta t)$. We normalize the optical flow and depth estimates for numerical stability. The optical flow and depth maps are rasterized simultaneously in the same forward pass with color.

5.5 Experiments

In this section, we first provide the implementation details of the proposed warp field regulation and then validate our proposed method on five dynamic scene datasets captured with different levels of camera movements. Our method outperforms baseline approaches in both static and dynamic camera settings, achieving state-of-the-art results in quantitative and qualitative evaluations.

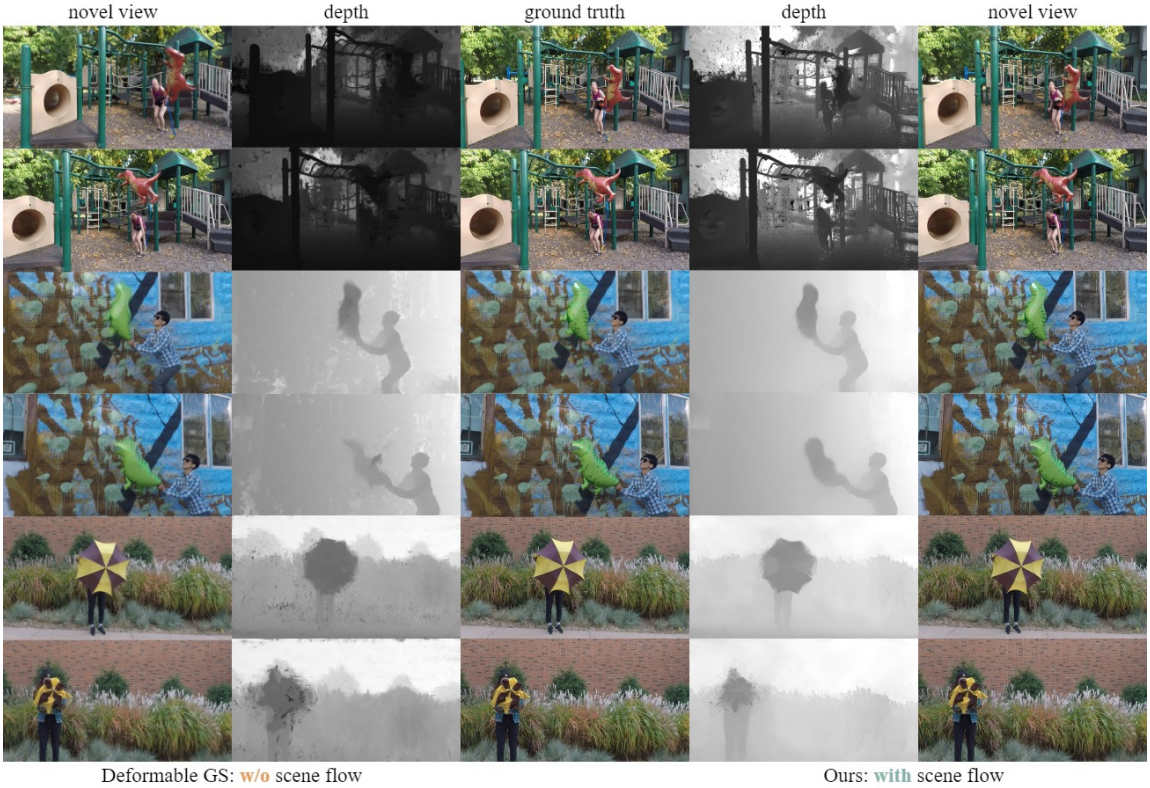


Figure 5.3: **Qualitative comparisons on the Dynamic Scenes dataset.** Compared to the baseline method, our approach can achieve superior rendering quality on real datasets with lower EMFs.

5.5.1 Implementation details

We implement the warp field using PyTorch [44], leveraging its `Autodiff` library for gradient and Jacobian computations. The framework is optimized with Adam [22]

as with 3DGS [21] on a NVIDIA A100 Tensor Core [39]. We use the `Torchdiffeq` [7] library for numerical integration. Modified from Deformable GS, the inputs to the warp field include 3D Gaussian positions ($\boldsymbol{\mu}$), time (t), and 3D Gaussian rotations ($\boldsymbol{q}$) in quaternions. Velocity derivations are enabled by removing the stop gradient operation from the network inputs. Positional encoding [36] is applied to extend inputs' frequency band [79].

5.5.2 Evaluation

Here we analyze the performance of our method quantitatively and qualitatively. We aim to study if the analytical scene flow regularization on the warp field helps disambiguate the dynamic scene geometry and promote the reconstruction of low EMF scenes.

Benchmarked datasets. We evaluate our method using real-world monocular datasets with varying camera motion. These include DyCheck dataset [14], captured with handheld monocular cameras; Dynamic Scene [74], captured by a stationary multi-view 8 camera rig with significant scene motion; Plenoptic Video [26], captured using a static rig with 21 GoPro cameras [16]; Hypernerf [43], which captures objects with moving topologies from a moving camera, suitable for quasi-static reconstruction; and sequences from DAVIS 2017 [45] dataset containing near-static monocular videos.

Quantitative baseline comparisons. We evaluate our method's novel-view synthesis capabilities on Plenoptic Videos, Hypernerf, Dynamic Scenes, and DyCheck, ordered by decreasing Effective Multi-view Factor (EMF) [14]. These datasets present increasing challenges, with DyCheck posing the greatest difficulty due to its low EMF and unreliable object motion. We first assess our method on Hypernerf and Plenoptics with higher EMFs against recent methods 3DGS [21], Deformable GS [73], 4DGaussians [70], and MA-GS [17]. We report the standard metrics LPIPS [78], SSIM [68], and PSNR [69] in Table 5.5 and 5.4. We see that our method consistently outperforms these Gaussian baselines on scenes with higher EMFs.

To further assess our method's performance on monocular sequences with lower EMFs, we evaluate its performance against recent methods Marbles [54], NSFF [28], NR-NeRF [56], Nerfies [42], and Flow-supervised NeRF [60], see Table 5.1 and 5.2. These comparisons highlight the effectiveness of our analytical scene flow

Table 5.1: **Quantitative evaluation of novel view synthesis on the Dynamic Scenes dataset.** See Sec. 5.5.2 for descriptions of the baselines.

method	Playground			Balloon1			Balloon2			Umbrella		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF	24.69	0.889	0.065	24.36	0.891	0.061	30.59	0.953	0.030	24.40	0.847	0.088
NR-NeRF	14.16	0.337	0.363	15.98	0.444	0.277	20.49	0.731	0.348	20.20	0.526	0.315
Nerfies	22.18	0.802	0.133	23.36	0.852	0.102	24.91	0.864	0.089	24.29	0.803	0.169
(w flow)	22.39	0.812	0.109	24.36	0.865	0.107	25.82	0.899	0.081	24.25	0.813	0.123
Flow-sup. NeRF	16.70	0.597	0.168	19.53	0.654	0.175	20.13	0.719	0.113	18.00	0.597	0.148
Deformable GS	24.82	0.646	0.343	22.40	0.833	0.137	24.19	0.818	0.153	22.35	0.711	0.186
4DGaussians	21.39	**0.776**	0.204	24.48	**0.849**	0.144	24.72	0.801	0.219	21.29	0.560	0.332
Ours	**26.34**	0.756	**0.184**	**26.35**	0.848	**0.133**	**25.89**	**0.911**	**0.151**	**23.02**	**0.746**	**0.176**

