Warsaw University of Technology

FACULTY OF ELECTRONICS AND INFORMATION TECHNOLOGY

PhD Thesis

in the discipline of Information and Communication Technology

Controllability of Radiance Fields with Sparse Information

Kacper Kania, M. Sc.

supervisor
Tomasz Trzciński, Prof. PhD DSc

assistant supervisor
Marek Kowalski, PhD

WARSZAWA 2025

Abstract

This thesis is a series of publications that introduce novel methods for neural rendering using limited information, focusing on Neural Radiance Fields (NeRFs) and 3D Gaussian Splatting (3DGS). It explores how these models construct 3D representations from 2D images and demonstrates ways to condition these representations for generating high-quality renderings of 3D objects.

We present that both NeRFs and 3DGS representations are not *controllable* in an interpretable way—once trained, one can no longer change the content of the scene such that output reflects the changes intended in the input. We propose four complementary techniques that use simple, interpretable inputs derived from sparse training data and extend these methods to perform effectively in few-shot learning scenarios.

We begin by examining the area of Neural Radiance Fields addressing limitations in existing approaches and presenting contributions to achieve controllable radiance fields. By incorporating partial and sparse data during training, our NeRF extension—CoNeRF—leverages the smoothness of neural networks to produce controllable, high-quality images across general scene types. We show that sparse, manual annotations obtainable in the matter of a minute are sufficient to provide an interpretable control over the NeRF's output thanks to the spectral bias of neural networks. We support the argument with extensive evaluation results, showcasing that CoNeRF is among the first steps to fully controllable radiance fields.

Modeling fine-grained deformations, like wrinkling on human faces, and effects those deformation have on the surrounding space require extensive, high-quality data annotations in multi-view videos. To overcome this reliance on the high data volume, we introduce a new method, BlendFields, for rendering fine-grained deformations within neural radiance fields in a few-shot, multi-view setting. This approach learns internal deformation templates, which are blended smoothly during inference, significantly improving image quality compared to existing baselines and enabling effective rendering from limited input images. To achieve that, we leverage the volume representation which takes form of tetrahedral cages. Using traditional techniques in computer graphics, we compute physically-based deformation quantities that provide smooth and interpretable interpolation between known "extreme" deformations. We show in our experiments that BlendFields achieves higher quality of renderings from just a few shots which contrasts to previous works that rely on hundreds of frames in a single capture.

Moreover, we develop a novel model, LumiGauss, for controlling radiance fields through environmental lighting for "in-the-wild" captures under varying daylight conditions. By incorporating precomputed radiance transfer, well-known in computer graphics, this model enables physically plausible scene relighting and provides users with intuitive control over lighting in reconstructed scenes. We achieve this goal despite the fact that each training image is captured under different lighting conditions and does not contain any information about light presented in the scene.

Many applications of neural rendering require high computation throughput, thus in the last work of this thesis we focus on adaptable computation efficiency with our CLoG. It applies a coarse-to-fine learning strategy for 3D Gaussian Splatting, which upscales a 2D grid storing Gaussian latent representations. This strategy achieves competitive results while allowing deployment on various computational devices with minimal quality loss. As demonstrated in our experiments, CLoG becomes increasingly efficient during inference with fewer Gaussians.

This research advances the state of the art in controllable neural radiance fields and expands their application to few-shot learning scenarios. These techniques enhance the possibilities for neural rendering of objects in the scene from limited information, and open new directions for future research in the field.

Keywords: Neural Rendering, Neural Radiance Fields, Few-Shot Learning, Human Rendering, Partial Information, Gaussian Splatting

Streszczenie

Ta praca doktorska składa się z serii publikacji, które wprowadzają nowe metody renderowania neuronowego przy użyciu ograniczonej informacji, koncentrując się na Neuronowych Polach Radiancji (NeRF) i Rozsmarowywaniu Gaussów 3D (3DGS). Bada ona, w jaki sposób te modele konstruują reprezentacje 3D z obrazów 2D, i demonstruje sposoby warunkowania tych reprezentacji do generowania wysokiej jakości renderów obiektów 3D.

Przedstawiamy, że zarówno reprezentacje NeRF, jak i 3DGS nie są *kontrolowalne* w interpretowalny sposób—po wytrenowaniu nie można już zmienić zawartości sceny tak, aby wynik odzwierciedlał zamierzone zmiany w danych wejściowych. Proponujemy cztery komplementarne techniki, które wykorzystują proste, interpretowalne dane wejściowe pochodzące z rzadkich danych treningowych i rozszerzają te metody, aby skutecznie działały w scenariuszach uczenia się z kilku przykładów.

Zaczynamy od analizy obszaru Neuronowych Pól Radiancji, zajmując się ograniczeniami istniejących podejść i przedstawiając wkład w osiągnięcie kontrolowalnych pól radiancji. Poprzez włączenie częściowych, rzadkich danych podczas treningu, nasze rozszerzenie NeRFów—CoNeRF—wykorzystuje gładkość sieci neuronowych do tworzenia kontrolowalnych obrazów wysokiej jakości dla różnych typów scen. Pokazujemy, że rzadkie, ręczne adnotacje, które można uzyskać w ciągu minuty, są wystarczające do zapewnienia interpretowalnej kontroli nad wynikiem NeRFów dzięki tendencji sieci neuronowych do faworyzowania niskich częstotliwości. Podpieramy ten argument obszernymi wynikami ewaluacji, pokazując, że CoNeRF jest jednym z pierwszych kroków w kierunku w pełni kontrolowalnych pól radiancji.

Modelowanie drobnych deformacji, jak marszczenie się na twarzy, oraz efektów, które te deformacje mają na otaczającą przestrzeń, wymaga wielu wysokiej jakości adnotacji w obrębie ujęć z wielu kamer. Aby zredukować tę zależność jakości obrazów od ilości danych, przedstawiamy nową metodę, BlendFields, która modeluje te deformacje przy pomocy kilku ujęć z wielu kamer. To podejście uczy się wewnętrznych szablonów deformacji, które są płynnie mieszane w czasie wnioskowania, znacznie poprawiając jakość obrazu w porównaniu do istniejących podejść i umożliwiając efektywne renderowanie z ograniczonej liczby obrazów wejściowych. W tym celu, wykorzystujemy reprezentację przestrzeni, która przyjmuje formę sieci czworościanów. Używając tradycyjnych technik w grafice komputerowej, obliczamy wielkości deformacji oparte na fizycznych równaniach, które zapewniają płynną i interpretowalną interpolację między

znanymi “granicznymi” deformacjami. Pokazujemy w naszych eksperymentach, że BlendFields osiąga wyższą jakość renderów z zaledwie kilku ujęć, co kontrastuje z poprzednimi pracami, które opierają się na nagraniach składających się z setek klatek.

Ponadto prezentujemy model do kontrolowania pól radiancji poprzez oświetlenie środowiskowe dla ujęć „w warunkach naturalnych” przy różnych warunkach oświetleniowych—LumiGauss. Poprzez użycie wyliczonego transferu radiancji, dobrze znanego w grafice komputerowej, model ten umożliwia fizycznie wiarygodne przeliczanie oświetlenia sceny i zapewnia użytkownikom intuicyjną kontrolę nad oświetleniem w zrekonstruowanych scenach. Osiągamy ten cel mimo faktu, że każdy obraz treningowy jest wykonany w różnych warunkach oświetleniowych i nie zawiera żadnych informacji o świetle obecnym w scenie.

Wiele zastosowań renderowania neuronowego wymaga wysokiej wydajności, dlatego w ostatniej pracy tej rozprawy koncentrujemy się na adaptowalnej efektywności obliczeniowej z naszym CLoG. Stosuje on strategię uczenia „od ogółu do szczegółu” dla Rozsmarowywania Gaussów 3D, która skaluje siatkę 2D przechowującą ukryte reprezentacje Gaussów. Ta strategia osiąga konkurencyjne wyniki, jednocześnie umożliwiając wdrożenie na urządzeniach z różnymi zdolnościami obliczeniowymi przy minimalnej utracie jakości. Jak wykazujemy w naszych eksperymentach, CLoG jest tym wydajniejszy im mniej Gaussów jest zadanych jako wejście.

Te badania rozwijają najnowsze osiągnięcia w dziedzinie kontrolowalnych neuronowych pól radiancji i rozszerzają ich zastosowanie do scenariuszy uczenia się z kilku przykładów. Nasze metody zwiększają możliwości renderowania neuronowego obiektów w scenie z ograniczonych informacji i otwierają nowe kierunki dla przyszłych badań w tej dziedzinie.

Słowa kluczowe: Renderowanie Sieciami Neuronowymi, Neuronowe Pola Radiancji, Uczenie z Kilku Ujęć, Renderowanie Awatarów, Warunkowanie Częściową Informacją, Rozsmarowywanie Gaussów

Publications in this thesis

Title	Authors	Venue	Status
CoNeRF: Controllable Neural Radiance Fields	**Kacper Kania**, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, Andrea Tagliasacchi	CVPR 2022	Accepted
BlendFields: Few-Shot Example-Driven Facial Modeling	**Kacper Kania**, Stephan J. Garbin, Andrea Taglisacchi, Virginia Estellers, Kwang Moo Yi, Julien Valentin, Tomasz Trzciński, Marek Kowalski	CVPR 2023	Accepted
LumiGauss: High-Fidelity Outdoor Relighting with 2D Gaussian Splatting	Joanna Kaleta, **Kacper Kania**, Tomasz Trzciński, Marek Kowalski	WACV 2025	Accepted
CLoG: Leveraging UV Space for Continuous Levels of Detail	**Kacper Kania**, Rawal Khirodkar, Shunsuke Saito, Kwang Moo Yi, Julieta Martinez	ICCV 2025	Under Review

Table 1. Publications – We present the list of publications that are included in this thesis. We show the records in the chronological order, together with their titles, authors, venues and the status at the moment of writing this thesis.

Contents

Chapter 1

Introduction

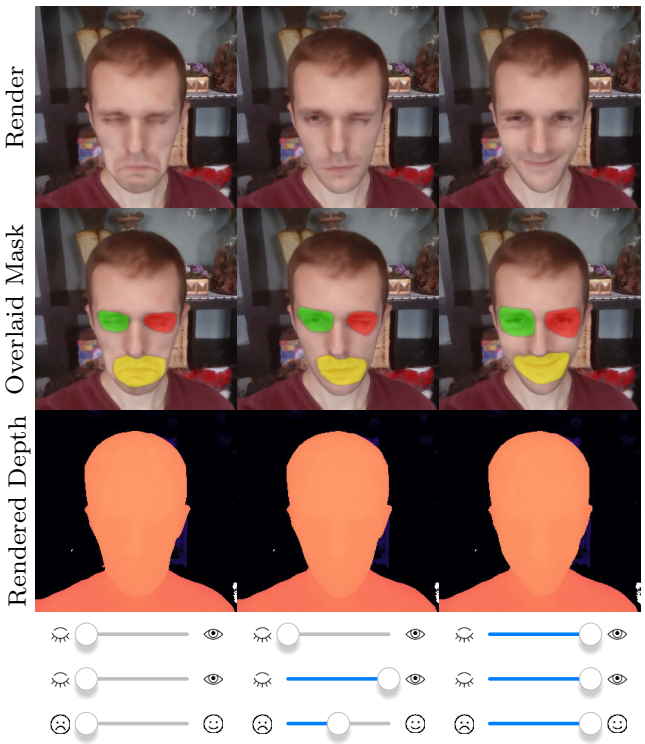
With the advent of deep learning, researchers have explored various ways to apply it to computer graphics. One of the most recent and promising approaches is neural rendering, a field that combines deep learning and computer graphics to generate realistic images of 3D scenes. A prominent neural rendering technique is the neural radiance field (NeRF [104]), which represents a 3D scene as a continuous function mapping 3D coordinates to radiance values. NeRF has demonstrated impressive results in synthesizing photorealistic images but suffers from high memory and computational demands, making it challenging to scale to large scenes. Moreover, its implicit representation prohibits its use in real-time applications such as game engines.

To address these limitations, Kerbl et al. [71] introduced 3D Gaussian Splatting (3DGS), a neural rendering technique that represents a 3D scene as a set of 3D Gaussians, which are splatted into image space using the algorithm proposed by Zwicker et al. [225]. Unlike NeRF, 3DGS can render large scenes in real time. It enables rendering millions of points at interactive frame rates on a single GPU, making it a viable option for real-time applications. However, both techniques share a common problem that limits their application in real-world scenarios—lack of an *interpretable controllability*.

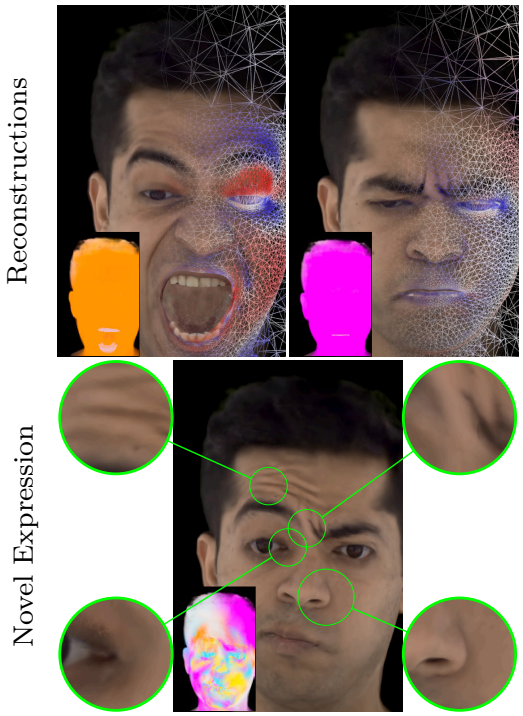
This thesis addresses the challenges of achieving controllability in neural rendering, with a focus on Neural Radiance Fields (NeRFs) and Gaussian Splatting (3DGS). The work presented here reflects the research journey as we tackled key obstacles in controlling texture, expression, lighting, and computational efficiency in radiance fields. In the following sections, we describe the motivation behind this work, outline the research questions that guide our inquiry, summarize my contributions, and finally, provide an overview of the thesis structure.

1.1 Motivation and Challenges

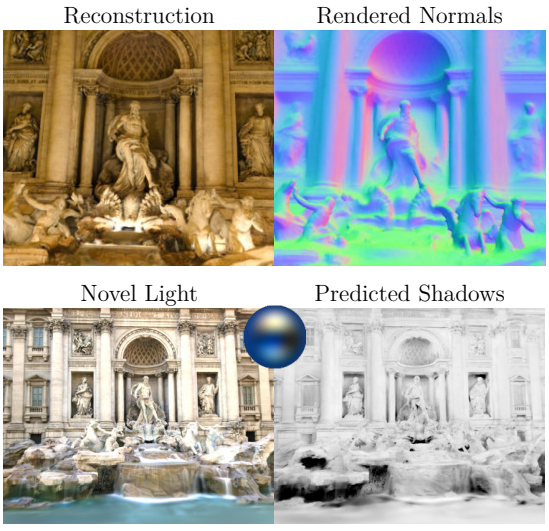
NeRF and 3DGS are powerful techniques for generating realistic images. However, once trained, their representations become uncontrollable—editing the underlying representation has an unpredictable effect on the output images. This lack of flexibility makes it challenging



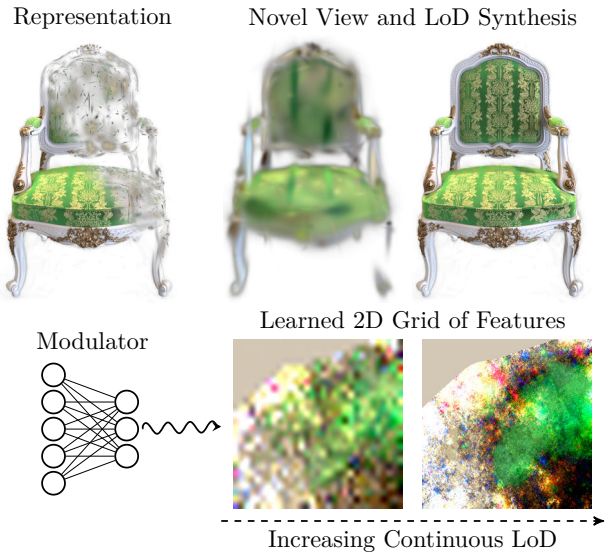
(a) CoNeRF – control with sparse annotations



(b) BlendFields – control with physical features



(c) LumiGauss – control with physical constraints



(d) CLoG – control with structural constraints

Figure 1. Teaser – This thesis introduces a series of publications that advance the state-of-the-art in controllable radiance field methods. We present the following papers: CoNeRF [69] in Fig. 1a, BlendFields [66] in Fig. 1b, LumiGauss [65] in Fig. 1c and CLoG in Fig. 1d. Together, these complementary approaches enable unprecedented control over 3D content through sparse annotations (CoNeRF and BlendFields), dynamic environment lighting (LumiGauss), and optimized computational efficiency for single-image rendering (CLoG), addressing key challenges in modern neural rendering systems.

to control key properties such as lighting, subject attributes, and scene composition, which are crucial for practical applications in graphics and vision. Enabling controllability in these models is essential to expanding their usability beyond passive scene reconstruction. This thesis aims to address this challenge by exploring methods to introduce intuitive and efficient control mechanisms.

Controllability is a fundamental requirement in computer graphics, especially in artistic workflows that rely on iterative refinement. Consider a game artist designing a 3D character who decides to use volumetric representations instead of meshes in their game. The artist's rationale is that NeRF and 3DGS reconstructions enable unprecedented immersion with low cost while creating hyperrealistic meshes and textures require an extensive effort, infeasible for large-scale products like games. This reason makes radiance field representations an appealing alternative to the computer graphics pipeline. However, modifying a trained NeRF or 3DGS without disrupting its realism is complex, often requiring extensive retraining. This limitation hinders real-time edits and reduces their viability for interactive applications.

Addressing this issue requires novel approaches that integrate controllability without compromising rendering quality or efficiency. In this thesis, we explore techniques that bridge the gap between neural representations and the adaptability demanded by real-world applications. We present the overview of our proposed solutions in Sec. 1.1.

Data-driven controllability. After the introduction of NeRFs [104], NeRFies [119] pioneered high-quality reconstructions from casual video sequences of dynamic scenes. However, NeRFies initially provided only limited control, relying on simple linear interpolation of latent embeddings to represent time.

HyperNeRF [120] addressed this limitation by introducing a high-dimensional embedding space, which is projected onto a lower-dimensional representation to enable more interpretable scene traversal. Despite this improvement, the projection remains non-linear and uninterpretable, making the effects of adjustments unpredictable and difficult to control.

Template-based controllability. Applying the approaches mentioned above to large-scale object deformations, such as jumping jacks, or to miniscule changes, such as wrinkling effects, is ineffective. To overcome the former issue, data-driven approaches like Fang et al. [34] use multi-scale voxel structures to model deformations. However, these methods perform well primarily in synthetic settings [124]. The latter issue is tackled by methods such as EditNeRF [90] and FigNeRF [189] which focus on static elements, restricting control to colors and textures.

An alternative strategy relies on template-based models, where a 3D mesh is used to canonicalize deformed points [223]. However, these methods are highly sensitive to registration accuracy, which is inherently imperfect [35, 222]. Even minor misalignments are amplified in radiance field models, which assume a perfectly calibrated scene. To compensate, existing

approaches introduce latent spaces [45, 69, 99] that require thousands of frames to capture high-fidelity deformations, making them impractical for data-efficient learning.

Physically-based illumination controllability. Beyond controlling shape and texture, realistic scene illumination is essential for practical applications. However, inverse rendering is inherently ill-posed, as multiple lighting conditions can produce identical observations [121]. Some approaches rely on datasets of a single object under diverse lighting conditions [19, 135, 200] which is infeasible for thousands of objects. These methods either fail to disentangle albedo from lighting [19, 200] or require auxiliary networks to predict shadows [135], thereby limiting their practicality.

Efficiency controllability. Volumetric representations are computationally expensive compared to traditional meshes. Training a NeRF model on a single scene can take several days on an NVIDIA V100 GPU, with rendering times exceeding 60 seconds per image—far from practical applications. While various speed-up methods exist [42, 54, 109, 130, 203], they often trade off memory, quality, or speed.

3D Gaussian Splatting (3DGS) [71] has emerged as a powerful alternative, offering high-quality rendering at interactive frame rates. However, achieving these speeds still requires high-end GPUs. We see promise in 3DGS as a canonical 3D representation, similar to meshes, yet its adaptability across different computational resources remains an open challenge.

1.2 Research Objectives

In this thesis, we explore diverse avenues to achieve radiance field controllability, addressing the challenges introduced above. Our goal is to answer the following research questions:

(**RQ 1**) Can a Neural Radiance Field (NeRF) be endowed with controllability by leveraging sparse annotations in the training dataset? How many annotations are sufficient to enable smooth interpolation between controlled values?

(**RQ 2**) Are extreme facial expressions, as documented in the literature, adequate for learning expression-dependent details that extrapolate to unseen expressions during training?

(**RQ 3**) Is it possible to learn an underlying radiance transfer function of a scene from images captured in unconstrained, "in-the-wild" settings? Can this transfer function generalize to novel environment maps?

(**RQ 4**) How can we learn a single 3D Gaussian Splatting (3DGS) representation that is adaptable to different computational regimes at inference time in a feed-forward manner?

Each question represents one of many possible directions for achieving controllability in radiance fields. In this thesis, we categorize our methods according to four control aspects: texture [66, 69], shape and expression [66], lighting [65], and resource usage [67]. Integrating

the solutions presented below addresses texture, expression, lighting, and resource adaptability, offering a comprehensive solution to radiance field controllability.

1.2.1 Controlling the Object from Partial Information

To address (**RQ 1**), we introduce CoNeRF [69], an approach that leverages sparse annotations to control subject attributes in a post-hoc manner. Early NeRF-based methods [90] were simple models—they overfit to a single subject and require retraining for new subjects—with limited editability. More sophisticated models, such as NeRFactor [217], operate only on simple, calibrated scenes and cannot handle motion. Inspired by HyperNeRF [120], CoNeRF revisits the use of a low-dimensional latent space. We introduce our concept of weak supervision to enable interpretable controllability of that space. By assuming that only a few sparse annotations (including region-specific labels) are provided, and by exploiting the smoothness of the MLPs used in NeRFs [161], the annotation signal is propagated across similar frames. This approach not only enables semantic segmentation of radiance fields but also decouples attribute controls. Such coarse annotations that can be generated in a few minutes suffice to achieve both novel view synthesis and novel attribute manipulations, though the annotation complexity grows with the number of controlled attributes. Our method enables flexible, interpretable edits to NeRFs while maintaining high rendering quality.

The article describing CoNeRF was published at the Conference on Computer Vision and Pattern Recognition (CVPR) 2022.

1.2.2 Controlling the Expression from Few-Shot Learning

In response to (**RQ 2**), we propose BlendFields [66] to handle dynamic facial expressions and effects it has on face due to the surface compression and decompression. Although CoNeRF is capable of rendering motions given sufficient data and annotations, acquiring such data for complex motions (*e.g.*, speaking full sentences) is challenging. Prior methods like VolTeMorph [41] employ face template models (such as FLAME [84]) to derive a canonical representation by using a tetrahedral cage that moves points to a fixed "neutral" position. However, this procedure is insufficient to capture realistic facial features like wrinkles. Our novel contribution, BlendFields, builds on VolTeMorph and introduces a computation of the deformation gradient for each tetrahedron—a physically based, interpretable quantity that guides smooth transitions in facial textures. Each NeRF branch that build BlendFields' model is overfit to a specific extreme expression, and the model predicts face bases conditioned on the expression vector to output the final color. This framework yields spatially coherent, expression-dependent details even when provided with only a few extreme expressions. It also significantly reduces data requirements while maintaining high fidelity, making it a practical solution for real-world applications.

Our work on BlendFields was accepted at CVPR 2023.

1.2.3 Controlling the Light from Unconstrained Images

For (**RQ 3**), we introduce LumiGauss [65] to tackle the challenge of illumination controllability. Previous approaches assumed idealized capture conditions with constant camera exposure and lighting, resulting in blended colors that preclude dynamic light changes. In contrast, our approach decouples the intrinsic color of the subject from lighting effects by learning the radiance transfer function directly from unconstrained photo collections. We employ 2D Gaussian Splatting (2DGS) [57] to obtain a smooth and spatially coherent surface—a task that is difficult with 3DGS [71]. By endowing the Gaussians with additional features corresponding to radiance transfer, expressed as Spherical Harmonics [46], LumiGauss renders realistic scenes and shadows that adapt to novel environment maps, achieving higher fidelity than prior approaches. By integrating precomputed radiance transfer techniques [128, 145], LumiGauss also achieves controllable, physically-plausible lighting.

This work was accepted at WACV 2025.

1.2.4 Controlling the Level of Details in One Model

Finally, to address (**RQ 4**), we propose CLoG [67] as a solution to the high computational demands of current radiance field models. All the contributions above typically require training dedicated models on high-end hardware. Inspired by the gaming industry's Levels of Detail (LoD) techniques, CLoG trains a single Gaussian Splatting model that can be modulated at inference time to adapt to various computational regimes with minimal loss in rendering quality. By constraining the number of Gaussians in 3DGS [71] to a fixed count—reshaped into a 2D grid—and using a coarse-to-fine training protocol, the model learns a high-quality volumetric representation. In a subsequent training stage, we spatially sort the Gaussians so that similar descriptors are adjacent, forming a low-frequency image that is upsampled using an off-the-shelf continuous architecture [169]. Our experiments demonstrate that even with as few as 2,000 Gaussians (compared to the typical 10^5–10^6), the object remains recognizable and the model adapts continuously—eliminating the need to retrain for different LoD settings. CLoG learns the underlying representation once and dynamically adapts it in a feed-forward manner.

Our work is currently under review at ICCV 2025.

1.3 Contributions

Based on these research questions, my contributions, corresponding to the projects introduced above, are summarized in four major points.

CoNeRF: Controllable Neural Radiance Fields. I introduced the idea of using partial annotations as a control signal for learning controllable radiance fields. For this work, I developed the entire Python codebase, designed the final CoNeRF architecture, created all the datasets,

conducted the experiments, and wrote the initial version of the article. This work laid the foundation for controlling subject attributes post-hoc and motivated further exploration into handling larger deformations.

BlendFields: Few-Shot Example-Driven Facial Modeling. To overcome CoNeRF's limitations in modeling large deformations and miniscule expression-dependent changes, I extended the approach in BlendFields—a method currently included in a US patent submission. I designed and performed early experiments that exposed the shortcomings of existing condition-based approaches for realistic facial wrinkle modeling in a few-shot setting. Motivated by discussions with my collaborators, I explored the use of wrinkle-maps technique known in computer graphics within neural radiance fields, devised the branched architecture, and executed the experimental evaluations. I also authored the majority of the paper's content and contributed substantially to its refinement.

LumiGauss: High-Fidelity Outdoor Relighting with 2D Gaussian Splatting. For LumiGauss, which addresses the challenge of illumination controllability, I played a supervisory role. I contributed to the design of the final architecture, co-authored the revised version of the paper, corrected figures, and oversaw further improvements. This work decouples intrinsic subject color from environmental lighting by learning the radiance transfer function from unconstrained images.

CLoG: Leveraging UV Space for Continuous Levels of Detail. I developed a solution for adapting a single 3D Gaussian Splatting model to varying computational regimes. In CLoG, I designed and performed the initial experiments that identified the primary research goal, built the complete architecture and training regime, implemented the full codebase, executed all experiments, and wrote the initial version of the article. This method enables continuous adaptation to different levels of detail in a feed-forward manner, eliminating the need for retraining when hardware constraints change.

Collectively, these contributions explore complementary strategies for achieving radiance field controllability. They not only address distinct control dimensions—texture, expression, lighting, and resource efficiency—but also interrelate to form a cohesive framework for advancing neural rendering.

1.4 Thesis Outline

The thesis is organized into four main chapters, each corresponding to one of the research questions and contributions described above. In Chapter 2, we introduce CoNeRF, our approach for controlling neural radiance fields using partial annotations. Next, Chapter 3 presents BlendFields, a method that generates realistic, expression-dependent textures from just a few

multi-view frames. Chapter 4 details LumiGauss, a Gaussian Splatting model that learns a radiance transfer function to enable novel lighting effects in unconstrained settings. Finally, Chapter 5 describes CLoG, a general approach that leverages Gaussian Splatting to achieve continuous levels-of-details across different computational regimes. We conclude in Chapter 6 with a discussion of future research directions and the broader impact of this work.

1.5 Publications not included in the thesis

I attach a list of articles that I co-authored and which are related to and can be used to in neural rendering approaches but do not form parts of the thesis:

- **Kania**, **K.**, Zięba, M., and Kajdanowicz, T., "UCSG-NET – Unsupervised Discovering of Constructive Solid Geometry Tree," *NeurIPS*, vol. 33, pp. 8776–8786, 2020,
- **Kania**, **K.**, Kowalski, M., and Trzciński, T., "TrajeVAE: Controllable Human Motion Generation from Trajectories," *arXiv preprint arXiv:2104.00351*, 2021,
- Stypułkowski, M., **Kania**, **K.**, Zamorski, M., Zięba, M., Trzciński, T., and Chorowski, J., "Representing Point Clouds with Generative Conditional Invertible Flow Networks," *Pattern Recognition Letters*, vol. 150, pp. 26–32, 2021,
- Xia, S., Yue, J., **Kania**, **K.**, Fang, L., Tagliasacchi, A., Yi, K. M., and Sun, W., "Densify Your Labels: Unsupervised Clustering with Bipartite Matching for Weakly Supervised Point Cloud Segmentation," *arXiv preprint arXiv:2312.06799*, 2023,
- Esposito, S., Xu, Q., **Kania**, **K.**, Hewitt, C., Mariotti, O., Petikam, L., Valentin, J., Onken, A., and Mac Aodha, O., "GeoGen: Geometry-Aware Generative Modeling via Signed Distance Functions," in *CVPRW*, 2024, pp. 7479–7488,
- Spurek, P., Winczowski, S., Zięba, M., Trzciński, T., **Kania**, **K.**, and Mazur, M., "Modeling 3D Surfaces with a Locally Conditioned Atlas," in *ICCS*, Springer, 2024, pp. 100–115.

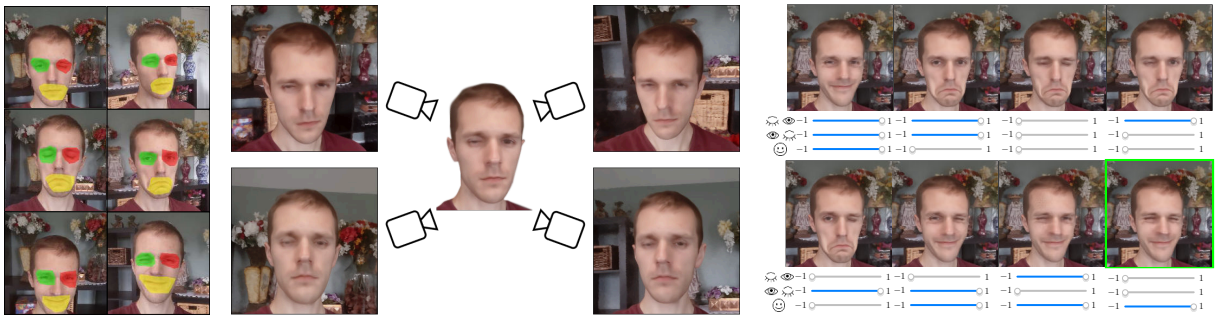
Chapter 2

CoNeRF: Controllable Neural Radiance Fields

In this chapter, we extend neural 3D representations to allow for intuitive and interpretable user control beyond novel view rendering (i.e. camera control). We allow the user to annotate which part of the scene one wishes to control with just a small number of mask annotations in the training images. Our key idea is to treat the attributes as latent variables that are regressed by the neural network given the scene encoding. This leads to a few-shot learning framework, where attributes are discovered automatically by the framework, when annotations are not provided. We apply our method to various scenes with different types of controllable attributes (e.g. expression control on human faces, or state control in movement of inanimate objects). Overall, we demonstrate, to the best of our knowledge, for the first time novel view and novel attribute re-rendering of scenes from a single video.

2.1 Introduction

Neural radiance field (NeRF) [104] methods have recently gained popularity thanks to their ability to render photorealistic novel-view images [99, 118, 120, 212]. In order to widen the scope to other possible applications, such as digital media production, a natural question is whether these methods could be extended to enable *direct* and *intuitive* control by a digital artist, or even a casual user. However, current techniques only allow coarse-grain controls over materials [217], color [61], or object placement [198], or only support changes that they are designed to deal with, such as shape deformations on a learned shape space of chairs [90], or are limited to facial expressions encoded by an explicit face model [38]. By contrast, we are interested in *fine-grained* control without limiting ourselves to a specific class of objects or their properties. For example, given a self-portrait video, we would like to be able to control individual *attributes* (e.g. whether the mouth is open or closed); see Fig. 2. We would like to achieve this objective with minimal user intervention, without the need of specialized capture setups [89].



(a) annotations **(b)** novel views **(c)** novel attributes

Figure 2. Teaser – We train a controllable neural radiance field from multiple views of a dynamic 3D scene, under varying poses and attributes; in this example eye being open/closed and mouth smiling/frowning. Given only six annotations (a), our method provides full control over the scene appearance, allowing us to synthesize (b) novel views and (c) novel attributes, including attribute combinations that were *never seen* in the training data (green box).

However, it is unclear how fine-grained control can be achieved, as current state-of-the-art models [120] encode the structure of the 3D scene in a *single* and *not interpretable* latent code. For the example of face manipulation, one could attempt to resolve this problem by providing *dense* supervision by matching images to the corresponding Facial Action Coding System (FACS) [30] action units. Unfortunately, this would require either an automatic annotation process or careful and extensive per-frame human annotations, making the process expensive, generally unwieldy, and, most importantly, domain-specific. Automated tools for domain-agnostic latent disentanglement are a very active topic of research in machine learning [18, 55, 56], but no effective plug-and-play solution exists yet.

Conversely, we borrow ideas from 3D morphable models (3DMM) [11], and in particular to recent extensions that achieve local control by *spatial disentanglement* of control attributes [110, 184]. Rather than having a single global code controlling the expression of the *entire* face, we would like to have a set of *local* "attributes", each controlling the corresponding *localized* appearance; more specifically, we assume spatial quasi-conditional independence of attributes [184]. For our example in Fig. 2, we seek an attribute capable to control the appearance of the mouth, another to control the appearance of the eye, etc.

Thus, we introduce a learning framework denoted CoNeRF (i.e. Controllable NeRF) that is capable of achieving this objective with just *few-shot* supervision. As illustrated in Fig. 2, given a single one-minute video, and with as little as two annotations per attribute, CoNeRF allows *fine-grained*, *direct*, and *interpretable* control over attributes. Our core idea is to provide, on top of the ground truth attribute tuple, *sparse* 2D mask annotations that specify which region of the image an attribute controls, in spirit of Interactive Digital Photomontage [2] and Video Sprites [139]. Further, by treating attributes as latent variables within the framework, the mask annotations can be automatically propagated to the whole input video. Thanks to the quasi-conditional independence of attributes, our technique allows us to synthesize expressions that

were *never* seen at training time; e.g. the input video never contained a frame where both eyes were closed and the actor had a smiling expression; see Fig. 2 (green box).

Contributions To summarize, our CoNeRF method[1]:

- provides *direct*, *intuitive*, and *fine-grained* control over 3D neural representations encoded as NeRF;
- achieves this via *few-shot* supervision, *e.g.*, just a handful of annotations in the form of attribute values and corresponding 2D mask are needed for a one minute video;
- while inspired by domain-specific facial animation research [184], it provides a *domain-agnostic* technique.

2.2 Related works

Neural Radiance Fields [104] provide high-quality renderings of scenes from novel views with just a few exemplar images captured by a handheld device. Various extensions have been suggested to date. These include ones that focus on improving the quality of results [99, 118, 120, 212], ones that allow a single model to be used for multiple scenes [141, 167], and some considering controllability of the rendering output at a coarse level [49, 90, 189, 198, 205, 217], as we detail next.

In more detail, existing works enable only compositional control of object location [198, 205], and recent extensions also allow for finer-grain reproduction of global illumination effects [49]. NeRFactor [217] shows one can model albedos and BRDFs, and shadows, which can be used to, *e.g.*, edit material, but the manipulation they support is limited to what is modeled through the rendering equation. CodeNeRF [61] and EditNeRF [90] showed that one can edit NeRF models by modifying the shape and appearance encoding, but they require a curated dataset of objects viewed under different views and colors. HyperNeRF [120], on the other hand can adapt to unseen changes specific to the scene, but learns an arbitrary attribute (ambient) space that cannot be supervised, and, as we show in Sec. 2.4, cannot be easily related to specific local attribute within the scene for controllability.

Explicit supervision One can also condition NeRF representations [38] with face attribute predicted by pre-trained face tracking networks, such as Face2Face [165]. Similarly, for human bodies, A-NeRF [149] and NARF [115] use the SMPL [94] model to generate interpretable pose parameters, and Neural Actor [89] further includes normal and texture maps for more detailed rendering. While these models result in controllable NeRF, they are limited to domain-specific control and the availability of a heavily engineered control model.

[1] Code and dataset are released here.

Controllable neural implicits Controllability of neural 3D *implicit* representations has also been addressed by the research community. Many works have limited focus on learning *human* neural implicit representations while enabling the control via SMPL parameters [94], or linear blend skinning weights [3, 26, 53, 97, 102, 137, 219, 224]. Some initial attempts at learned disentangled of shape and poses have also been made in A-SDF [108], allowing behavior control of the output geometry (*e.g.* doors open vs. closed) while maintaining the general shape. However, the approach is limited to controlling SE(3) articulation of objects, and requires dense 3D supervision.

2.2.1 Neural Radiance Field (NeRF)

For completeness, we briefly discuss NeRF before diving into the details of our method. A Neural Radiance Field captures a volumetric representation of a specific scene within the weights of a neural network. As input, it receives a sample position $\mathbf{x}$ and a view direction $\mathbf{v}$ and outputs the density of the scene σ at position $\mathbf{x}$ as well as the color $\mathbf{c}$ at position $\mathbf{x}$ as seen from view direction $\mathbf{v}$. One then renders image pixels $\mathbf{C}$ via volume rendering [64]. In more detail, $\mathbf{x}$ is defined by observing rays $\mathbf{r}(t)$ as $\mathbf{x} = \mathbf{r}(t)$, where t parameterizes at which point of the ray you are computing for. One then renders the color of each pixel $\mathbf{C}(\mathbf{r})$ by computing

$$\mathbf{C}\left(\mathbf{r}\right) = \int_{t_n}^{t_f} T(t)\sigma\left(\mathbf{r}(t)\right)\mathbf{c}\left(\mathbf{r}(t), \mathbf{v}\right) dt \ , \tag{1}$$

where $\mathbf{v}$ is the viewing angle of the ray $\mathbf{r}$, t_n and t_f are the near and far planes of the rendering volume, and

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right) \ , \tag{2}$$

is the accumulated transmittance. Integration in Eq. (1) is typically done via numerical integration [104].

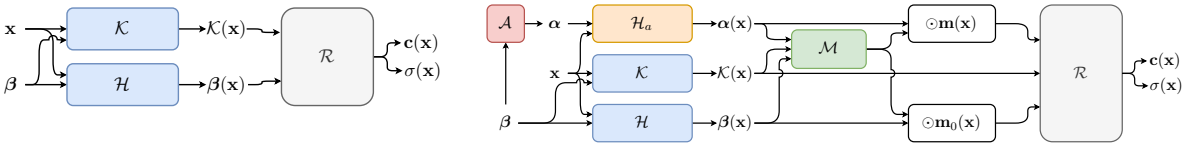
2.2.2 HyperNeRF

Note that in its original formulation Eq. (1) is only able to model *static* scenes. Various recent works [118, 120, 166] have been proposed to explicitly account for possible appearance changes in a scene (for example, temporal changes in a video). To achieve this, they introduce the notion of *canonical hyperspace* – more formally given a 3D query point $\mathbf{x}$ and the collection $\boldsymbol{\theta}$ of all parameters that describe the model, they define:

$$\mathcal{K}(\mathbf{x}) \equiv \mathcal{K}(\mathbf{x}; \boldsymbol{\beta}, \boldsymbol{\theta}), \qquad \text{Canonicalizer} \tag{3}$$

$$\boldsymbol{\beta}(\mathbf{x}) \equiv \mathcal{H}(\mathbf{x}; \boldsymbol{\beta}, \boldsymbol{\theta}), \qquad \text{Hyper Map} \tag{4}$$

$$\mathbf{c}(\mathbf{x}), \sigma(\mathbf{x}) = \mathcal{R}(\mathcal{K}(\mathbf{x}), \boldsymbol{\beta}(\mathbf{x}); \boldsymbol{\theta}). \qquad \text{Hyper NeRF} \tag{5}$$



(a) Neural radiance field with lifting [120] **(b)** Controllable neural radiance field (our method)

Figure 3. Framework – We depict in (a) the HyperNeRF [120] formulation, and (b) our Controllable-NeRF (CoNeRF). In (a), both point coordinates $\mathbf{x}$ and latent representation $\boldsymbol{\beta}$ are respectively processed by a canonicalizer $\mathcal{K}$ and a hyper map $\mathcal{H}$, which are then turned into radiance and density field values by $\mathcal{R}$. In (b), we introduce regressors $\mathcal{A}$ and $\mathcal{M}$ that regress the attribute and the corresponding mask that enable few-shot attribute-based control of the NeRF model. See Sec. 2.3.3 for details.

where the location is canonicalized via a canonicalizer $\mathcal{K}$, and the appearances, represented by $\boldsymbol{\beta}$, are mapped to a hyperspace via $\mathcal{H}$, which are then utilized by another neural network $\mathcal{R}$ to retrieve the color $\mathbf{c}$ and the density σ at the query location. Note throughout this paper we denote $\boldsymbol{\beta}$ to indicates a latent code, while $\boldsymbol{\beta}(\mathbf{x})$ to indicate the corresponding field generated by the hypermap lifting. With this latent lifting, these methods render the scene via Eq. (1). Note that the original NeRF model can be thought of the case where $\mathcal{K}$ and $\mathcal{H}$ are identity mappings.

2.3 Controllable NeRF (CoNeRF)

Given a collection of C color images $\{\mathbf{C}_c\} \in [0,1]^{W\times H\times 3}$, we train our controllable neural radiance field model by an auto-decoding optimization [117] whose losses can be grouped into two main subsets:

$$\underset{\boldsymbol{\theta}=\boldsymbol{\theta},\{\boldsymbol{\beta}_c\}}{\arg\min}\ \underbrace{\ell_{\text{rep}}(\boldsymbol{\theta};\{\mathbf{C}_c\})}_{\text{Sec. 2.3.1}} + \underbrace{\ell_{\text{ctrl}}(\boldsymbol{\theta};\{\mathbf{M}^{\text{gt}}_{c,a}\},\{\alpha^{\text{gt}}_{c,a}\})}_{\text{Sec. 2.3.2}}. \tag{6}$$

The first group consists of the classical HyperNeRF [120] auto-decoder losses, attempting to optimize neural network parameters $\boldsymbol{\theta}$ jointly with latent codes $\{\boldsymbol{\beta}_c\}$ to *reproduce* the corresponding input images $\{\mathbf{C}_c\}$:

$$\ell_{\text{rep}}(\cdot) = \ell_{\text{recon}}(\boldsymbol{\theta},\{\boldsymbol{\beta}_c\};\{\mathbf{C}_c\}) + \ell_{\text{enc}}(\{\boldsymbol{\beta}_c\}). \tag{7}$$

The latter allow us to inject *explicit control* into the representation, and are our core contribution:

$$\ell_{\text{ctrl}}(\cdot) = \ell_{\text{mask}}(\boldsymbol{\theta},\{\boldsymbol{\beta}_c\};\{\mathbf{M}^{\text{gt}}_{c,a}\}) \qquad \text{g.t. masks} \tag{8}$$

$$+\ \ell_{\text{attr}}(\boldsymbol{\theta},\{\boldsymbol{\beta}_c\};\{\alpha^{\text{gt}}_{c,a}\}). \qquad \text{g.t. attributes} \tag{9}$$

As mentioned earlier in Sec. 2.1, we aim for a neural 3D appearance model that is controlled by a collection of attributes $\boldsymbol{\alpha}=\{\alpha_a\}$, and we expect each image to be a manifestation of a different value of attributes, that is, each image $\mathbf{C}_c$, and hence each latent code $\boldsymbol{\beta}_c$, will have a

corresponding attribute $\boldsymbol{\alpha}_c$. The learnable connection between latent codes $\boldsymbol{\beta}$ and the attributes $\boldsymbol{\alpha}$, which we represent via regressors, is detailed in Sec. 2.3.3.

2.3.1 Reconstruction losses

The primary loss guiding the training of the NeRF model is the reconstruction loss, which simply aims to reconstruct observations $\{\mathbf{C}_c\}$. As in other neural radiance field models [99, 104, 118, 120] we simply minimize the L2 photometric reconstruction error with respect to ground truth images:

$$\ell_{\text{recon}}(\cdot) = \sum_c \mathbb{E}_{\mathbf{r}\sim\mathbf{C}_c}\left[\left\|\mathbf{C}(\mathbf{r};\boldsymbol{\beta}_c,\boldsymbol{\theta}) - \mathbf{C}^{\text{gt}}(\mathbf{r})\right\|_2^2\right]. \tag{10}$$

As is typical in auto-decoders, and following [117], we impose a zero-mean Gaussian prior on the latent codes $\{\boldsymbol{\beta}_c\}$:

$$\ell_{\text{enc}}(\cdot) = \sum_c \|\boldsymbol{\beta}_c\|_2^2. \tag{11}$$

2.3.2 Control losses

The user defines a *discrete* set of A number of attributes that they seek to control, that are *sparsely* supervised across frames—we only supervise attributes *when* we have an annotation, and let others be discovered on their own throughout the training process, as guided by Eq. (7). More specifically, for a particular image $\mathbf{C}_c$, and a particular attribute α_a, the user specifies the quantities:

- $\alpha_{c,a} \in [-1, 1]$: specifying the value for the a-th attribute in the c-th image; see the *sliders* in Fig. 2;
- $\mathbf{M}_{c,a} \in [0, 1]^{W\times H}$: roughly specifying the image region that is controlled by the a-th attribute in the c-th image; see the *masks* in Fig. 2.

To formalize sparse supervision, we employ an indicator function $\mathbb{1}_{c,a}$, where $\mathbb{1}_{c,a} = 1$ if an annotation for attribute a for image c is provided, otherwise $\mathbb{1}_{c,a} = 0$. We then write the loss for *attribute* supervision as:

$$\ell_{\text{attr}}(\cdot) = \sum_c \sum_a \mathbb{1}_{c,a} |\alpha_{c,a} - \alpha_{c,a}^{\text{gt}}|^2. \tag{12}$$

For the mask few-shot supervision, we employ the volume rendering in Eq. (20) to project the 3D volumetric neural mask field $\mathbf{m}_a(\mathbf{x})$ into image space, and then supervise it as:

$$\ell_{\text{mask}}(\cdot) = \sum_{c,a} \mathbb{1}_{c,a}\, \mathbb{E}_{\mathbf{r}}\left[\text{CE}\left(\mathbf{M}(\mathbf{r};\boldsymbol{\beta}_c,\boldsymbol{\theta}), \mathbf{M}_{c,a}^{\text{gt}}(\mathbf{r})\right)\right], \tag{13}$$

where $\text{CE}(\cdot,\cdot)$ denotes cross entropy, and the field $\sigma(\mathbf{x})$ in Eq. (20) is learned by minimizing Eq. (10). Importantly, as we do not wish for Eq. (13) to interfere with the training of the underlying 3D representation learned through Eq. (10), we *stop gradients* in Eq. (13) w.r.t.

$\sigma(\mathbf{x})$. Furthermore, in practice, because the attribute mask vs. background distribution can be highly imbalanced depending on which attribute the user is trying to control (*e.g.* an eye only covers a very small portion of an image), we employ a *focal loss* [87] in place of the standard cross entropy loss.

2.3.3 Controlling and rendering images

In what follows, we drop the image subscript c to simplify notation without any loss of generality. Given a B-dimensional latent code $\boldsymbol{\beta}$ representing the 3D scene behind an image, we derive a mapping to our A attributes via a neural map $\mathcal{A}$ with learnable parameters $\boldsymbol{\theta}$:

$$\{\alpha_a\} = \mathcal{A}(\boldsymbol{\beta}; \boldsymbol{\theta}), \quad \mathcal{A} : \mathbb{R}^B \rightarrow [0,1]^A, \tag{14}$$

where these correspond to the *sliders* in Fig. 2. In the same spirit of Eq. (4), to allow for complex topological changes that may not be represented by the change in a single scalar value alone, we lift the attributes to a hyperspace. In addition, since each attribute governs different aspects of the scene, we employ *per-attribute* learnable hypermaps $\{\mathcal{H}_a\}$, which we write:

$$\alpha_a(\mathbf{x}) = \mathcal{H}_a(\mathbf{x}, \alpha_a; \boldsymbol{\theta}) \quad \mathcal{H}_a : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^d. \tag{15}$$

Note that while α_a is a scalar *value*, $\alpha_a(\mathbf{x})$ is a *field* that can be queried at any point $\mathbf{x}$ in space. These fields are concatenated to form $\boldsymbol{\alpha}(\mathbf{x}) = \{\alpha_a(\mathbf{x})\}$.

We then provide all this information to generate an *attribute masking field* via a network $\mathcal{M}(\cdot; \boldsymbol{\theta})$. This field determines which attribute *attends* to which position in space $\mathbf{x}$:

$$\mathbf{m}_0(\mathbf{x}) \oplus \mathbf{m}(\mathbf{x}) = \mathcal{M}(\mathcal{K}(\mathbf{x}), \boldsymbol{\beta}(\mathbf{x}), \boldsymbol{\alpha}(\mathbf{x}); \boldsymbol{\theta}), \tag{16}$$

$$\mathcal{M} : \mathbb{R}^3 \times \mathbb{R}^B \times \mathbb{R}^{A \times d} \rightarrow \mathbb{R}_+^{A+1}, \tag{17}$$

where $\oplus$ is a concatenation operator, $\mathbf{m}(\mathbf{x})$=$\{\mathbf{m}_a(\mathbf{x})\}$, and the additional mask $\mathbf{m}_0(\mathbf{x})$ denotes space that is not affected by *any* attribute. Note that because the mask location should be affected by both the particular attribute of interest (*e.g.*, the selected eye status) and the global appearance of the scene (*e.g.*, head movement), $\mathcal{M}$ takes both $\boldsymbol{\beta}(\mathbf{x})$ and $\boldsymbol{\alpha}(\mathbf{x})$ as input in addition to $\mathcal{K}(\mathbf{x})$. In addition, because the mask is modeling the attention related to attributes, collectively, these masks satisfy the partition of unity property:

$$\mathbf{m}_0(\mathbf{x}) + \Sigma_a[\mathbf{m}_a(\mathbf{x})] = 1 \quad \forall \mathbf{x} \in \mathbb{R}^3. \tag{18}$$

Finally, in a similar spirit to Eq. (5), all of this information is processed by a neural network that produces the desired radiance and density fields used in volume rendering:

$$\left.\begin{matrix}\mathbf{c}(\mathbf{x}) \\ \sigma(\mathbf{x})\end{matrix}\right\} = \mathcal{R}(\mathcal{K}(\mathbf{x}), \underbrace{\mathbf{m}(\mathbf{x}) \odot \boldsymbol{\alpha}(\mathbf{x})}_{\text{attribute controls}}, \underbrace{\mathbf{m}_0(\mathbf{x}) \cdot \boldsymbol{\beta}(\mathbf{x})}_{\text{everything else}}; \boldsymbol{\theta}). \tag{19}$$

In particular, note that $\mathbf{m}(\mathbf{x})=0$ implies $\mathbf{m}_0(\mathbf{x})=1$, hence our solution has the capability of reverting to classical HyperNeRF Eq. (5), where all change in the scene is globally encoded in $\boldsymbol{\beta}(\mathbf{x})$. Finally, these fields can be used to render the mask in image space, following a process analogous to volume rendering of radiance:

$$\mathbf{M}(\mathbf{r}; \boldsymbol{\theta}) = \int_{t_n}^{t_f} T(t) \cdot \sigma(\mathbf{r}(t)) \cdot [\mathbf{m}_0(\mathbf{r}(t)) \oplus \mathbf{m}(\mathbf{r}(t))]\, dt. \tag{20}$$

We depict our inference flow in Fig. 3 (b).

2.3.4 Implementation details

We implement our method for NeRF based on the JAX [13] implementation of HyperNeRF [120]. We use both the scheduled windowed positional encoding and weight initialization of [118], as well as the coarse-to-fine training strategy [120].

Besides the newly added networks, we follow the same architecture as HyperNeRF. For the attribute network $\mathcal{A}$ we use a six-layer multi-layer perceptron (MLP) with 32 neurons at each layer, with a skip connection at the fifth layer, following [118, 120]. For the lifting network $\mathcal{H}_a$, we use the same architecture as $\mathcal{H}$, except for the input and output dimension sizes. For the masking network $\mathcal{M}$ we use a four-layer MLP with 128 neurons at each layer, followed by an additional 64 neuron layer with a skip connection. The network $\mathcal{R}$ also shares the same architecture as HyperNeRF, but with a different input dimension size to accommodate for the changes our method introduces.

2D implementation To show that our idea is not limited to neural radiance fields, we also test a 2D version of our framework that can be used to directly represent images, without going through volume rendering. We use the same architecture and training procedure as in the NeRF case, with the exception that we do not predict the density σ, and we also do not have the notion of depth—each ray is directly the pixel. We center crop each video and resize each frame to be 128×128.

Hyperparameters We train all our NeRF models with 480×270 images and with 128 samples per ray. We train for 250k iterations with a batch size of 512 rays. During training, we maintain that 10% of rays are sampled from annotated images. We set $\ell_{\text{attr}} = 10^{-1}$, $\ell_{\text{mask}} = 10^{-2}$ and $\ell_{\text{enc}} = 10^{-4}$. For the number of hyper dimensions we set $d = 8$. For the 2D implementation

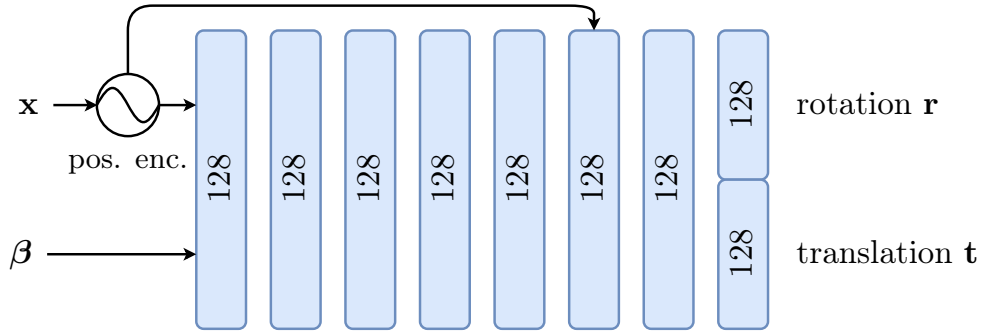


Figure 4. The canonicalization network takes positionally encoded raw coordinates $\mathbf{x}$ and learnable per-image latent code $\boldsymbol{\beta}$ and outputs rotation $\mathbf{r}$ expressed as a quaternion and translation $\mathbf{t}$. We rigidly transform each point $\mathbf{x}$ with an affine transform using both output. We use windowed positional encoding [118] for $\mathbf{x}$ with 8 components, linearly increasing contribution of components throughout 80k steps. We initialize the last layer to small values so the network can learn a base structure of the data.

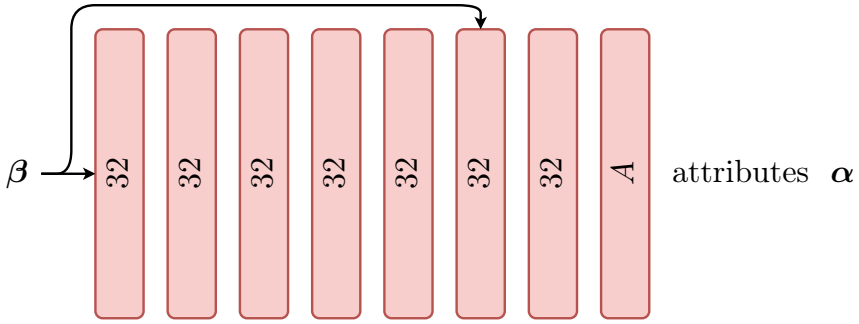


Figure 5. The attribute map $\mathcal{A}$ takes a per-image learnable latent code $\boldsymbol{\beta}$ and outputs A attributes $\boldsymbol{\alpha}$.

experiments, we sample 64 random images from the scene and further subsample 1024 pixels from each of them. For all experiments we use Adam [75] with learning rate 10^{-4} exponentially decaying to 10^{-5} in 250k iterations. Training a single model takes around 12 hours on an NVIDIA V100 GPU.

Architecture details We present architecture of: canonicalizer $\mathcal{K}$ in Fig. 4, attribute map $\mathcal{A}$ in Fig. 5, hypermap $\mathcal{H}$ in Fig. 6, per-attribute hypermap in Fig. 7, mask prediction network in Fig. 8 and the rendering network in Fig. 9. Each network contains only fully connected layers. Hidden layers use ReLU activation function. Colors of figures correspond to colors of blocks in Fig. 3b.

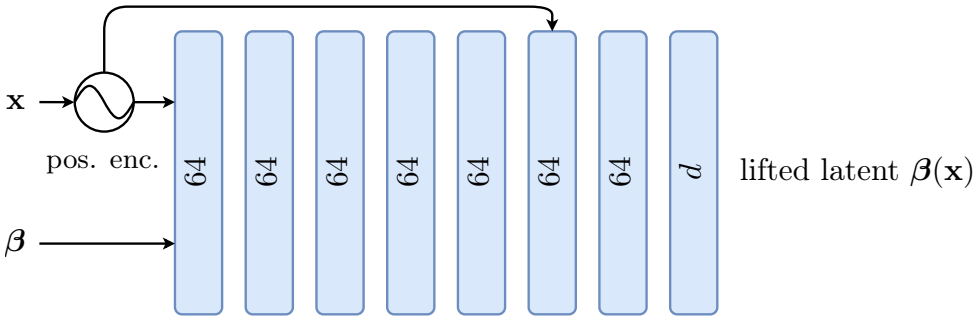


Figure 6. The network predicting lifted latent code $\boldsymbol{\beta}$, takes per-image $\boldsymbol{\beta}$ as an input, positionally encoded raw points $\boldsymbol{\beta}$ and outputs a lifted code of size d. We use only one sine component to encode $\mathbf{x}$.

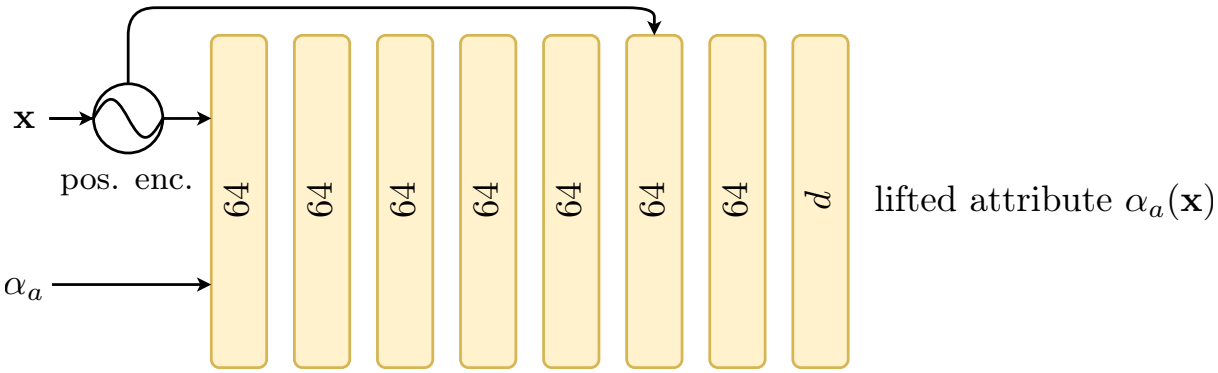


Figure 7. Per-attributes hypermaps take an attribute together with encoded $\mathbf{x}$ coordinates and output lifted $\alpha_a(\mathbf{x})$ ambient code of size d. We encode $\mathbf{x}$ with only single component.

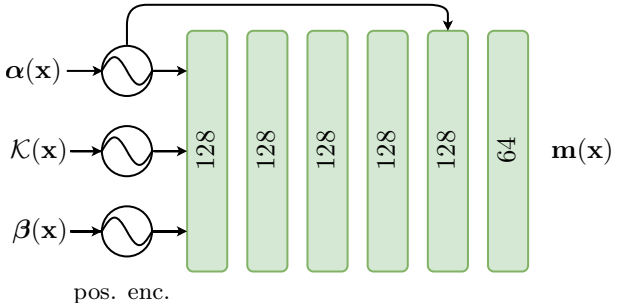


Figure 8. Masking network $\mathcal{M}$ take lifted attributes $\boldsymbol{\alpha}(\mathbf{x})$, lifted latent code $\boldsymbol{\beta}(\mathbf{x})$ and canonicalized points $\mathcal{K}(\mathbf{x})$. We transform $\boldsymbol{\alpha}(\mathbf{x})$ and $\boldsymbol{\beta}(\mathbf{x})$ through a windowed positional encoding where we start at 1k-th step linearly increasing a single sine component for the next 10k steps. Points $\mathcal{K}(\mathbf{x})$ are encoded with 8 components. The output is activated with a sigmoid function. We realize $\mathbf{m}_0(\mathbf{x})$ as $\mathbf{m}(\mathbf{x})_0 = 1 - \sum_{a \in A} \mathbf{m}_a(\mathbf{x})$, and clip the output to ensure the values range to be in $[0, 1]$. Note that while the network shares similarities with the radiance field prediction part $\mathcal{R}$, it is not conditioned on view directions and appearance codes.

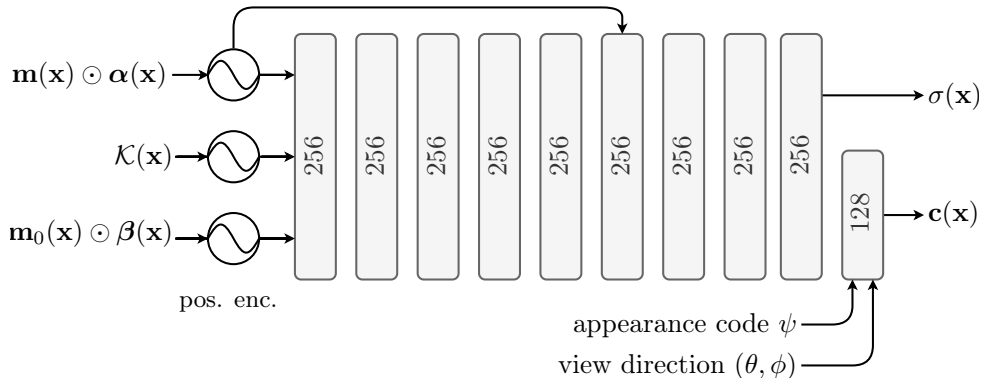


Figure 9. The radiance field prediction network predicts RGB colors $\mathbf{c}(\mathbf{x})$ and density values $\sigma(\mathbf{x})$ from canonicalized points. We encode points $\mathbf{x}$ with 8 sine components and linearly increase contribution of a single component in $\boldsymbol{\alpha}(\mathbf{x})$ and $\boldsymbol{\beta}(\mathbf{x})$ from 1k to 11k step. Per-point predicted predicted attributes $\boldsymbol{\alpha}(\mathbf{x})$ and lifted latent code $\boldsymbol{\beta}(\mathbf{x})$ are masked by a mask predicted from the masking network depicted in Fig. 8. The final linear layer takes additional per-image learnable appearance code ψ to account for any visual variations that cannot be explained by the rest of the framework (*e.g.* changes in lighting). The code can discarded during evaluation. The same layer is additionally conditioned on the positionally encoded view directions. We activate the color output with a sigmoid function.

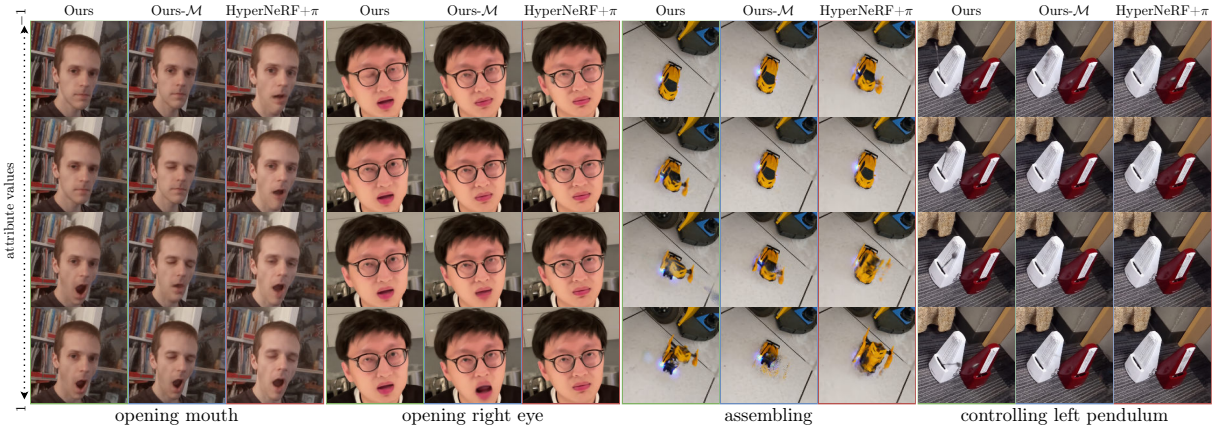


Figure 10. Novel view and novel attribute synthesis on real data – We synthesize scenes from a novel view and with a novel attribute combination, not seen during training. A naive extension of HyperNeRF, HyperNeRF+π fails to disentangle attributes and results in a modification of the scene irrespectively of attribute meaning *e.g.*, opening mouth results in closing eyes at the same time. Ours-$\mathcal{M}$ improves the results, but does not disentangles the attribute space, as successfully done by our complete method. The differences between these methods can even lead to complete failure cases, as shown in the metronome and the toy car case.

2.4 Results

2.4.1 Datasets and baselines

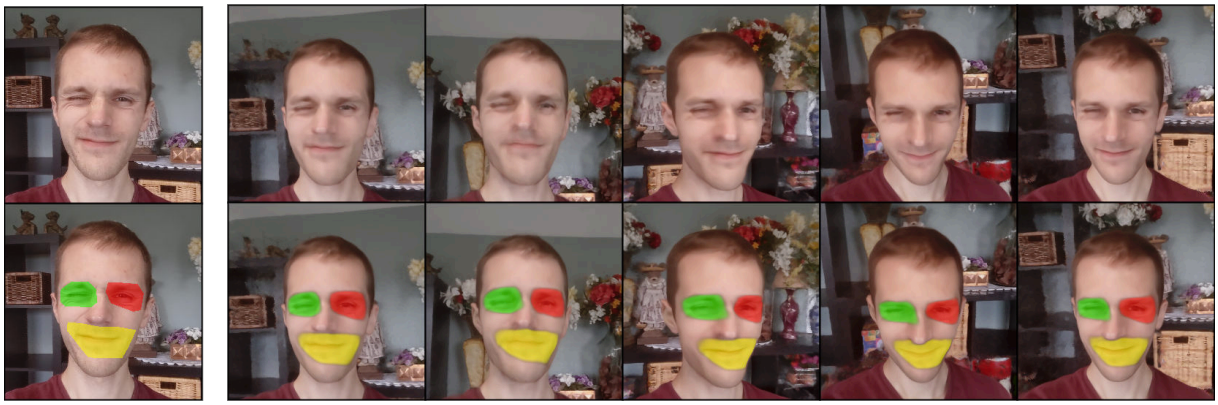
We evaluate our method on two datasets: real video sequences captured with a smartphone (*real dataset*) and synthetically rendered sequences (*synthetic dataset*). Here we introduce those datasets and the baselines for our approach.

Real dataset Each of the seven real sequences is 1 minute long and was captured either with a Google Pixel 3a or an Apple iPhone 13 Pro. Four of them consists of people performing different facial expressions including smiling, frowning, closing or opening eyes, and opening mouth. For the other three, we captured a toy car changing its shape (*a.k.a.* Transformer), a single metronome, and two metronomes beating with different rates. For one of the four videos depicting people, to use it for the 2D implementation case, we captured it with a static camera that shows a frontal view of the person. All other sequences feature camera motions showing front and sides of the object in the center of the scene. For videos with human subjects, the subjects signed a participant consent form, which was approved by a research ethics board. We informed the participants that their data will be altered with our method.

We extract frames at 15 FPS which gives approximately 900 frames per capture. Because novel attribute synthesis via user control on real scenes does not have a ground truth view—we aim to create scenes with unseen attribute combinations—the benefit of our method is best seen qualitatively. Nonetheless, to quantitatively evaluate the rendering quality, we interpolate between two frames and evaluate its quality. In more detail, to minimize the chance of the dynamic nature of the scene interfering with this assessment, we use every other frame as a test frame for the interpolation task.

For all human videos, we define three attributes—one for the status of each of the two eyes, and one for the mouth. We annotate only six frames per video in this case, specifically the frames that contain the extremes of each attribute (*e.g.*, left eye fully open). For the toy car, we set the shape of the toy car to be an attribute, and annotate two extremes from two different view points—when the toy is in robot-mode and when it is in car-mode from its left and right side. For the metronomes, we consider the state of the pendulum to be the attribute and annotate the two frames with the two extremes for the single metronome case, and seven frames for the two metronome case as the pendulums of the two metronomes are often close to each other and required special annotations for these close-up cases; see Fig. 10.

Synthetic dataset Since the lack of ground-truth data renders measuring the quality of novel attribute synthesis infeasible in practice, we leverage Kubric software [47] to generate synthetic dataset, where we know exactly the state of each object in the scene. We create a simple scene where three 3D objects, the teapot [163], the Stanford bunny [15], and Suzanne [156], are placed within the scene and are rendered with varying surface colors, which are our attributes;



(a) annotation **(b)** unannotated views

Figure 11. Annotation example – We provide only a rough annotation for each attribute, which is enough for the method to discover the mask for each attribute across all views automatically. Bottom row shows masks overlaid on the image.

see Fig. 12. We generate 900 frames for training and 900 frames for testing. To ensure that the attribute combination during training is not seen in the test scene, we set the attributes to be synchronized for the training split, and desynchronized for the test split. We further render the test split from different camera positions than the training split to account for novel views. We randomly sample 5% of the frames with a given attribute for each object to be set as the ground-truth attribute. During validation, we use attribute values directly to predict the image.

Baselines To evaluate the reconstruction quality of our method, CoNeRF, we compare it with four different baselines: ① standard NeRF [104]; ② NeRF+Latent, a simple extension to NeRF where we concatenate each coordinate $\mathbf{x}$ with a learnable latent code $\boldsymbol{\beta}$ to support appearance changes of the scene; ③ Nerfies [118]; and ④ HyperNeRF[2] [120]. Additionally, as existing methods do not support attribute-based control with a few-shot supervision, we create another baseline ⑤ by extending HyperNeRF with a simple linear regressor π that regresses $\boldsymbol{\beta}_c$ given $\boldsymbol{\alpha}_c$. We call this baseline HyperNeRF+π. To further show the importance of masking, we also compare our approach against a stripped-down version of our method, Ours-$\mathcal{M}$, where we disable the part of our pipeline responsible for masking. All baselines that utilize annotations were trained with the same sparse labels as our method.

2.4.2 Comparison with the baselines

Qualitative highlights We first show qualitative examples of novel attribute and view synthesis on the real dataset in Fig. 10. Our method allows for controlling the selected attribute without changing other aspects of the image—our control is disentangled. This disentanglement allows our method to generate images with attribute combinations that were not seen at training

[2]We use the version with dynamic plane slicing as it consistently outperforms the axis-aligned strategy; see [120] for more details.

Method	PSNR↑	MS-SSIM↑	LPIPS↓
HyperNeRF+π	25.963	0.854	0.158
Ours-$\mathcal{M}$	27.868	0.898	0.155
Ours	**32.394**	**0.972**	**0.139**

Table 2. Novel view and novel attributes results – We report average PSNR, MS-SSIM, and LPIPS values for novel view and novel attribute synthesis on synthetic data. Our method gives the best results.

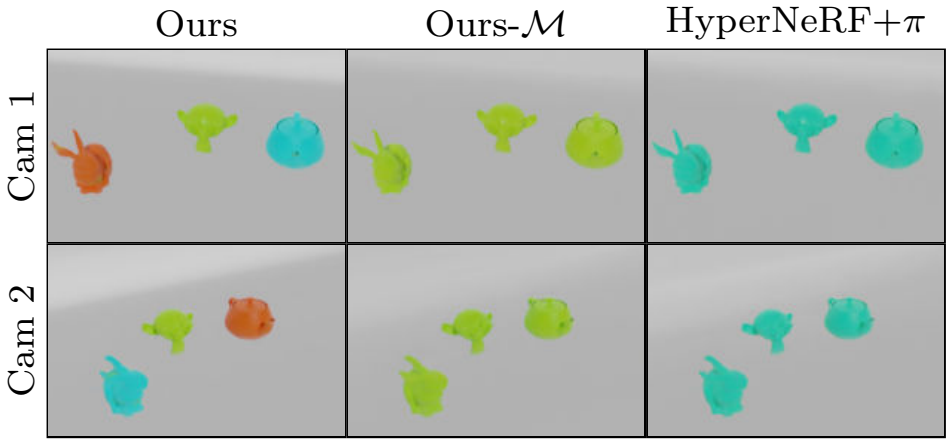


Figure 12. Novel view and novel attribute synthesis on synthetic data – We show examples of novel view and novel attribute synthesis on synthetic data. The scene is composed of three objects, where the color of each object is their attribute. Our method provides control over the color of each object independently, whereas both HyperNeRF+π and Ours-$\mathcal{M}$ fail to deliver controllability and results in all three objects having the same attribute in the rendered scene.

time. On the contrary, as there is no incentive for the learned embeddings of HyperNeRF to be disentangled, the simple regression strategy of HyperNeRF+π results in entangled control, where when one tries to close/open the mouth it ends up affecting the eyes. The same phenomenon happens also for Ours-$\mathcal{M}$. Moreover, due to the complexity of motions in the scene, HyperNeRF+π fails completely to render novel views of the toy car, whereas our method, with only four annotated frames, successfully provides both controllability and high-quality renderings.

Note that in all of these sequences, we provide highly sparse annotations and yet our method learns how each attribute should influence the appearance of the scene. In Fig. 11, we show an example annotation and how the method finds the mask for unannotated views.

Quantitative results on synthetic dataset To complete the qualitative evaluation of our method, we provide results using synthetic dataset with available ground truth. We measure Peak Signal-to-Noise Ratio (PSNR), Multi-scale Structural Similarity (MS-SSIM) [178], and Learned Perceptual Image Patch Similarity (LPIPS) [214] and report them in Tab. 2. With only

Method	PSNR ↑	MS-SSIM ↑	LPIPS ↓
NeRF	28.795	0.951	0.210
NeRF + Latent	32.653	0.981	0.182
NeRFies	32.274	0.981	0.180
HyperNeRF	32.520	0.981	0.169
Ours-$\mathcal{M}$	32.061	0.979	0.167
Ours	32.342	0.981	0.168

Table 3. Quantitative results (interpolation) – We report results in terms of PSNR, MS-SSIM, and LPIPS for the interpolation task. These results are obtained for interpolated view synthesis only, not for novel attribute rendering. Our method provides similar performance in terms of rendering quality, but with controllability.

5% of the annotations, our method provides the best novel-view and novel-attribute synthesis results, as reconfirmed by the qualitative examples in Tab. 2. As shown, neither HyperNeRF+π nor Ours-$\mathcal{M}$ is able to provide good results in this case, as without disentangled control of each attribute, the novel attribute and view settings of each test frame cannot be synthesized properly.

Interpolation task To further verify that our rendering quality does not degrade with the introduction of controllability, we evaluate our method on a frame interpolation task without any attribute control. Unsurprisingly, as shown in Tab. 3, all methods that support dynamic scenes work similarly, including ours for interpolation. Note that for the interpolation task, we interpolate every other frame, in order to minimize the chance of attributes affecting the evaluation. Here, we are purely interested in the rendering quality from a novel view.

2.4.3 Direct 2D rendering

To verify how our approach generalizes beyond NeRF models and volume rendering, we apply our method to videos taken from a single view point, creating a 2D rendering task. We show in Fig. 13 a proof-of-concept for employing our approach outside of NeRF applications to allow controllable neural generative models.

2.4.4 Ablation study

Loss functions In Tab. 4, we show how each loss term affects the network's performance, contributing to performance improvements. When rendering novel views with novel attributes, the full formulation is a must, as without all loss terms the performance drops significantly—for example, results without ℓ_{mask} is similar to Ours-$\mathcal{M}$ results in Tab. 2 and Fig. 12. In the case of the interpolation task, the additional loss functions for controllability have no significant effect on the rendering quality. In other words, our controllability losses **do not interfere** with the rendering quality, other than imbuing the framework with controllability.

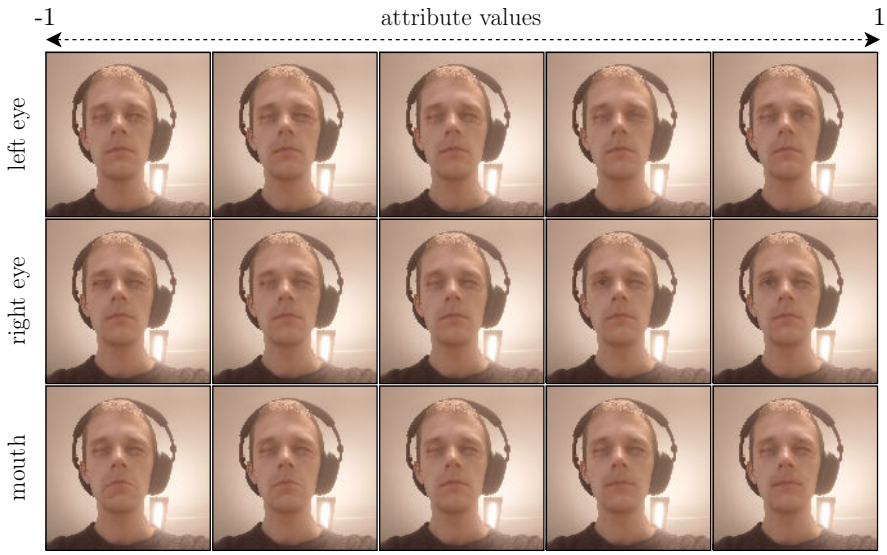


Figure 13. 2D image generation example – Our framework also generalizes to direct generation of 2D images. Here we show novel attribute synthesis for a webcam video of a person making expressions. Each individual part of the scene is correctly controlled according to the attribute values.