Table 5.2: **Quantitative results on the DyCheck dataset.**

Method Windmill	Apple	Spin	Block	Teddy	Paper
NSFF	17.54	18.38	16.61	13.65	17.34
Marbles	17.57	15.49	16.88	13.57	18.67
4D Gaussians	15.41	14.41	11.28	12.36	15.60
Ours	16.03	16.71	15.46	13.60	17.41
Ours - pose refined	**22.61**	**24.09**	**23.27**	**17.78**	**19.91**

Table 5.3: **Quantitative results for the Deformable GS method.**

Method	PSNR ↑	SSIM ↑	LPIPS ↓
DefGS w/ flow	20.97	0.690	0.31
DefGS w/ depth	21.17	0.673	0.29
DefGS w/ both	23.94	0.702	0.28
DefGS	24.82	0.646	0.34
Ours	**26.34**	**0.756**	**0.18**

approach compared to other flow-guided rendering frameworks. Notably, our method outperforms NSFF, which uses a similar supervision style but a different scene flow derivation.

Table 5.4: **Quantitative evaluation of novel view synthesis on the Plenoptic Videos dataset.** See Sec 5.5.2 for an analysis of the performance.

method	cook spinach			cut roasted beef			sear steak			Mean		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Deformable GS	32.97	0.947	0.087	30.72	0.941	0.090	33.68	0.955	0.079	32.46	0.948	0.085
4DGaussians	31.98	0.938	**0.056**	31.56	0.939	**0.062**	31.20	0.949	**0.045**	31.58	0.942	**0.055**
MA-GS	32.10	0.937	0.060	32.56	0.941	0.059	31.60	0.951	0.045	32.09	0.943	0.053
Ours	**33.91**	**0.951**	0.064	**32.40**	**0.957**	0.084	**34.02**	**0.963**	0.057	**33.44**	**0.954**	0.068

Comparisons with direct depth and flow Supervision. We note that it is essential to understand the performance improvements over regularizing the deformable network directly with depth and flow priors. We include ablation experiments with Deformable GS supervised with depth and flow losses, see Table 5.3. The results suggest that directly applying depth and flow regularizations encourages the scene to overfit to training viewpoints, leading to degraded performance. With depth regularizations, the scene geometries become more prominent, with a sacrifice in motion smoothness. Regularizing offsets between two frames can enforce accurate piecewise

Table 5.5: **Quantitative evaluation on the HyperNeRF dataset.** See Sec. 5.5.2 for detailed explanations.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS	20.84	0.70	0.45
Deformable GS	26.47	<u>0.79</u>	<u>0.29</u>
4DGaussians	<u>26.98</u>	0.78	0.31
Ours	**27.38**	**0.81**	**0.26**

Table 5.6: **Ablation study on scene flow decomposition in "Playground" from Dynamic Scenes.** See Sec. 5.5.2 for design details.

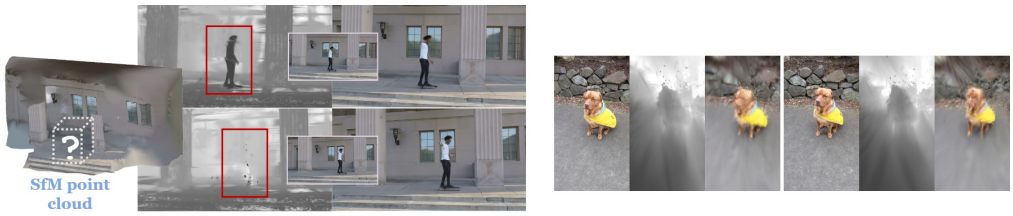
Method	PSNR ↑	SSIM ↑	LPIPS ↓
w/ Depth Sup.	24.76	0.711	0.274
w/ Flow Sup.	24.37	0.695	0.229
Ours	**26.34**	**0.756**	**0.184**

motion but still cannot generalize well to the discontinued unseen viewpoints. In Table 5.6, we ablate the analytical regularization and report the outcomes of the scene. As shown, each of the scene flow components helps in achieving high-quality synthesis. These ablation experiments reaffirm the validity of the proposed warp field regularization.

Trajectories from scene flow fields. We visualize the scene flow trajectories from Dynamic Scenes Dataset sequences to assess the quality and smoothness of the derived flow field. The Gaussians are subsampled from the dynamic regions of the canonical Gaussian space, and then time integrated to produce their respective displacements across time, visualized as colored trajectories in Fig. 5.2. We note that the trajectories are smooth and follow the expected motions of the dynamic objects. The visualized trajectories of the sequences demonstrate the framework's potential to be extended for tracking in dynamic 3D Gaussian scenes.

Geometric consistency. Fig. 5.3 compares our method with Deformable GS, which learns Gaussians' time dependency without scene flow regularization. While Deformable GS renders visually accurate novel views, the underlying scene structures are inaccurate, leading to blurry dynamic objects and artifacts in depth maps. Our method, leveraging analytical scene flow regularization, achieves more accurate geometries, reflected in both visual and quantitative results (Table 5.1). This also enables accurate Gaussian motion tracking (Fig. 5.2).

Static and dynamic motion separation. Our approach formulates deformations within a canonical Gaussian space. To assess the fidelity of our method in separating static and dynamic regions, we visualize the travel distance of each Gaussian from its canonical projection to a queried viewing camera in Fig. 5.2. The plots are color-coded by the absolute travel distance, with yellow indicating larger distances and purple indicating smaller distances.



(a) Unstable reconstruction when SfM fails to reconstruct dynamic region.

(b) Inaccurate camera poses lead to degraded rendering.

Figure 5.4: Our method struggles with insufficient point cloud initialization due to reliance on non-rigid warping of scene geometry. In the "skating" scene, this results in unstable geometry at certain angles. Inaccurate camera calibrations also degrade our method, as shown in the "toby-sit" sequence, where miscalibration distorts the scene geometry.

In the Playground scene featuring human-object interactions, the visualized travel distance is greatest at the far-side boundaries of the moving object as it recedes from the human. The scene's background is correctly rendered as static with minimal motion. This visualization demonstrates the effectiveness of our method in accurately identifying dynamic regions. Moreover, these results suggest promising future directions for optimizing dynamic scene rendering by filtering out static regions from the optimization process, potentially reducing computational costs.

5.6 Limitations

Sensitivity to SfM Initialization: Our approach is sensitive to the quality of the Structure-from-Motion (SfM) Gaussian space initialization. Accurate initialization is crucial for 3D Gaussian Splatting, particularly without other scene priors. As shown in Fig. 5.4.a, reconstruction quality deteriorates when dynamic regions have insufficient initialized points.