	Real (interpolation)			Synthetic (novel view & attr.)		
Model	PSNR ↑	MS-SSIM ↑	LPIPS ↓	PSNR ↑	MS-SSIM ↑	LPIPS ↓
Base (ℓ_{recon})	32.457	0.981	0.168	24.407	0.718	0.173
$+\ell_{\text{enc}}$	32.478	0.982	0.167	27.018	0.871	0.164
$+\ell_{\text{enc}} + \ell_{\text{attr}}$	32.254	0.981	0.167	27.322	0.873	0.147
$+\ell_{\text{enc}} + \ell_{\text{attr}} + \ell_{\text{mask}}$	32.342	0.981	0.168	**32.394**	**0.972**	**0.139**

Table 4. Effect of loss functions – We report the rendering quality of our method as we procedurally introduce the loss terms. For controlled rendering with novel views and attributes (synthetic data), each loss term adds to the rendering quality, with the ℓ_{mask} being critical. For the novel view rendering on real data, addition of loss functions for controllability do not have a significant effect on the rendering quality—they do no harm.

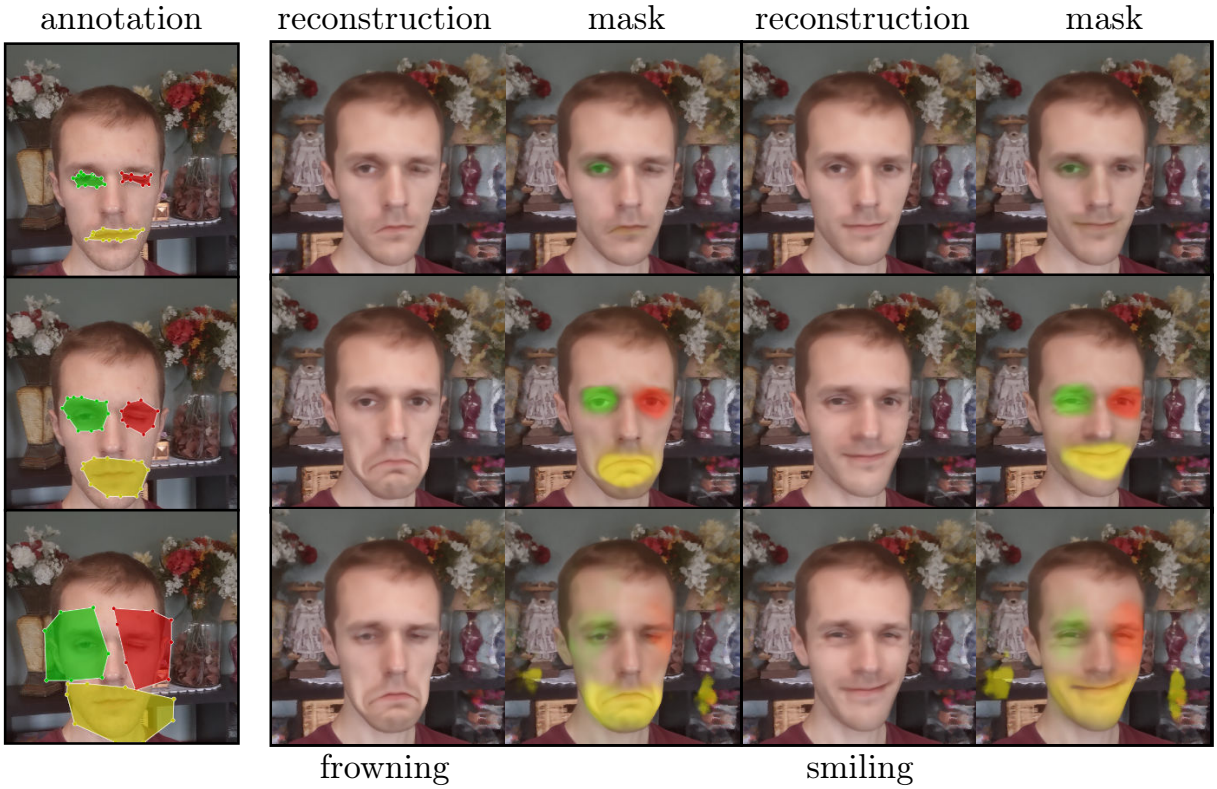
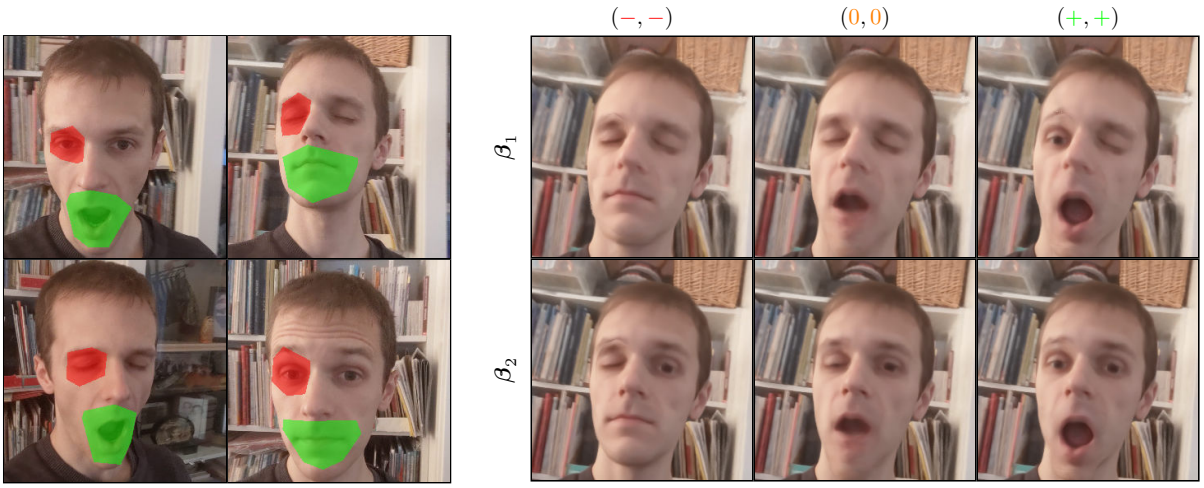


Figure 14. Effect of annotation quality – Our method is moderately robust to the quality of annotations. We visualize the results for two expressions: frowning and smiling, while keeping both eyes in a neutral position. Even with wildly varying annotations as shown, the reconstructions are reasonably controlled, with the exception of the top row, where we show a case where the annotations is too restrictive, resulting in the annotation being ignored for one eye. We show in bottom row also an interesting case, where the mask is large enough to start capturing the correlation among mouth expressions and the eye.



(a) annotated samples **(b)** renderings

Figure 15. Example with unannotated attributes – We show an example of how our method performs when a part of the image changes appearance, but is not annotated. With the annotations in (a), we synthesize the scene with novel view and attributes in (b), where the two rows are with different $\boldsymbol{\beta}$ configurations. We denote the attribute configuration on the top of each column in (b). As shown, the change that is not annotated is simply encoded in the per-image encoding $\boldsymbol{\beta}$.

Quality of few shot supervision We test how sensitive our method is against the quality of annotation supervision. In Fig. 14 we demonstrate how each annotation leads to the final rendering quality. Our framework is robust to a moderate degree to the inaccuracies in the annotations. However, when they are too restrictive, the mask may collapse, as shown on the top row. Too large of a mask could also lead to moderate entanglement of attributes, as shown in the bottom row. Still, in all cases, our method provides a reasonable control over what is annotated.

Unannotated attributes A natural question to ask is then what happens with the unannotated changes that may exist in the scene. In Fig. 15 we show how the method performs when annotating only parts of the appearance change within the scene. The unannotated changes of the scene get encoded as $\boldsymbol{\beta}$, as in the case of HyperNeRF [120].

Failure Cases We identify two modes of failure cases in our approach and present them in Fig. 16. In some cases with particular mask annotations, our model can struggle with controlling elements that occupy small space in the image. The problem is especially visible for controlling pendulum movement or opening and closing eyes. In the former, pendulum disappears and reappears in different places. In the latter, the control of eyes is periodic and there are two distant values in $[-1, 1]$ that produce opening eyes. While with careful annotations we noticed that the problem is mostly preventable, this problem may occur in practice.

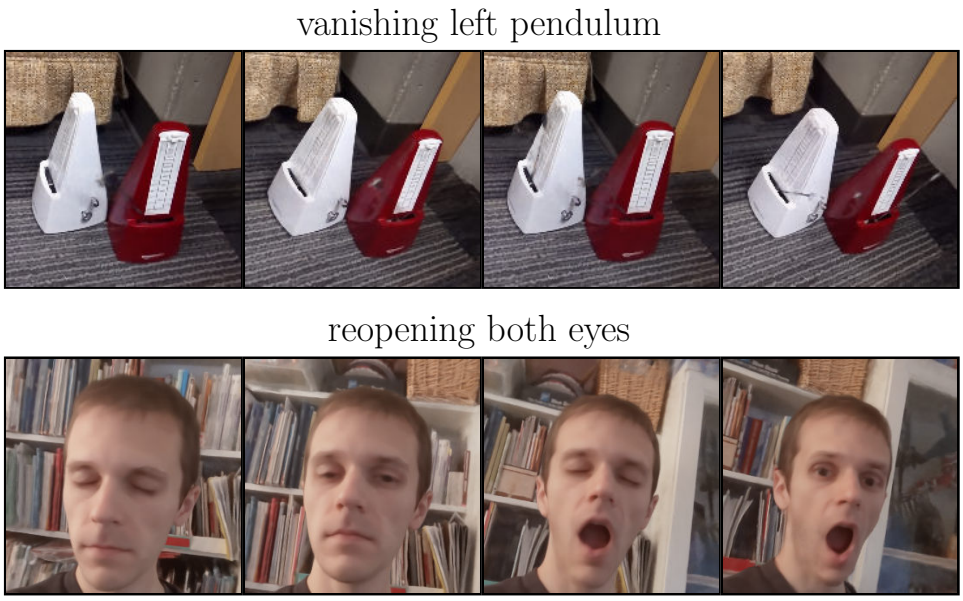


Figure 16. Failure cases – Our model may learn spurious interpolations for controlled elements that occupy little space in the image and with insufficient/careless annotations. For the metronome, due to the fast motion of the pendulum and its specularity, without careful annotation our method may simply learn its motion blur or sometimes even completely ignore the pendulum. In the face example, this may result in the eye blinking multiple times while interpolating between the attribute values of -1 and 1. Both cases are preventable with more careful annotations and by annotating more frames.

2.5 Conclusions

We have introduced CoNeRF, an intuitive controllable NeRF model that can be trained with few-shot annotations in the form of attribute masks. The core contribution of our method is that we represent attributes as localized masks, which are then treated as latent variables within the framework. To do so we regress the attribute and their corresponding masks with neural networks. This leads to a few-shot learning setup, where the network learns to regress provided annotations, and if they are not provided for a given image, proper attributes and masked are discovered throughout training automatically. We have shown that our method allows users to easily annotate what to control and how, within a single video simply by annotating a few frames, which then allows rendering of the scene from novel views and with novel attributes, at high quality.

Limitations While our method delivers controllability to NeRF models, there is room for improvement. First, our disentanglement of attribute strictly relies on the locality assumption—if multiple attributes act on a single pixel, our method is likely to have entangled outcomes when rendering with different attributes. An interesting direction would therefore be to incorporate manifold disentanglement approaches [83, 216] to our method. Second, while very few, we still require sparse annotations. Unsupervised discovery of controllable attributes, for example as in

[77], in a scene remains yet to be explored. Lastly, we resort to user intuition on which frames should be annotated—we heuristically choose frames with extreme attributes (*e.g.*, mouth fully open). While this is a valid strategy, an interesting direction for future research would be to employ active learning techniques for this purpose [10, 134]

Chapter 3

BlendFields: Few-Shot Example-Driven Facial Modeling

Generating faithful visualizations of human faces requires capturing both coarse and fine-level details of the face geometry and appearance. Existing methods are either data-driven, requiring an extensive corpus of data not publicly accessible to the research community, or fail to capture fine details because they rely on geometric face models that cannot represent fine-grained details in texture with a mesh discretization and linear deformation designed to model only a coarse face geometry. This chapter introduces a method that bridges this gap by drawing inspiration from traditional computer graphics techniques. Unseen expressions are modeled by blending appearance from a sparse set of extreme poses. This blending is performed by measuring local volumetric changes in those expressions and locally reproducing their appearance whenever a similar expression is performed at test time. We show that our method generalizes to unseen expressions, adding fine-grained effects on top of smooth volumetric deformations of a face, and demonstrate how it generalizes beyond faces.

3.1 Introduction

Recent advances in neural rendering of 3D scenes [164] offer 3D reconstructions of unprecedented quality [104] with an ever-increasing degree of control [69, 90]. Human faces are of particular interest to the research community [4, 38, 40, 41] due to their application in creating realistic digital doubles [98, 164, 211, 220].

To render facial expressions not observed during training, current solutions [4, 38, 40, 41] rely on *parametric* face models [11]. These allow radiance fields [104] to be controlled by facial parameters estimated by off-the-shelf face trackers [84]. However, parametric models primarily capture smooth deformations and lead to digital doubles that lack realism because fine-grained and expression-dependent phenomena like wrinkles are not faithfully reproduced.

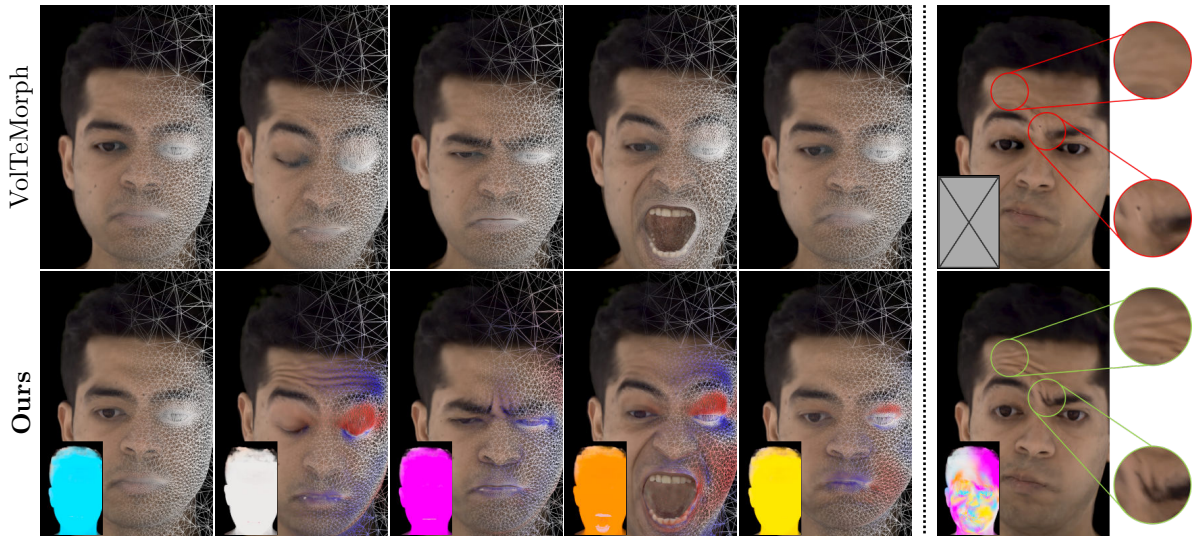


Figure 17. Teaser – Given five multi-view frames of different expressions, our approach generates a model capable of capturing the fine-grained details of a novel expression beyond the resolution of the underlying face model [41] (top right corner). This is achieved by *blending* the radiance fields computed for individual expressions, where the blending coefficients are modulated accordingly to *local* volumetric changes. These volumetric changes are measured as the difference in the tetrahedral volume of a mesh that deforms with the expression (■ increase, ■ decrease, and ■ no change in volume). Such an approach allows *BlendFields* to render sharp, expression-dependent details of the face without increasing the resolution of the mesh (bottom right corner).

Authentic Volumetric Avatars (AVA) [16] overcomes this issue by learning from a large multi-view dataset of synchronized and calibrated images captured under controlled lighting. Their dataset covers a series of dynamic facial expressions and multiple subjects. However, this dataset remains unavailable to the public and is expensive to reproduce. Additionally, training models from such a large amount of data requires significant compute resources. To democratize digital face avatars, more efficient solutions in terms of hardware, data, and compute are necessary.

We address the efficiency concerns by building on the recent works in Neural Radiance Fields [41, 195, 208]. In particular, we extend VolTeMorph [41] to render facial details learned from images of a sparse set of expressions. To achieve this, we draw inspiration from blend-shape correctives [80], which are often used in computer graphics as a data-driven way to correct potential mismatches between a simplified model and the complex phenomena it aims to represent. In our setting, this mismatch is caused by the low-frequency deformations that the tetrahedral mesh from VolTeMorph [41], designed for real-time performance, can capture, and the high-frequency nature of expression wrinkles.

We train multiple radiance fields, one for each of the K sparse expressions present in the input data, and blend them to correct the low-frequency estimate provided by VolTeMorph [41]; see Fig. 17. We call our method BlendFields since it resembles the way blend shapes are employed in 3DMMs [11]. To keep K small (*i.e.*, to maintain a few-shot regime), we perform local blending to exploit the known correlation between wrinkles and changes in local differential properties [60, 129]. Using the dynamic geometry of [41], local changes in differential properties can be easily extracted by analyzing the tetrahedral representation underlying the corrective blendfields of our model.

Contributions We outline the main qualitative differences between our approach and related works in Tab. 5, and our empirical evaluations confirm these advantages. In summary, we:

- extend VolTeMorph [41] to enable modeling of high-frequency information, such as expression wrinkles in a few-shot setting;
- introduce correctives [11] to neural field representations and activate them according to local deformations [129];
- make this topic more accessible with an alternative to techniques that are data and compute-intensive [16];
- show that our model generalizes beyond facial modeling, for example, in the modeling of wrinkles on a deformable object made of rubber.

3.2 Related Works

Neural Radiance Fields (NeRF) [104] is a method for generating 3D content from images taken with commodity cameras. It has prompted many follow-up works [7, 8, 58, 99, 103, 119, 132, 150, 160, 171, 188, 212] and a major change in the field for its photorealism. The main limitations

	NeRF [104]	NeRFies [119]	HyperNeRF [120]	NeRFace [38]	NHA [45]	AVA [16]	VolTeMorph [41]	**Ours**
Applicability beyond faces	✓	✓	✓	✗	✗	✗	✓	✓
Interpretable control	✗	✗	✗	✓	✓	✗	✓	✓
Data efficiency	✗	✓	✓	✗	✓	✗	✓	✓
Expression-dependent changes	✗	✗	✓	✓	✓	✓	✗	✓
Generalizability to unknown expressions	✗	✗	✗	✓	✓	✗	✓	✓

Table 5. Comparison – We compare several methods to our approach. Other methods fall short in data efficiency and applicability. For example, AVA [16] requires 3.1 million training images while VolTeMorph [41] cannot model expression-dependent wrinkles realistically.

of NeRF are its rendering speed, being constrained to static scenes, and lack of ways to control the scene. Rendering speed has been successfully addressed by multiple follow-up works [42, 54, 203]. Works solving the limitation to static scenes[4, 5, 62, 114, 175, 180, 187, 218] and adding explicit control [20, 69, 74, 153, 173, 197, 208] have limited resolution or require large amounts of training data because they rely on controllable coarse models of the scene (*e.g.*, 3DMM face model [11]) or a conditioning signal [120]. Methods built on an explicit model are more accessible because they require less training data but are limited by the model's resolution. Our technique finds a sweet spot between these two regimes by using a limited amount of data to learn details missing in the controlled model and combining them together. Our experiments focus on faces because high-quality data and 3DMM face models are publicly available, which are the key component for creating digital humans.

3.2.1 Radiance Fields

Volumetric representations [172] have grown in popularity because they can represent complex geometries like hair more accurately than mesh-based ones. Neural Radiance Fields (NeRFs) [104] model a radiance volume with a coordinate-based MLP learned from posed images. The MLP predicts density $\sigma(\mathbf{x})$ and color $\mathbf{c}(\mathbf{x}, \mathbf{v})$ for each point $\mathbf{x}$ in the volume and view direction $\mathbf{v}$ of a given camera. To supervise the radiance volume with the input images, each image pixel is associated with a ray $\mathbf{r}(t)$ cast from the camera center to the pixel, and samples along the ray are accumulated to determine the value of the image pixel $C(\mathbf{r})$:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{v}) dt, \tag{21}$$

where t_n and t_f are near and far planes, and

$$T(t) = \exp\left(- \int_{t_n}^{t} \sigma(\mathbf{r}(s)) ds \right), \tag{22}$$

is the transmittance function [157]. The weights of the MLP are optimized to minimize the mean squared reconstruction error between the target pixel and the output pixel. Several methods have shown that replacing the implicit functions approximated with an MLP for a function

discretized on an explicit voxel grid results in a significant rendering and training speed-up [42, 54, 88, 151, 203].

3.2.2 Animating Radiance Fields

Several works exist to animate the scene represented as a NeRF. D-NeRF uses an implicit deformation model that maps sample positions back to a canonical space [124], but it cannot generalize to unseen deformations. Several works [38, 119, 120, 166] additionally account for changes in the observed scenes with a per-image latent code to model changes in color as well as shape, but it is unclear how to generalize the latents when animating a sequence without input images. Similarly, works focusing on faces [4, 38, 40, 221] use parameters of a face model to condition NeRF's MLP, or learn a latent space of images and geometry [16, 92, 93, 98, 101, 174] that does not extrapolate beyond expressions seen during training.

In contrast to these approaches, we focus on using as little temporal training data as possible (*i.e.* five frames) while ensuring generalization. For this reason, we build our method on top of VolTeMorph [41], that uses a parametric model of the face to track the deformation of points in a volume around the face and builds a radiance field controlled by the parameters of a 3DMM. After training, the user can render an image for any expression of the face model. However, the approach cannot generate expression-dependent high-frequency details; see Fig. 17.

Similarly, NeRF-Editing [208] and NeRF Cages [195] propose to use tetrahedral meshes to deform a single-frame NeRF reconstruction. The resolution of the rendered scenes in these methods is limited by the resolution of the tetrahedral cage, which is constrained to a few thousand elements.

3.2.3 Tetrahedral Cages

To apply parametric mesh models, it is necessary to extend them to the volume to support the volumetric representation of NeRF. Tetrahedral cages are a common choice for their simplicity and ubiquity in computer graphics [41, 195, 197]. For example, VolTeMorph uses dense landmarks [183] to fit a parametric face model whose blendshapes have been extended to a tetrahedral cage with finite elements method [22]. These cages can be quickly deformed and raytraced [105] using parallel computation on GPUs [24] while driving the volume into the target pose and allowing early ray termination for fast rendering. We further leverage the tetrahedral cage and use its differential properties [60], such as a local volume change, to model high-frequency details. For example, a change from one expression to another changes the volume of tetrahedra in regions where wrinkle formation takes place while it remains unchanged in flat areas. We can use this change in volume to select which of the trained NeRF expressions should be used for each tetrahedron to render high-frequency details.

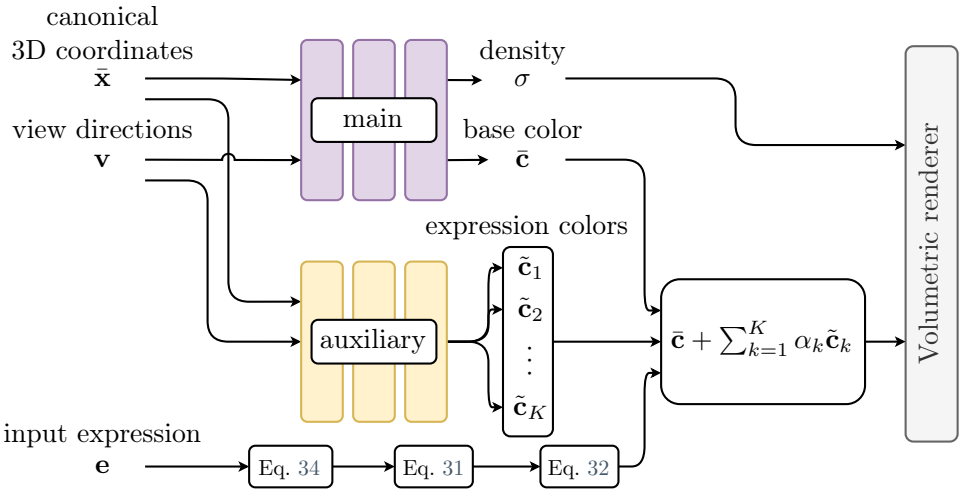


Figure 18. BlendFields – We implement our approach as a volumetric model, where the *appearance* (*i.e.* radiance) is the sum of the main appearance corrected by blending a small set of K expression-specific appearances. These appearances are learnt from extreme expressions, and then blended at test-time according to blend weights computed as a function of the input expression $\mathbf{e}$.

3.2.4 Concurrent Works

Gao *et al.* [40] and Xu *et al.* [196] also use an interpolation between known expressions to combine multiple neural radiance fields trained for those expressions. However, their approach interpolates between grids of latent vectors [109] globally. The interpolation weights are taken from blendshape coefficients.

Zielonka *et al.* [223] use a parametric head model to canonicalize 3D points similarly to our ends. However, instead of building a tetrahedral cage around the head, they smoothly assign each face triangle to 3D points. Then they canonicalize points using transformations that each of the assigned triangles undergoes for a given expression. They concatenate 3D points with the expression code from FLAME [84] to model expression-dependent effects.

3.3 Method

We introduce a volumetric model that can be driven by input expressions and visualize it in in Fig. 18. We start this section by explaining our model and how we train and drive it with novel expressions utilizing parametric face models (Sec. 3.3.1). We then discuss how to compute measures of volume expansion and compression in the tetrahedra to combine volumetric models of different expressions (Sec. 3.3.2) and how we remove artifacts in out-of-distribution settings (Sec. 3.3.3). We conclude this section with implementation details (Sec. 3.3.4).

3.3.1 Our model

Given a neutral expression $\bar{\mathbf{e}}$, and a collection of posed images $\{C_c\}$ of this expression from multiple views, VolTeMorph [41] employs a map $\mathcal{T}$ to fetch the density and radiance[1] for a new expression $\mathbf{e}$ from the *canonical* frame defined by expression $\bar{\mathbf{e}}$:

$$\mathbf{c}(\mathbf{x};\mathbf{e}) = \bar{\mathbf{c}}(\bar{\mathbf{x}}), \quad \bar{\mathbf{x}} = \mathcal{T}(\mathbf{x};\mathbf{e}\rightarrow\bar{\mathbf{e}}) \tag{23}$$

$$\sigma(\mathbf{x};\mathbf{e}) = \bar{\sigma}(\bar{\mathbf{x}}), \quad \bar{\mathbf{x}} = \mathcal{T}(\mathbf{x};\mathbf{e}\rightarrow\bar{\mathbf{e}}) \tag{24}$$

$$\ell_{\text{rgb}} = \mathbb{E}_{C\sim\{C_c\}}\, \mathbb{E}_{\mathbf{r}\sim C}\, \ell^{\mathbf{r}}_{\text{rgb}} \tag{25}$$

$$\ell^{\mathbf{r}}_{\text{rgb}} = \|C(\mathbf{r};\mathbf{e}) - C(\mathbf{r})\|_2^2, \tag{26}$$

where $C(\mathbf{r};\mathbf{e})$ is a pixel color produced by our model conditioned on the input expression $\mathbf{e}$, $C(\mathbf{r})$ is the ground-truth pixel color, and the mapping $\mathcal{T}$ is computed from smooth deformations of a tetrahedral mesh to render unseen expressions $\mathbf{e}$. We use expression vectors $\mathbf{e}$ from parametric face models, such as FLAME [84, 182]. However, as neither density nor radiance change with $\mathbf{e}$, changes in appearance are limited to the low-frequency deformations that $\mathcal{T}$ can express. For example, this model cannot capture high-frequency dynamic features like expression wrinkles. We overcome this limitation by conditioning radiance on expression. For this purpose, we assume radiance to be the sum of a template radiance (*i.e.* rest pose appearance of a subject) and K residual radiances (*i.e.* details belonging to corresponding facial expressions):

$$\mathbf{c}(\mathbf{x};\mathbf{e}) = \bar{\mathbf{c}}(\mathbf{x}) + \sum_{k=k}^{K} \alpha_k(\mathbf{x};\mathbf{e}) \cdot \tilde{\mathbf{c}}_k(\mathbf{x}), \tag{27}$$

We call our model *blend fields*, as it resembles the way in which blending is employed in 3D morphable models [11] or in wrinkle maps [116]. Note that we assume that pose-dependent geometry can be effectively modeled as a convex combination of colors $[\tilde{\mathbf{c}}(\mathbf{x})]_{k=1}^{K}$, since we employ the same density fields as in Eq. (24). In what follows, for convenience, we denote the vector field of blending coefficients as $\boldsymbol{\alpha}(\mathbf{x})=[\alpha_k(\mathbf{x})]_{k=1}^{K}$.

Training the model We train our model by assuming that we have access to a small set of K images $\{\mathbf{C}_k\}$ (example in Fig. 19), each corresponding to an "extreme" expression $\{\mathbf{e}_k\}$, and minimize the loss:

$$\ell_{\text{rgb}} = \mathbb{E}_k\, \mathbb{E}_{\mathbf{r}}\, \|C_K(\mathbf{r};\mathbf{e}_k) - C_k(\mathbf{r})\|_2^2 \tag{28}$$

$$\text{where} \quad \forall\mathbf{x},\ \boldsymbol{\alpha}(\mathbf{x}) = \mathbb{1}_k, \tag{29}$$

where $\mathbb{1}_k$ is the indicator vector, which has value one at the k-th position and zeroes elsewhere, and C_K represents the output of integrating the radiances in Eq. (27) along a ray.

[1] We omit view-dependent effects to simplify notation but include them in our implementation.

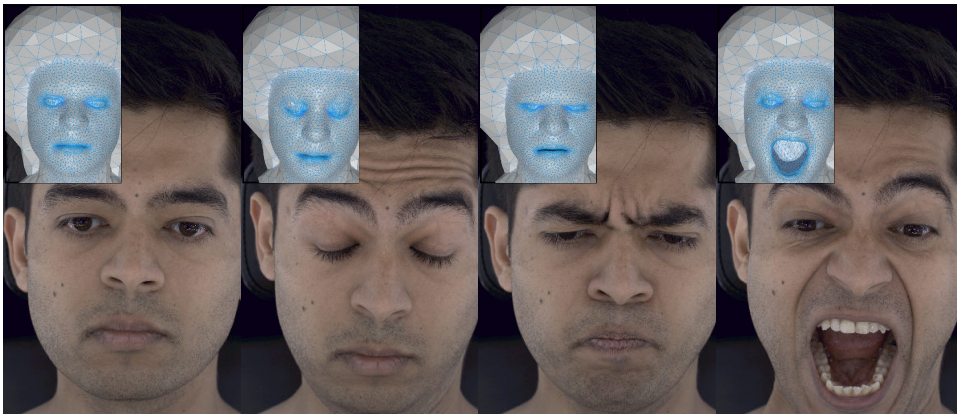


Figure 19. Data – We represent the data as a multi-view, multi-expression images. For each of these images, we obtain parameters of a parametric model, such as FLAME [84] to get: an expression vector $\mathbf{e}$ and a tetrahedral mesh described by vertices $\mathbf{V}(\mathbf{e})$. We highlight that our approach works for any object if a rough mesh and its descriptor are already provided.

Driving the model To control our model given a novel expression $\mathbf{e}$, we need to map the input expression code to the corresponding blendfield $\boldsymbol{\alpha}(\mathbf{x})$. We parameterize the blend field as a vector field discretized on the vertices $\mathbf{V}(\mathbf{e})$ of our tetrahedral mesh, where the vertices deform according to the given expression. The field is discretized on vertices, but it can be queried within tetrahedra using linear FEM bases [106]. Our core intuition is that when the (local) geometry of the mesh matches the local geometry in one of the input expressions, the corresponding expression blend weight should be locally activated. More formally, let $\mathbf{v}\in\mathbf{V}$ be a vertex in the tetrahedra and $\mathcal{G}(\mathbf{v})$ a local measure of volume on the vertex described in Sec. 3.3.2, then

$$\mathcal{G}(\mathbf{v}(\mathbf{e}))\approx\mathcal{G}(\mathbf{v}(\mathbf{e}_k)) \implies \boldsymbol{\alpha}(\mathbf{v}(\mathbf{e})) \approx \mathbb{1}_k. \tag{30}$$

To achieve this we first define a *local* similarity measure:

$$[\Delta\mathcal{G}_k(\mathbf{v}(\mathbf{e}))] = [\|\mathcal{G}(\mathbf{v}(\mathbf{e}))-\mathcal{G}(\mathbf{v}(\mathbf{e}_k))\|_2^2] \in \mathbb{R}^K \tag{31}$$

and then gate it with softmax (with temperature $\tau{=}10^6$) to obtain vertex blend weights:

$$\boldsymbol{\alpha}(\mathbf{v}(\mathbf{e})) = \mathrm{softmax}_\tau\{\Delta\mathcal{G}_k(\mathbf{v}(\mathbf{e}))\} \in [0,1]^K \tag{32}$$

which realizes Eq. (30), as well as preserves the typically desirable characteristics of blend weights:

- *partition of unity*: $\forall\mathbf{x}\ \boldsymbol{\alpha}(\mathbf{x}) \in [0,1]^K$ and $\|\boldsymbol{\alpha}(\mathbf{x})\|_1{=}1$
- *activations sparsity*: minimizers of $\|\boldsymbol{\alpha}(\mathbf{x})\|_0$

where the former ensures any reconstructed result is a *convex* combination of input data, and the latter prevents destructive interference [59].

# expr.	Casual Expressions			Novel Pose Synthesis		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
K=1	27.5834	0.9028	0.0834	28.7589	0.9147	0.0806
K=2	27.6783	0.9026	0.0856	29.2859	0.9186	0.0803
K=3	27.9137	0.9054	0.0819	29.8551	0.9279	0.0728
K=4	27.8140	0.9055	0.0815	30.1543	0.9336	0.0701
K=5	28.0254	0.9110	0.0778	30.4721	0.9372	0.0688
K=6	28.0517	0.9091	0.0813	–	–	–
K=7	28.2004	0.9115	0.0823	–	–	–
K=8	28.2542	0.9124	0.0830	–	–	–

Table 6. Number of training expressions – We ablate over the number of training expressions. We evaluate the model on the captures from the Multiface dataset [185]. We run the model for each possible expression combination for a given K and average the results. The best results are colored in ■ and the second best in ■. Increasing the number of available training expressions consistently improves the results. However, using K=5 expressions saturates the quality and using K>5 brings diminishing improvements. We do not report "Novel Pose Synthesis" for K>5 as we use validation expressions and poses to train those models (refer to Sec. 3.4.1 for more details).

3.3.2 Local geometry descriptor

Let us consider a tetrahedron as the matrix formed by its vertices $\mathbf{T}=\{\mathbf{v}_i\} \in \mathbb{R}^{3\times4}$, and its edge matrix as $\mathbf{D} = [\mathbf{v}_3 - \mathbf{v}_0, \mathbf{v}_2 - \mathbf{v}_0, \mathbf{v}_1 - \mathbf{v}_0]$. Let us denote $\bar{\mathbf{D}}$ as the edge matrix in rest pose and $\mathbf{D}$ as one of the deformed tetrahedra (*i.e.*, due to expression). From classical FEM literature, we can then compute the change in volume of the tetrahedra from the determinant of its deformation gradient [60]:

$$\Delta\mathcal{V}(\mathbf{T}) = \det(\mathbf{D} \cdot \bar{\mathbf{D}}^{-1}) \tag{33}$$

We then build a local volumetric descriptor for a specific (deformed) vertex $\mathbf{v}(\mathbf{e})$ by concatenating the changes in volumes of neighboring (deformed) tetrahedra:

$$\mathcal{G}(\mathbf{v}(\mathbf{e})) = \bigoplus_{\mathbf{T}\in\mathcal{N}(\mathbf{v})} \Delta\mathcal{V}(\mathbf{T}(\mathbf{e})), \tag{34}$$

where $\bigoplus$ denotes concatenation and $\mathcal{N}(\mathbf{v})$ topological neighborhood of a vertex $\mathbf{v}$.

3.3.3 Blend-field smoothness

High-frequency spatial changes in blendfields can cause visual artifacts, see Fig. 20. We overcome this issue by applying a small amount of smoothing to the blendfield. Let us denote with $\mathbf{A}=\{\boldsymbol{\alpha}(\mathbf{v}_v)\}$ the matrix of blend fields defined on all mesh vertices, and with $\mathbf{L}$ the Laplace-Beltrami operator for the tetrahedral mesh induced by linear bases [60]. We exploit the fact

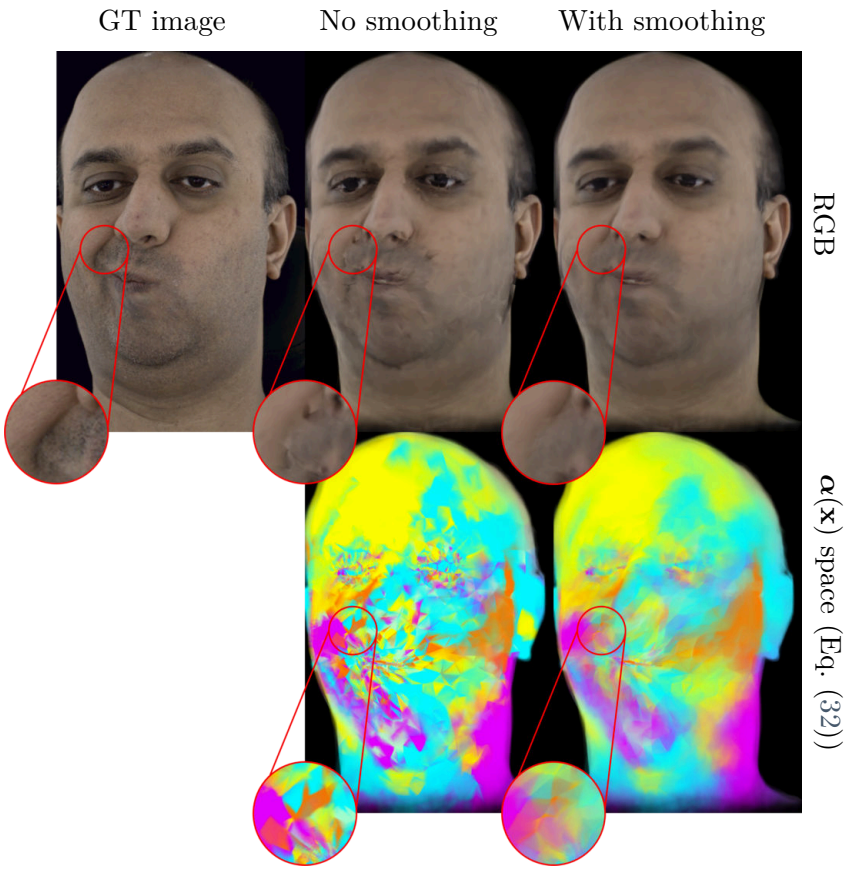


Figure 20. Laplacian smoothing – To combat artifacts stemming from calculating weights α across multiple expressions, which may assign different expressions to neighboring tetrahedra, we apply Laplacian smoothing [27]. As seen in the bottom row, smoothing gives a more consistent expression assignment.

that at test-time, the field is discretized on the mesh vertices, execute a diffusion process on the tetrahedral manifold, and, to avoid instability problems, implement it via backward Euler [27]:

$$\mathbf{A}^{\text{diff}} = (\mathbf{I} - \lambda_{\text{diff}}\mathbf{L})^{-1}\mathbf{A}^{n}. \tag{35}$$

3.3.4 Implementation details

We build on VolTeMorph [41] and use its volumetric 3DMM face model. However, the same methodology can be used with other tetrahedral cages built on top of 3DMM face models. The face model is created by extending the blendshapes of the parametric 3DMM face model [182] to a tetrahedral cage that defines the support in the neural radiance field. It has four bones controlling global rotation, the neck and the eyes with linear blend skinning, 224 expression blendshapes, and 256 identity blendshapes. Our face radiance fields are thus controlled and posed with the identity, expression, and pose parameters of the 3DMM face model [182], can be estimated by a real-time face tracking system like [183], and generalize convincingly to expressions representable by the face model.

Method	Real Data						Synthetic Data		
	Casual Expressions			Novel Pose Synthesis			Novel Pose Synthesis		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [104]	23.6465	0.7384	0.2209	25.6696	0.8127	0.1861	13.7210	0.6868	0.3113
Conditioned NeRF [104]	22.9106	0.7162	0.2029	24.7283	0.7927	0.1682	19.5971	0.8138	0.1545
NeRFies [119]	22.6571	0.7105	0.2271	24.8376	0.7990	0.1884	19.3042	0.8081	0.1591
HyperNeRF-AP [120]	22.6219	0.7087	0.2236	24.7119	0.7931	0.1848	19.3557	0.8132	0.1563
HyperNeRF-DS [120]	22.9299	0.7182	0.2241	24.9909	0.8007	0.1860	19.4637	0.8159	0.1526
VolTeMorph$_1$ [41]	24.9939	0.8358	0.1164	26.7526	0.8749	0.0954	26.7033	0.9500	0.0394
VolTeMorph$_{avg}$[41]	26.9209	0.8912	0.1105	28.6866	0.9176	0.0982	30.2107	0.9815	0.0387
BlendFields	27.5977	0.9056	0.0854	29.7372	0.9311	0.0782	32.7949	0.9882	0.0221

Table 7. Quantitative results – We compare BlendFields to other related approaches. We split the real data into two settings: one with casual expressions of subjects and the other with novel, static expressions. For the real data, we only compute metrics on the face region, which we separate using an off-the-shelf face segmentation network [182]. Please refer to the Tab. 9 for the results that include the background in the metrics as well. We average results across frames and subjects. VolTeMorph$_{avg}$ [41] is trained on all frames, while VolTeMorph$_1$ is trained on a single frame. HyperNeRF-AP/-DS follows the design principles from Park *et al.* [120]. The best results are colored in ■ and second best results in ■. BlendFields performs best in most of the datasets and metrics. Please note that HyperNeRF-AP/DS and NeRFies predict a dense deformation field designed for dense data. However, our input data consists of a few static frames only where the deformation field leads to severe overfitting.

Training. During training, we sample rays from a single frame to avoid out-of-memory issues when evaluating the tetrahedral mesh for multiple frames. Each batch contains 1024 rays. We sample N_{coarse}=128 points along a single ray during the coarse sampling and $N_{\text{importance}}$=64 for the importance sampling. We train the network to minimize the loss in Eq. (28) and sparsity losses with standard weights used in VolTeMorph [41, 54]. We train the methods for 5×10^5 steps using Adam [75] optimizer with learning rate 5×10^{-4} decaying exponentially by factor of 0.1 every 5×10^5 steps.

Inference. During inference, we leverage the underlying mesh to sample points around tetrahedra hit by a single ray. Therefore, we perform a single-stage sampling with $N=N_{\text{coarse}}+N_{\text{importance}}$ samples along the ray. When extracting the features (Eq. (34)), we consider $|\mathcal{N}(\mathbf{v})|$=20 neighbors. For the Laplacian smoothing, we set λ_{diff}=0.1 and perform a single iteration step. Geometric-related operations impose negligible computational overhead.

3.4 Experiments

We evaluate all methods on data of four subjects from the publicly available Multiface dataset [185]. We track the face for eight manually-selected "extreme" expressions. We then select K=5 expressions the combinations of which show as many wrinkles as possible. Each subject was captured with ≈38 cameras which gives ≈190 training images per subject [2]. We use Peak Signal To Noise Ratio (PSNR) [6], Structural Similarity Index (SSIM) [178] and Learned

[2]To train BlendFields for a single subject we use ≈ 0.006% of the dataset used by AVA [16].

Parameter	Real Data						Synthetic Data		
	Casual Expressions			Novel Pose Synthesis			Novel Pose Synthesis		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
$\lvert\mathcal{N}(\mathbf{v})\rvert = 1$	27.5620	0.9043	0.0893	29.7269	0.9306	0.0815	32.2371	0.9882	0.0234
$\lvert\mathcal{N}(\mathbf{v})\rvert = 5$	27.5880	0.9054	0.0864	29.7548	0.9312	0.0789	32.2900	0.9882	0.0231
$\lvert\mathcal{N}(\mathbf{v})\rvert = 10$	27.5933	0.9054	0.0859	29.7456	0.9312	0.0785	32.3324	0.9882	0.0230
$\lvert\mathcal{N}(\mathbf{v})\rvert = 20$	27.5977	0.9056	0.0854	29.7372	0.9311	0.0782	32.7949	0.9887	0.0221
Without smoothing	27.2535	0.8959	0.0939	29.3726	0.9233	0.0846	32.2452	0.9876	0.0238
With smoothing	27.5977	0.9056	0.0854	29.7372	0.9311	0.0782	32.7949	0.9887	0.0221

Table 8. Ablation study – First, we check the effect of the neighborhood size $|\mathcal{N}(\mathbf{v})|$ on the results. Below that, we compare the effect of smoothing. The best results are colored in ■ and the second best in ■. For the real dataset, changing the neighborhood size gives inconsistent results, while smoothing improves the rendering quality. In the synthetic scenario, setting $|\mathcal{N}(\mathbf{v})|$=20 and the Laplacian smoothing consistently gives the best results. The discrepancy between real and synthetic datasets is caused by inaccurate face tracking for the former. We describe this issue in detail in Sec. 3.4.4.

Perceptual Image Patch Similarity (LPIPS) [215] to measure the performance of the models. Each of the rendered images has a resolution of 334×512 pixels.

As baselines, we use the following approaches: the original, static NeRF [104], NeRF conditioned on an expression code concatenated with input points $\mathbf{x}$, NeRFies [119], HyperNeRF[3] [120], and VolTeMorph [41]. We replace the learnable code in NeRFies and HyperNeRF with the expression code $\mathbf{e}$ from the parametric model. Since VolTeMorph can be trained on multiple frames, which should lead to averaging of the output colors, we split it into two regimes: one trained on the most extreme expression[4] (VolTeMorph_1) and the another trained on all available expressions ($\text{VolTeMorph}_{\text{avg}}$)[5]. We use both of these baselines as VolTeMorph was originally designed for a single-frame scenario. By using two versions, we show that it is not trivial to extend it to multiple expressions.

3.4.1 Realistic Human Captures

Novel expression synthesis. We extract eight multi-view frames from the Multiface dataset [185], each of a different expression. Five of these expressions serve as training data, and the rest are used for evaluation. We present in Fig. 25 example training frames for one of the subjects. Each frame is a multi-view frame captured with ≈35 cameras (the number of available cameras varied slightly between subjects).

After training, we can extrapolate from the training expressions by modifying the expression vector $\mathbf{e}$. We use the remaining three expressions: moving mouth left and right, and puffing cheeks, to evaluate the capability of the models to reconstruct other expressions. In Fig. 21 we show that BlendFields is the only method capable of rendering convincing wrinkles

[3] We use two architectures proposed by Park *et al.* [120].

[4] We manually select one frame that has the most visible wrinkles.

[5] We do not compare to NeRFace [38] and NHA [45] as VolTeMorph [41] performs better quantitatively than these methods.

Method	Casual Expressions			Novel Pose Synthesis		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [104]	22.0060	0.6556	0.3222	23.8077	0.7448	0.2779
Conditioned NeRF [104]	21.0846	0.6280	0.3042	22.9991	0.7261	0.2362
NeRFies [119]	20.7004	0.6076	0.3579	23.0123	0.7253	0.2840
HyperNeRF-AP [120]	20.8105	0.6214	0.3504	22.8193	0.7185	0.2689
HyperNeRF-DS [120]	20.8847	0.6111	0.3656	23.0075	0.7259	0.2729
VolTeMorph$_1$ [41]	21.3265	0.7091	0.2706	22.3007	0.7795	0.2281
VolTeMorph$_{avg}$[41]	22.0759	0.7755	0.2615	23.8974	0.8458	0.2302
BlendFields	22.8982	0.7954	0.2256	24.4432	0.8477	0.2052

Table 9. Quantitative results without masking – Similarly to Tab. 7, we compare BlendFields to other related approaches. However, we calculate the results over the whole image space, without removing the background. BlendFields and VolTeMorph [41] model the background as a separate NeRF-based [104] network. The points that do not fall into the tetrahedral mesh are assigned to the background. As the network overfits to sparse training views, it poorly extrapolates to novel expressions (as the new head pose or expression may reveal some unknown parts of the background) and views. At the same time, all other baselines do not have any mechanism to disambiguate the background and the foreground.

dynamically, depending on the input expression. BlendFields performs favorably compared to the baselines (see Tab. 7).

Casual expressions. The Multiface dataset contains sequences where the subject follows a script of expressions to show during the capture. Each of these captures contains between 1000 and 2000 frames. This experiment tests whether a model can interpolate between the training expressions smoothly and generalize beyond the training data. Quantitative results are shown in Tab. 7. Our approach performs best all the settings.

Additionally, we compare BlendFields and the baselines similarly to Sec. 3.4.1. However, in this experiment, we deliberately include the background in metric calculation. We show the results in Tab. 9. In all the cases, BlendFields performs best even though the method was not designed to model the background accurately. Additionally, as HyperNeRF [120], NeRFies [119], and NeRF [104] do not have any mechanism to disambiguate between the foreground and the background, the metrics are significantly worse when including the latter.

We show in Fig. 24 results of baselines that do not rely on parametric models of the face [84]. Compared to BlendFields, they cannot render high-fidelity faces. The issue comes from the assumed data sparsity—those approaches rely on the interpolation in the training data. As we assume access to just a few frames, there is no continuity in the training data that would guide them to interpolate between known expressions. BlendFields presents superior results given novel expressions even with such a sparse dataset.

3.4.2 Modeling Objects Beyond Faces

We show that our method can be applied beyond face modeling. We prepare two datasets containing 96 views per frame of bending and twisting cylinders made of a rubber-like material (24 and 72 temporal frames, respectively). When bent or twisted, the cylinders reveal pose-dependent details. The expression vector $\mathbf{e}$ now encodes time: 0 if the cylinder is in the canonical pose, 1 if it is posed, and any values between $[0, 1]$ for the transitioning stage. We select expressions $\{0, 0.5, 1.0\}$ as a training set (for VolTeMorph$_1$ we use 1.0 only). For evaluation, we take every fourth frame from the full sequence using cameras from the bottom and both sides of the object. We take the mesh directly from Houdini [193], which we use for wrinkle simulation, and render the images in Blender [23]. We show quantitative results in Tab. 7 for the bending cylinder, and a comparison of the inferred images in Fig. 22 for the twisted one[6]. BlendFields accurately captures the transition from the rest configuration to the deformed state of the cylinder, rendering high-frequency details where required. All other approaches struggle with interpolation between states. VolTeMorph$_1$ (trained on a single extreme pose) renders wrinkles even when the cylinder is not twisted.

3.4.3 Ablations

Contributed components We check how the neighborhood size $|\mathcal{N}(\mathbf{v})|$ and the application of the smoothing influence the performance of our method. We show the results in Tab. 8. BlendFields works best in most cases when considering a relatively wide neighborhood for the tetrahedral features[7]. Laplacian smoothing consistently improves the quality across all the datasets (see Fig. 20).

Ablating number of expressions We ablate over the number of used expressions during the training. To evaluate the effect of the number of expressions, we add consecutive frames to the training set (starting from a single, neutral one), *i.e.*, the training set has $k{<}K$ expressions. We train BlendFields for such a set for each subject separately. We then average the results for a given k across subjects. We present the results in Tab. 6. When selecting the training expressions, we aim to choose those that show all wrinkles when combined. We can see from Fig. 23 that if removed, *e.g.*, the expressions with eyebrows raised, then the model cannot render wrinkles on the forehead. In summary, increasing the number of expressions improves the quality results with diminishing returns when $K{>}5$, while $K{=}5$ provides a sufficient trade-off between the data capture cost and the quality.

3.4.4 Failure Cases

While BlendFields offers significant advantages for rendering realistic and dynamic high-frequency details, it falls short in some scenarios (see Fig. 26). One of the issues arises when the contrast between wrinkles and the subject's skin color is low. In those instances, we observe a much longer time to convergence. Moreover, as we build BlendFields on VolTeMorph, we also inherit some of its problems. Namely, the method heavily relies on the initial fit of the parametric model—any inaccuracy leads to ghosting artifacts or details on the face that jump between frames.

3.5 Conclusions

We present a general approach, BlendFields, for rendering high-frequency expression-dependent details using NeRFs. BlendFields draws inspiration from classical computer graphics by blending expressions from the training data to render expressions unseen during training. We show that BlendFields renders images in a controllable and interpretable manner for novel expressions and can be applied to render human avatars learned from publicly available datasets.

[6] Our motivation is that it is easier to show pose-dependent deformations on twisting as it affects the object globally, while the bending cannot be modeled by all the baselines due to the non-stationary effects.

[7] Larger neighborhood sizes caused out-of-memory errors on our NVIDIA 2080Ti GPU.

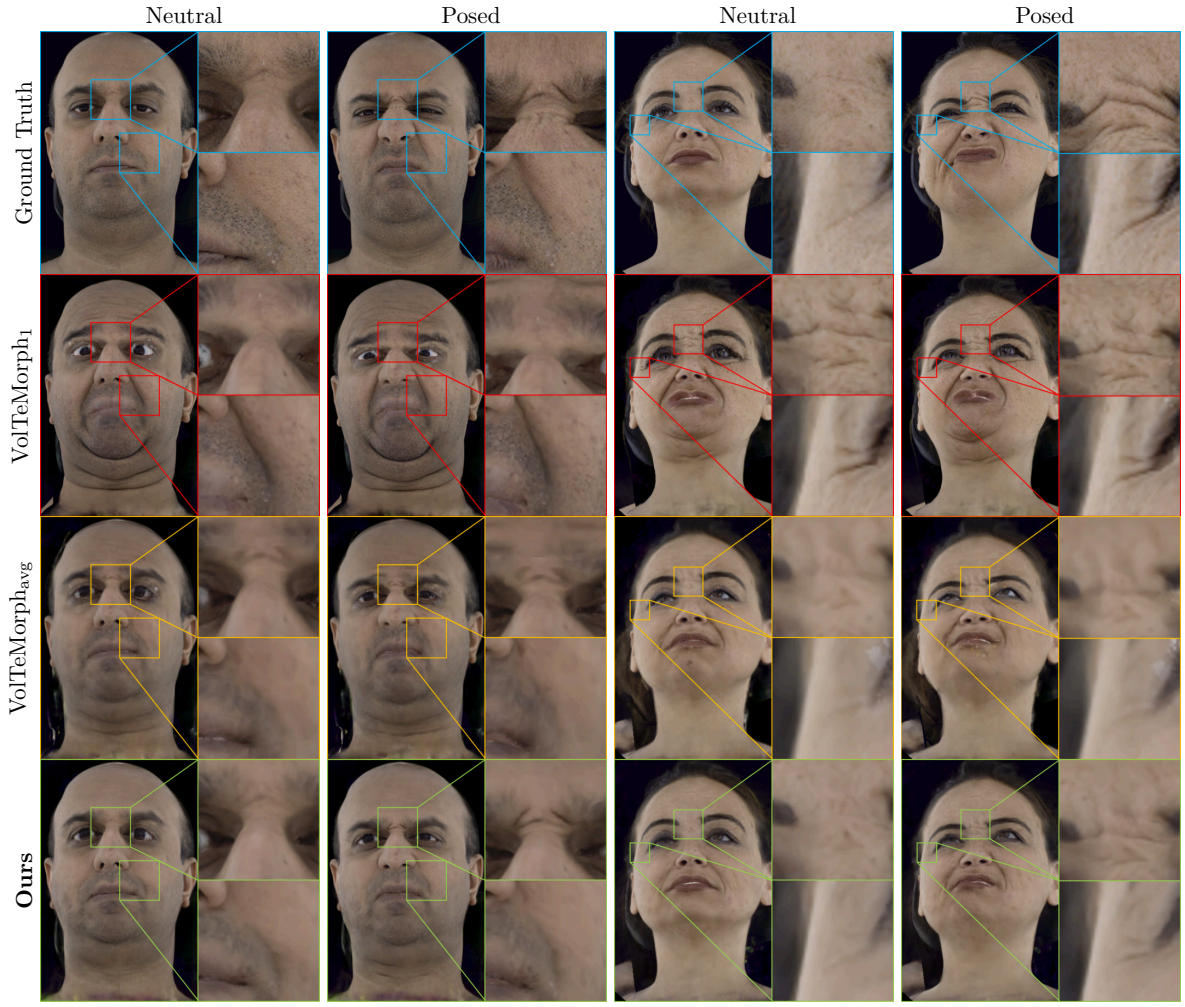


Figure 21. Novel expression synthesis – We compare qualitatively BlendFields with selected baselines (vertical) across two selected subjects (horizontal). Firstly, we show a neutral pose of the subject and then any of the available expressions. To our surprise, VolTeMorph$_{\text{avg}}$ trained on multiple frames renders some details but with much lower fidelity. We argue that VolTeMorph$_{\text{arg}}$ considers rendering wrinkles as artifacts that depend on the view direction (see Eq. (21)). VolTeMorph$_1$ is limited to producing the wrinkles it was trained for. In contrast to those baselines, **BlendFields** captures the details and generalizes outside of the distribution.

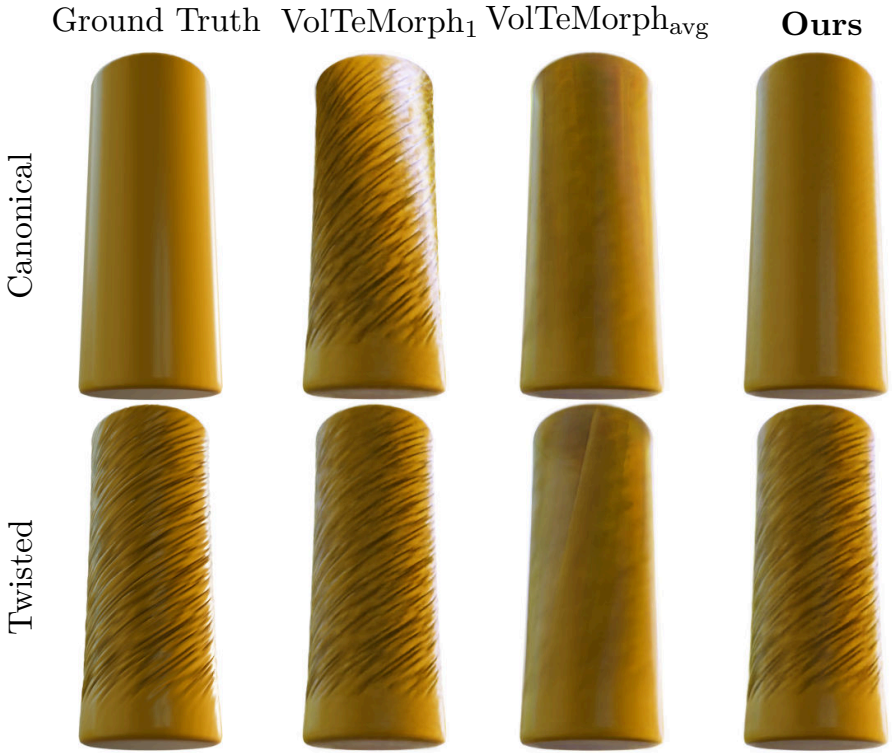


Figure 22. Qualitative results on synthetic dataset – For a simple dataset, baselines cannot model high-frequency, pose-dependent details. VolTeMorph$_1$ renders wrinkles for the straight pose as well, as it is trained for the twisted cylinder only, while VolTeMorph$_{avg}$ averages out the texture.

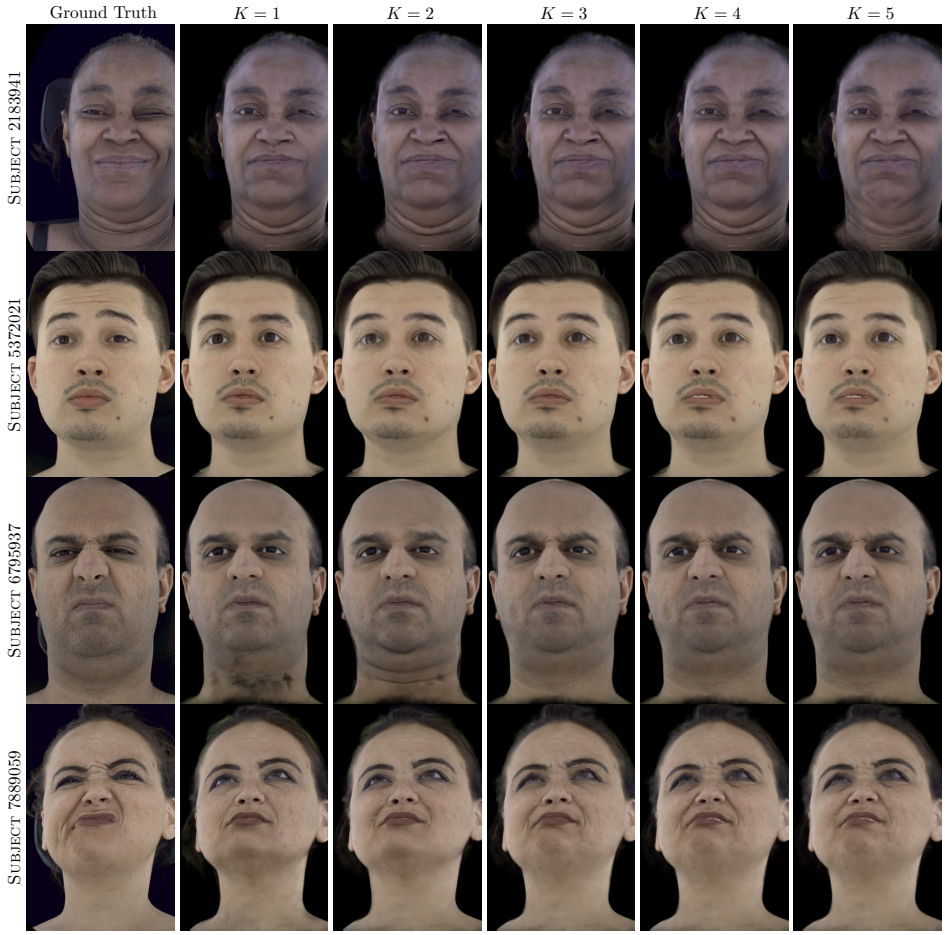


Figure 23. Qualitative ablation over the number of training expressions – We show qualitatively how the number of training expressions K affects the rendering quality. The first row shows the ground truth images. All other consecutive rows show the images rendered with BlendFields while increasing the number of training expressions. The last row, K=5 corresponds to the results presented in the main part of the article. The subject's naming follows the convention introduced in the Multiface repository [185]. Please refer to Tab. 6 for quantitative results.

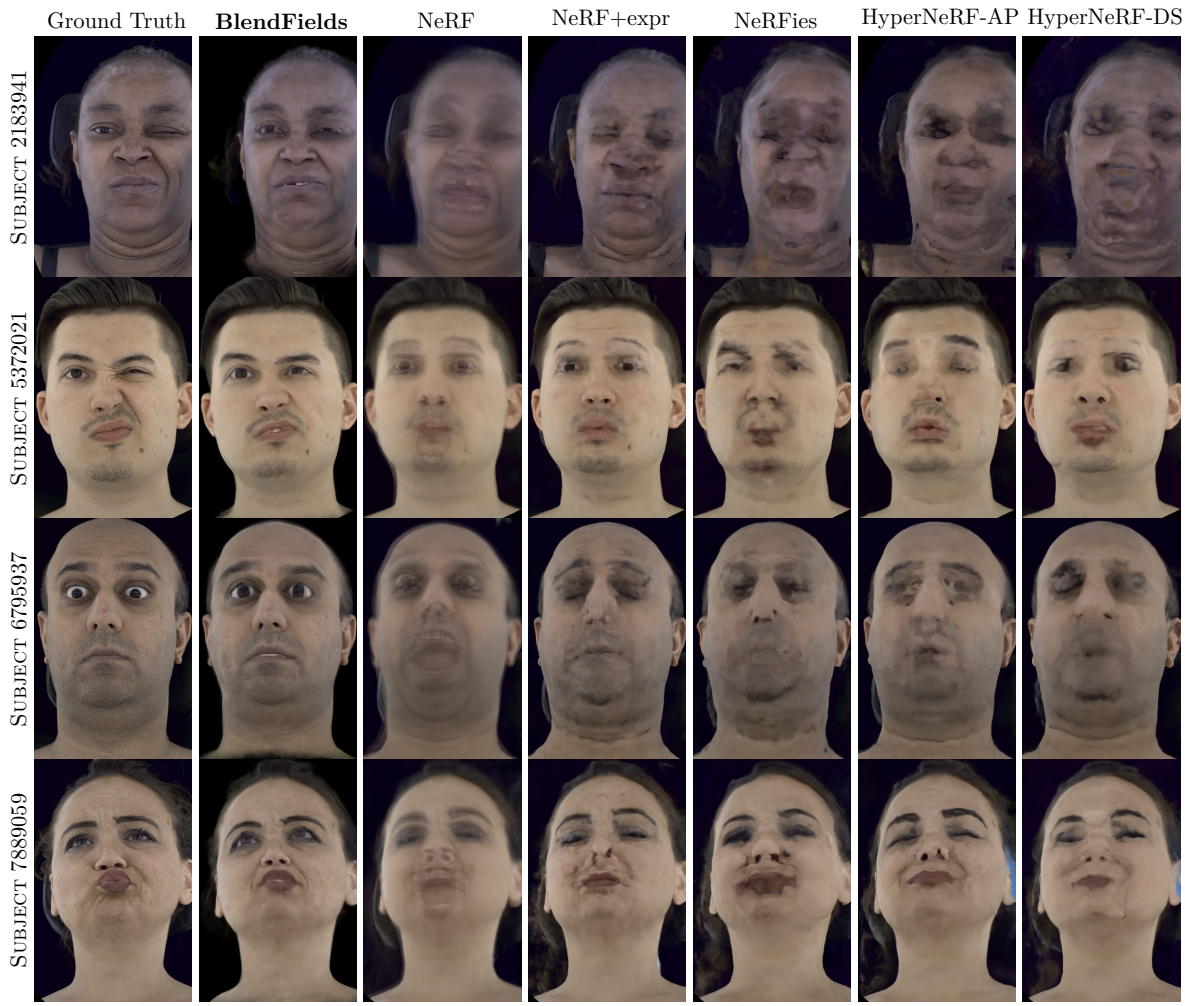


Figure 24. Comparison to strictly data-driven approaches – We compare BlendFields to other baselines that do not rely on mesh-driven rendering: NeRF [104], NeRF conditioned on the expression code (NeRF+expr) [104], NeRFies [119], and HyperNeRF-AP/DS [120]. As a static model, NeRF converges to an average face from available (K=5) expressions. All other baselines exhibit severe artifacts compared to BlendFields. Those baselines rely on the data continuity in the training set (*e.g.*, from a video), and cannot generalize to any other expression.

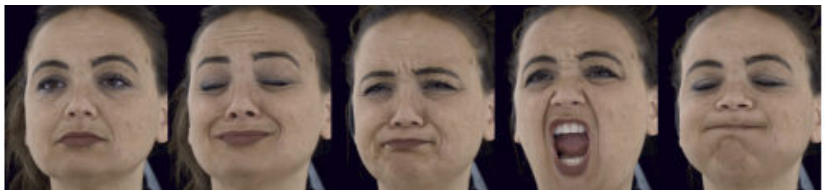


Figure 25. Training frames – In Sec. 3.4, we show results for the BlendFields trained on K=5 expressions. The images represent these expressions for one of the subjects. For each subject, we selected similar expressions to show all possible wrinkles when combined. Please note that we also include a "neutral" expression (the first from the left)—it is necessary to enable the learning of a face without any wrinkles.

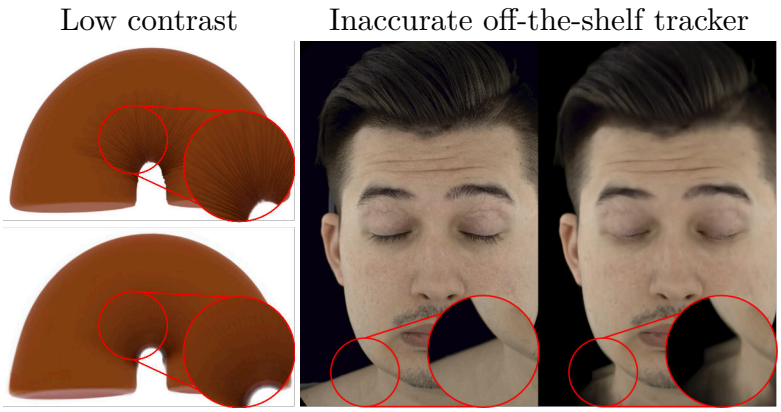


Figure 26. Failure cases – We show failure cases for our proposed approach. *Left:* In the presence of wrinkles in low-contrast images, BlendFields takes longer to converge to make wrinkles visible. We show the ground truth on the top, and rendering after training $7{\times}10^5$ steps on the bottom. In contrast, we rendered images in Fig. 22 after $2{\times}10^5$ steps. *Right:* BlendFields inherits issues from VolTeMorph [41], which relies on the initial fit of the face mesh. If the fit is inaccurate, artifacts appear in the final render.

Chapter 4

LumiGauss: Relightable Gaussian Splatting in the Wild

Decoupling lighting from geometry using unconstrained photo collections is notoriously challenging. Solving it would benefit many users as creating complex 3D assets takes days of manual labor. Many previous works have attempted to address this issue, often at the expense of output fidelity, which questions the practicality of such methods. In this chapter, we introduce LumiGauss—a technique that tackles 3D reconstruction of scenes and environmental lighting through 2D Gaussian Splatting. Our approach yields high-quality scene reconstructions and enables realistic lighting synthesis under novel environment maps. We also propose a method for enhancing the quality of shadows, common in outdoor scenes, by exploiting spherical harmonics properties. Our approach facilitates seamless integration with game engines and enables the use of fast precomputed radiance transfer. We validate our method on the NeRF-OSR dataset, demonstrating superior performance over baseline methods. Moreover, LumiGauss can synthesize realistic images for unseen environment maps.

4.1 Introduction

The colors emitted by objects are a combination of a spectrum of the light hitting the object and the material properties of that object. The light hitting the object's surface is a sum of the light scattered in the medium and bounced from neighboring objects [181]. In computer graphics, we often simplify this effect and decouple it into two entities: an intrinsic object's color or *albedo* and an omnidirectional texture representing the illumination [128]—*environment map*. Acquiring those assets enables the designing of realistic scenes in games or movies.

In many scenarios, creating realistic albedo textures and environment maps requires skilled technicians and artists to be involved in the process. To democratize it, the previous approaches [43, 135, 179] tried to use photographs taken with commodity cameras and *invert* the

Figure 27. Teaser – LumiGauss reconstructs environment maps and object surfaces from *in-the-wild* images. Our model decouples the scene color and its normals (*second and fourth column in the top row*). At inference, it can synthesize novel views (*bottom row*) and realistic lighting (*first and third columns in the bottom*) with high-fidelity shadows (*second and fourth columns in the bottom*).

capturing process to recover albedo and an environment map. Given the abundance of casual, in-the-wild photographs available on the Internet, solving that issue is of high importance.

Recent advancements in reconstruction in-the-wild include NeRF-in-the-Wild [99] (NeRF-W). NeRF-W leverages neural radiance fields [104] which reconstruct a scene given its photos with calibrated cameras. NeRF-W can further work in realistic scenarios where the pictures come from the *in-the-wild* collections—the images in such may differ in the lighting conditions or scene content. However, NeRF-W and its follow-up works, HA-NeRF [19] and CR-NeRF [200], cannot decouple the object's albedo and the environment map, making it difficult to use in practice. NeRF-OSR [135] approaches that problem, but its shading model requires neural network execution at runtime, making integration with graphics engines difficult, and the reconstruction quality leaves space for improvement.

3D Gaussian Splatting [71] (3DGS) solves one of the main bottlenecks of NeRF - the training speed and output fidelity. In contrast to NeRFs, 3DGS models the scene as a composition of 3D Gaussians attributed with colors and opacity which are rasterized, or *splatted*, to render the output image. Recovering an object's surface from them requires specialized training techniques [48]. On the other hand, 2DGS [57] proposes reformulating 3D Gaussians as their 2D alternative where one of the axes is collapsed. The final scene representation ends up being composed of 2D *surfels* which provide a flat surface crucial for our relighting approach.

In this work, we propose LumiGauss, a method that uses 2DGS [57] to perform inverse graphics on images taken in the wild. In contrast to past approaches, our method is imbued with fast training and inference speed while maintaining high-quality renderings and being easy to integrate with graphics engines. In our method, the light is modeled as a combination of an environment map and a radiance transfer function that represents which parts of the

environment map illuminate a given surfel—both are modeled by spherical harmonics [128]. This approach allows for modeling shadows, which is our main goal, but also has the potential to represent light reflected off of other objects. The output from LumiGauss enables both novel view synthesis and relighting using environment maps beyond those available during training. Leveraging the possibilities offered by the precomputed radiance transfer, our representation integrates seamlessly into game engines, enabling fast and efficient relighting.

Our contributions:

- We repurpose 2D Gaussian Splatting for an inverse graphics pipeline in an in-the-wild setting. With our approach, we recover high-quality albedo and environment maps.
- To enable shadows we learn the radiance transfer function for each 2D splat and represent it using spherical harmonics.
- Finally, we demonstrate that our reconstructed environment maps can be effectively used to relight arbitrary objects within graphic engines.

4.2 Related Works

Relighting Relighting outdoor scenes is a key challenge in computer graphics and VR/AR. Early works [9, 28, 50, 78, 155, 168, 191] used training-free methods like statistical inference. Deep learning approaches, such as Yu *et al.* [206] with a neural renderer, and Philip *et al.* [122] with proxy geometry, face limitations in reconstruction quality and viewpoint flexibility.

NeRF-based methods [104] enabled simultaneous viewpoint and lighting changes. However, methods like [147, 209, 217] handle a single illumination only or specific illumination setup during. Others are object-specific, such as for faces [154]. Many unconstrained photo collection methods focus on appearance, not lighting, complicating integration with other graphical components [19, 81, 99, 200].

NeRF-based approaches, such as [125] and [51], focus on inverse rendering for outdoor scenes, particularly in applications like autonomous driving. However, these methods are designed for single video sequences rather than unstructured photo collections. Rudnev *et al.* [135] proposed a method for relighting landmarks from unconstrained photo collections, using NeRF with external lighting extraction. Similarly, [82] compresses the per-image illumination into a disentangled latent vector. Wang *et al.* [179] target static scenes and works with unconstrained photo collections but rely on costly mesh extraction. Some methods incorporate additional priors, environmental assumptions, or regularizations [152, 199]. Gardner *et al.* [43] leverage externally trained models to provide environmental lighting priors. Despite their potential, these methods cannot be used in real-time applications due to NeRF's slow training and rendering times.

In contrast, the TensoRF-based approach by Chang *et al.* [17] aligns time information and sun direction with images for relighting, eliminating the need for external lighting models. While this method is faster than NeRF, it still lacks seamless integration with graphics engines and is unsuitable for synthetic light integration.

Notable Gaussian Splatting works designed for unconstrained photo collections [25, 71, 176, 192, 210] focused on appearance editing, not seamless graphical component integration. Relightable Gaussian approaches, like [39, 86, 143], tackle material decomposition but are not adapted to handle varying lighting conditions of *in-the-wild* training setup. Radiance transfer properties, employed in a similar way to LumiGauss, are utilized in [136, 213]. However, these methods rely on a burdensome dataset setup, restricting their applicability to specific use cases.

Gaussian Splatting Kerbl *et al.* [71] introduced a notion of using learnable 3D Gaussian primitives from point clouds. Those Gaussians are parametrized with 3D covariance matrix Σ_k and their location $\mathbf{t}_k$:

$$\mathcal{G}(\mathbf{t}) = \exp(\frac{1}{2}(\mathbf{t}-\mathbf{t}_k)^\top \Sigma_k^{-1}(\mathbf{t}-\mathbf{t}_k)), \tag{36}$$

where the covariance matrix is factorized into a scaling diagonal matrix $\mathbf{s}_k$ and a rotation matrix $\mathbf{R}_k$ as $\Sigma_k{=}\mathbf{R}_k\mathbf{s}_k\mathbf{s}_k^\top\mathbf{R}_k^\top$. An image is rendered with a splatting operator $\mathcal{S}(\cdot)$ which projects Gaussians into the camera coordinates with a world-to-camera matrix and then to image plane with a local affine transformation [225]:

$$\mathcal{S}(c_c \mid \mathcal{G}) = \sum_{k=1}^{K} \mathbf{c}_k o_k \mathcal{G}_k \prod_{j=1}^{k-1}(1-o_j\mathcal{G}_k). \tag{37}$$

The operator produces an RGB image, given a calibrated camera matrix c_c and their additional Gaussians' attributes: their colors $\mathbf{c}$ and opacities o. Attributes are learned using a stochastic gradient descent.

Huan *et al.* [57] argues that 3DGS although producing high-quality images, the implicit surface representation is noisy, limiting its applicability in relighting scenarios. They propose using 2D Gaussians instead to create smooth, coherent meshes thanks to their exact 2D surfel projection. We leverage that representation in our LumiGauss—a relightable model that decouples albedo, environment light and shadows thanks to our proposed physical constraints.