Camera Parameter Sensitivity: Our method is sensitive to inaccuracies in bundle-adjusted camera intrinsic and extrinsic parameters. Errors in these parameters lead to inaccurate rasterization projections, causing noisy geometry or failed reconstruction, as shown in Fig. 5.4.b.

Computational Complexity: The pseudo-inverse operation for velocity field computation requires $(N \times d)$ square matrix decompositions, where N is the number

of Gaussians, and d is the number of Gaussian parameters. Potential mitigations include optimized implementations or pre-filtering to remove static Gaussians.

5.7 Conclusion

This chapter presents a method for incorporating scene flow regularization into deformable Gaussian Splatting. We derive an analytical scene flow representation, drawing upon the theoretical foundations of rigid transformation warp fields in point cloud registration. Our approach significantly enhances the structural fidelity of the underlying dynamic Gaussian geometry, enabling reconstructions of scenes with rapid motions.

Comparison with other Deformable Gaussian Splatting variants demonstrates that regularizing on the warp-field derived scene flow produces more accurate dynamic object geometries and improved motion separation due to its continuous-time regularization. Additionally, comparison with flow-supervised NeRFs reveals the advantages of using Gaussians' explicit scene representations.

While our method exhibits limitations as discussed in the previous section, the accurate geometries learned from dynamic scenes open up promising possibilities for future applications, including 3D tracking from casual hand-held device captures and in-the-wild dynamic scene rendering.

Chapter 6

Distractor-Free 4D Gaussian Splatting

GS-LK reduces geometric ambiguities caused by limited camera motion and monocular inputs, improving both reconstruction and tracking. However, this only addresses part of the problem. Many scenes contain ambiguity not in how they are observed, but in what they contain. Transient objects, reflections, and occlusions are common in real-world video and often interfere with reconstruction. While prior methods attempt to handle such distractors, they assume the underlying scene is static and aim to remove all dynamic content. This strategy fails when dynamic elements are actually the subject of interest. In these cases, the challenge is not just to separate static from dynamic, but to identify which motion is consistent and reconstructable and which is transient and ambiguous. This chapter introduces Focus4DGS, an uncertainty-aware extension of 4D Gaussian Splatting that addresses this problem directly. By analyzing the stability of the learned motion field, Focus4DGS reconstructs dynamic subjects while ignoring unreconstructable transients, enabling high-quality results in unconstrained dynamic environments.

6.1 Introduction

Reconstructing 3D scenes from real-world video is a longstanding challenge in computer vision. Despite substantial progress, one core difficulty remains. Real-world video

is inherently unconstrained. Scenes often include transient objects, reflections, and other forms of visual ambiguity that degrade reconstruction quality. Recent methods aim to address this issue by disentangling the true scene from these distractors.

These approaches, whether based on implicit or explicit representations, typically assume that the scene of interest is static. Their goal is to recover a clean background by removing all dynamic elements, which are treated as noise. This assumption fails in scenarios where dynamic entities are the actual subjects of interest. In such settings, removing all dynamic content results in the unintended loss of key information and prevents meaningful reconstruction.

This limitation highlights a gap in current methods. Existing techniques are not equipped to reconstruct scenes that are fully dynamic while also containing distracting motion. The core challenge is not just to separate static from dynamic elements. Instead, the real need is to distinguish between motion that is meaningful and consistent, and motion that is ambiguous and unstructured. Addressing this challenge requires a change in perspective. Rather than removing motion entirely, we must identify which parts contain recoverable structure and which do not.

We present Focus4DGS, a distractor removal framework designed to operate in fully dynamic environments. It models uncertainty explicitly in order to separate consistent motion that can be reconstructed from transient effects that cannot. The method decomposes each scene into two complementary components. Core Gaussians model the primary dynamic subject with high fidelity. Per-view Distractor Gaussians account for transient objects, occlusions, and regions with high uncertainty. An uncertainty-aware optimization process guides this decomposition to ensure that the main reconstruction remains stable, detailed, and unaffected by distractors.

Our main contributions are

- A new reconstruction strategy for unconstrained dynamic scenes that separates reconstructable motion from unreconstructable motion
- A framework for uncertainty-aware 4D Gaussian Splatting based on a Core-and-Distractor decomposition of the scene
- Empirical validation showing that our uncertainty-guided decomposition achieves performance comparable to specialized baselines, highlighting its robustness and generalizability.

6.2 Related Works

6.2.1 Scene Representations

Consistent with prior chapters, we adopt 3D Gaussian Splatting (3DGS) [21] as the foundational framework for scene representation. A detailed overview of this method is provided in Chapter 2.

To handle motion in dynamic environments, we extend 3DGS following the strategies outlined in Chapters 4 and 5. Existing dynamic extensions, such as 4D Gaussian Splatting [70] and Deformable 3D Gaussians [73], typically address motion by learning deformation fields that map Gaussians from a static canonical space to time-varying positions. While effective, this canonical-to-deformed formulation introduces new challenges. Highly dynamic or transient regions often interfere with the reconstruction of nearby regions that exhibit more stable and coherent motion. Addressing this interference is a central focus of the present chapter.

6.2.2 Handling Distractors

Recent research addresses the challenge of transient distractors within predominantly static scene assumptions. Approaches generally fall into three categories based on their distractor-handling strategies:

Photometric and Uncertainty-Based Methods identify distractors using reconstruction error or predictive uncertainty. NeRF-W [33] separates the scene into static and transient components and applies uncertainty-aware rendering to suppress distractor influence. RobustNeRF [50] reframes the problem using robust statistics, iteratively masking high-error regions to stabilize training.

Semantic Feature-Based Methods replace photometric cues with semantic information from pretrained vision models. NeRF On-the-go [49] uses DINOv2-derived features to isolate distractors in novel-view synthesis. Within the 3DGS framework, SpotLessSplats [51] applies unsupervised clustering on text-to-image embeddings to group transient elements. Robust 3D Gaussian Splatting [57] further enhances masking by incorporating SAM-derived segmentations.

Decompositional Methods explicitly separate static and transient content

into distinct representational layers. DeSplat [66] models static structure with global Gaussians and transient phenomena with per-view splats. HybridGS [19] adopts a similar strategy, combining persistent 3D Gaussians with view-specific 2D Gaussians to handle distractors.

While these methods are effective in their respective contexts, they share a common limitation. All assume that the scene of interest is static and aim to recover a clean, singular reconstruction. Our work moves beyond this assumption. We address the more complex problem of reconstructing dynamic scenes that also contain unreconstructable dynamic content, a challenge that remains open in prior literature.

6.3 Method

To robustly reconstruct a dynamic scene in the presence of transient distractors, we propose Focus4DGS. Our framework learns to decompose the scene into two distinct geometric representations. A single, coherent set of dynamic Gaussians models the primary subject, while separate, per-view sets of Gaussians capture transient phenomena. Crucially, this decomposition is guided not by explicit masks but by an uncertainty signal derived directly from the learned dynamics. An overview of the full pipeline is shown in Figure 6.1.

6.3.1 Decomposed 4D Gaussian Splatting

Our method represents the scene with two distinct sets of 3D Gaussians, extending the decompositional approach introduced for static scenes in DeSplat [66].

Core Gaussians The first component, the Core Gaussians G_{core}, models the stable scene content, including the static background and the primary dynamic subject. These Gaussians are defined in a canonical space and made dynamic via a learned deformation field N, implemented as an MLP. For a given canonical position μ_i and time t, the MLP predicts deformation offsets:

$$N(\mu_i, t) = (\Delta\mu_i, \Delta R_i, \Delta S_i) \tag{6.1}$$

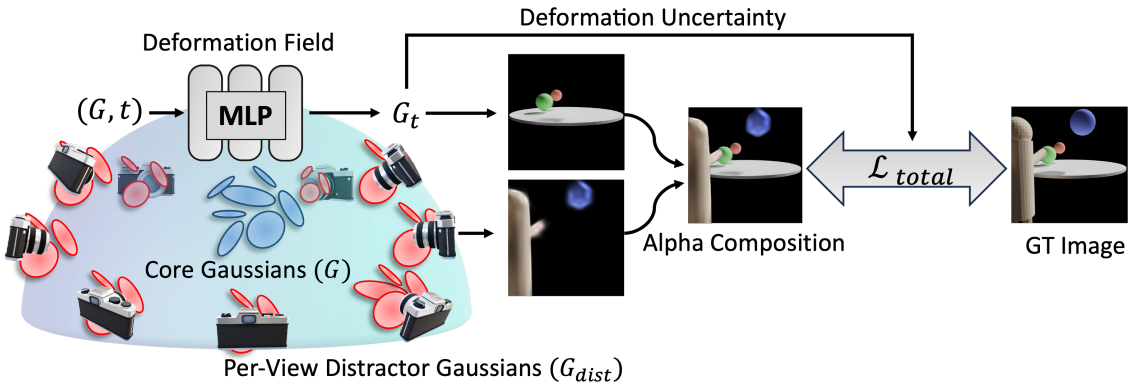


Figure 6.1: Overview of the Focus4DGS pipeline.

These offsets update the position, rotation, and scale of each Gaussian to reflect its configuration at time t, enabling complex, non-rigid motion modeling.

Distractor Gaussians We adopt the formulation introduced in DeSplat [66], where each training view maintains its own independent set of Distractor Gaussians $G_{\text{dist},n}$ to model view-dependent transients such as occluders and regions of high uncertainty. These Gaussians are initialized on a view-aligned plane and are not shared across views.

Initialization. For each camera view n with camera-to-world rotation R_n and translation $\mathbf{t}_n$, we initialize K distractor Gaussians by sampling their positions on a 2D plane in front of the camera. Each distractor's 3D location $\boldsymbol{\mu}_d$ is computed as:

$$\boldsymbol{\mu}_d = \mathbf{t}_n - R_n^\top \mathbf{u}, \quad \text{where} \quad \mathbf{u} = \begin{bmatrix} \rho u \\ \rho v \\ \rho \end{bmatrix}, \quad u, v \sim \mathcal{U}(0, 1) \tag{6.2}$$

Here, ρ is a small constant that defines the depth of the plane in front of the camera (we use $\rho = 0.02$). The vector $\mathbf{u}$ is sampled in camera space and defines distractor locations on a 2D plane. The color of each distractor is modeled as an RGB vector $\mathbf{c} \in \mathbb{R}^3$, without spherical harmonics, reflecting their purely view-dependent nature.

Densification. Densification is also handled per view, following the strategy from DeSplat. We apply Adaptive Density Control (ADC) [21] based on view-space

gradient magnitudes, but only trigger updates for view n when a sample counter s_n exceeds a threshold S. This allows each view to independently grow or prune its distractor set, preserving localized transient modeling without interfering with other views.

Compositing As in DeSplat [66], we render the core and distractor layers separately and then combine them through alpha compositing. For each frame at time t, the final image is computed as

$$c_{\text{comp}} = c_{\text{dist}} + (1 - \alpha_{\text{dist}})c_{\text{core}} \tag{6.3}$$

where c_{dist} and α_{dist} are the accumulated color and opacity of the distractor layer. This ensures that distractor content correctly occludes the core scene where necessary, producing a unified rendering for supervision.

6.3.2 Guiding Decomposition with Scene Dynamics

A standard photometric loss is insufficient to enforce a meaningful separation, as it would encourage the powerful Core model G_{core} to overfit to all visual phenomena. To solve this, we introduce a dual-signal regularization strategy that leverages both internal motion stability and external temporal consistency. This strategy creates a principled handoff mechanism, ensuring that the Core Gaussians model only what is stable and consistent, while offloading phenomena that are too challenging to the per-view Distractor Gaussians.

Jacobian-Based Motion Uncertainty Our first guiding signal measures the internal stability of the learned motion. We propose that unstable, chaotic, or difficult-to-model motion corresponds to high sensitivity in the learned deformation field, $\mathcal{N}$. We quantify this sensitivity by analyzing the Jacobian of the network's output with respect to its inputs.

For each Core Gaussian g_i, we compute the Jacobian of the predicted offsets $(\Delta\mu_i, \Delta q_i, \Delta s_i)$ with respect to the inputs (canonical position μ_i and time t). For example, for the position offset $\Delta\mu$, the Jacobian $J_{\Delta\mu}$ is the matrix of partial

Figure 6.2: Visualization of Jacobian-based motion uncertainty. In this scene, the subject lifts a pair of weights, causing rapid arm motion. Due to their speed and complex deformation, the arms exhibit high motion uncertainty, as captured by our Jacobian-derived metric. Brighter regions indicate higher uncertainty, highlighting areas where the deformation field is less stable and harder to model.

derivatives:

$$J_{\Delta\mu} = \frac{\partial \Delta\mu}{\partial(\mu, t)}$$

Our uncertainty score is derived from the sample covariance of the Jacobian's rows. The rows of the Jacobian represent the gradients of each output dimension with respect to the inputs; high variance among these gradients suggests an unstable and unpredictable mapping. The uncertainty contribution from a single parameter type (e.g., position) is the trace of this covariance matrix:

$$U_{\Delta\mu}(g_i) = \mathrm{Tr}\left(\mathrm{Cov}(J_{\Delta\mu})\right) \tag{6.4}$$

The trace of a covariance matrix is equivalent to the sum of the variances of its constituent variables, thus providing a measure of the total sensitivity of the deformation. In practice, we compute this Jacobian using a stochastically perturbed time input $t' = t + \epsilon$ for a more robust local estimate.