4.3 Method

4.3.1 Preliminaries on Radiance Transfer

The rendering equation, in its simplified form [46], is an integral function that represents light $L(\mathbf{x}, \boldsymbol{\omega}_o)$ exiting point $\mathbf{x}$ along the vector $\boldsymbol{\omega}_o$:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = \int_s f_r(x, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i) L_i(\mathbf{x}, \boldsymbol{\omega}_i) D(\mathbf{x}, \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i \tag{38}$$

where $f_r(\cdot)$ is a BRDF function, $L_i(\cdot)$ an incoming light along the vector $\boldsymbol{\omega}_i$, and $D(\cdot)$ is a radiance transfer function. Intuitively, $f_r(\cdot)$ represents the surface material, $L_i(\cdot)$ represents

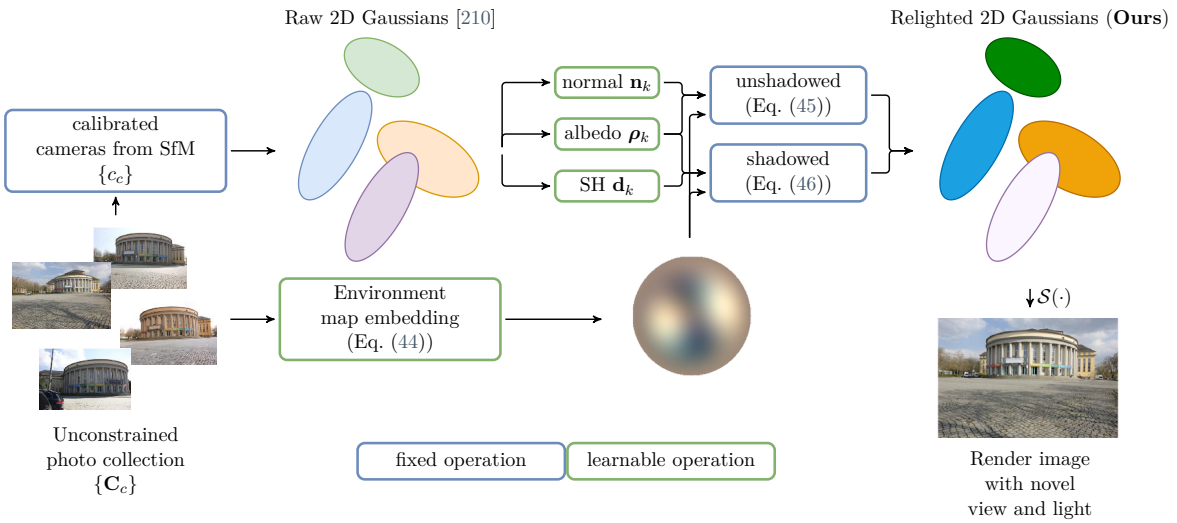


Figure 28. Pipeline – LumiGauss learns the relightable 2D Gaussian [210] representation from unconstrained photo collection with variable camera parameters and lighting conditions. Each of k Gaussians holds: a normal $\mathbf{n}_k$, albedo $\boldsymbol{\rho}_k$, and learnable transfer function $\mathbf{d}_k$. Our contributed method composes the Gaussians in two modes—*shadowed* and *unshadowed*. The *shadowed* model reconstructs additional shadows (see Fig. 27) on top of the unshadowed model thanks to our proposed use of a radiance transfer function. The Gaussians are splatted [71, 210] to render the output image in a novel view and light.

the intensity and color of the illumination, and $D(\cdot)$ is a term that takes into account shadows or light reflections from other surfaces. Depending on the formulation of those functions, the rendering equation can range from a straightforward and inaccurate light model to a highly complex and accurate one.

Unshadowed model One example of a reflection model that can be represented with Eq. (38) is the diffuse surface reflection model, also known as *dot product lighting*. A diffuse BRDF reflects light uniformly, making the lighting view-independent and simplifying the BRDF as follows:

$$L_D(\mathbf{x}) = \frac{\rho(\mathbf{x})}{\pi} \int_s L_i(\mathbf{x}, \boldsymbol{\omega}_i) \max(\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i, 0) d\boldsymbol{\omega}_i \tag{39}$$

where $\rho(\cdot)$ is the surface albedo, $\mathbf{n}(\mathbf{x})$ a surface normal at the point x. Shadows are neglected.

The incoming light $L_i(\mathbf{x}, \boldsymbol{\omega}_i)$ can be represented in several ways. In this work, we assume that the scene is illuminated with an **omnidirectional environment map** that is parametrized using spherical harmonics (SH) of degree n with $(n+1)^2$ coefficients. Because the environment map is positioned infinitely far from the scene, the light is position-independent, and thus, the rendering equation is further simplified:

$$L_U(\mathbf{x}) = \frac{\rho(\mathbf{x})}{\pi} \int_s L_i(\boldsymbol{\omega}_i) \max(\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i, 0) d\boldsymbol{\omega}_i \tag{40}$$

With illumination parametrized with SH, we can evaluate the integral in Eq. (40) using a closed-form solution from Eq. (12) in [128]. From this point onward, we refer to rendering with Eq. (40) as *unshadowed*.

Shadowed model In addition to the *unshadowed* lighting model, we propose a *shadowed* model, where $D(\mathbf{x}, \boldsymbol{\omega}_i)$ is parameterized using spherical harmonics (SH) and learned from training data. In $D(\mathbf{x}, \boldsymbol{\omega}_i)$, SH represents a spherical signal that quantifies the light arriving from each direction of the environment map to an associated point in space. The *shadowed* model is derived by replacing the dot product term in Eq. (40):

$$L_S(\mathbf{x}) = \frac{\rho(\mathbf{x})}{\pi} \int_s L_i(\boldsymbol{\omega}_i) D(\mathbf{x}, \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i. \tag{41}$$

In addition to modeling shadows, this approach also has the potential to model the interreflection of light between objects in the scene.

Using SH of the same degree for both the environment map and transfer function allows efficient evaluation of the rendering equation Eq. (41). A key SH property simplifies the integral of two SH-based functions to a dot product of their coefficients, thanks to SH orthogonality. With this property Eq. (41) can be re-written as:

$$L_S(\mathbf{x}) = \frac{\rho(\mathbf{x})}{\pi} \mathbf{l} \cdot \mathbf{d}, \tag{42}$$

where $\mathbf{l} \in R^{(n+1)^2}$ are the SH coefficients of $L_i(\boldsymbol{\omega}_i)$ and $\mathbf{d} \in R^{(n+1)^2}$ are the SH coefficients of $D(\mathbf{x}, \boldsymbol{\omega}_i)$. Please see [46, 145] for derivation. This property is commonly used in real-time rendering where the radiance transfer function is pre-computed and only Eq. (42) is evaluated at runtime.

4.3.2 LumiGauss

LumiGauss creates a 3D representation of a relightable model using 2D Gaussians [57] from $c \leq C$ images taken *in-the-wild* $\{\mathcal{I}_c\}_{c=1}^{C}$ with associated calibrated cameras $\{c_c\}_{c=1}^{C}$. Our goal is to find Gaussian parameters $\mathcal{G}{=}\{\mathbf{t}_k, \mathbf{R}_k, \mathbf{s}_k, o_k, \boldsymbol{\rho}_k, \mathbf{d}_k\}_{k=1}^{K}$ that after the rasterization [71] recreate those images. We optimize Gaussians by minimizing the objective:

$$\underset{\mathcal{G}, \{\boldsymbol{\beta}_c\}, \boldsymbol{\theta}}{\arg\min}\, \mathbb{E}_{c_c \sim \{c_c\}} \underbrace{\ell_{\text{rgb}}(\mathcal{S}(c_c \mid \mathcal{G}, \{\boldsymbol{\beta}_c\}, \boldsymbol{\theta}), \mathbf{C}_c)}_{\text{Sec. 4.3.4}} + \underbrace{\mathcal{R}(\mathcal{G})}_{\text{Sec. 4.3.3}}, \tag{43}$$

where $\{\boldsymbol{\beta}_c\}{=}\{\mathbf{e}_c\}_{c=1}^{C}$ is a set of scene-dependent, learnable environment embeddings, ℓ_{rgb} is a photometric objective that compares the rendered image from an operator $\mathcal{S}(\cdot)$ (Eq. (37)), and $\mathcal{R}$ are additional regularization terms. In contrast to 2DGS [57], for each Gaussian we model

the base color $\boldsymbol{\rho}$ as diffuse[1], and introduce SH coefficients for the transfer function $\mathbf{d}$[2]. 2DGS provides smooth normals that make relighting possible.

In what follows, we drop the dependence of functional forms on the positions $\mathbf{x}$ we introduced in Sec. 4.3.1 for brevity.

Relighting To handle the diverse lighting conditions in *in-the-wild* images, we associate each training image with a learnable latent code $\mathbf{e}_c$ that encodes its lighting conditions. Using this embedding, we predict the environment map coefficients via an MLP:

$$\mathbf{l}_c = \text{MLP}(\mathbf{e}_c|\boldsymbol{\theta}), \tag{44}$$

where $\mathbf{l}_c \in \mathbb{R}^{3\times(n+1)^2}$ represents the SH coefficients of the environment map, and n=2 is the SH degree. As shown in [128], second-order SH is sufficient to approximate environment lighting in many scenarios.

The predicted illumination is used in the rendering process in one of two ways: *unshadowed* and *shadowed*. Those two approaches correspond to Eq. (40) and Eq. (42) respectively, and are described below.

Unshadowed model For the unshadowed scenario, we follow Eq. (40), which integrates light over the hemisphere in the direction of the surface normal. The color $\mathbf{c}_k$, *radiance*, for each Gaussian $\mathcal{G}_k$ given its normal $\mathbf{n}_k$ and the illumination parameters $\mathbf{l}_c$ equates to:

$$\mathbf{c}_k = \boldsymbol{\rho}_k \odot \underbrace{\mathbf{n}_k^t M(\mathbf{l}_k)\mathbf{n}_k}_{\text{unshadowed irradiance}}, \tag{45}$$

where M is a 4×4 matrix derived from the SH parameters of the environment map. It is the closed form solution of the integral in Eq. (40), please see Eq. (12) in [128] for details.

This simple yet effective model already imbues the model with relighting capabilities. However, as described in Fig. 29 it does not capture shadows correctly, limiting the output's fidelity.

Shadowed model To effectively capture shadows in the model, we redefine the output color of a Gaussian as $\tilde{\mathbf{c}}_k$, a function of learnable radiance transfer function D_k parametrized by spherical harmonics $\mathbf{d}_k \in \mathbb{R}^{(n+1)^2}$, light $\mathbf{l}_c$ and albedo $\boldsymbol{\rho}_k$. Using a learned radiance transfer function (instead of fixing it to capture light from the hemisphere above the normal as we do in *unshadowed*) allows for creating shadows, as described in Sec. 4.3.1. Overall, following Eq. (42),

[1] As per Sec. 4.3.1, view-dependent effects are not modeled in diffuse reflections.
[2] These coefficients correspond to a single channel in practice.

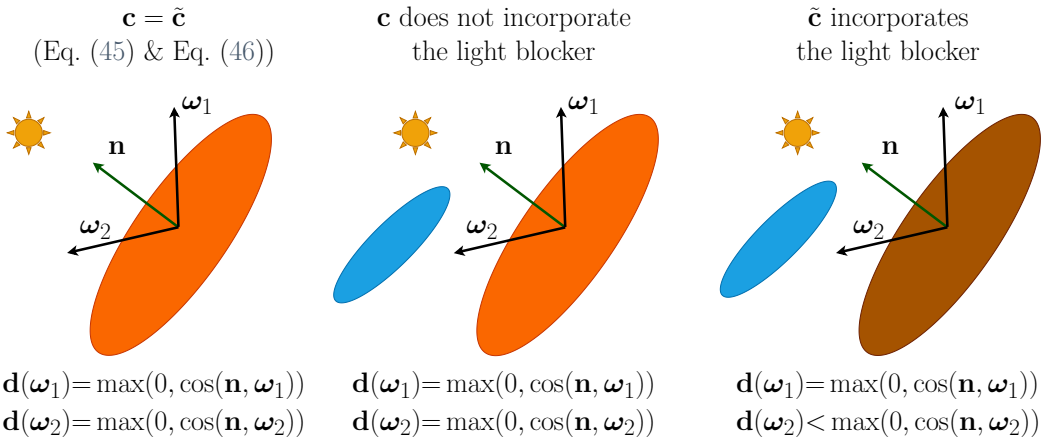


Figure 29. Unshadowed $\mathbf{c}$ (Eq. (45)) and shadowed $\tilde{\mathbf{c}}$ (Eq. (46)) may give the same output color if a Gaussian is fully exposed to the environment light. In the case of any occluder, $\mathbf{c}$ does not handle, and the color does not change. However, our proposed $\tilde{\mathbf{c}}$ properly reacts to the occluder and makes the output color darker.

the output shadowed color or *radiance* reduces to:

$$\tilde{\mathbf{c}}_k = \boldsymbol{\rho}_k \odot \underbrace{\sum_{i=1}^{(n+1)^2} \mathbf{l}_c^i \cdot \mathbf{d}_k^i}_{\text{shadowed irradiance}} , \tag{46}$$

As we show later in the experiments, the addition of shadows leads to more accurate relighting. Additionally, it does not require learnable MLP to reconstruct shadows at the inference stage, differentiating it from NeRF-OSR [135] and making our approach applicable to rendering engines directly.

4.3.3 Physical constraints

The regularizations proposed in 2DGS [57] keep the Gaussians close to the surface and smooth locally, which is crucial in our relighting scenario. Aside from them, we propose new loss terms based on the physical light properties that restrict the optimization from achieving degenerate, *non-relightable* cases. We restrict radiance transfer D_k function to remain within the range of 0 to 1, where 0 indicates complete shadowing and 1 signifies full exposure to lighting:

$$\begin{aligned} \ell_{0\text{-}1} = \mathbb{E}_k \mathbb{E}_{\boldsymbol{\omega}_i} [\| \max(D_k(\boldsymbol{\omega}_i), 1) - 1\|_2^2 \\ + \| \min(D_k(\boldsymbol{\omega}_i), 0)\|_2^2], \end{aligned} \tag{47}$$

and allow the environment light to remain in the $\mathbb{R}_+$ domain:

$$\ell_+ = \mathbb{E}_k \mathbb{E}_{\boldsymbol{\omega}_i} \| \min(L_c(\boldsymbol{\omega}_i), 0)\|_2^2, \tag{48}$$

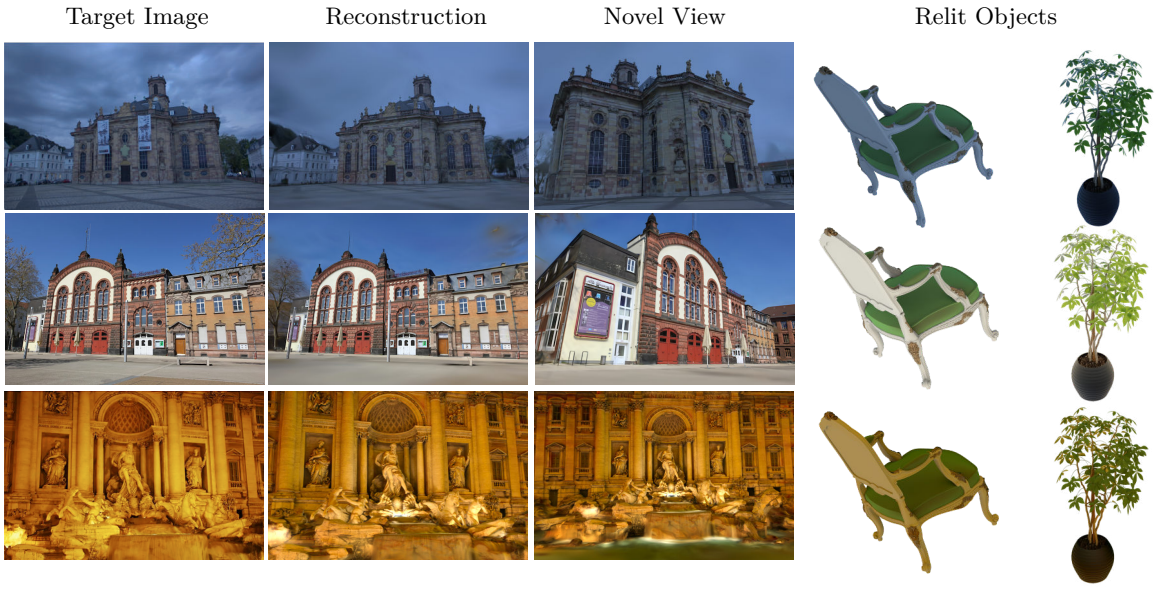


Figure 30. Scene reconstruction and relightning – Reconstruction and relighting capabilities of LumiGauss. LumiGauss reproduces sharp and clean landmarks, and the learned environment lighting enables accurate scene relighting. We use learned environment maps to relight the scene from novel viewpoints and then relight arbitrary objects within a graphics engine.

which allows the environment light to brighten the scene arbitrarily.

The shadowed radiance transfer should remain close to the unshadowed version. If not, the shadowed version might include light from any direction, resulting in degenerate solutions and incorrect relighting. We visualize the shadowed and unshadowed transfer functions in Fig. 29. To address this issue, we propose the following loss function:

$$\ell_{\bullet\leftrightarrow\circ} = \mathbb{E}_k \mathbb{E}_{\boldsymbol{\omega}_i} \| \max(\mathbf{n}_k \cdot \boldsymbol{\omega}_i, 0) - D_k(\boldsymbol{\omega}_i) \|_2^2, \tag{49}$$

The applied transfer function inherently accounts for shadows and interreflections. To focus specifically on modeling shadows and restrict the use of Eq. (46) for other cases, we impose a loss function ensuring that shadowed radiance should not be brighter than unshadowed one:

$$\ell_{\bullet} = \mathbb{E}_k \mathbb{E}_{\boldsymbol{\omega}_i} \| \max(D_k(\boldsymbol{\omega}_i) - \max(\mathbf{n}_k \cdot \boldsymbol{\omega}_i, 0), 0) \|_2^2, \tag{50}$$

Those losses are weighted with scalars $\{\lambda_{1,\dots,4}\}$ fixed across experiments and contribute to our regularization term:

$$\mathcal{R}(\mathcal{G}) = \lambda_{0\text{-}1}\ell_{0\text{-}1} + \lambda_{+}\ell_{+} + \lambda_{\bullet\leftrightarrow\circ}\ell_{\bullet\leftrightarrow\circ} + \lambda_{\bullet}\ell_{\bullet} \tag{51}$$

Calculating it exactly requires us to compute the expectation over the hemisphere $\mathbb{S}^2$. Instead, we approximate the expectations over directions $\boldsymbol{\omega}_i$ with a Monte Carlo estimator by randomly sampling the SH lobe with N samples at each training step.

Method	Landwehrplatz				Ludwigskirche				Staatstheater			
	PSNR ↑	MSE ↓	MAE ↓	SSIM ↑	PSNR ↑	MSE ↓	MAE ↓	SSIM ↑	PSNR ↑	MSE ↓	MAE ↓	SSIM ↑
Yu et al. $_{u/s}$[206]	15.17	0.033	0.133	0.376	17.87	0.017	0.097	0.378	15.28	0.032	0.138	0.385
Philip et al. [122]	12.28	0.062	0.179	0.319	16.63	0.023	0.113	0.367	12.34	0.065	0.200	0.272
NeRF-OSR [135]	16.65	0.024	0.114	0.501	18.72	0.014	0.090	0.468	15.43	0.029	0.133	0.517
NeRF-OSR* [135]	15.66	0.029	-	-	19.34	0.012	-	-	16.35	0.027	-	-
SR-TensoRF [17]	16.74	0.024	0.093	0.653	17.30	0.021	0.096	0.542	15.43	0.030	0.111	0.632
FEGR [179]	17.57	0.018	-	-	21.53	0.007	-	-	17.00	0.023	-	-
SOL-NeRF [152]	17.58	0.028	-	0.618	21.23	0.008	-	0.749	18.18	0.019	-	0.680
NeuSky [43]	18.31	0.016	-	-	22.50	0.005	-	-	16.66	0.023	-	-
Ours	18.01	0.017	0.096	0.778	19.59	0.012	0.085	0.700	17.02	0.021	0.107	0.729
Yu et al. [206]	15.84	0.028	0.123	0.392	18.71	0.014	0.088	0.400	15.43	0.031	0.136	0.363
Philip et al. $_{d/s}$ [122]	12.85	0.054	0.169	0.164	17.37	0.019	0.105	0.429	11.85	0.070	0.210	0.184
NeRF-OSR $_{d/s}$ [135]	17.38	0.021	0.106	0.576	19.86	0.011	0.080	0.626	15.83	0.026	0.128	0.556
Ours $_{d/s}$	18.40	0.016	0.094	0.746	20.13	0.011	0.080	0.727	17.24	0.020	0.105	0.715
Ours †	15.03	0.034	0.139	0.58	19.34	0.015	0.094	0.693	16.09	0.028	0.124	0.665
Ours ‡	17.59	0.019	0.100	0.733	19.05	0.016	0.097	0.680	16.83	0.022	0.110	0.694
Ours $\setminus \ell_{0\text{-}1}$	18.30	0.016	0.095	0.744	20.15	0.010	0.080	0.734	17.25	0.020	0.105	0.712
Ours $\setminus \ell_{+}$	17.35	0.020	0.104	0.728	20.17	0.012	0.081	0.729	17.10	0.020	0.106	0.703

Table 10. Quantitative results – Comparison between our LumiGauss and selected baselines for two different. We report the reconstruction quality regarding PSNR, MSE, MAE, SSIM on full and 4x downsampled image resolutions. *u/s* denotes using upsampled, *d/s* downsampled images for the evaluation, and the last delimited area presents the ablation study on downsampled data. We denote NeRF-OSR [135] results reproduced by FEGR [179] with *. We use † to further annotate our approach where we remove loss terms $\ell_{\bullet \leftrightarrow \circ}$, $\ell_{\text{rec}}(\circ)$ from the second training stage. In ‡, we omit the first training stage. Compared to the baselines, LumiGauss achieves reconstructions of high fidelity. It reliably produces smooth surfaces and sharp edges, reflected in its high SSIM values. Additionally, our proposed components either enhance reconstruction or preserve physical accuracy without negatively impacting the results. Please note, that NeuSky [43] is a concurrent work, published prior to the WACV's deadline at ECCV 2024.

4.3.4 Reconstruction

We render images using the splatting algorithm $\mathcal{S}(\cdot)$ proposed in 2DGS [57]. We compare the rendered images with ground-truth $\{\mathbf{C}_c\}$ taken with $\{c_c\}$ cameras. Our method builds on 2DGS [57] and therefore our reconstruction loss ℓ_{rgb} follows the following term:

$$\ell_{\text{rgb}} = \lambda_{\text{rec}}(\bullet)\ell_{\text{rec}}(\bullet) + \lambda_{\text{rec}}(\circ)\ell_{\text{rec}}(\circ), \tag{52}$$

$$\ell_{\text{rec}}(\{\bullet, \circ\}) = \ell_1(\{\bullet, \circ\}) + \lambda_{\text{D-SSIM}}\ell_{\text{D-SSIM}}(\{\bullet, \circ\}), \tag{53}$$

where the ℓ_1 is the L_1 loss comparing either the image rendered from our shadowed or unshadowed models and $\ell_{\text{D-SSIM}}$ is a differentiable D-SSIM [177] further improving the quality. We use $\lambda_{\text{D-SSIM}}$=0.2 throughout all the experiments. Our proposed $\ell_{\text{rec}}(\circ)$ resembles a pretraining stage. As the more complex shadowed model lands in local minima if trained from scratch, we initiate the training with $\lambda_{\text{rec}}(\bullet)$=0.0 and $\lambda_{\text{rec}}(\circ)$=1.0. Once the simpler model converges, we switch $\lambda_{\text{rec}}(\bullet)$=1.0 and $\lambda_{\text{rec}}(\circ)$ to a small value so as not to deteriorate the quality of the model. In short, the shadowed model explains the parts of an image with shadows, which the unshadowed could not with its simpler lighting model.

Method	Training time	FPS
NeRF-OSR [135]	31h	0.003
NeuSky [43]	14h	0.004
Ours	1h 20min	20.7

Table 11. Performance comparison – Training time and inference speed comparison between the baselines and our LumiGauss.

4.3.5 Implementation details

The appearance embedding vector is set to a size of 24 dimensions. For predicting the environment map, we use MLP with 3 fully-connected layers of size 64. We trained all models for 40000 iterations, the first training stage is set to 20000 iterations. The learning rate for MLP and embedding is set to 0.002, which after first training stage is reduced to 0.0002. We train gaussian spherical harmonics with a learning rate of 0.002. We set the loss function weights as follows: for $\lambda_{0\text{-}1} = 0.001$, for $\lambda_{+} = 0.05$, for $\lambda_{\bullet\leftrightarrow\circ} \in \{1.0, 10.0\}$, for $\lambda_{\bullet} = 10.0$. In the second training stage we set $\lambda_{\bullet} = 0.001$.

We adhere to the original Gaussian splatting densification and pruning protocols, with a densification interval of 500 iterations and an opacity reset interval of 3000 iterations. We apply regularizations to align Gaussians with surfaces, as originally described in [57]. Additionally, we utilize the dual visibility concept proposed in [57]. This ensures that the Gaussians are always correctly oriented towards the camera. Dual visibility effectively produces consistent world normals, with visible normals being consistent and non-visible ones contributing minimally to the rendering. Regularization of Spherical Harmonics $\mathbf{d}_k$ is dependent on gaussian normals. Since normals are rotated to always face the camera, to maintain alignment between each Gaussian's normal and its associated $\mathbf{d}_k$, we also rotate $\mathbf{d}_k$ accordingly.

4.4 Experiments

4.4.1 Datasets and baselines

To evaluate our approach, we followed the protocol from NeRF-OSR [135] using ground truth environment maps. We use the official data split for Staatstheatert, Landwehrplatz, and Ludwigskirche. We use segmentation masks for test images provided in the OSR dataset and calculate MSE, MAE, SSIM and PSNR on masked regions only. We compare LumiGauss against several NeRF-based baselines[3] and TensoRF baseline.

Occluders. To exclude occluders from training images we use masks provided with OSR dataset [135].

[3] We include the concurrent NeuSky [43] which has been published officially after the WACV deadline.

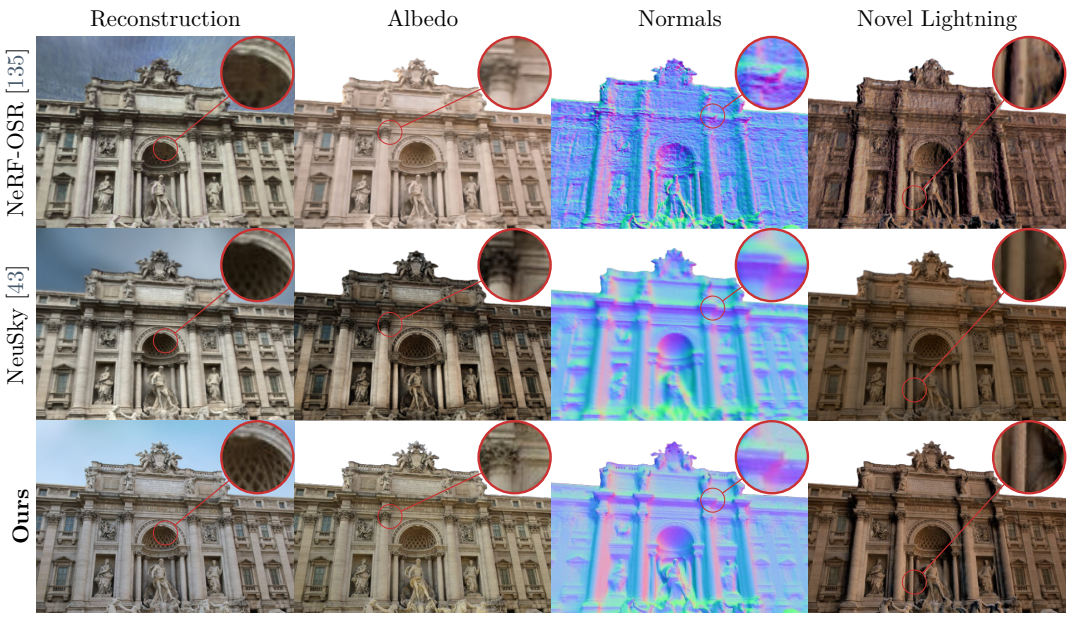


Figure 31. Qualitative comparison of albedo, normals, and relighting under similar lighting conditions on Trevi Fountain. LumiGauss produces albedo with fewer baked-in shadows, sharp normals, smooth surfaces, and more accurate novel lighting compared to the baselines. Results for NeuSky originally reported in [43]. Please, zoom in for details.

Test set. We test our approach on 5 viewpoints for each scene, as it was originally proposed in [135]. For testing, we use test masks provided by [135] and we stricly follow their evaluation protocol. For SSIM, we report the average value over the segmentation mask, utilizing the scikit-image implementation with a window size of 5 and eroding the segmentation mask by the same window size to exclude the influence of pixels outside the mask on the metric value.

Testing with ground truth environment map. The authors of [43] made an effort to recover steps for environment map preprocessing and alignment. The preprocessing step is available in their repository, accessible at this link. The detailed discussion on SOL-NeRF [152] approach to environment map alignment is included in the NeuSky main paper [43] and also confirmed with SOL-NeRF authors.

4.4.2 Scene reconstruction and relighting

We present the qualitative results in Tab. 10 and quantitative in Figs. 30 and 31. As Yu *et al.* [206] evaluates their model on downsampled images, we show the metric values on downsampled (d/s), and upsampled (u/s) to identify the quality differences. As we can see, LumiGauss performs better or on par with the baselines. NeuSky [43] is a concurrent work which models the environment maps and the sky using a prior, pretrained model.

As our backbone, 2DGS [57] incorporates priors to produce sharp edges and smooth surfaces, our model inherently performs better as expressed by SSIM. Please also see the zoom-ins in Fig. 31. Those shape reconstruction qualities allow us to relight the scene with high fidelity.

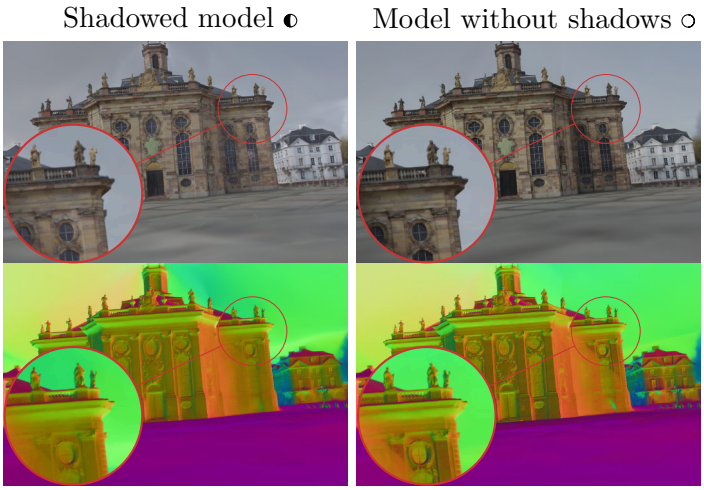


Figure 32. Effects of shadowed training – We show the comparison of **albedo** between the shadowed (*left*) and unshadowed (*right*) models. The albedo in the shadowed training is brighter with fewer shadows. The shadowed model recovers more accurate normals.

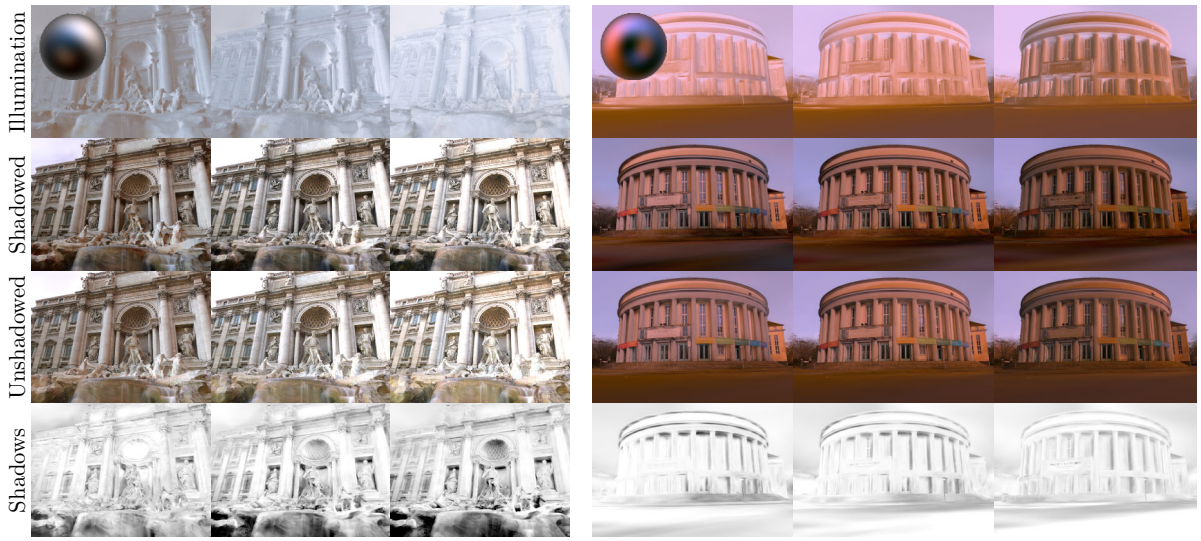


Figure 33. Environment map rotation – The top row shows the illumination entering the scene. The second and third rows display the shadowed and unshadowed renderings, respectively. The last row represents the approximate predicted shadows. Please zoom in for details.

We demonstrate that in Fig. 33 where one can see that our method effectively relights landmarks under various lighting conditions. We finally visualize the rendered shadows produced thanks to our proposed physical constraints at training time. Since LumiGauss does not predict shadows explicitly, we visualize them as grayscaled difference of output irradiances between the *unshadowed* (Eq. (45)) and *shadowed* (Eq. (46)) to approximate shadow effects:

$$\max(\mathfrak{g}(\mathbf{c}_k \oslash \boldsymbol{\rho}_k - \tilde{\mathbf{c}}_k \oslash \boldsymbol{\rho}_k), 0), \tag{54}$$

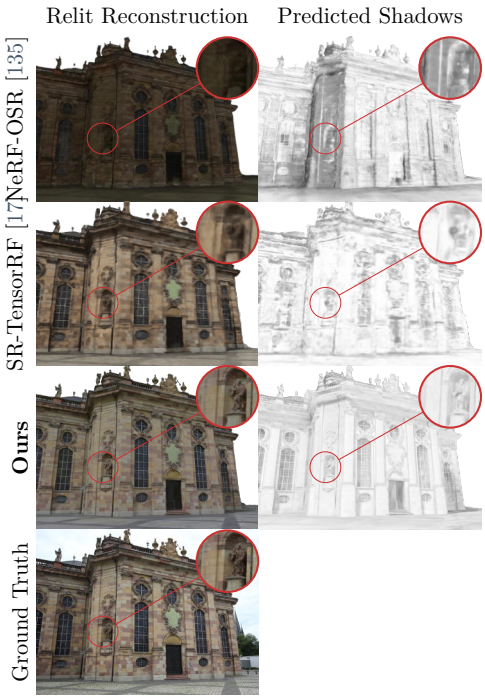


Figure 34. Qualitative comparison of scene reconstruction for the selected photo session – Results for NeRF-OSR [135] and LumiGauss were generated using ground truth enviroment maps for selected photo session, while ST-TensoRF [17] used extracted timestamp. Results for NeRF-OSR and SR-TensorF reported originally in [17].

where $\oslash$ is an element-wise division and $\mathfrak{g}(\cdot)$ converts from the RGB space to the grayscale space.

We also display the illumination in Fig. 33 to differentiate between shadows and dark illumination from the environment map.

Additionally, in Fig. 34 we show the qualitative comparison of our method, NeRF-OSR, and SR-TensoRF. We show the landmark relit with ground truth envoronment map for NeRF-OSR and LumiGauss. SR-TensoRF reconstructs ground truth using only daytime (timestamp).

In Fig. 36, we show the qualitative comparison of our method, NeRF-OSR, and SR-TensoRF. We use the *default synthetic* environment map provided by [135]. This environment map was used for visualisation purposes in [17]. We use it to ensure a fair comparison and consistency with results from concurrent works. We also present albedo and normals extracted from the reconstructed scene. Please note that our model produces much cleaner results. Compared to the baselines, it reconstructs sharp features in small elements of the buildings, which is also reflected in the quantitative results Tab. 10. LumiGauss also gracefully smooths out the elements of scenes that are variable across the images, such as trees and clouds. On the other hand, NeRF-OSR and SR-TensoRF produce artifacts that negatively impact the output reconstructions.

In Fig. 35 we present additional comparison with concurrent works. We focus on normal and albedo quality.

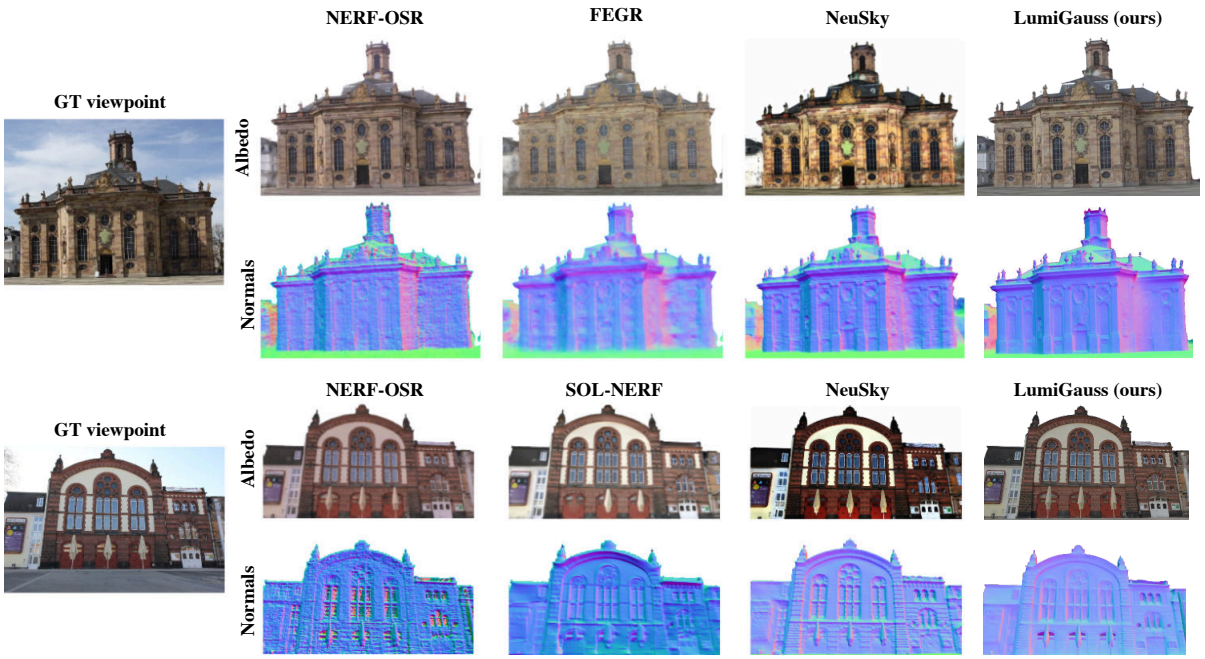


Figure 35. Qualitative comparison of predicted albedo and rendered normals. Results for NeRF-OSR, FEGR, SOL-NERF, NeuSky reported originally in [43].