The total uncertainty score for a Gaussian g_i is the sum of the scores for each parameter type:

$$U(g_i) = U_{\Delta\mu}(g_i) + U_{\Delta q}(g_i) + U_{\Delta s}(g_i) \tag{6.5}$$

Figure 6.2 visualizes the per-gaussian uncertainty scores on a synthetic scene.

Uncertainty-Guided Optimization Once high-uncertainty regions are detected, the next step is to **suppress** their influence in the core model, encouraging a **handover** to the per-view distractor model. This is achieved through a combined optimization framework. We adopt the alpha-based regularization from DeSplat [66], which encourages a sparse distractor layer ($\lambda_d|\alpha_{dist}|$) and an opaque core layer ($\lambda_s|1-\alpha_{core}|$):

$$\mathcal{L}_{alpha} = \lambda_s|1 - \alpha_{core}| + \lambda_d|\alpha_{dist}| \tag{6.6}$$

Given this, we introduce an uncertainty-aware regularizer that directly penalizes core Gaussians for being both uncertain and opaque. This uncertainty loss is formulated as:

$$\mathcal{L}_{unc} = \sum_{g_i \in G_{core}} U(g_i) \cdot \alpha_{core,i} \tag{6.7}$$

This term creates a competition: a Gaussian can either have a stable motion field (low $U(g_i)$) or it must become transparent (low $\alpha_{core,i}$) to minimize the loss. This explicitly enforces the suppress mechanism on unstable parts of the core model. As a core region is suppressed, the photometric error in that area would increase. The per-view distractor Gaussians, which are rendered on top, are then optimized to take over and explain this residual appearance, thus completing the handover.

The final loss function combines these terms:

$$\mathcal{L}_{total} = \mathcal{L}_{photo} + \mathcal{L}_{alpha} + \lambda_{unc}\mathcal{L}_{unc} \tag{6.8}$$

This framework ensures that the Core Gaussians model only what is stable and consistent, while systematically offloading transient or unreconstructable phenomena to the per-view Distractor Gaussians.

6.4 Experiments

To validate Focus4DGS, we conducted a series of experiments on both synthetic and real-world datasets. Our evaluation is designed to assess the framework's ability to distinguish between reconstructable and unreconstructable motion, thereby preserving the integrity of the primary dynamic subject in unconstrained environments.

6.4.1 Experimental Setup

Datasets. To evaluate our method, we use a combination of custom synthetic datasets, designed to test our core hypothesis, and established real-world benchmarks.

(a) Bouncing Balls (b) Squatting

Figure 6.3: Example synthetic datasets.

As shown in Figure 6.3, our synthetic evaluation includes a **Bouncing Balls** scene with both predictable and chaotic motion, and a **Squatting** scene where a primary dynamic subject is occluded by a transient running person. For quantitative comparison, we also use standard real-world static scene benchmarks where any dynamic elements are considered distractors.

Baselines. We compare Focus4DGS against state-of-the-art methods tailored to different conditions. For fully dynamic scenes, we compare against leading reconstruction methods: Deformable 3D Gaussians [73] and 4D Gaussian Splatting [70]. For static scenes with dynamic distractors, we compare against specialized distractor-removal techniques: DeSplat [66], SpotLessSplats [51], WildGaussians [23], and NeRF On-the-go [49].

Evaluation Metrics. We quantify performance using three standard image-quality metrics: Peak Signal-to-Noise Ratio (PSNR)(↑), Structural Similarity Index (SSIM)(↑), and Learned Perceptual Image Patch Similarity (LPIPS)(↓).

Implementation Details. Our method is implemented in PyTorch [44] and trained on a single NVIDIA H200 GPU. We train for 60k iterations. Each camera is initialized with $K = 1000$ distractor Gaussians using the strategy outlined in 6.3.1, which is consistent with [66]. We perform ADC on the distractor gaussians after all training images have been seen for $S = 10$ times.

6.4.2 Results and Analysis

We now present our experimental results, beginning with a qualitative comparison on our synthetic and real-world test cases, followed by a quantitative analysis.

Qualitative Comparison. Our uncertainty-guided decomposition strategy proves highly effective at isolating primary dynamic subjects from visual distractors. As shown in Figures 6.4 and 6.5, competing methods like DeformableGS attempt to model all motion indiscriminately, resulting in corrupted geometry and significant artifacts when faced with inconsistent trajectories. In contrast, Focus4DGS correctly identifies these hard-to-model regions, assigns them high uncertainty, and delegates them to the per-view distractor representation. This principled separation allows the core model to produce a clean and stable reconstruction of the primary subject. We observe similar success on real-world videos (Figure 6.6), where our method effectively models the core content while capturing transient occluders and ambiguous motion in the separate distractor splats, preserving the integrity of the main reconstruction.

Quantitative Comparison. To demonstrate the versatility of our framework, we evaluated it on established static-scene benchmarks where all dynamic content is treated as transient. The results are presented in Table 6.1. While our method is designed for fully dynamic scenes, it remains competitive with specialized, state-of-the-art distractor removal techniques. This shows that our uncertainty mechanism is flexible enough to successfully isolate a static background by treating all dynamic content as high-uncertainty, confirming the robustness of our approach.

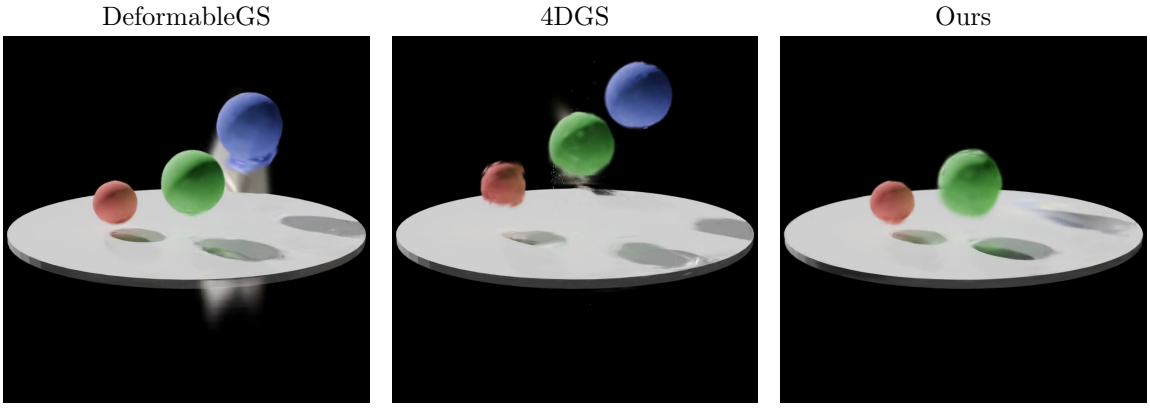


Figure 6.4: Qualitative comparison on the Bouncing Ball scene. Prior methods struggle to handle dynamic elements like the runner and the blue ball, resulting in noticeable artifacts and degraded scene quality.

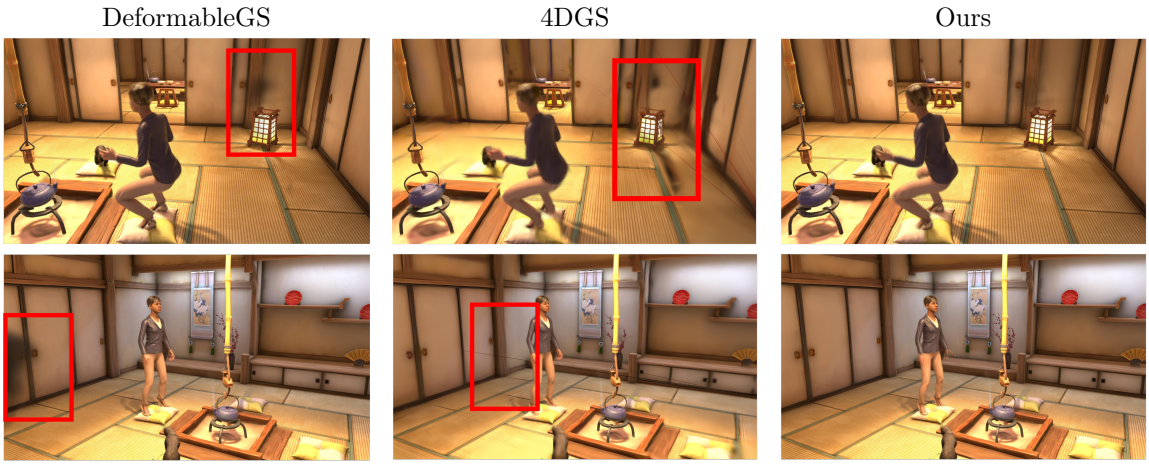


Figure 6.5: Qualitative comparison on synthetic squat scene.

Method	Mountain			Fountain			Corner			Patio		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF On-the-go	20.15	0.64	0.26	20.11	0.61	0.31	24.22	0.81	0.19	20.78	0.75	0.22
SpotLessSplats	22.53	0.77	0.18	22.81	0.80	0.15	26.43	0.90	0.10	22.24	0.86	0.10
WildGaussians	20.43	0.65	0.26	20.81	0.66	0.22	24.16	0.82	0.20	21.44	0.80	0.14
DeSplat	19.59	0.71	0.17	20.27	0.68	0.17	26.05	0.88	0.09	20.89	0.81	0.11
Focus4DGS (ours)	19.04	0.54	0.43	19.64	0.59	0.33	24.31	0.83	0.19	20.69	0.74	0.29

Table 6.1: Quantitative comparison on real static scenes with distractors. Our method, designed for the more general task of dynamic scenes, performs competitively against specialized approaches.

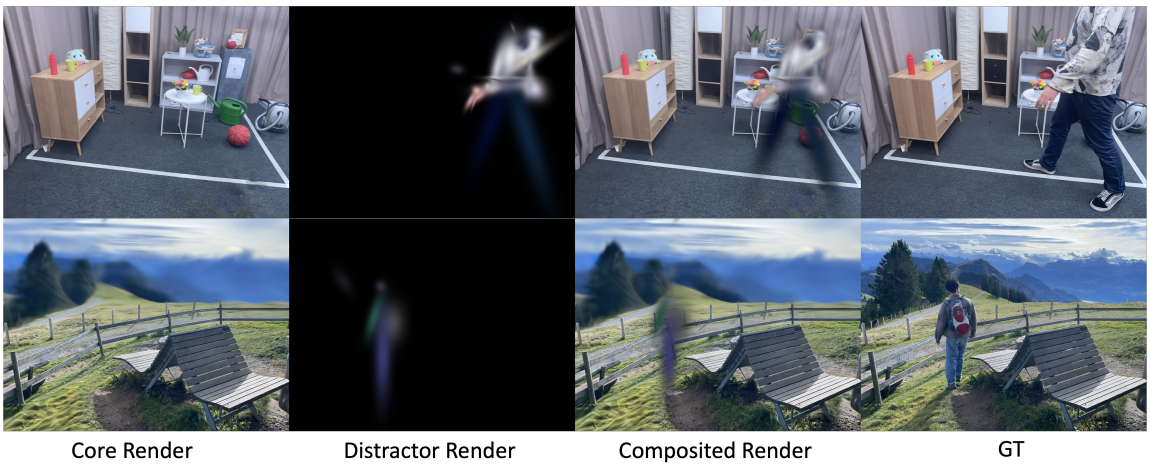


Figure 6.6: Qualitative results on real static scenes.

6.5 Conclusion

This chapter introduced Focus4DGS, a novel framework for robustly reconstructing dynamic scenes in the presence of visual distractors. Our work redefines the challenge from a simple static-versus-dynamic separation to a more nuanced distinction between reconstructable and unreconstructable motion. The key to our approach is a decompositional architecture that models a scene with a primary, dynamic set of Core Gaussians and auxiliary per-view Distractor Gaussians.

Crucially, this decomposition is not supervised by explicit masks but is guided by an uncertainty signal derived from the stability of the learned motion field itself. By analyzing the Jacobian of the deformation network, we produce a per-Gaussian uncertainty score that allows the model to identify and isolate regions of inconsistent or chaotic motion. An uncertainty-aware loss function then encourages the model to hand over these challenging regions to the distractor representation, preserving the integrity of the core reconstruction. Experiments show that this method effectively disentangles the primary dynamic subject from transient phenomena, enabling high-fidelity results in unconstrained environments where previous methods fail.

Chapter 7

Conclusions

7.1 Limitations and Future Work

The work presented in this thesis advances the state of the art. However, they have certain limitations that present clear opportunities for future research.

Dependency of Pre-computation and Initialization. DyLiN requires a pre-trained teacher NeRF model, which can be expensive to obtain. Furthermore, all of our Gaussian Splatting-based methods, particularly GS-LK, are sensitive to the quality of the Structure-from-Motion (SfM) point cloud and the accuracy of the camera parameters.

Modeling Extreme Dynamics and Materials. Methods like CoGS may struggle with extremely non-rigid deformations or large-scale movements in monocular settings. Additionally, challenges remain for the Gaussian Splatting based methods in accurately modeling objects with highly complex or view-dependent materials, such as shiny or transparent surfaces.

Generative Inpainting. For Focus4DGS, generative inpainting methods could be applied to fill the holes left behind after distractors are removed, leading to more complete and visually plausible scene reconstructions.

Preliminary Validation of Focus4DGS. While Chapter 6 introduces the complete framework for Focus4DGS, its experimental validation has not yet been completed. The method has not been thoroughly tested on a diverse range of challenging real-world scenes, which is essential to fully assess its practical effectiveness.

7.2 Summary

This thesis addressed fundamental challenges in dynamic 3D scene reconstruction, introducing four novel methods that advance the field toward real-time, controllable, and robust performance in unconstrained environments. Our contributions provide a comprehensive pipeline for reasoning about dynamic geometry, from improving foundational rendering speed to enabling high-level scene manipulation and understanding.