In Fig. 38 we present additional results of novel view synthesis and comparison with concurrent works. Similarly to NeRF-OSR, we relight our scenes with the **default synthetic** map provided by NeRF-OSR for visualization purposes. This environment map does not correspond to GT images.

4.4.3 Ablations

We prioritize enhancing relighting capabilities over accurate appearance recreation during the optimization process, contrasting with recent Gaussian splatting methods that target novel view synthesis based on unconstrained photo collections [25, 192, 210]. Consequently, our ablation study primarily focuses on the degradation of relighting capabilities when removing any of the proposed components. We compare shadowed and unshadowed modeling and investigate the contributions of each loss term. We present the results in Tab. 10.

Gaussians can optimize to shadowed surfaces and represent shadows as normals and albedo colors (effect known as albedo/illumination ambiguity). Therefore, gains from separating shadows from lightning are not visible in metrics computed on a limited data subset. We noticed that adding a shadowed version can help restore proper albedo and normal vectors of surfaces that during the training were distorted or had low brightness (see Fig. 32).

In Fig. 37 we present renders from training without selected regularization terms.

Figure 36. Qualitative results – Showcase of novel view synthesis using shadowed radiance transfer. We present albedo and normals produced by our method. Our method generates much sharper renderings. Please see zoom-ins to see details on the quality difference, such as surface smoothness and edge sharpness of small building elements. We use visual results for SR-TensoRF and NeRF-OSR presented originally in [17]. Please note that, in this comparison the environment map used to create renders for NeRF-OSR and LumiGauss **does not match** the illumination in ground truth. LumiGauss and NeRF-OSR employ the **default** environment map provided by NeRF-OSR **for clear visualisation purpose only**. SR-TensoRF do not rely on any environment map, instead it utilizes daytime information.

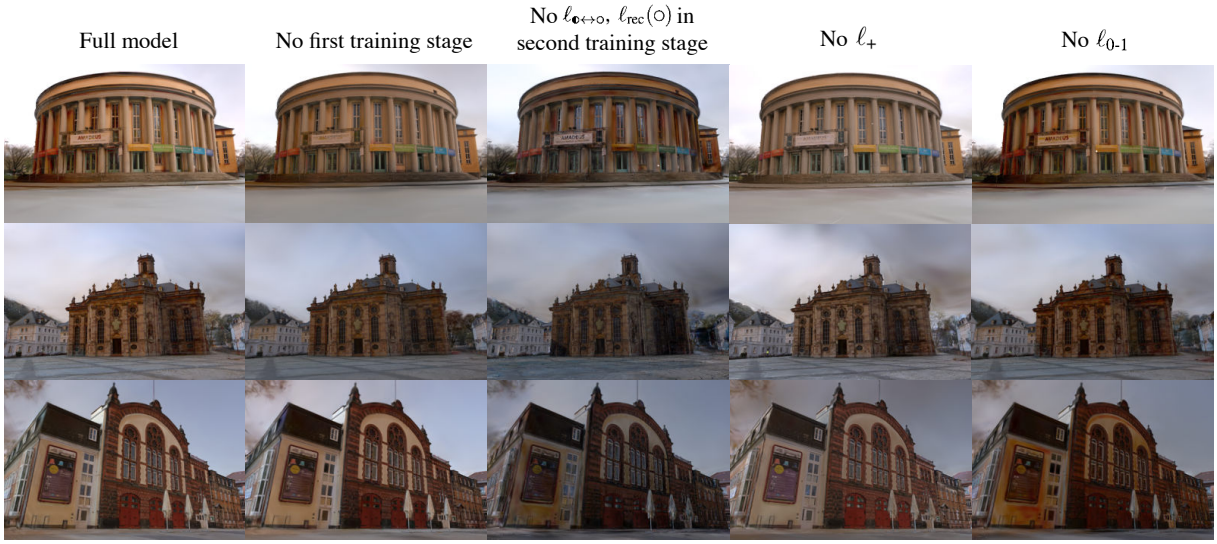


Figure 37. Ablation study for relightning with external environment map – The full model results in the clearest render. The strongest quality drop is observed when components restricting D_k are omitted.

4.4.4 Performance comparison

We compare LumiGauss' efficiency with two NeRF baselines. Our method achieves plausible relightning results while being orders of magnitudes faster both in terms of training and inference as shown in Tab. 11.

Figure 38. Qualitative comparison – Additional novel viewpoints. Results for NeRF-OSR and SR-TensoRF originally reported in [17]. Please note, in this comparison renders for NeRF-OSR and LumiGauss **do not have to** reconstruct ground truth.

4.4.5 Limitations

We identify the following limitations of our approach. Notably, surface albedo and normals may attempt to simulate shadows in scenarios with hard and frequent shadows. This can pose challenges for shadow training, especially when shadows are visible in several training images, potentially hindering the accurate representation of surface normals. Incorporating priors for environment light and shadowing could further enhance disentanglement and light transport modeling as presented in the concurrent NeuSky [43]. While we assume diffuse albedo, valid for most outdoor cases, shadows can appear unnaturally on reflective surfaces such as windows. Separate background optimization could enhance the synthesis of scenes with extensive sky areas. Finally, our shadow modeling baked-in the spherical harmonics representations is non trivial to extend to dynamic applications, such as autonomous driving.

4.5 Conclusions

We present LumiGauss—the method capable of decoupling environment lighting and albedo of objects from images *in-the-wild*. To this end, we apply 2DGS [57] to reconstruct the object's surface accurately and then use our proposed training components that correctly disentangle light properties from the rendered colors. As we show in the experiments, our approach achieves better reconstruction results than the baselines. We also present that one of our contributions—modeling shadows via leveraging Spherical Harmonics properties—provides shadows of high fidelity that react appropriately to changing environment light. LumiGauss is a novel approach in the direction of inverting the rendering process from images *in-the-wild*, reconstructing high-quality scene properties without sacrificing the fidelity of the output.

Chapter 5

CLoG: Leveraging UV Space for Continuous Levels of Detail

Gaussian Splatting has become popular thanks to its ability to render high-quality novel views from casually captured videos efficiently in real time. However, to achieve high-quality renderings, many Gaussians are used, which limits the devices that can render these representations. In this chapter, we introduce a method to dynamically change the number of Gaussians used on demand, according to where the representation is deployed and how many Gaussians can be supported. Importantly, we achieve this with a single representation that is trained only once—a representation that embeds a continuous notion of levels of detail. Specifically, we train a 2D-to-3D mapping—a multi-channel 2D image—of 3D Gaussian features, which is then interpreted by a small Multi-Layer Perceptron (MLP) to the characteristics of the Gaussian, *i.e.*, center, covariance, opacity, and color. When training, we make sure that downsampled versions of this image also render the scene well by introducing a modulator that adjusts the 2D map of features after downsampling. Thus, at inference time, this 2D image can be downsampled to any resolution of choice, rendering any number of 3D Gaussians. We demonstrate the performance of our method on NeRF [104], DTU [1] and AVA [100] at with various number of Gaussians, outperforming the state-of-the-art 3D Gaussian Splatting representations that support levels of detail.

5.1 Introduction

Since the introduction of Neural Radiance Fields (NeRF) [104], 3D computer vision, especially image-based reconstruction of 3D scenes, has seen a massive improvement in the quality of novel-view renderings. NeRFs store the radiance values of a scene within a neural representation such as Multi-Layer Perceptrons (MLP) [104] and hash grids [109], and render them via volume rendering.

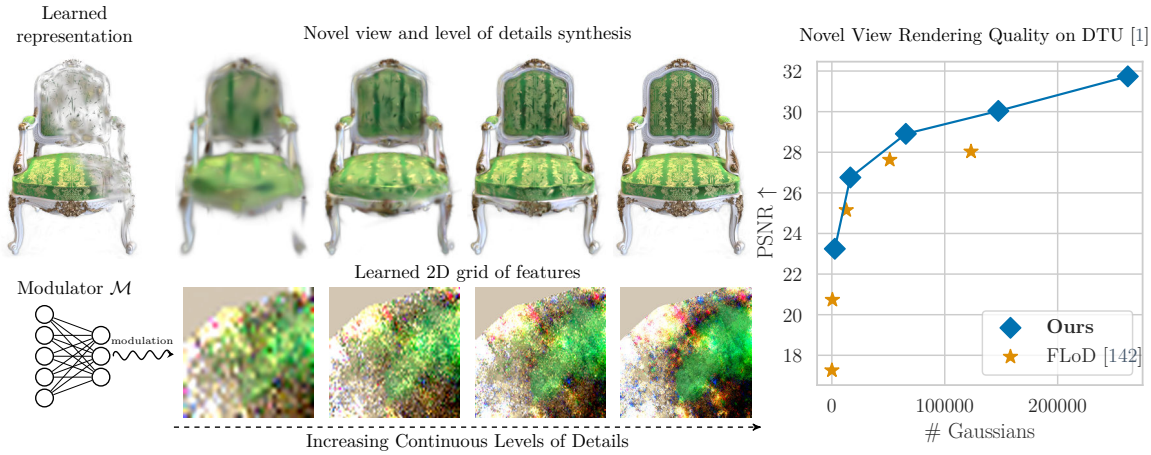


Figure 39. Teaser – We present a novel method to introduce continuous levels of detail (LoD) for Gaussian Splatting. We represent Gaussians as a set of vectors $\mathbf{\Lambda}$ embedded onto a 2D grid, which we can easily subsample to obtain a desired level of detail. To account for any artifacts that can occur during downsampling, we train a modulator $\mathcal{M}$ that corrects such errors. Our method provides a continuous change of the LoD and provides the best trade-off between the number of Gaussians and rendering quality.

More recently, 3D Gaussian Splatting [71] has become arguably one of the de-facto standards when it comes to storing radiance values, which are then *rasterized*, *i.e.*, "splatted", to images instead of using volume rendering. While 3D Gaussian Splatting offers faster rendering with enhanced rendering quality, it comes at the cost of requiring millions of Gaussian primitives to be trained. Researchers have thus attempted to reduce the number of Gaussian primitives, either via finding more compact configurations that allow similar rendering quality [32, 79, 111, 112], or concurrently to our work, via introducing Levels of Detail (LoD) [142, 144]. The former approach maintains rendering quality but is constrained by its fixed number of Gaussians during training, preventing dynamic adaptation to the budget available for deployment—they also typically still require many Gaussians. The latter resolves this issue, but these concurrent solutions only support a discrete, predefined set of levels, limiting their application, for example to continuous levels of details used in gaming [170] or streaming.

In this chapter, we present Continuous Level of Gaussians (CLoG), a novel method that introduces continuous levels of detail to 3D Gaussian Splatting. Our key idea is to learn a mapping [36] of 3D Gaussians, representing them as uniform 2D grid of points akin to pixels. Specifically, we create a 2D image of trainable feature representations, which are then interpreted by a light-weight Multi-Layer Perceptron (MLP) to the center, variance, color, and opacity of Gaussians. The learnable representation enables continuous level-of-detail control through direct resampling of the 2D feature map to the desired level of detail. Although the 2D maps are high-dimensional features, simply downsampling them does not yield renderings of high-quality. To address this limitation, we introduce a modulator MLP that dynamically alters the downsampled feature values based on the target level of detail, ensuring high-quality rendering.

We train CLoG in two stages. In the first stage, we jointly train the Gaussians, the 2D feature image and the interpretive MLP, by progressively growing the 2D image to the highest resolution. After training, because we rely on downsampling, which assumes nearby samples are alike, we sort [107] the 2D image such that Gaussians with similar properties are nearby. Now, with the highest resolution being pre-trained and fixed, we train the modulator by first subsampling the 2D image, with appropriate blurring to a desired randomly chosen level of detail, then applying the modulator and using the modulated 2D maps to render the scene. We then optimize the modulator weights so that this rendering becomes faithful to the training images.

Our evaluations show that CLoG maintains rendering quality comparable to baseline methods at their highest resolution, and is able to outperform alternative representations that provide LoD.

In summary, our contributions are:

- We introduce Continuous Levels of Gaussians CLoG, a novel method that enables continuous levels of detail for 3D Gaussian Splatting.
- Our key idea lies in leveraging 2D continuous maps that support efficient subsampling for level-of-detail control.
- We develop a progressive training strategy that grows 2D maps from noise, followed by spatial sorting for optimal structure. Additionally, we introduce a level-of-detail-conditioned modulator that refines downsampled 2D maps for high-quality rendering.
- We demonstrate how our method dynamically adapts the underlying representation to accommodate the desired compute and memory budget.

5.2 Related Work

We first briefly discuss work on NeRFs [104] and 3D Gaussian Splatting [71], then discuss works that focus on levels of details.

Neural Radiance Fields. Neural Radiance Fields (NeRFs) [104] were introduced as a way to incorporate volume rendering within a neural rendering pipeline. They use a Multi-Layer Perceptron (MLP) to encode the radiance field values at each point in the modeling space, which is then volume rendered to images. Because this volume rendering process is differentiable, given a set of images of the same scene with known camera poses, NeRFs can be trained via back-propagating through the rendering process for a faithful reconstruction of the training images.

A core limitation of NeRFs is their required compute, which easily took multiple days on high-end GPUs for optimal quality. Follow-up works thus proposed various ways, using small MLPs divided into various regions [42], using explicit octree representations [37, 203], and multi-level hashgrids [109] that eventually allowed NeRFs to run in real-time. Further enhancements, for example using quantized tables [158] were also suggested. Interestingly, this further allowed

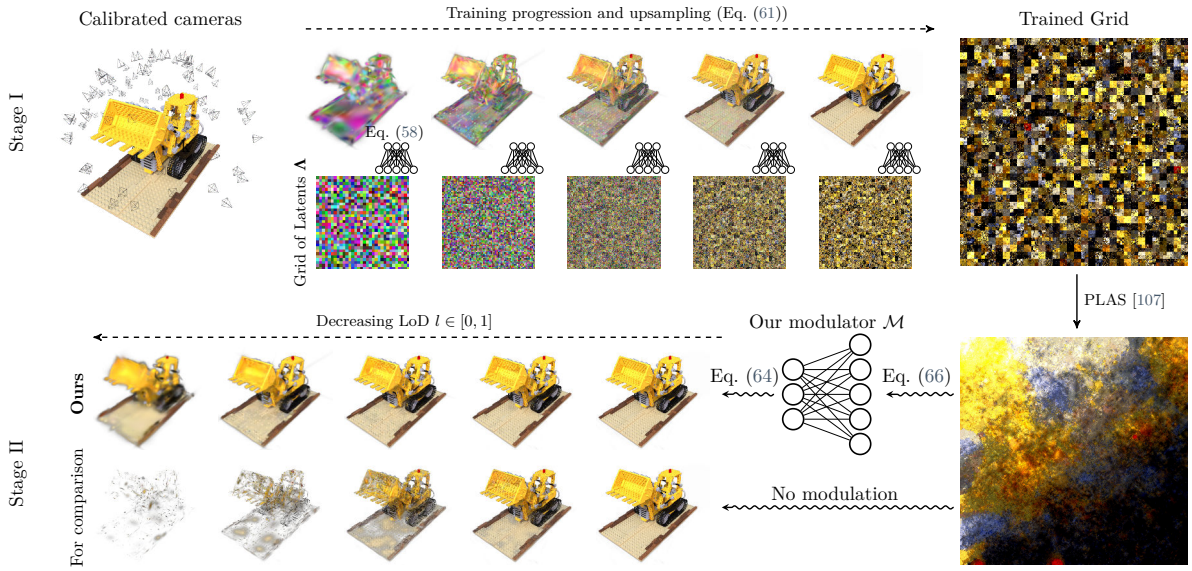


Figure 40. Pipeline – We represent the scene with a 2D UV map containing features that an MLP interprets 3D Gaussians. We first train the UV map at the highest resolution and sort it to ensure spatial coherence of similar features, enabling effective subsampling. For any desired level of detail (LoD), we obtain the corresponding UV map by downsampling and processing it through a modulator that adapts the features to map correctly to 3D Gaussians at reduced density. This approach enables support for *arbitrary* levels of detail.

NeRF representations to be 'streamed' at desired resolutions [29, 131], effectively providing levels-of-detail, similarly to how Neural Geometric Level of Detail [159] provided LoDs for neural 3D shape representations.

Gaussian Splatting. Even with a hash grid-based implementation [109], the quality of NeRF renderings highly depends on how points are sampled along the light ray. On the other hand, 3D Gaussian Splatting (3DGS) [71] allows a deterministic rendering process that does not require this sampling process. Rather, they represent the radiance values of a 3D scene via 3D Gaussians, that are then *rasterized* to images efficiently and differentiably. 3DGS thus provide super real-time rendering, much faster than the fastest NeRF methods. Since the inception of 3DGS, much progress has been made to improve its rendering quality [14, 44, 57, 63, 72, 91, 126, 138, 207], to distill Gaussians into 3D voxels [91, 95, 133] for efficient rendering, or to limit the number of Gaussians used [32, 33, 79, 107, 111, 112]. However, these methods focus on creating a 'fixed' compact representation for each scene, and always aim for maximum rendering quality—they do not have a notion of levels of detail.

Levels of Detail. Levels of Detail is a well-known technique in computer graphics for adapting a target 3D object (usually represented as mesh) to a given compute budget [21, 96, 113]. In the gaming industry, it is commonly used to replace complex 3D objects with simpler representations depending on the distance from the player's view. This method reduces the overall scene

complexity without sacrificing its fidelity. Additionally, they can be used to deliver the object at multiple compute budgets.

In the realm of neural representations, NGLOD [159] proposed to produce a signed distance field in a continuous matter. In a similar spirit, Variable Bitrate Neural Fields [158] quantize the latent vectors of the radiance field to enable field streaming. These methods, however, are not trivial to extend to 3DGS, which optimizes each of the Gaussians independently and uses non-differentiable heuristics to minimize the reconstruction error.

Octree-GS [133] introduced levels of detail with Gaussian splatting by learning an adaptive partition of 3D space (an octree) that naturally induces a hierarchy of details into the scene. The method is closely related to Scaffold-GS [95], as they both start from points obtained from Structure-from-Motion [140] for their representation, but expands their view-dependent rendering to select anchors and Gaussian parameter offsets from multiple LoDs during inference. A recent method, FLoD [142], introduces a less structured approach that simply trains using multiple stages, where each stage is dedicated to a 'level' of detail, and optionally selects a subset of levels at runtime. During training, FLoD constrains the scale of the Gaussians at each level, effectively using large Gaussians for coarser levels, later learning fine-grained details with smaller Gaussians. While these strategies do allow for levels of detail with 3DGS, they only do so at fixed pre-defined intervals, limiting their applicability. On the other hand, CLoG is capable of continuously change the levels of details at inference time.

In our work, we draw inspiration from Mixed Volumetric Primitives (MVP) [93, 127] and Relightable Gaussian Codec Avatars (RGCA) [136] and approach the problem of 3D reconstruction from the perspective of a deterministic, pixel-aligned 2D to 3D mapping. Our key idea is that once this 2D-to-3D mapping is learned, we can subsample the 2D image to create a continuous LoD. Using 2D images as a structure for 3D Gaussians was also explored in TextureGS [194], where this idea was used to disentangle texture from appearance post-hoc scene editability. Here, we show that such a structure can be repurposed for different ends—it can be used to group Gaussians together, creating a representation that amenable to continuous levels of detail for 3D Gaussian Splatting.

5.3 Method

In this section, we briefly review Gaussian Splatting and then explain our method—Continuous Levels of Gaussians.

5.3.1 Preliminary: 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [71] is a volumetric model $\mathcal{G}=\{\boldsymbol{\mu}_i, \boldsymbol{q}_i, \boldsymbol{s}_i, o_i, \boldsymbol{c}_i\}_{i=1}^{G}$ parametrized as means $\boldsymbol{\mu} \in \mathbb{R}^3$, rotations as quaternions $\boldsymbol{q} \in \mathbb{R}^4$, scales $\boldsymbol{s} \in \mathbb{R}^3$, opacities $o \in [0, 1]$, and colors[1]

[1] In practice, we follow the original 3DGS implementation and predict Spherical Harmonics coefficients but we keep describing colors directly for simplicity.

$\boldsymbol{c} \in \mathbb{R}^3$ from a set of calibrated images $\{\mathbf{C}\}_{c\sim\mathcal{C}}$ with camera parameters $\mathcal{C}$. Kerbl et al. [71] use properties of splatting the 3D Gaussians on the image space introduced by Zwicker et al. [225]. 3DGS applies a rendering operator $\mathcal{R}(\cdot)$ which takes the set of Gaussian parameters $\mathcal{G}$ and renders an image $\hat{\mathcal{I}}$ from a novel viewpoint $\hat{c}$. The rendering operator is realized as an alpha-blending from traditional volume rendering:

$$\hat{\mathcal{I}}_{\hat{c}} = \mathcal{R}(\mathcal{G}, \hat{c}) = \sum_{i=1}^{N} \boldsymbol{c}_i o_i \prod_{j=1}^{i-1}(1 - o_j), \tag{55}$$

To efficiently render novel images, 3DGS uses a GPU-based implementation of tiling and sorting to obtain an ordered set of splatted 2D Gaussians, leveraging the identity for the i-th Gaussian:

$$\Sigma_i = r(\boldsymbol{q}_i)\,\mathrm{diag}(\boldsymbol{s}_i)\,\mathrm{diag}(\boldsymbol{s}_i)^\top r(\boldsymbol{q}_i)^\top,$$

where $\Sigma \in \mathbb{R}^{3\times3}$ is the covariance matrix of the Gaussian, $r(\cdot) : \mathbb{R}^4 \mapsto \mathbb{R}^{3\times3}$ converts a quaternion $\boldsymbol{q}$ into a rotation matrix, and $\mathrm{diag}(\cdot) : \mathbb{R}^3 \mapsto \mathbb{R}^{3\times3}$ diagonalizes a vector of scales $\boldsymbol{s}$. The Gaussian parameters are optimized by minimizing the reconstruction loss:

$$\ell_{\text{rec}} = (1 - \alpha_{\text{rec}})\ell_1 + \alpha_{\text{rec}}\ell_{\text{D-SSIM}}, \tag{56}$$

where ℓ_1 is the per-pixel L_1 loss, $\ell_{\text{D-SSIM}}$ is a differentiable SSIM [71] and α_{rec} is a loss weighting term. Please refer to the work of Kerbl et al. [71] for an in-depth derivation of the rendering equation used in 3DGS.

5.3.2 Continuous Level of Gaussians

We seek a set of 3D Gaussian parameters $\mathcal{G}$ that are able to reconstruct the input images under a given level-of-detail factor $l \in [0, 1]$ [159]. Unlike previous work, which usually refers to the level of detail in frequency octaves of the input data [133, 142], we allow the model to take any continuous value in the given range.

We represent the 3D Gaussians on a 2D grid as evenly spaced points (pixels). This allows us to leverage texturing techniques, such as learnable *mipmapping*, to adjust the resolution of the grid and, therefore, the number of output Gaussians. Specifically, as shown in Fig. 40, we train CLoG to represent the scene by decoding 3D Gaussians from latent features $\boldsymbol{\Lambda}$ placed on a 2D map with associated learnable 3D coordinates $\boldsymbol{x}^{\mathsf{3D}}$ in an auto-decoding fashion [12].

Additionally, as we use downsampled 2D map to enable LoDs, we use a learnable modulator $\mathcal{M}$ that uses the Gaussian parameters $\mathcal{G} = \{\boldsymbol{\Lambda}, \boldsymbol{x}^{\mathsf{3D}}\}$ and modulates them into $\tilde{\mathcal{G}} = \{\tilde{\boldsymbol{\Lambda}}, \boldsymbol{x}^{\mathsf{3D}}\}$. The primary role of this modulator is to compensate for the fewer number of Gaussians after downsampling. The final output image is rendered via the alpha-blending

from Eq. (55) by applying the operator:

$$\hat{\mathcal{I}}_{\hat{c}} = \mathcal{R}\left(\{\tilde{\boldsymbol{\Lambda}}, \boldsymbol{x}^{\mathsf{3D}}\}, \hat{c}\right). \tag{57}$$

Our training process consists of two stages. First, we optimize the 3D Gaussians using the highest-resolution UV map to achieve optimal reconstruction quality. Subsequently, we fix the high-resolution UV map and train the modulator to enable effective level-of-detail control, focusing on maintaining quality at lower resolutions. We describe each stage in detail in the following subsections.

5.3.3 Stage I – Training the Gaussian Representation

We parametrize our Gaussians as learnable D-dimensional latents $\boldsymbol{\Lambda} \in \mathbb{R}^{(W\times H)\times D}$, embedded on a 2D (image) plane of $W \times H$ dimensions, and the corresponding 3D coordinates $\boldsymbol{x}^{\mathsf{3D}} \in \mathbb{R}^3$. We decode $\boldsymbol{\Lambda}$ into Gaussian attributes $\{\boldsymbol{\mu}_i, \boldsymbol{q}_i, \boldsymbol{s}_i, o_i, \boldsymbol{c}_i\}_{i=1}^{G}$ with an MLP, *i.e.*, those attributes become functions of the latent vectors $\boldsymbol{\lambda}_i \in \boldsymbol{\Lambda}$:

$$\begin{aligned}
\boldsymbol{\mu}_i(\boldsymbol{\lambda}_i) &= \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\boldsymbol{\Theta}), \quad \boldsymbol{q}_i(\boldsymbol{\lambda}_i) = \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\boldsymbol{\Theta}),\\
\boldsymbol{s}_i(\boldsymbol{\lambda}_i) &= \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\boldsymbol{\Theta}), \quad o_i(\boldsymbol{\lambda}_i) = \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\boldsymbol{\Theta}),\\
\boldsymbol{c}_i(\boldsymbol{\lambda}_i) &= \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\boldsymbol{\Theta}).
\end{aligned} \tag{58}$$

At this stage, we are treating our 2D image plane as a hash table for 3D Gaussians with the number of entries equal to the number of Gaussians, akin to DINER [190]. We optimize $\{\boldsymbol{\Lambda}, \boldsymbol{\Theta}, \boldsymbol{x}^{\mathsf{3D}}\}$ by minimizing the objective:

$$\underset{\{\boldsymbol{\Lambda},\boldsymbol{\Theta},\boldsymbol{x}^{\mathsf{3D}}\}}{\arg\min}\ \ell_{\mathrm{rec}}(\boldsymbol{\Lambda}, \boldsymbol{\Theta}) + \Omega(\boldsymbol{\Lambda}, \boldsymbol{\Theta}), \tag{59}$$

where ℓ_{rec} deals with measuring the per-pixel discrepancy between the rendered and ground truth images, while $\Omega(\cdot)$ is a regularizer imposed on Gaussian descriptors $\boldsymbol{\Lambda}$ and parameters of the MLPs $\boldsymbol{\Theta}$, that regularizes the scale and the opacity of Gaussians. Specifically, we minimize $\|o_i(\boldsymbol{\lambda}_i)\|_1$ so that Gaussians are encouraged to either be completely opaque or transparent, and $\|\boldsymbol{s}_i(\boldsymbol{\lambda}_i)\|_2^2$ of the Gaussians to keep them small.

Initialization and exploration. One downside of such representation is that it is then non-trivial to initialize 3D Gaussians, *e.g.*, using Structure-from-Motion (SfM) points as in other 3DGS approaches [71, 95, 133, 138, 142], as an initial bijective mapping needs to be set. Instead, we opt to simply sample the $\boldsymbol{x}^{\mathsf{3D}}$ coordinates from a uniform distribution $\boldsymbol{x}^{\mathsf{3D}} \sim \mathcal{U}(-\epsilon, \epsilon)$.

Following Kheradmand et al. [73], we introduce controlled noise to transparent Gaussians to encourage exploration, effectively eliminating the performance gap while maintaining training stability. We apply normally-distributed noise $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to Gaussians centers during

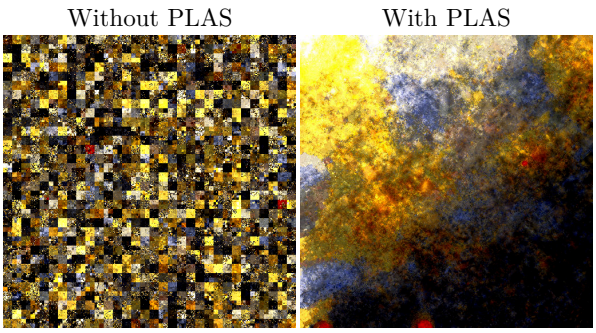


Figure 41. Effect of using PLAS [107] – We demonstrate the effect of sorting by visualizing the decoded Gaussian colors in the UV map, for the `lego` sequence from the NeRF Synthetic [104] dataset. After sorting, the UV map has smoother color transitions, confirming the successful spatial clustering of similar Gaussians.

optimization as a function of their covariances [73]:

$$\tilde{\boldsymbol{\mu}}' = \alpha_{\text{lr}} \cdot \sigma\left(-k(o-t)\right) \cdot \Sigma\boldsymbol{\eta}, \tag{60}$$

where we adopt the hyperparameters from Kheradmand *et al.* [73].

Growing. 3DGS [71] employs a gradual scene population strategy by selectively duplicating Gaussians deemed more "useful" based on heuristics. This approach prevents computational and memory waste by avoiding Gaussian placement in less significant regions. In our framework, this process naturally translates to the "upsampling" operations.

We initialize training with a grid of dimensions $W' \times H'$ where $W' \ll W$ and $H' \ll H$. At fixed intervals of k steps during training, we double the grid resolution until reaching the target dimensions $W \times H$. This progressive growth follows:

$$\boldsymbol{x}^{\text{3D}} \leftarrow \text{upsample}(\boldsymbol{x}^{\text{3D}},\ 2) \quad \boldsymbol{\Lambda} \leftarrow \text{upsample}(\boldsymbol{\Lambda},\ 2), \tag{61}$$

where upsample$(\cdot)$ is the upsampling operator, which ensures that new Gaussians inherit features and coordinates from their parents.

Pruning (relocating). While 3DGS [71] removes low-opacity Gaussians due to their minimal rendering contribution, we adopt the relocation strategy of Kheradmand et al. [73]. We replace Gaussians with opacities $o < \tau_o$ by cloning more effective ones. Following Bulò et al. [14], to allocate computational budget to where it would be most useful, we look at how much each Gaussian is contributing to reconstruction error e and clone those that exceed a threshold $e > \tau_e$. We compute each Gaussian's error e as its contribution to the overall $\ell_{\text{D-SSIM}}$ loss.

Sorting. After our 2D representation is trained, as mentioned in Sec. 5.3.2, we downsample it to generate 2D maps for lower LoDs. This process requires neighboring features to exhibit

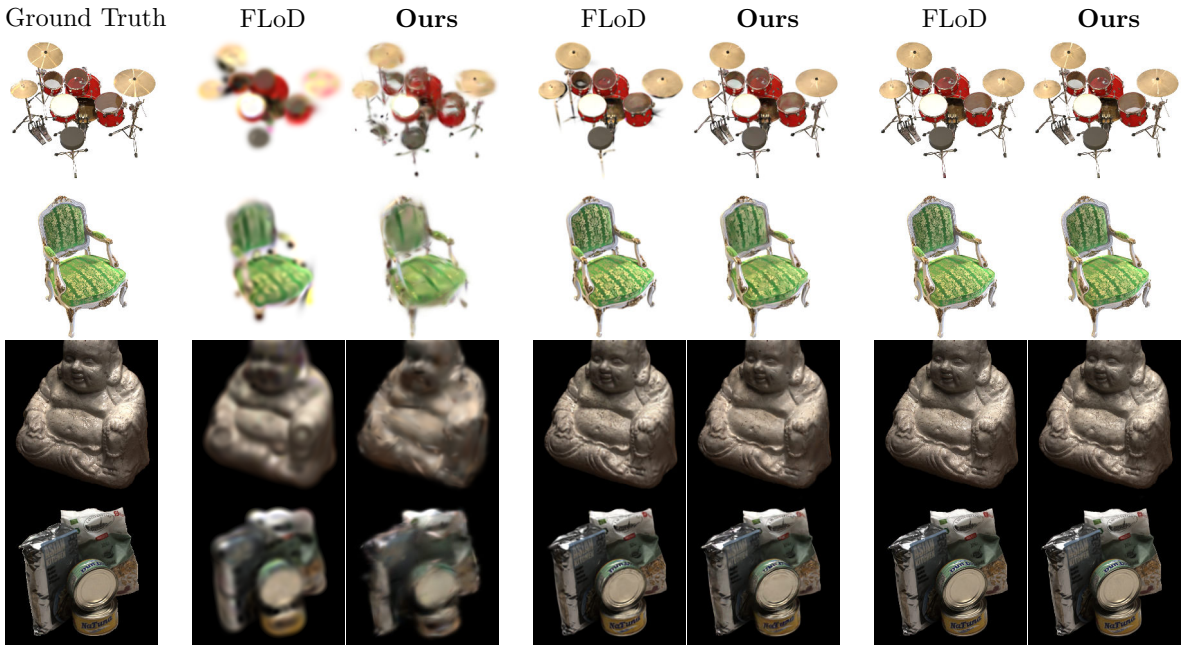


Figure 42. Qualitative highlight (varying levels of detail) – We provide example renderings for various levels of detail, denoted by the number of Gaussians used. As shown, our method is able to provide high-quality reconstructions at various levels, and provides a better tradeoff than FLoD [142]. We showcase the results for $l \in \{0.1, 0.5, 1.0\}$ for our method, and LoD=$\{2, 3, 5\}$ for FLoD to roughly match the number of Gaussians.

similarity. To achieve this, we apply the PLAS algorithm [107] to latents $\mathbf{\Lambda}$. Notably, we perform this sorting only *once* at the end of Stage-I training, unlike in [107] who apply it multiple times during training, which increases computational overhead. Fig. 41 demonstrates PLAS's impact on the feature space by visualizing decoded Gaussian colors $\boldsymbol{c}$ at their corresponding pixel locations. The post-sorting visualization reveals smooth color transitions, indicating successful spatial coherence of similar features.

5.3.4 Stage II – Training the Modulator

With the highest-resolution UV map trained, we then downsample it to a given LoD l. We use l to first downsample our latents $\mathbf{\Lambda}$ and learned 3D coordinates $\boldsymbol{x}^{\mathsf{3D}}$ to the given resolution as:

$$\begin{aligned} \mathbf{\Lambda}_{\downarrow} &= \text{downsample}(\mathbf{\Lambda}, l), \\ \boldsymbol{x}^{\mathsf{3D}}_{\downarrow} &= \text{downsample}(\boldsymbol{x}^{\mathsf{3D}}, l), \end{aligned} \tag{62}$$

producing attributes $\mathbf{\Lambda}_{\downarrow}, \boldsymbol{x}^{\mathsf{3D}}_{\downarrow} \in \mathbb{R}^{(W_{\downarrow} \times H_{\downarrow}) \times D}$ where:

$$W_{\downarrow} = \lfloor W \cdot l \rfloor \quad H_{\downarrow} = \lfloor H \cdot l \rfloor. \tag{63}$$

From here on, we denote Gaussian attributes decoded from $\boldsymbol{\lambda}_\downarrow \in \boldsymbol{\Lambda}_\downarrow$ as $\{\boldsymbol{\mu}_\downarrow, \boldsymbol{q}_\downarrow, \boldsymbol{s}_\downarrow, o_\downarrow, \boldsymbol{c}_\downarrow\}$, where we drop the i-th index for brevity, and the resulting number of Gaussians is denoted as $W_\downarrow \times H_\downarrow = G_\downarrow$

Since using $l < 1$ produces a smaller number of Gaussians than those contained in the entire set ($G_\downarrow < G$), the Gaussian parameters $\boldsymbol{\Lambda}_\downarrow$ need to be refined to compensate for missing Gaussians. We introduce a parametrized latent modulator $\mathcal{M}$ with parameters $\boldsymbol{\Theta}^{\mathcal{M}}$, conditioned on the level of detail l that produces modulation vectors $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^D$. A single Gaussian descriptor $\boldsymbol{\lambda}$ is modulated then as:

$$\boldsymbol{\beta}(\boldsymbol{\lambda}), \boldsymbol{\gamma}(\boldsymbol{\lambda}) = \mathcal{M}(\boldsymbol{\lambda}_\downarrow, l\,;\, \boldsymbol{\Theta}^{\mathcal{M}}) \tag{64}$$

$$\tilde{\boldsymbol{\lambda}} = \sigma(\boldsymbol{\gamma}(\boldsymbol{\lambda})) \odot \boldsymbol{\lambda} + \boldsymbol{\beta}(\boldsymbol{\lambda}), \tag{65}$$

where $\sigma(\cdot)$ is a sigmoid activation function. Inspired by Continuous Upsampling Filters [169], we implement $\mathcal{M}$ as an MLP that takes a latent $\boldsymbol{\lambda}$ concatenated with its corresponding 2D coordinate $\boldsymbol{x}^{\mathsf{2D}} \in [0,1]^2$ and the LoD value l. We additionally pass $\boldsymbol{x}^{\mathsf{2D}}$ and l through a positional embedding function $f(\cdot)$ defined as in traditional NeRF approaches [104]. Overall, our $\mathcal{M}$ has the following form:

$$\mathcal{M}(\boldsymbol{\lambda}, l\,;\, \boldsymbol{\Theta}^{\mathcal{M}}) = \mathrm{MLP}(\boldsymbol{\lambda}, f(\boldsymbol{x}^{\mathsf{2D}}), f(l)\,;\, \boldsymbol{\Theta}^{\mathcal{M}}). \tag{66}$$

We optimize $\boldsymbol{\Theta}^{\mathcal{M}}$ by minimizing the reconstruction objective (as in Eq. (59)) and corresponding regularization terms:

$$\arg\min_{\boldsymbol{\Theta}^{\mathcal{M}}} \ell^{\mathcal{M}}_{\mathrm{rec}}(\boldsymbol{\Lambda}, \boldsymbol{\Theta}, \boldsymbol{\Theta}^{\mathcal{M}}) + \Omega^{\mathcal{M}}(\boldsymbol{\Lambda}, \boldsymbol{\Theta}^{\mathcal{M}}), \tag{67}$$

where $\Omega^{\mathcal{M}}$ is the modulator-specific regularization term where we minimize $\|1 - \boldsymbol{\beta}(\boldsymbol{\lambda})\|_2^2$ and $\|\boldsymbol{\gamma}(\boldsymbol{\lambda})\|_2^2$, such that the modulator is encouraged to perform an identity transform.

5.3.5 Implementation details

For the latents $\boldsymbol{\Lambda}$, we use the descriptor dimension D=16×6 meaning that each attribute consists of its own descriptor. We represent colors $\boldsymbol{c}$ as Spherical Harmonics of the second degree. For the decoding MLPs in Eq. (58), we use two layers, each with 64 neurons. For the modulator $\mathcal{M}$ we use an MLP with six layers, each with 256 neurons. We encode $\boldsymbol{x}^{\mathsf{2D}}$ using the hash grid from InstantNGP [109]. We use a hashgrid size of 2^{19} elements which holds 16 levels of 2-dimensional feature vectors. For the scale component we use frequency encodings [104]—we use 8 frequencies, increasing by a power of 2 starting from 1. Regarding hyperparemters, we set them identically for all experiments. We use a regularization strength of 10^{-2} for $\|o_i(\boldsymbol{\lambda}_i)\|_1$, 10^{-4} for $\|\boldsymbol{s}_i(\boldsymbol{\lambda}_i)\|_2^2$, 10^{-2} for both $\|1 - \boldsymbol{\beta}_i(\boldsymbol{\lambda}_i)\|_2^2$ and $\|\boldsymbol{\gamma}_i(\boldsymbol{\lambda}_i)\|_2^2$, all set empirically.