Our work began by tackling the issue of performance with **DyLiN**, the first Light Field Network capable of modeling dynamic scenes. By leveraging knowledge distillation, DyLiN achieves an order-of-magnitude increase in rendering speed over previous methods while maintaining high visual fidelity. Building on the efficiency of explicit representations, we then introduced **CoGS**, a framework that extends 3D Gaussian Splatting to enable intuitive, fine-grained control of dynamic elements without requiring dense supervision. To address geometric ambiguities in challenging capture conditions, **GS-LK** integrates classical Lucas-Kanade principles with Gaussian Splatting. This hybrid approach regularizes the scene's motion via an analytically derived scene flow, significantly improving tracking and reconstruction from monocular videos with limited camera movement. Finally, **Focus4DGS** confronts the problem of real-world visual clutter by learning to separate consistent, reconstructable motion from transient distractors. By explicitly modeling uncertainty in the learned motion field, it decomposes the scene into a stable core representation and transient per-view distractor Gaussians, allowing it to robustly reconstruct the primary dynamic subject while gracefully ignoring ambiguous visual phenomena.

Bibliography

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust efficient point cloud registration using pointnet. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, 2019. doi: 10.1109/CVPR.2019.00733. 5.2, 5.3, 5.4.1

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, et al. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proc. IEEE/CVF ICCV*, pages 5855–5864, 2021. 3.1

[3] Eduard Batschelet. Über die numerische Auflösung von Randwertproblemen bei elliptischen partiellen Differentialgleichungen. *ZAMP Zeitschrift für angewandte Mathematik und Physik*, 3(3), 1952. ISSN 00442275. doi: 10.1007/BF02008824. 5.4.2

[4] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model Compression. In *Proc. 12th ACM SIGKDD ICKDDM*, pages 535–541, 2006. 3.2

[5] Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. 2023. doi: 10.1109/cvpr52729.2023.00021. 5.2

[6] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning Efficient Object Detection Models with Knowledge Distillation. *Adv. NeurIPS*, 30, 2017. 3.2

[7] Ricky T. Q. Chen. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`. 5.5.1

[8] Shuo Chen, Binbin Yan, Xinzhu Sang, Duo Chen, Peng Wang, Xiao Guo, Chongli Zhong, and Huaming Wan. Bidirectional Optical Flow NeRF: High Accuracy and High Quality under Fewer Views. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, volume 37, 2023. doi: 10.1609/aaai.v37i1.25109. 5.1, 5.2

[9] Zhiqin Chen. *IM-NET: Learning implicit fields for generative shape modeling*. PhD thesis, Applied Sciences: School of Computing Science, 2019. 3.1

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *Proc. IEEE/CVF CVPR*, pages 12882–12891, 2022. 3.1

[11] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, et al. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *arXiv:2205.15285*, 2022. 3.1, 3.2, 3.4.3, 3.1, 3.2, 3.3c, 3.3h, 3.3, 3.4c, 4.1, 4.2, 4.7c, 4.7h, 4.8c, 4.8h, 5.2

[12] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, et al. Plenoxels: Radiance Fields Without Neural Networks. In *Proc. IEEE/CVF CVPR*, pages 5501–5510, 2022. 3.1, 3.4.3, 3.1, 3.3, 4.1

[13] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. In *Proc. IEEE/CVF CVPR*, pages 8649–8658, 2021. 3.1, 3.2, 5.2

[14] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 33768–33780. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/dab5a29f6614ec47ea0ca85c140226fd-Paper-Conference.pdf. 5.1, 5.5.2

[15] Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. GaussianFlow: Splatting Gaussian Dynamics for 4D Content Creation. 3 2024. 5.2, 5.4.2

[16] Inc. GoPro. Go Pro Hero 7, 2023. 5.5.2

[17] Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-aware 3D Gaussian Splatting for Efficient Dynamic Scene Reconstruction. 3 2024. 5.2, 5.4.2, 5.5.2

[18] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *20th ICPR*, pages 2366–2369. IEEE, 2010. 3.4.4, 4.4.3

[19] Lubin Fan Bojian Wu Yujing Lou Renjie Chen Ligang Liu Jieping Ye Jingyu Lin, Jiaqi Gu. Hybridgs: Decoupling transients and statics with 2d and 3d gaussian splatting. In *Arxiv*, 2024. 6.2.2

[20] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. CoNeRF: Controllable Neural Radiance Fields. In *Proc. IEEE/CVF CVPR*, pages 18623–18632, 2022. 3.4.3, 4.2, 4.4.1

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions*

on Graphics (ToG), 42(4):1–14, 2023. 2.2.3, 2.2.3, 2.2.3, 4.1, 4.3.1, 4.1, 4.2, 4.4.2, 4.7d, 4.7i, 4.8d, 4.8i, 5.1, 5.2, 5.5.1, 5.5.2, 6.2.1, 6.3.1

[22] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014. 3.4.2, 4.4.2, 5.5.1

[23] Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. WildGaussians: 3D gaussian splatting in the wild. *NeurIPS*, 2024. 6.4.1

[24] Marc Levoy and Pat Hanrahan. Light Field Rendering. In *Proc. 23rd CGIT*, pages 31–42, 1996. 3.2

[25] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural Scene Flow Prior. In *Advances in Neural Information Processing Systems*, volume 10, 2021. 5.2

[26] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Pointnetlk revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12763–12772, June 2021. 5.2, 5.4.1, 5.5.2

[27] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast Neural Scene Flow. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9878–9890, 2023. 5.2

[28] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 3.4.3, 3.2, 3.3, 4.2, 5.1, 5.2, 5.5.2

[29] David B Lindell, Julien NP Martel, and Gordon Wetzstein. AutoInt: Anutomatic Integration for Fast Neural Volume Rendering. In *Proc. IEEE/CVF CVPR*, pages 14556–14565, 2021. 3.1, 3.2

[30] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. *Adv. NeurIPS*, 33:15651–15663, 2020. 3.1, 3.2

[31] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, et al. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019. 3.4.3, 3.2, 3.3, 4.2

[32] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 4.3.1, 4.3.2, 5.1, 5.2, 5.3

[33] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, et al. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proc. IEEE/CVF CVPR*, pages 7210–7219, 2021. 3.1, 6.2.2

[34] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proc. IEEE/CVF CVPR*, pages 4460–4470, 2019. 3.1

[35] Natan Micheletti, Jim H Chandler, and Stuart N Lane. Structure from Motion (SfM) Photogrammetry. *British Society for Geomorphology Geomorphological Techniques*, 2(2):1–12, 2015. ISSN 2047-0371. doi: 10.5194/isprsarchives-XL-5-W4-37-2015. 5.1

[36] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Commun. ACM*, 65(1):99–106, 2021. 2.2.1, 3.1, 3.4.3, 3.1, 3.2, 3.3, 4.1, 4.1, 4.2, 5.1, 5.2, 5.5.1

[37] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images. In *Proc. IEEE/CVF CVPR*, pages 16190–16199, 2022. 3.1

[38] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, et al. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *CGF*, volume 40, pages 45–59. Wiley Online Library, 2021. 3.1

[39] Nvidia Corporation. NVIDIA A100 Tensor Core, 2024. 5.5.1

[40] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *ICML*, pages 2642–2651. PMLR, 2017. 3.4.4, 4.4.3

[41] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proc. IEEE/CVF CVPR*, pages 165–174, 2019. 3.1

[42] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, et al. Nerfies: Deformable Neural Radiance Fields. In *Proc. IEEE/CVF ICCV*, pages 5865–5874, 2021. 3.1, 3.2, 3.3.1, 3.4.3, 3.2, 4.2, 5.2, 5.5.2

[43] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, et al. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Trans. Graph.*, 40(6):1–12, 2021. 3.1, 3.2, 3.3.1, 3.3.2, 3.4.1, 3.4.3, 3.2, 3.3, 3.4b, 3.5b, 3.5g, 4.4.1, 4.2, 4.8b, 4.8g, 4.4.3, 5.2, 5.5.2

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 4.4.2, 5.5.1, 6.4.1

[45] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, 2016. doi: 10.1109/CVPR.2016.85. 5.5.2

[46] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proc. IEEE/CVF CVPR*, pages 10318–10327, 2021. 2.3, 3.1, 3.2, 3.3.2, 3.4.1, 3.4.3, 3.4.3, 3.1, 3.3b, 3.3g, 3.3, 4.1, 4.4.1, 4.7b, 4.7g, 4.4.3, 5.2

[47] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, et al. DeRF: Decomposed Radiance Fields. In *Proc. IEEE/CVF CVPR*, pages 14153–14161, 2021. 3.1

[48] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *Proc. IEEE/CVF ICCV*, pages 14335–14345, 2021. 3.1, 3.2

[49] Weining Ren, Zihan Zhu, Boyang Sun, Jiaqi Chen, Marc Pollefeys, and Songyou Peng. Nerf on-the-go: Exploiting uncertainty for distractor-free nerfs in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 6.2.2, 6.4.1

[50] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J. Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20626–20636, June 2023. 6.2.2

[51] Sara Sabour, Lily Goli, George Kopanas, Mark Matthews, Dmitry Lagun, Leonidas Guibas, Alec Jacobson, David J. Fleet, and Andrea Tagliasacchi. SpotLessSplats: Ignoring distractors in 3d gaussian splatting. *arXiv:2406.20055*, 2024. 6.2.2, 6.4.1

[52] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4.3.1, 5.1

[53] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering. In *Proc. NeurIPS*, 2021. 3.1

[54] Colton Stearns, Adam W. Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic gaussian marbles for novel view synthesis of casual monocular videos. In *ArXiv*, 2024. 5.5.2

[55] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *Proc. IEEE/CVF*

CVPR, 2022. 3.4.3, 3.1, 3.3, 4.1

[56] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, et al. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video. In *Proc. IEEE/CVF ICCV*, pages 12959–12970, 2021. 3.1, 3.2, 5.2, 5.5.2

[57] Paul Ungermann, Armin Ettenhofer, Matthias Nießner, and Barbara Roessle. Robust 3d gaussian splatting for novel view synthesis in presence of distractors. In *DAGM German Conference on Pattern Recognition*, pages 153–167. Springer, 2024. 6.2.2

[58] S. Vedula, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005. doi: 10.1109/TPAMI.2005.63. 5.4

[59] Bruce A. Wallace. Merging and transformation of raster images for cartoon animation. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1981*, 1981. doi: 10.1145/800224.806813. 5.2

[60] Chaoyang Wang, Lachlan Ewen MacDonald, Laszlo A Jeni, and Simon Lucey. Flow supervision for deformable nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21128–21137, 2023. 5.2, 5.4.1, 5.5.2

[61] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative Distillation for Ultra-Resolution Universal Style Transfer. In *Proc. IEEE/CVF CVPR*, pages 1860–1869, 2020. 3.2

[62] Huan Wang, Jian Ren, Zeng Huang, Kyle Olszewski, Menglei Chai, et al. R2L: Distilling Neural Radiance Field to Neural Light Field for Efficient Novel View Synthesis. *arXiv:2203.17261*, 2022. 2.2.2, 3.1, 3.2, 3.3.2, 3.4.2, 3.4.2, 3.5.1

[63] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, et al. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-time. In *Proc. IEEE/CVF CVPR*, pages 13524–13534, 2022. 3.1, 3.2

[64] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 3.2

[65] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, et al. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Adv. NeurIPS*, 34:27171–27183, 2021. 3.1

[66] Yihao Wang, Marcus Klasson, Matias Turkulainen, Shuzhe Wang, Juho Kannala, and Arno Solin. DeSplat: Decomposed Gaussian splatting for distractor-free ren-

dering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 6.2.2, 6.3.1, 6.3.1, 6.3.1, 6.3.2, 6.4.1, 6.4.1

[67] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The 37th Asilomar SSC*, volume 2, pages 1398–1402. Ieee, 2003. 3.4.4

[68] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004. 3.4.4, 4.4.3, 5.5.2

[69] Stephen Welstead. *Fractal and Wavelet Image Compression Techniques*. 2009. doi: 10.1117/3.353798. 5.5.2

[70] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2.3, 5.2, 5.3, 5.5.2, 6.2.1, 6.4.1

[71] Liuyue Xie, Joel Julin, Koichiro Niinuma, and Laszlo A. Jeni. Gaussian splatting lucas-kanade, 2025. URL https://arxiv.org/abs/2407.11309. 1.3

[72] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, et al. Point-NeRF: Point-based Neural Radiance Fields. In *Proc. IEEE/CVF CVPR*, pages 5438–5448, 2022. 3.1

[73] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2.3, 4.3.1, 4.4.3, 5.1, 5.2, 5.3, 5.4.2, 5.5.2, 6.2.1, 6.4.1

[74] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.00538. 5.5.2

[75] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, et al. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *Proc. IEEE/CVF ICCV*, pages 5752–5761, 2021. 3.1

[76] Heng Yu, Joel Julin, Zoltan A Milacski, Koichiro Niinuma, and Laszlo A Jeni. Cogs: Controllable gaussian splatting. *arXiv*, 2023. 1.3

[77] Heng Yu, Joel Julin, Zoltan A Milacski, Koichiro Niinuma, and László A Jeni. Dylin: Making light field networks dynamic. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12397–12406, 2023. 1.3, 4.3.2, 5.2

[78] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. IEEE/CVF CVPR*, pages 586–595, 2018. 3.4.4, 4.4.3, 5.5.2

[79] Jianqiao Zheng, Sameera Ramasinghe, Xueqian Li, and Simon Lucey. Trading positional complexity vs. deepness in coordinate networks. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 5.5.1

[80] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. doi: 10.1109/VISUAL.2001.964490. 2.2.3