We train the representation starting with a 32^2 grid. We first warmup the training for 500 steps, then progressively grow it by doubling its size every 2000 steps. We train both stages

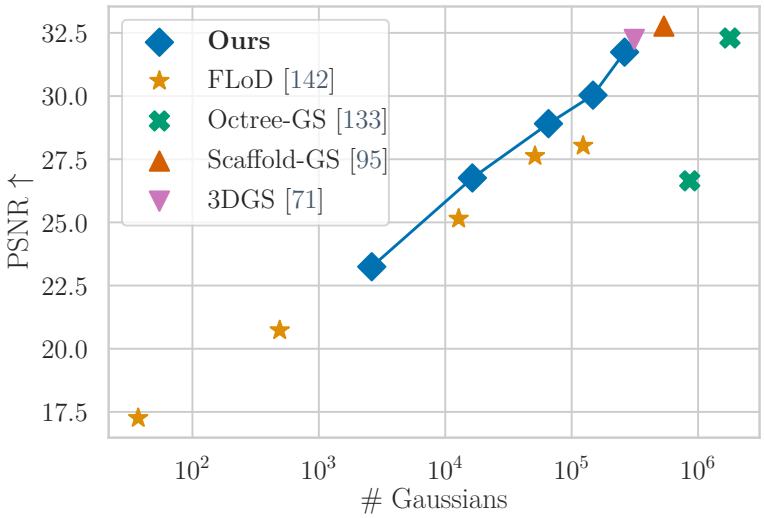


Figure 43. Rendering quality vs number of Gaussians for the DTU dataset [1] – We show the PSNR values for each method with respect to the number of Gaussians (level of detail). Our method is able to provide a continuous level of detail, and provides the best rendering quality for the same budget. Our finest level provides rendering quality that is on par with the traditional baselines.

Method	AVA [100]			Blender [104]			DTU [1]		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	22.840 ± 1.291	0.872 ± 0.030	0.175 ± 0.019	33.783 ± 3.692	0.970 ± 0.027	0.031 ± 0.031	31.758 ± 3.966	0.944 ± 0.030	0.063 ± 0.027
Octree-GS [133]	22.845 ± 1.229	0.871 ± 0.035	0.173 ± 0.020	32.268 ± 3.622	0.961 ± 0.028	0.042 ± 0.030	31.803 ± 3.751	0.941 ± 0.030	0.069 ± 0.027
Scaffold-GS [95]	18.664 ± 1.192	0.729 ± 0.047	0.364 ± 0.035	33.697 ± 3.630	0.968 ± 0.028	0.033 ± 0.031	32.224 ± 3.788	0.945 ± 0.027	0.065 ± 0.025
FLoD [142]	22.295 ± 1.465	0.871 ± 0.030	0.191 ± 0.020	29.480 ± 8.381	0.923 ± 0.087	0.075 ± 0.079	29.572 ± 7.597	0.897 ± 0.118	0.106 ± 0.089
Ours	27.896 ± 2.236	0.858 ± 0.047	0.209 ± 0.031	31.797 ± 3.341	0.963 ± 0.031	0.046 ± 0.041	30.727 ± 3.948	0.936 ± 0.035	0.085 ± 0.036

Table 12. Quantitative results (finest level of detail) – We report the rendering quality our method, at the finest level of detail, compared to other baselines. we provide comparable rendering quality, and additionally allow continuous levels of detail. Note that NeRF Synthetic [104] dataset results for FLoD [142] should be interpreted with caution as it performs excessively poorly in some sequences.

Mode	Method	LoD=1	LoD=2	LoD=3	LoD=4
Decoding	FLoD	1.091 ± 0.041	2.429 ± 0.080	6.500 ± 0.496	14.183 ± 1.246
	Ours	3.337 ± 0.106	16.415 ± 0.081	39.771 ± 0.152	62.275 ± 0.209
Inference	FLoD	2.317 ± 0.013	4.047 ± 0.015	5.887 ± 0.026	7.400 ± 0.056
	Ours	**1.228 ± 0.027**	**1.219 ± 0.015**	**1.231 ± 0.021**	**1.238 ± 0.049**

Table 13. Decoding and inference times – We present timing measurements in milliseconds averaged across all datasets for different LoDs. Decoding encompasses all steps required to obtain Gaussians' characteristics for splatting. Inference, on the other hand, measures the time needed to generate the final image. Faster times are in bold.

for 10^5 steps so the training takes 2×10^5 steps in total. We implement our method using the `gsplat` [202] library.

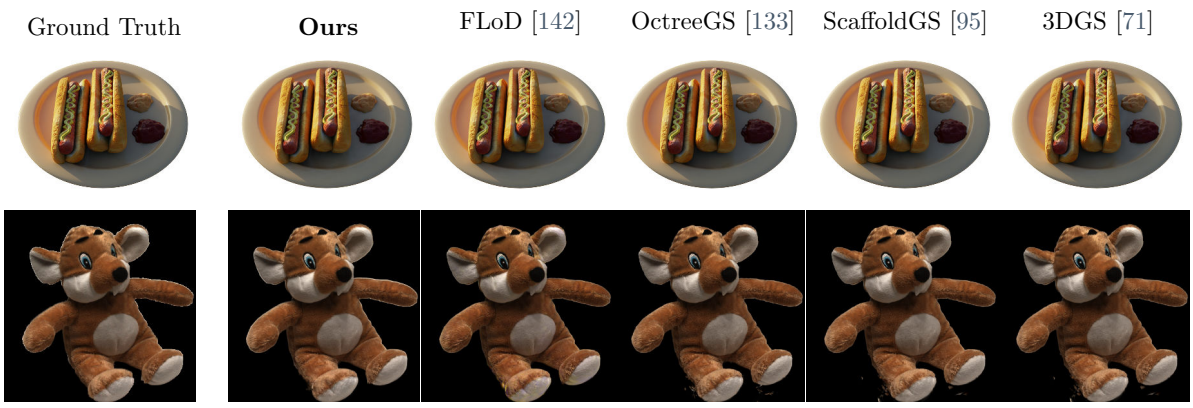


Figure 44. Qualitative examples (finest level of detail) – We show qualitative examples of our method, at the finest level of detail, compared to other baselines. Our method is able to provide comparable rendering quality at the finest level, and additionally allow continuous levels of detail as demonstrated in Fig. 42.

5.4 Results

We first discuss our evaluation setup, then present our results. We also provide ablation study on the design choices.

5.4.1 Experimental setup

Datasets. We evaluate our method on three different datasets. We use the NeRF Synthetic [104] as this dataset is commonly used in previous work. We also use the DTU [1] and BlendedMVS [201] datasets to evaluate performance on a non-synthetic dataset. Following the standard protocol [133], we use the provided foreground masks. Finally, we use the AVA [100] dataset to showcase that our model can be also applied when ground truth UV mapping is available.

Baselines. We evaluate method against other neural-based Gaussian Splatting, namely Scaffold-GS [95] and Octree-GS [133], and vanilla 3DGS [71]. Notably, Octree-GS implements a levels of detail mechanism which enables adapting the output to the available computational constraints. We additionally provide a comparison against FLoD [142], a concurrent method that trains several defined levels. For each baseline, we use default configurations provided by the authors, using the official implementations.

Metrics. We evaluate the novel-view rendering quality of each method via the standard metrics: PSNR, SSIM [177] and LPIPS [215]. To provide an idea of the compute/memory budget for each method, we also provide the number of Gaussians used to render images for each method. For Scaffold-GS [95] and Octree-GS [133], we measure the final number Gaussians

Method	Blender [104]				DTU [1]			
	PSNR ↑	SSIM ↑	LPIPS ↓	Number of Gaussians ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Number of Gaussians ↓
Octree-GS [133], LoD=0	22.025 ± 2.371	0.884 ± 0.035	0.115 ± 0.034	$4.722\times10^5 \pm 2.998\times10^5$	26.609 ± 4.240	0.863 ± 0.082	0.146 ± 0.069	$9.062\times10^5 \pm 4.178\times10^5$
Octree-GS [133], LoD=1	32.268 ± 3.622	0.961 ± 0.028	0.042 ± 0.030	$1.237\times10^6 \pm 4.392\times10^5$	31.803 ± 3.751	0.941 ± 0.030	0.069 ± 0.027	$1.809\times10^6 \pm 8.495\times10^5$
FLoD [142], LoD=0	16.815 ± 2.169	0.736 ± 0.069	0.236 ± 0.058	$0.810\times10^3 \pm 1.318\times10^3$	17.915 ± 3.212	0.670 ± 0.105	0.295 ± 0.079	$0.043\times10^3 \pm 0.029\times10^3$
FLoD [142], LoD=1	19.317 ± 2.631	0.799 ± 0.053	0.227 ± 0.058	$1.249\times10^3 \pm 1.132\times10^3$	22.083 ± 4.777	0.763 ± 0.111	0.291 ± 0.091	$0.632\times10^3 \pm 0.428\times10^3$
FLoD [142], LoD=2	23.338 ± 5.252	0.862 ± 0.066	0.156 ± 0.059	$1.117\times10^4 \pm 6.841\times10^3$	26.598 ± 6.897	0.833 ± 0.123	0.211 ± 0.097	$1.389\times10^4 \pm 6.795\times10^3$
FLoD [142], LoD=3	27.880 ± 7.470	0.915 ± 0.084	0.088 ± 0.074	$5.927\times10^4 \pm 3.830\times10^4$	29.122 ± 7.623	0.889 ± 0.117	0.129 ± 0.087	$6.157\times10^4 \pm 4.403\times10^4$
FLoD [142], LoD=4	29.480 ± 8.381	0.923 ± 0.087	0.075 ± 0.079	$1.879\times10^5 \pm 1.411\times10^5$	29.572 ± 7.597	0.897 ± 0.118	0.106 ± 0.089	$1.654\times10^5 \pm 1.676\times10^5$
Ours, LoD=0	21.109 ± 1.767	0.870 ± 0.037	0.197 ± 0.047	$2.601\times10^3 \pm 0$	23.648 ± 3.183	0.798 ± 0.087	0.283 ± 0.090	$2.601\times10^3 \pm 0$
Ours, LoD=1	24.535 ± 2.250	0.897 ± 0.035	0.136 ± 0.039	$1.638\times10^4 \pm 0$	26.985 ± 3.526	0.851 ± 0.078	0.222 ± 0.077	$1.638\times10^4 \pm 0$
Ours, LoD=2	27.401 ± 2.571	0.934 ± 0.023	0.087 ± 0.027	$6.554\times10^4 \pm 0$	29.083 ± 3.680	0.895 ± 0.058	0.163 ± 0.057	$6.554\times10^4 \pm 0$
Ours, LoD=3	29.238 ± 2.745	0.955 ± 0.017	0.058 ± 0.019	$1.475\times10^5 \pm 0$	30.173 ± 3.864	0.921 ± 0.042	0.120 ± 0.039	$1.475\times10^5 \pm 0$
Ours, LoD=4	32.076 ± 3.423	0.974 ± 0.014	0.032 ± 0.015	$2.621\times10^5 \pm 0$	31.766 ± 4.353	0.946 ± 0.032	0.076 ± 0.025	$2.621\times10^5 \pm 0$

Table 14. Quantitative results (varying levels of detail) – We report the performance of each method that supports different levels of detail. Our method provides the best tradeoff in terms of number of Gaussians vs rendering quality. At the finest resolution, our method is comparable to the original 3DGS [71] and also Octree-GS [133]. Note that NeRF Synthetic [104] dataset results for FLoD [142] should be interpreted with caution as it performs excessively poorly in some sequences.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	28.330 ± 3.202	0.921 ± 0.040	0.090 ± 0.037
Octree-GS [133]	15.058 ± 6.695	0.533 ± 0.199	0.197 ± 0.242
Scaffold-GS [95]	16.241 ± 7.944	0.592 ± 0.216	0.175 ± 0.205
FLoD [142]	18.636 ± 8.502	0.671 ± 0.232	0.323 ± 0.187
Ours	24.254 ± 3.043	0.769 ± 0.083	0.256 ± 0.069

Table 15. Quantitative results (finest level of detail) on BlendedMVS [201] – We show the rendering quality of our method and the baselines at the finest level of detail for BlendedMVS [201]. Compared to the datasets shown in Tab. 12, our approach achieves better performance that the baselines that provide a mechanism for varying levels of details.

Model	DTU [1] @ LoD=0		
	PSNR ↑	SSIM ↑	LPIPS ↓
– $\mathcal{M}$	14.556 ± 3.769	0.627 ± 0.114	0.297 ± 0.084
– $\tilde{\boldsymbol{\mu}}'$	23.498 ± 3.007	0.788 ± 0.098	0.290 ± 0.098
– $\Omega(\cdot)$	23.421 ± 2.932	0.784 ± 0.098	0.296 ± 0.102
– PLAS [107]	23.326 ± 2.908	0.784 ± 0.097	0.297 ± 0.100
– resampling [14]	22.465 ± 2.821	0.775 ± 0.099	0.305 ± 0.101
– Full method	23.648 ± 3.183	0.798 ± 0.087	0.283 ± 0.090

Table 16. Ablation study – We compare our method without the modulator $\mathcal{M}$, exploration noise $\tilde{\boldsymbol{\mu}}'$, regularizers $\Omega(\cdot)$, sorting using PLAS [107], and resampling based on the method of Bulò et al. [14]. We show the results at the coarsest LoD with about 2.6 thousand Gaussians. Our model comprising all the introduced components performs best on average. We mark best results with ■ and second best with ■.

used for rendering, which is by default ten times the number of point cloud points used in the representation.

5.4.2 Continuous Level of Details

As the main focus of our work is to introduce continous levels of detail, we first demonstrate its performance qualitatively (Fig. 42) and quantitatively (Fig. 43 and Tab. 14). As shown in Fig. 42, our method is able to render scenes at varying levels of detail. We additionally provide results for a concurrent work, FLoD [142], which is also able to provide levels of detail at predefined set of levels. Our method can render at *any* number of Gaussians between 32^2 and 512^2, and provides sharper renderings.

We note that results for FLoD [142] for the NeRF Synthetic [104] dataset should be interpreted with caution, as they performed particularly badly for `ficus` and `mic` sequences, despite our

Method	AVA [100]																				
	AAN112			ADL311			AEY864			AJR151			ANX726			APP152			AYE877		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	25.184	0.914	0.183	21.597	0.825	0.170	23.163	0.864	0.172	23.295	0.871	0.152	24.032	0.892	0.148	21.258	0.814	0.185	23.784	0.872	0.156
Octree-GS [133]	24.898	0.909	0.186	21.768	0.827	0.173	23.033	0.868	0.170	23.632	0.873	0.154	24.412	0.891	0.143	21.200	0.777	0.195	23.666	0.872	0.155
Scaffold-GS [95]	20.536	0.786	0.364	17.495	0.654	0.347	17.480	0.687	0.401	19.100	0.725	0.345	18.715	0.716	0.363	19.215	0.673	0.337	18.854	0.706	0.377
FLoD [142]	25.238	0.916	0.267	21.846	0.839	0.231	23.086	0.844	0.299	23.837	0.872	0.247	24.364	0.879	0.239	21.449	0.814	0.264	23.517	0.848	0.275
Ours	29.530	0.901	0.256	27.740	0.850	0.183	26.518	0.816	0.214	24.382	0.808	0.272	26.618	0.847	0.227	31.049	0.892	0.161	30.796	0.904	0.175

Method	AVA [100]																				
	BDF920			BGR645			BHC106			BHK376			BJM420			BLY735			BMP511			BPM833		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	22.365	0.864	0.169	21.987	0.887	0.185	21.481	0.862	0.205	25.320	0.924	0.156	23.233	0.902	0.187	22.904	0.884	0.151	21.249	0.861	0.191	21.742	0.838	0.210
Octree-GS [133]	22.532	0.874	0.171	21.787	0.883	0.182	21.494	0.859	0.204	24.892	0.927	0.150	23.357	0.903	0.178	22.851	0.889	0.144	21.362	0.871	0.186	21.799	0.834	0.208
Scaffold-GS [95]	17.987	0.734	0.345	19.082	0.778	0.313	18.354	0.728	0.402	21.666	0.832	0.298	18.513	0.782	0.370	17.129	0.720	0.365	17.120	0.723	0.400	18.713	0.683	0.438
FLoD [142]	22.704	0.873	0.249	21.954	0.893	0.210	21.444	0.848	0.314	25.277	0.929	0.212	23.199	0.901	0.256	22.762	0.867	0.257	21.172	0.859	0.291	21.614	0.792	0.356
Ours	26.683	0.843	0.201	26.465	0.877	0.221	26.270	0.811	0.219	33.195	0.950	0.210	27.076	0.884	0.213	28.599	0.870	0.165	26.991	0.858	0.182	26.529	0.752	0.228

Method	Blender [104]																				
	chair			drums			ficus			hotdog			lego			materials			mic			ship		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	35.634	0.988	0.010	26.280	0.955	0.037	35.486	0.987	0.012	38.016	0.985	0.020	35.997	0.982	0.017	30.464	0.960	0.037	36.669	0.992	0.006	31.719	0.907	0.107
Octree-GS [133]	34.924	0.985	0.013	25.180	0.938	0.060	30.668	0.971	0.030	37.840	0.984	0.023	35.342	0.980	0.019	30.433	0.959	0.041	32.567	0.976	0.035	31.188	0.899	0.113
Scaffold-GS [95]	35.218	0.986	0.013	26.366	0.950	0.045	34.882	0.985	0.014	37.959	0.984	0.022	35.728	0.981	0.018	30.715	0.961	0.040	37.172	0.992	0.008	31.535	0.903	0.107
FLoD [142]	35.711	0.987	0.011	26.203	0.953	0.039	35.281	0.986	0.012	38.197	0.985	0.022	35.993	0.981	0.018	14.406	0.721	0.240	18.190	0.867	0.150	31.858	0.906	0.108
Ours	34.152	0.981	0.023	25.290	0.944	0.057	31.501	0.978	0.025	36.723	0.984	0.030	34.009	0.980	0.025	29.452	0.958	0.048	33.442	0.988	0.012	29.805	0.889	0.148

Method	DTU [1]																				
	24			37			40			55			63			65			69		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	27.539	0.924	0.072	24.470	0.893	0.082	27.048	0.875	0.137	30.570	0.936	0.052	34.371	0.971	0.035	29.554	0.949	0.069	28.028	0.913	0.097
Octree-GS [133]	28.096	0.907	0.099	24.954	0.893	0.091	26.844	0.868	0.140	29.912	0.940	0.059	34.117	0.960	0.046	30.205	0.947	0.072	28.995	0.921	0.101
Scaffold-GS [95]	28.314	0.911	0.093	25.440	0.894	0.084	27.643	0.888	0.130	30.751	0.944	0.054	34.272	0.961	0.046	30.396	0.950	0.070	29.191	0.927	0.091
FLoD [142]	29.460	0.937	0.059	9.896	0.608	0.333	28.882	0.897	0.124	24.720	0.845	0.166	35.817	0.973	0.035	30.802	0.959	0.062	29.684	0.941	0.087
Ours	26.261	0.905	0.116	23.121	0.872	0.122	25.726	0.861	0.182	29.490	0.933	0.080	29.467	0.918	0.058	29.682	0.951	0.086	28.304	0.913	0.122

Method	DTU [1]																				
	83			97			105			106			110			114			118			122		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	38.681	0.984	0.028	30.210	0.951	0.053	33.384	0.960	0.053	33.988	0.949	0.060	34.217	0.968	0.062	30.732	0.945	0.060	36.593	0.968	0.043	36.991	0.967	0.034
Octree-GS [133]	38.328	0.980	0.029	30.271	0.943	0.063	33.075	0.956	0.060	34.342	0.952	0.066	34.140	0.963	0.065	30.531	0.942	0.065	36.526	0.964	0.048	36.707	0.970	0.037
Scaffold-GS [95]	38.723	0.982	0.030	30.237	0.947	0.057	33.799	0.959	0.060	34.664	0.955	0.061	34.506	0.967	0.063	30.787	0.947	0.062	36.983	0.971	0.044	37.659	0.974	0.032
FLoD [142]	38.596	0.983	0.033	30.818	0.954	0.051	34.318	0.964	0.060	15.360	0.621	0.283	35.475	0.973	0.060	31.162	0.954	0.052	30.791	0.873	0.153	37.797	0.979	0.028
Ours	34.404	0.979	0.040	29.638	0.945	0.074	31.457	0.960	0.072	33.888	0.953	0.074	34.738	0.969	0.066	30.666	0.940	0.082	36.869	0.973	0.054	37.199	0.975	0.043

Method	BlendedMVS [201]																				
	59d265			59f363			5a3ca9			5a6400			5a6464			5aa235			5ab871			5adc6b		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [71]	27.554	0.902	0.077	31.809	0.968	0.048	28.098	0.927	0.110	27.490	0.927	0.082	26.610	0.939	0.064	21.938	0.827	0.176	30.204	0.937	0.094	32.939	0.945	0.070
Octree-GS [133]	8.796	0.409	0.489	28.366	0.922	0.097	18.336	0.632	0.397	16.363	0.511	-0.016	14.392	0.573	0.327	17.862	0.566	0.516	14.119	0.509	0.515	4.781	0.190	0.647
Scaffold-GS [95]	8.796	0.409	0.489	31.305	0.958	0.058	17.540	0.622	0.375	16.924	0.637	0.328	10.706	0.599	0.313	19.625	0.685	0.345	26.593	0.859	0.170	4.781	0.190	0.647
FLoD [142]	23.896	0.719	0.256	29.684	0.938	0.109	25.268	0.854	0.193	23.954	0.856	0.167	10.760	0.599	0.313	21.737	0.749	0.293	9.009	0.460	0.609	4.782	0.190	0.647
Ours	23.330	0.653	0.321	19.011	0.731	0.274	25.158	0.818	0.225	26.025	0.831	0.212	24.378	0.864	0.152	20.385	0.629	0.391	27.328	0.846	0.211	28.420	0.783	0.260

Table 17. Per-subject quantitative results (finest level of detail) – We show the performance of the methods presented in the main part of the paper for each of the datasets' subjects. We found that for some examples Scaffold-GS [95] and FLoD [142] fails to produce crisp results and FLoD [142] outputs blank images on `ficus` and `mic` datasets from NeRF Synthetic [104], despite having the same hyperparameter and coordinate initialization values across datasets. That matter requires further investigation, potentially proposing a new approach to stabilizing their performance.

Model	DTU [1] @ l=0.25			DTU [1] @ l=0.50			DTU [1] @ l=0.75			DTU [1] @ l=1.00		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
– $\mathcal{M}$	16.311 ± 3.650	0.702 ± 0.103	0.260 ± 0.080	20.872 ± 3.304	0.795 ± 0.086	0.211 ± 0.072	25.990 ± 3.309	0.870 ± 0.064	0.146 ± 0.057	30.664 ± 3.972	0.931 ± 0.042	0.084 ± 0.036
– $\hat{\boldsymbol{\mu}}'$	26.168 ± 3.267	0.836 ± 0.092	0.233 ± 0.089	27.719 ± 3.471	0.874 ± 0.074	0.180 ± 0.076	28.433 ± 3.584	0.897 ± 0.060	0.140 ± 0.063	29.072 ± 3.887	0.917 ± 0.048	0.104 ± 0.047
– $\Omega(\cdot)$	26.461 ± 3.282	0.836 ± 0.094	0.233 ± 0.092	28.183 ± 3.519	0.873 ± 0.073	0.176 ± 0.072	29.222 ± 3.682	0.908 ± 0.054	0.131 ± 0.055	30.577 ± 4.275	0.931 ± 0.042	0.087 ± 0.037
– PLAS [107]	26.569 ± 3.244	0.840 ± 0.090	0.228 ± 0.087	28.350 ± 3.466	0.885 ± 0.068	0.169 ± 0.068	29.557 ± 3.619	0.912 ± 0.050	0.126 ± 0.053	30.639 ± 4.046	0.934 ± 0.037	0.085 ± 0.036
– resampling [14]	25.861 ± 3.186	0.829 ± 0.094	0.246 ± 0.091	27.776 ± 3.424	0.869 ± 0.076	0.192 ± 0.078	28.774 ± 3.518	0.900 ± 0.061	0.152 ± 0.066	30.399 ± 4.076	0.931 ± 0.042	0.102 ± 0.050
Full method	26.317 ± 3.242	0.833 ± 0.089	0.232 ± 0.089	28.253 ± 3.430	0.878 ± 0.070	0.173 ± 0.070	29.282 ± 3.527	0.908 ± 0.052	0.129 ± 0.055	30.727 ± 3.948	0.936 ± 0.035	0.085 ± 0.036

Table 18. Ablation study for varying levels of detail – We present additional ablation study performed over multiple level of details. We mark best results with ■ and second best with ■.

best efforts using the official implementation. We present metrics for each subject in the dataset and additional renders in Tab. 17.

In Fig. 43, we summarize the quantitative results in Tab. 14 for the DTU dataset. We use DTU only for this comparison as FLoD does not work well for the two sequences from the NeRF Synthetic dataset. As shown, our method provides the best rendering quality given a predetermined number of Gaussians. FLoD [142] provides slightly inferior performance compared to ours, and while Octree-GS is able to be used to extract a LOD, its quality degrades quickly. Finally, at the finest level, our results are comparable to traditional 3D Gaussian Splatting. We emphasize once more, that our results are obtained with a *single* underlying representation for all LoDs, simply rendered at a different level of detail.

We present additional, per-subject quantitative results in Tab. 17 and extended ablation studies over additional scales $l \in \{0.25, 0.5, 0.75, 1.0\}$ (for reference, Tab. 16 involves l=1.0) Tab. 18. We also include more reconstruction samples and comparison to baselines in Fig. 46 and the visualization the LoD progression has on the results in Fig. 45.

5.4.3 At the finest level

To demonstrate that our method is also able to provide fine details if necessary, we provide a summary of the results at the finest level of detail. We provide qualitative examples in Figs. 44 and 47 and a summary in Tabs. 12 and 15. Our method provides comparable rendering quality to other state-of-the-art Gaussian Splatting methods, and on top, is able to provide continuous levels of detail. Additionally, we note that our method trains given a pre-defined budget, which makes deployment also easy.

5.4.4 Ablation study

To motivate our design choices, we provide an ablation study where various components of our method are disabled. Specifically, we look into the impact of the modulator $\mathcal{M}$, the noise in $\tilde{\boldsymbol{\mu}}'$ Eq. (60), the sorting with PLAS [107] in the feature space for Stage II training, the use of regularizers $\Omega(\cdot)$, and disabling error-based relocation [14]. Others can trivially be disabled, and for the modulator $\mathcal{M}$ we use directly the downsampled descriptors $\boldsymbol{\Lambda}$ without refinement, *i.e.*, we apply $\boldsymbol{\Lambda}_{\downarrow}$ instead of $\tilde{\boldsymbol{\Lambda}}$ in the rendering equation (Eq. (55)). We present our results in Tab. 16.

5.5 Conclusions

We presented Continuous Levels of Gaussians (CLoG), a novel approach that enables continuous levels of detail for 3D Gaussian Splatting. Our method embeds Gaussians on a 2D map that can be dynamically downsampled to any resolution to provide levels of detail. To account for using fewer Gaussians, we introduce a modulator network that adapts Gaussian features to lower resolutions. We demonstrated that CLoG achieves comparable quality to existing methods

at their highest resolution while providing the unique ability to smoothly transition between different levels of detail, with the best trade-off between quality and the number of Gaussians.

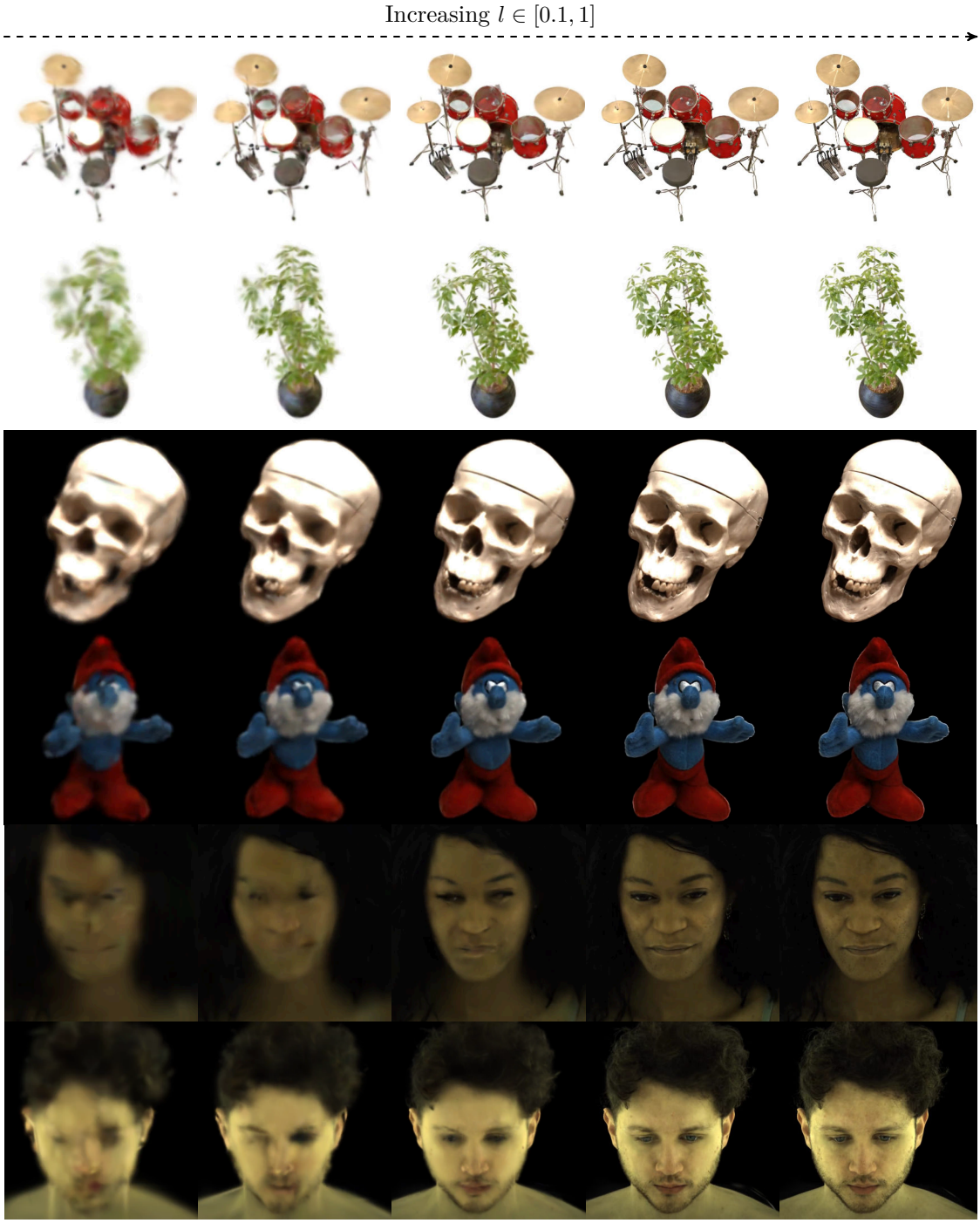


Figure 45. Additional qualitative examples (progressive level of details) – We showcase how our method reconstructs different samples under a given LoD l. The bottom row shows the number of Gaussians used to render an image. We use $l \in \{0.1, 0.28, 0.46, 0.82, 1.0\}$ levels od detail.

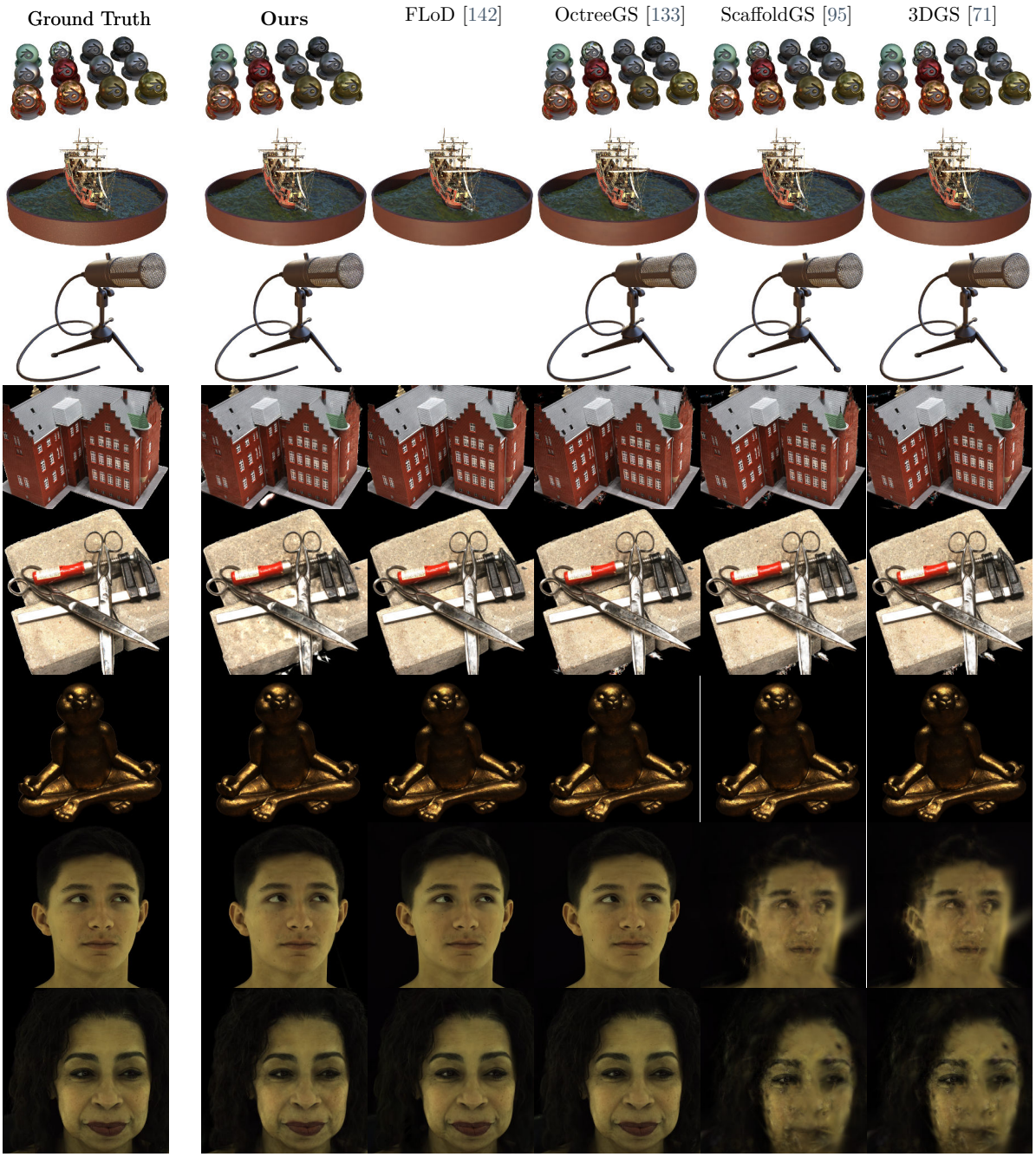


Figure 46. Additional qualitative examples (finest level of details) – We report additional qualitative results of our method and the baselines. Empty images for FLoD [142] denote subjects where the method failed to converge. However, it is worth noting that its quality is on par with OctreeGS [133], ScaffoldGS [95] and 3DGS [71] on all other samples. Please notice that ScaffoldGS [95] and 3DGS [71] produce artifacts for AVA (two last rows) which we argue come from insufficient coverage of cameras around the subjects and lack of the scale regularization mechanisms present in other approaches.

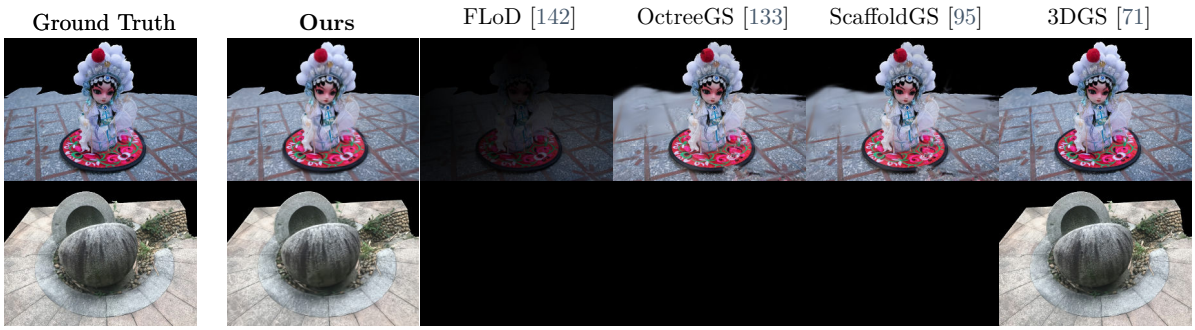


Figure 47. Qualitative examples for BlendedMVS [201] (finest level of details) – We showcase additional novel-view synthesis results for different methods on BlendedMVS [201]. It is worth noting that FLoD [142], OctreeGS [133] and ScaffoldGS [95] and their hyperparameters are notoriously difficult to tune to obtain results of reasonable quality. Lack of convergence is displayed as black rectangles, while FLoD outputs an artifact in front of the camera.

Chapter 6

Conclusions and Future Directions

In this thesis, we have advanced the field of realistic 3D object rendering through a series of methodological advancements in the use of sparse conditioning signals. Our research has addressed four fundamental challenges in neural rendering: interpretable control, animation fidelity, lighting decomposition, and computational efficiency. Each contribution builds upon the previous ones while opening new avenues for investigation.

We began by tackling the challenge of controllability in Neural Radiance Fields (NeRFs) through our CoNeRF model. Unlike concurrent approaches that sacrifice quality for control, CoNeRF achieves interpretable manipulation of 3D representations while maintaining high-fidelity output. This work laid the foundation for more sophisticated control mechanisms in neural rendering.

Building on these control capabilities, we developed BlendFields to address the complex challenge of realistic animation. This method transforms multi-view static captures into expressive human head avatars, generating physically-plausible, expression-dependent skin deformations in a few-shot regime. By incorporating physically-based tetrahedral deformation features, BlendFields achieves precise skin shading adjustments that surpass existing methods dependent on extensive learned conditioning signals.

To enhance visual realism of the light in reconstructed scenes, we tackled the challenge of entangled appearance in trained radiance fields. Our LumiGauss model introduces a data-driven approach to learning lighting features, constraining the output color to be a product of intrinsic subject color and lighting attributes. This enables intuitive post-training scene relighting without requiring the extensive multi-condition datasets needed by methods such as Saito et al. [136].

Finally, we addressed the practical constraints of deployment with our CLoG method for adapting 3D Gaussian Splats (3DGS) representations. By modeling 3D Gaussians on a 2D grid and applying PLAS [107] for frequency-compatible reorganization, we enabled continuous representation adaptation during inference. This achievement maintains performance parity with fixed-scale approaches while offering unprecedented flexibility in computational resource management.

6.1 Impact and Broader Implications

These contributions collectively address the research questions outlined in Chapter 1, establishing a comprehensive framework for next-generation radiance fields. Our advances enable:

- Intuitive control through sparse annotation strokes, making complex 3D manipulation accessible to non-experts,
- High-fidelity avatar animation with realistic expression-dependent features from minimal training data,
- Flexible scene relighting using compact sets of varied lighting conditions,
- Adaptive representation optimization for diverse computational environments.

Beyond these technical achievements, our work has broader implications for the democratization of neural rendering technologies. By reducing data requirements and computational demands while increasing controllability, we have taken significant steps toward making NeRFs and 3DGS more accessible to the public. This accessibility could catalyze their adoption in fields like 3D medical imaging, where precise control and efficient deployment are crucial. Furthermore, our frameworks provide building blocks for future innovations in human-centric neural rendering.

6.2 Future Work and Open Questions

Our research advances have unveiled several promising directions that warrant further investigation. We identify three fundamental questions that could drive future innovations in neural rendering:

Representation-Agnostic Control Methods. The rapid evolution of neural rendering architectures raises fundamental questions about the generalizability of our control mechanisms. This thesis was conducted during a period of significant advancement in both NeRFs and 3DGS, leading to methods like CoNeRF being initially designed for NeRFs' architectures. The challenge lies in the fundamental differences between representations: NeRFs utilize Fourier-transformed coordinates for high-frequency detail generation, enabling field modulation through frequency manipulation, while spatial encodings like InstantNGP's [109] hash grids offer high expressivity but present control challenges due to spatial decoherence. Recent progress, including Neuralangelo [85] and the successful adaptation of CoNeRF to 3DGS by Yu et al. [204], suggests the potential for unified control frameworks across representations.

Expanding Control Modalities. While we have explored various control approaches—from sparse annotations to lighting and computational adaptation—the potential landscape of control modalities remains largely unexplored. Emerging research directions point to exciting possibilities: text-prompted asset generation [162], parametric human control [76], shadow manipulation [152], and intelligent representation selection [52]. Each of these directions presents

unique challenges and opportunities for extending our control frameworks. This thesis represents an initial exploration in the rapidly evolving domain of radiance field control, with numerous promising avenues yet to be investigated.

Integration with Generative Models. Perhaps the most transformative potential lies in combining our control methods with generative modeling. Current generative approaches for NeRFs and 3DGS demand substantial computational resources [162] and primarily adapt image-based techniques through text prompts. We envision integrating our control mechanisms with score distillation objectives [123], enabling joint conditioning of both radiance fields and diffusion processes. While no comprehensive solution exists yet, success in this direction could revolutionize content creation across industries, from digital art to gaming, while potentially reducing computational demands through more targeted generation processes.

Bibliography

[1] Aanæs, H., Jensen, R. R., Vogiatzis, G., Tola, E., and Dahl, A. B., "Large-Scale Data for Multiple-View Stereopsis," *IJCV*, pp. 1–16, 2016 (cit. on pp. 67, 68, 77–80).

[2] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M., "Interactive Digital Photomontage," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04, Los Angeles, California: Association for Computing Machinery, 2004, pp. 294–302, ISBN: 9781450378239. DOI: 10.1145/1186562.1015718. [Online]. Available: https://doi.org/10.1145/1186562.1015718 (cit. on p. 10).

[3] Alldieck, T., Xu, H., and Sminchisescu, C., "imGHUM: Implicit Generative Models of 3D Human Shape and Articulated Pose," in *CVPR*, 2021 (cit. on p. 12).

[4] Athar, S., Xu, Z., Sunkavalli, K., Shechtman, E., and Shu, Z., "RigNeRF: Fully Controllable Neural 3D Portraits," in *CVPR*, 2022, pp. 20 364–20 373 (cit. on pp. 29, 32, 33).

[5] Attal, B., Laidlaw, E., Gokaslan, A., Kim, C., Richardt, C., Tompkin, J., and O'Toole, M., "TöRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis," *NeurIPS*, vol. 34, pp. 26 289–26 301, 2021 (cit. on p. 32).

[6] Avcibas, I., Sankur, B., and Sayood, K., "Statistical evaluation of image quality measures," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 206–223, 2002 (cit. on p. 39).

[7] Barron, J., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P., "Mip-NeRF: A Multiscale Representation for Anti-aliasing Neural Radiance Fields," in *ICCV*, 2021, pp. 5855–5864 (cit. on p. 31).

[8] Barron, J., Mildenhall, B., Verbin, D., Srinivasan, P., and Hedman, P., "Mip-NeRF 360: Unbounded Anti-aliased Neural Radiance Fields," in *CVPR*, 2022, pp. 5470–5479 (cit. on p. 31).

[9] Barron, J. T. and Malik, J., "Shape, illumination, and reflectance from shading," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 8, pp. 1670–1687, 2014 (cit. on p. 51).

[10] Belharbi, S., Ben Ayed, I., McCaffrey, L., and Granger, E., "Deep Active Learning for Joint Classification & Segmentation with Weak Annotator," 2021 (cit. on p. 28).

[11] Blanz, V. and Vetter, T., "A Morphable Model For The Synthesis Of 3D Faces," in *CGIT*, 1999, pp. 187–194 (cit. on pp. 10, 29, 31, 32, 35).

[12] Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A., "Optimizing the Latent Space of Generative Networks," in *ICML*, 2017 (cit. on p. 72).

[13] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q., *JAX: Composable Transformations of Python+NumPy Programs*, http://github.com/google/jax, version 0.2.5, 2018 (cit. on p. 16).

[14] Bulò, S. R., Porzi, L., and Kontschieder, P., "Revising Densification in Gaussian Splatting," in *ECCV*, 2024 (cit. on pp. 70, 74, 79–81).

[15] *Bunny 3D model*, https://graphics.stanford.edu/~mdfisher/Data/Meshes/bunny.obj, Accessed: 2021-11-16 (cit. on p. 20).

[16] Cao, C., Simon, T., Kim, J. K., Schwartz, G., Zollhoefer, M., Saito, S.-S., Lombardi, S., Wei, S.-E., Belko, D., Yu, S.-I., et al., "Authentic Volumetric Avatars from a Phone Scan," *ToG*, vol. 41, no. 4, pp. 1–19, 2022 (cit. on pp. 31–33, 39).

[17] Chang, Y., Kim, Y., Seo, S., Yi, J., and Kwak, N., "Fast sun-aligned outdoor scene relighting based on tensorf," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 3626–3636 (cit. on pp. 51, 58, 62, 64, 65).

[18] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," in *NeurIPS*, 2016 (cit. on p. 10).

[19] Chen, X., Zhang, Q., Li, X., Chen, Y., Feng, Y., Wang, X., and Wang, J., "Hallucinated Neural Radiance Fields in the Wild," in *CVPR*, 2022, pp. 12 943–12 952 (cit. on pp. 4, 50, 51).

[20] Cheng, Z., Chai, M., Ren, J., Lee, H.-Y., Olszewski, K., Huang, Z., Maji, S., and Tulyakov, S., "Cross-Modal 3D Shape Generation and Manipulation," in *ECCV*, Springer, 2022, pp. 303–321 (cit. on p. 32).

[21] Clark, J. H., "Hierarchical geometric models for visible surface algorithms," *Communications of the ACM*, vol. 19, no. 10, pp. 547–554, 1976 (cit. on p. 70).

[22] Clough, R. W., "The Finite Element Method in Plane Stress Analysis," in *Conference on Electronic Computation*, 1960 (cit. on p. 33).

[23] Community, B. O., *Blender - a 3d modeling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022. [Online]. Available: http://www.blender.org (cit. on p. 42).

[24] Cook, S., *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*, 1st. 2012, ISBN: 9780124159334 (cit. on p. 33).

[25] Dahmani, H., Bennehar, M., Piasco, N., Roldao, L., and Tsishkou, D., "Swag: Splatting in the wild images with appearance-conditioned gaussians," in *European Conference on Computer Vision*, Springer, 2025, pp. 325–340 (cit. on pp. 52, 63).

[26] Deng, B., Lewis, J. P., Jeruzalski, T., Pons-Moll, G., Hinton, G., Norouzi, M., and Tagliasacchi, A., "NASA: Neural Articulated Shape Approximation," in *ECCV*, 2020 (cit. on p. 12).

[27] Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H., "Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow," in *CGIT*, 1999, pp. 317–324 (cit. on p. 38).

[28] Duchêne, S., Riant, C., Chaurasia, G., Lopez-Moreno, J., Laffont, P.-Y., Popov, S., Bousseau, A., and Drettakis, G., "Multi-view intrinsic images of outdoors scenes with an application to relighting," *ACM Transactions on Graphics*, p. 16, 2015 (cit. on p. 51).

[29] Duckworth, D., Hedman, P., Reiser, C., Zhizhin, P., Thibert, J.-F., Lučić, M., Szeliski, R., and Barron, J. T., "SMERF: Streamable Memory Efficient Radiance Fields for Real-Time Large-Scene Exploration," *TOG*, vol. 43, no. 4, pp. 1–13, 2024 (cit. on p. 70).

[30] Ekman, P. and Rosenberg, E. L., *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)*. Oxford University Press, USA, 1997 (cit. on p. 10).

[31] Esposito, S., Xu, Q., **Kania**, **K.**, Hewitt, C., Mariotti, O., Petikam, L., Valentin, J., Onken, A., and Mac Aodha, O., "GeoGen: Geometry-Aware Generative Modeling via Signed Distance Functions," in *CVPRW*, 2024, pp. 7479–7488 (cit. on p. 8).

[32] Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., and Wang, Z., "LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS," in *NeurIPS*, 2024 (cit. on pp. 68, 70).

[33] Fang, G. and Wang, B., "Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians," in *ECCV*, 2024 (cit. on p. 70).

[34] Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., and Tian, Q., "Fast Dynamic Radiance Fields with Time-Aware Neural Voxels," in *SIGGRAPH Asia 2022 Conference Papers*, 2022, pp. 1–9 (cit. on p. 3).

[35] Feng, Y., Feng, H., Black, M. J., and Bolkart, T., "Learning an Animatable Detailed 3D Face Model from In-The-Wild Images," *ToG*, vol. 40, no. 4, pp. 1–13, 2021 (cit. on p. 3).

[36] Flavell, L., "Uv mapping," in *Beginning Blender: Open Source 3D Modeling, Animation, and Game Design*, Springer, 2010, pp. 97–122 (cit. on p. 68).

[37] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A., "Plenoxels: Radiance Fields without Neural Networks," in *CVPR*, 2022, pp. 5501–5510 (cit. on p. 69).

[38] Gafni, G., Thies, J., Zollhofer, M., and Nießner, M., "Dynamic Neural Radiance Fields for Monocular 4d Facial Avatar Reconstruction," in *CVPR*, 2021, pp. 8649–8658 (cit. on pp. 9, 11, 29, 32, 33, 40).

[39] Gao, J., Gu, C., Lin, Y., Zhu, H., Cao, X., Zhang, L., and Yao, Y., "Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing," *arXiv preprint arXiv:2311.16043*, 2023 (cit. on p. 52).

[40] Gao, X., Zhong, C., Xiang, J., Hong, Y., Guo, Y., and Zhang, J., "Reconstructing Personalized Semantic Facial NeRF Models From Monocular Video," *SIGGRAPH Asia*, vol. 41, no. 6, 2022 (cit. on pp. 29, 33, 34).

[41] Garbin, S. J., Kowalski, M., Estellers, V., Szymanowicz, S., Rezaeifar, S., Shen, J., Johnson, M. A., and Valentin, J., "VolTeMorph: Real-time, Controllable and Generalizable Animation of Volumetric Representations," in *CGS*, Wiley Online Library, vol. 43, 2024, e15117 (cit. on pp. 5, 29–33, 35, 38–41, 48).

[42] Garbin, S. J., Kowalski, M., Johnson, M., Shotton, J., and Valentin, J., "FastNeRF: High-Fidelity Neural Rendering at 200FPS," in *ICCV*, 2021, pp. 14 346–14 355 (cit. on pp. 4, 32, 33, 69).

[43] Gardner, J. A., Kashin, E., Egger, B., and Smith, W. A., "The sky's the limit: Relightable outdoor scenes via a sky-pixel constrained illumination prior and outside-in visibility," in *European Conference on Computer Vision*, Springer, 2025, pp. 126–143 (cit. on pp. 49, 51, 58–60, 63, 66).

[44] Girish, S., Gupta, K., and Shrivastava, A., "EAGLES: Efficient Accelerated 3D Gaussians with Lightweight Encoding," in *ECCV*, 2024 (cit. on p. 70).

[45] Grassal, P.-W., Prinzler, M., Leistner, T., Rother, C., Nießner, M., and Thies, J., "Neural Head Avatars From Monocular RGB Videos," in *CVPR*, 2022, pp. 18 653–18 664 (cit. on pp. 4, 32, 40).

[46] Green, R., "Spherical harmonic lighting: The gritty details," in *Archives of the game developers conference*, vol. 56, 2003, p. 4 (cit. on pp. 6, 52, 54).

[47] Greff, K., Tagliasacchi, A., Liu, D., and Laradji, I., *Kubric*, `http://github.com/google-research/kubric`, version 0.1.1, 2021 (cit. on p. 20).

[48] Guédon, A. and Lepetit, V., "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5354–5363 (cit. on p. 50).

[49] Guo, M., Fathi, A., Wu, J., and Funkhouser, T., "Object-Centric Neural Scene Rendering," 2020 (cit. on p. 11).

[50] Haber, T., Fuchs, C., Bekaer, P., Seidel, H.-P., Goesele, M., and Lensch, H. P., "Relighting objects from image collections," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 627–634 (cit. on p. 51).

[51] He, L., Li, L., Sun, W., Han, Z., Liu, Y., Zheng, S., Wang, J., and Li, K., "Neural radiance field in autonomous driving: A survey," *arXiv preprint arXiv:2404.13816*, 2024 (cit. on p. 51).

[52] He, S., Osman, Z., and Chaudhari, P., *From NeRFs to Gaussian Splats, and Back*, 2024. arXiv: 2405.09717 [cs.CV] (cit. on p. 88).

[53] He, T., Xu, Y., Saito, S., Soatto, S., and Tung, T., "ARCH++: Animation-Ready Clothed Human Reconstruction Revisited," in *ICCV*, 2021 (cit. on p. 12).

[54] Hedman, P., Srinivasan, P. P., Mildenhall, B., Barron, J. T., and Debevec, P., "Baking Neural Radiance Fields for Real-Time View Synthesis," in *ICCV*, 2021, pp. 5875–5884 (cit. on pp. 4, 32, 33, 39).

[55] Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A., "Towards a Definition of Disentangled Representations," 2018 (cit. on p. 10).

[56] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A., "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *ICLR*, 2017 (cit. on p. 10).

[57] Huang, B., Yu, Z., Chen, A., Geiger, A., and Gao, S., "2D Gaussian Splatting for Geometrically Accurate Radiance Fields," in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11 (cit. on pp. 6, 50, 52, 54, 56, 58–60, 66, 70).

[58] Huang, X., Zhang, Q., Feng, Y., Li, H., Wang, X., and Wang, Q., "HDR-NeRF: High Dynamic Range Neural Radiance Fields," in *CVPR*, 2022, pp. 18 398–18 408 (cit. on p. 31).

[59] Ichim, A. E., Bouaziz, S., and Pauly, M., "Dynamic 3D Avatar Creation from Hand-Held Video Input," *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015, ISSN: 0730-0301 (cit. on p. 36).

[60] Irving, G., Teran, J., and Fedkiw, R., "Invertible Finite Elements For Robust Simulation of Large Deformation," in *SIGGRAPH*, 2004, pp. 131–140 (cit. on pp. 31, 33, 37).

[61] Jang, W. and Agapito, L., "CodeNeRF: Disentangled Neural Radiance Fields for Object Categories," in *ICCV*, 2021 (cit. on pp. 9, 11).

[62] Jiang, W., Yi, K. M., Samei, G., Tuzel, O., and Ranjan, A., "NeuMan: Neural Human Radiance Field from a Single Video," in *ECCV*, Springer, 2022, pp. 402–418 (cit. on p. 32).

[63] Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., and Ma, Y., "GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces," in *CVPR*, 2024, pp. 5322–5332 (cit. on p. 70).

[64] Kajiya, J. T. and Von Herzen, B. P., "Ray Tracing Volume Densities," *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 165–174, 1984 (cit. on p. 12).

[65] Kaleta, J., **Kania**, **K.**, Trzcinski, T., and Kowalski, M., "LumiGauss: High-Fidelity Outdoor Relighting with 2D Gaussian Splatting," 2025 (cit. on pp. 2, 4, 6).

[66] **Kania**, **K.**, Garbin, S. J., Tagliasacchi, A., Estellers, V., Yi, K. M., Valentin, J., Trzciński, T., and Kowalski, M., "BlendFields: Few-Shot Example-Driven Facial Modeling," in *CVPR*, 2023, pp. 404–415 (cit. on pp. 2, 4, 5).

[67] **Kania**, **K.**, Khirodkar, R., Saito, S., Yi, K. M., and Martinez, J., "CLoG: Leveraging UV Space for Continuous Levels of Detail," 2024 (cit. on pp. 4, 6).

[68] **Kania**, **K.**, Kowalski, M., and Trzciński, T., "TrajeVAE: Controllable Human Motion Generation from Trajectories," *arXiv preprint arXiv:2104.00351*, 2021 (cit. on p. 8).

[69] **Kania**, **K.**, Yi, K. M., Kowalski, M., Trzciński, T., and Tagliasacchi, A., "CoNeRF: Controllable Neural Radiance Fields," in *CVPR*, 2022 (cit. on pp. 2, 4, 5, 29, 32).

[70] **Kania**, **K.**, Zięba, M., and Kajdanowicz, T., "UCSG-NET – Unsupervised Discovering of Constructive Solid Geometry Tree," *NeurIPS*, vol. 33, pp. 8776–8786, 2020 (cit. on p. 8).

[71] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G., "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *TOG*, vol. 42, no. 4, pp. 139–1, 2023 (cit. on pp. 1, 4, 6, 50, 52–54, 68–74, 77–80, 84, 85).

[72] Kerbl, B., Meuleman, A., Kopanas, G., Wimmer, M., Lanvin, A., and Drettakis, G., "A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 4, pp. 1–15, 2024 (cit. on p. 70).

[73] Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, J., Isack, H., Kar, A., Tagliasacchi, A., and Yi, K. M., "3D Gaussian Splatting as Markov Chain Monte Carlo," in *NeurIPS*, 2024 (cit. on pp. 73, 74).

[74] Kim, M., Ko, J., Cho, K., Choi, J., Choi, D., and Kim, S., "AE-NeRF: Auto-Encoding Neural Radiance Fields for 3D-Aware Object Manipulation," *arXiv preprint arXiv:2204.13426*, 2022 (cit. on p. 32).

[75] Kingma, D. P. and Ba, J., "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015 (cit. on pp. 17, 39).

[76] Kocabas, M., Chang, J.-H. R., Gabriel, J., Tuzel, O., and Ranjan, A., "HUGS: Human Gaussian Splats," in *CVPR*, 2024, pp. 505–515 (cit. on p. 88).

[77] Kulkarni, T., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A., and Mnih, V., "Unsupervised Learning of Object Keypoints for Perception and Control," in *NeurIPS*, 2019 (cit. on p. 28).

[78] Lalonde, J.-F., Efros, A. A., and Narasimhan, S. G., "Webcam clip art: Appearance and illuminant transfer from time-lapse sequences," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, pp. 1–10, 2009 (cit. on p. 51).

[79] Lee, J. C., Rho, D., Sun, X., Ko, J. H., and Park, E., "Compact 3D Gaussian Representation for Radiance Field," in *CVPR*, 2024, pp. 21 719–21 728 (cit. on pp. 68, 70).

[80] Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F., and Deng, Z., "Practice and Theory of Blendshape Facial Models," in *Eurographics 2014 - State of the Art Reports*, Lefebvre, S. and Spagnuolo, M., Eds., The Eurographics Association, 2014 (cit. on p. 31).

[81] Li, P., Wang, S., Yang, C., Liu, B., Qiu, W., and Wang, H., "Nerf-ms: Neural radiance fields with multi-sequence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 591–18 600 (cit. on p. 51).

[82] Li, Q., Guo, J., Fei, Y., Li, F., and Guo, Y., "Neulighting: Neural lighting for free viewpoint outdoor scene relighting with unconstrained photo collections," in *SIGGRAPH Asia 2022 Conference Papers*, 2022, pp. 1–9 (cit. on p. 51).

[83] Li, S. Z., Zang, Z., and Wu, L., "Markov-Lipschitz Deep Learning," 2020 (cit. on p. 27).

[84] Li, T., Bolkart, T., Black, M. J., Li, H., and Romero, J., "Learning a model of facial shape and expression from 4D scans," *SIGGRAPH Asia*, vol. 36, no. 6, 194:1–194:17, 2017 (cit. on pp. 5, 29, 34–36, 41).

[85] Li, Z., Müller, T., Evans, A., Taylor, R. H., Unberath, M., Liu, M.-Y., and Lin, C.-H., "Neuralangelo: High-Fidelity Neural Surface Reconstruction," in *CVPR*, 2023, pp. 8456–8465 (cit. on p. 88).

[86] Liang, Z., Zhang, Q., Feng, Y., Shan, Y., and Jia, K., "Gs-ir: 3d gaussian splatting for inverse rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 644–21 653 (cit. on p. 52).

[87] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P., "Focal Loss for Dense Object Detection," in *ICCV*, 2017 (cit. on p. 15).

[88] Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S., and Theobalt, C., "Neural Sparse Voxel Fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 651–15 663, 2020 (cit. on p. 33).

[89] Liu, L., Habermann, M., Rudnev, V., Sarkar, K., Gu, J., and Theobalt, C., "Neural Actor: Neural Free-view Synthesis of Human Actors with Pose Control," 2021 (cit. on pp. 9, 11).

[90] Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.-Y., and Russell, B., "Editing Conditional Radiance Fields," in *ICCV*, 2021, pp. 5773–5783 (cit. on pp. 3, 5, 9, 11, 29).

[91] Liu, Y., Guan, H., Luo, C., Fan, L., Peng, J., and Zhang, Z., "CityGaussian: Real-time High-quality Large-Scale Scene Rendering with Gaussians," in *ECCV*, 2024 (cit. on p. 70).

[92] Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., and Sheikh, Y., "Neural Volumes: Learning Dynamic Renderable Volumes from Images," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019, ISSN: 0730-0301 (cit. on p. 33).

[93] Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., and Saragih, J., "Mixture of Volumetric Primitives for Efficient Neural Rendering," *ACM TOG*, vol. 40, no. 4, pp. 1–13, 2021 (cit. on pp. 33, 71).

[94] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M., "SMPL: A Skinned Multi-Person Linear Model," *ACM Trans. Graph.*, vol. 34, no. 6, 2015 (cit. on pp. 11, 12).

[95] Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., and Dai, B., "Scaffold-GS: Structured 3D Gaussians for View-Adaptive Renderin," in *CVPR*, 2024, pp. 20 654–20 664 (cit. on pp. 70, 71, 73, 77–80, 84, 85).

[96] Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B., and Huebner, R., *Level of Detail for 3D Graphics*. Elsevier, 2002 (cit. on p. 70).

[97] Ma, Q., Saito, S., Yang, J., Tang, S., and Black, M. J., "SCALE: Modeling Clothed Humans with a Surface Codec of Articulated Local Elements," in *CVPR*, 2021 (cit. on p. 12).

[98] Ma, S., Simon, T., Saragih, J., Wang, D., Li, Y., De la Torre, F., and Sheikh, Y., "Pixel Codec Avatars," in *CVPR*, Jul. 2021, pp. 64–73 (cit. on pp. 29, 33).

[99] Martin-Brualla, R., Radwan, N., Sajjadi, M. S., Barron, J. T., Dosovitskiy, A., and Duckworth, D., "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7210–7219 (cit. on pp. 4, 9, 11, 14, 31, 50, 51).

[100] Martinez, J., Kim, E., Romero, J., Bagautdinov, T., Saito, S., Yu, S.-I., Anderson, S., Zollhöfer, M., Wang, T.-L., Bai, S., Li, C., Wei, S.-E., Joshi, R., Borsos, W., Simon, T., Saragih, J., Theodosis, P., Greene, A., Josyula, A., Maeta, S. M., Jewett, A. I., Venshtain, S., Heilman, C., Chen, Y.-T., Fu, S., Elshaer, M. E. A., Du, T., Wu, L., Chen, S.-C., Kang, K., Wu, M., Emad, Y., Longay, S., Brewer, A., Shah, H., Booth, J., Koska, T., Haidle, K., Andromalos, M., Hsu, J., Dauer, T., Selednik, P., Godisart, T., Ardisson, S., Cipperly, M., Humberston, B., Farr, L., Hansen, B., Guo, P., Braun, D., Krenn, S., Wen, H., Evans, L., Fadeeva, N., Stewart, M., Schwartz, G., Gupta, D., Moon, G., Guo, K., Dong, Y., Xu, Y., Shiratori, T., Prada, F., Pires, B. R., Peng, B., Buffalini, J., Trimble, A., McPhail, K., Schoeller, M., and Sheikh, Y., "Codec Avatar Studio: Paired Human Captures for Complete, Driveable, and Generalizable Avatars," *NeurIPS Track on Datasets and Benchmarks*, 2024 (cit. on pp. 67, 77, 78, 80).

[101] Mihajlovic, M., Bansal, A., Zollhoefer, M., Tang, S., and Saito, S., "KeypointNeRF: Generalizing Image-based Volumetric Avatars using Relative Spatial Encoding of Keypoints," in *ECCV*, 2022 (cit. on p. 33).

[102] Mihajlovic, M., Zhang, Y., Black, M. J., and Tang, S., "LEAP: Learning Articulated Occupancy of People," in *CVPR*, 2021 (cit. on p. 12).

[103] Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P., and Barron, J., "NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images," in *CVPR*, 2022, pp. 16 190–16 199 (cit. on p. 31).

[104] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021 (cit. on pp. 1, 3, 9, 11, 12, 14, 21, 29, 31, 32, 39–41, 47, 50, 51, 67, 69, 74, 76–80).

[105] Molino, N., Bridson, R., and Fedkiw, R., "Tetrahedral Mesh Generation for Deformable Bodies," 2003 (cit. on p. 33).

[106] Monk, P., "Finite Elements on Tetrahedra," in *Finite Element Methods for Maxwell's Equations*, Oxford, United Kingdom: Oxford University Press, Apr. 2003, pp. 99–154, ISBN: 9780198508885 (cit. on p. 36).

[107] Morgenstern, W., Barthel, F., Hilsmann, A., and Eisert, P., "Compact 3D Scene Representation via Self-Organizing Gaussian Grids," in *ECCV*, 2024 (cit. on pp. 69, 70, 74, 75, 79–81, 87).

[108] Mu, J., Qiu, W., Kortylewski, A., Yuille, A., Vasconcelos, N., and Wang, X., "A-SDF: Learning Disentangled Signed Distance Functions for Articulated Shape Representation," in *ICCV*, 2021 (cit. on p. 12).

[109] Müller, T., Evans, A., Schied, C., and Keller, A., "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ToG*, vol. 41, no. 4, 102:1–102:15, Jul. 2022 (cit. on pp. 4, 34, 67, 69, 70, 76, 88).

[110] Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., and Theobalt, C., "Sparse Localized Deformation Components," *ACM TOG*, vol. 32, no. 6, pp. 1–10, 2013 (cit. on p. 10).

[111] Niedermayr, S., Stumpfegger, J., and Westermann, R., "Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis," in *CVPR*, 2024, pp. 10 349–10 358 (cit. on pp. 68, 70).

[112] Niemeyer, M., Manhardt, F., Rakotosaona, M.-J., Oechsle, M., Duckworth, D., Gosula, R., Tateno, K., Bates, J., Kaeser, D., and Tombari, F., "RadSplat: Radiance Field-Informed Gaussian Splatting for Robust Real-Time Rendering with 900+ FPS," in *ECCV*, 2024 (cit. on pp. 68, 70).

[113] Nießner, M., Loop, C., Meyer, M., and Derose, T., "Feature-adaptive GPU rendering of Catmull-Clark subdivision surfaces," *TOG*, vol. 31, no. 1, pp. 1–11, 2012 (cit. on p. 70).

[114] Noguchi, A., Iqbal, U., Tremblay, J., Harada, T., and Gallo, O., "Watch It Move: Unsupervised Discovery of 3D Joints for Re-Posing of Articulated Objects," in *CVPR*, 2022, pp. 3677–3687 (cit. on p. 32).

[115] Noguchi, A., Sun, X., Lin, S., and Harada, T., "Neural Articulated Radiance Field," in *ICCV*, 2021 (cit. on p. 11).

[116] Oat, C., "Animated Wrinkle Maps," in *SIGGRAPH*, ser. SIGGRAPH '07, San Diego, California: Association for Computing Machinery, 2007, pp. 33–37, ISBN: 9781450318235 (cit. on p. 35).

[117] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S., "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *CVPR*, 2019 (cit. on pp. 13, 14).

[118] Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R., "Deformable Neural Radiance Fields," in *ICCV*, 2021 (cit. on pp. 9, 11, 12, 14, 16, 17, 21).

[119] Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R., "Nerfies: Deformable Neural Radiance Fields," in *ICCV*, 2021, pp. 5865–5874 (cit. on pp. 3, 31–33, 39–41, 47).

[120] Park, K., Sinha, U., Hedman, P., Barron, J. T., Bouaziz, S., Goldman, D. B., Martin-Brualla, R., and Seitz, S. M., "HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields," *ACM TOG*, vol. 40, no. 6, 2021 (cit. on pp. 3, 5, 9–14, 16, 21, 26, 32, 33, 39–41, 47).

[121] Patow, G. and Pueyo, X., "A Survey of Inverse Rendering Problems," in *Computer graphics forum*, Wiley Online Library, vol. 22, 2003, pp. 663–687 (cit. on p. 4).

[122] Philip, J., Gharbi, M., Zhou, T., Efros, A. A., and Drettakis, G., "Multi-view relighting using a geometry-aware network.," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 78–1, 2019 (cit. on pp. 51, 58).

[123] Poole, B., Jain, A., Barron, J. T., and Mildenhall, B., "DreamFusion: Text-to-3D using 2D Diffusion," in *ICLR*, 2023. [Online]. Available: https://openreview.net/forum?id=FjNys5c7VyY (cit. on p. 89).

[124] Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F., "D-NeRF: Neural Radiance Fields for Dynamic Scenes," in *CVPR*, 2021, pp. 10 318–10 327 (cit. on pp. 3, 33).

[125] Pun, A., Sun, G., Wang, J., Chen, Y., Yang, Z., Manivasagam, S., Ma, W.-C., and Urtasun, R., "Neural Lighting Simulation for Urban Scenes," in *NeurIPS*, 2023 (cit. on p. 51).

[126] Radl, L., Steiner, M., Parger, M., Weinrauch, A., Kerbl, B., and Steinberger, M., "StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering," *TOG*, vol. 43, no. 4, pp. 1–17, 2024 (cit. on p. 70).

[127] Raj, A., Zollhofer, M., Simon, T., Saragih, J., Saito, S., Hays, J., and Lombardi, S., "Pixel-aligned Volumetric Avatars," in *CVPR*, 2021, pp. 11 733–11 742 (cit. on p. 71).

[128] Ramamoorthi, R. and Hanrahan, P., "An efficient representation for irradiance environment maps," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 497–500 (cit. on pp. 6, 49, 51, 54, 55).

[129] Raman, C., Hewitt, C., Wood, E., and Baltrusaitis, T., "Mesh-Tension Driven Expression-Based Wrinkles for Synthetic Faces," in *WACV Workshop on Applications of Computer Vision*, 2023 (cit. on p. 31).

[130] Reiser, C., Peng, S., Liao, Y., and Geiger, A., "KiloNeRF: Speeding Up Neural Radiance Fields With Thousands of Tiny MLPs," in *ICCV*, 2021, pp. 14 335–14 345 (cit. on p. 4).

[131] Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., and Hedman, P., "MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes," *TOG*, vol. 42, no. 4, pp. 1–12, 2023 (cit. on p. 70).

[132] Rematas, K., Liu, A., Srinivasan, P., Barron, J., Tagliasacchi, A., Funkhouser, T., and Ferrari, V., "Urban Radiance Fields," in *CVPR*, 2022, pp. 12 932–12 942 (cit. on p. 31).

[133] Ren, K., Jiang, L., Lu, T., Yu, M., Xu, L., Ni, Z., and Dai, B., "Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians," *arXiv preprint arXiv:2403.17898*, 2024 (cit. on pp. 70–73, 77–80, 84, 85).

[134] Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X., "A Survey of Deep Active Learning," 2020 (cit. on p. 28).

[135] Rudnev, V., Elgharib, M., Smith, W., Liu, L., Golyanik, V., and Theobalt, C., "Neural Radiance Fields for Outdoor Scene Relighting," in *European Conference on Computer Vision*, Springer, 2022, pp. 615–631 (cit. on pp. 4, 49–51, 56, 58–60, 62, 64).

[136] Saito, S., Schwartz, G., Simon, T., Li, J., and Nam, G., "Relightable Gaussian Codec Avatars," in *CVPR*, 2024, pp. 130–141 (cit. on pp. 52, 71, 87).

[137] Saito, S., Yang, J., Ma, Q., and Black, M. J., "SCANimate: Weakly Supervised Learning of Skinned Clothed Avatar Networks," in *CVPR*, 2021 (cit. on p. 12).

[138] Saswat Mallick and Rahul Goel, Kerbl, B., Vicente Carrasco, F., Steinberger, M., and De La Torre, F., "Taming 3DGS: High-Quality Radiance Fields with Limited Resources," in *SIGGRAPH Asia 2024 Conference Papers*, 2024. DOI: 10.1145/3680528.3687694. [Online]. Available: https://humansensinglab.github.io/taming-3dgs/ (cit. on pp. 70, 73).

[139] Schödl, A. and Essa, I. A., "Controlled Animation of Video Sprites," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '02, San Antonio, Texas: Association for Computing Machinery, 2002, pp. 121–127, ISBN: 1581135734. DOI: 10.1145/545261.545281. [Online]. Available: https://doi.org/10.1145/545261.545281 (cit. on p. 10).

[140] Schonberger, J. L. and Frahm, J.-M., "Structure-from-Motion Revisited," in *CVPR*, 2016, pp. 4104–4113 (cit. on p. 71).

[141] Schwarz, K., Liao, Y., Niemeyer, M., and Geiger, A., "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis," in *NeurIPS*, 2020 (cit. on p. 11).

[142] Seo, Y., Choi, Y. S., Son, H. S., and Uh, Y., "FLoD: Integrating Flexible Level of Detail into 3D Gaussian Splatting for Customizable Rendering," *arXiv preprint arXiv:2408.12894*, 2024 (cit. on pp. 68, 71–73, 75, 77–81, 84, 85).

[143] Shi, Y., Wu, Y., Wu, C., Liu, X., Zhao, C., Feng, H., Liu, J., Zhang, L., Zhang, J., Zhou, B., et al., "GIR: 3D Gaussian Inverse Rendering for Relightable Scene Factorization," *arXiv preprint arXiv:2312.05133*, 2023 (cit. on p. 52).

[144] Shi, Y., Gasparini, S., Morin, G., and Ooi, W. T., "LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming," *arXiv preprint arXiv:2408.14823*, 2024 (cit. on p. 68).

[145] Slomp, M. P. B., Oliveira Neto, M. M. d., and Patrício, D. I., "A gentle introduction to precomputed radiance transfer," *Revista de informática teórica e aplicada. Porto Alegre. Vol. 13, n. 2 (2006), p. 131-160*, 2006 (cit. on pp. 6, 54).

[146] Spurek, P., Winczowski, S., Zięba, M., Trzciński, T., **Kania**, **K.**, and Mazur, M., "Modeling 3D Surfaces with a Locally Conditioned Atlas," in *ICCS*, Springer, 2024, pp. 100–115 (cit. on p. 8).

[147] Srinivasan, P. P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., and Barron, J. T., "NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis," in *CVPR*, 2021, pp. 7495–7504 (cit. on p. 51).

[148] Stypułkowski, M., **Kania**, **K.**, Zamorski, M., Zięba, M., Trzciński, T., and Chorowski, J., "Representing Point Clouds with Generative Conditional Invertible Flow Networks," *Pattern Recognition Letters*, vol. 150, pp. 26–32, 2021 (cit. on p. 8).

[149] Su, S.-Y., Yu, F., Zollhöfer, M., and Rhodin, H., "A-NeRF: A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose," in *NeurIPS*, 2021 (cit. on p. 11).

[150] Suhail, M., Esteves, C., Sigal, L., and Makadia, A., "Light Field Neural Rendering," in *CVPR*, 2022, pp. 8269–8279 (cit. on p. 31).

[151] Sun, C., Sun, M., and Chen, H., "Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction," in *CVPR*, 2021, pp. 5449–5459 (cit. on p. 33).

[152] Sun, J.-M., Wu, T., Yang, Y.-L., Lai, Y.-K., and Gao, L., "SOL-NeRF: Sunlight Modeling for Outdoor Scene Decomposition and Relighting," in *SIGGRAPH Asia 2023 Conference Papers*, ser. SA '23, Sydney, NSW, Australia: Association for Computing Machinery, 2023, ISBN: 9798400703157. DOI: 10.1145/3610548.3618143. [Online]. Available: https://doi.org/10.1145/3610548.3618143 (cit. on pp. 51, 58, 60, 88).

[153] Sun, J., Wang, X., Zhang, Y., Li, X., Zhang, Q., Liu, Y., and Wang, J., "FENeRF: Face Editing in Neural Radiance Fields," in *CVPR*, 2022, pp. 7672–7682 (cit. on p. 32).

[154] Sun, T., Lin, K.-E., Bi, S., Xu, Z., and Ramamoorthi, R., "NeLF: Neural Light-transport Field for Portrait View Synthesis and Relighting," in *Eurographics Symposium on Rendering*, 2021 (cit. on p. 51).

[155] Sunkavalli, K., Matusik, W., Pfister, H., and Rusinkiewicz, S., "Factored time-lapse video," in *ACM SIGGRAPH 2007 papers*, 2007, 101–es (cit. on p. 51).

[156] *Suzanne 3D model*, https://github.com/OpenGLInsights/OpenGLInsightsCode/blob/master/Chapter%2026%20Indexing%20Multiple%20Vertex%20Arrays/article/suzanne.obj, Accessed: 2021-11-16 (cit. on p. 20).

[157] Tagliasacchi, A. and Mildenhall, B., "Volume Rendering Digest (for NeRF)," *arXiv preprint arXiv:2209.02417*, 2022 (cit. on p. 32).

[158] Takikawa, T., Evans, A., Tremblay, J., Müller, T., McGuire, M., Jacobson, A., and Fidler, S., "Variable Bitrate Neural Fields," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–9 (cit. on pp. 69, 71).

[159] Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., and Fidler, S., "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes," in *CVPR*, 2021, pp. 11 358–11 367 (cit. on pp. 70–72).

[160] Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P., Barron, J., and Kretzschmar, H., "Block-NeRF: Scalable Large Scene Neural View Synthesis," in *CVPR*, 2022, pp. 8248–8258 (cit. on p. 31).

[161] Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R., "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," in *NeurIPS*, 2020 (cit. on p. 5).

[162] Tang, J., Ren, J., Zhou, H., Liu, Z., and Zeng, G., "DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation," in *ICLR*, 2024 (cit. on pp. 88, 89).

[163] *Teapot 3D model*, https://graphics.stanford.edu/courses/cs148-10-summer/as3/code/as3/teapot.obj, Accessed: 2021-11-16 (cit. on p. 20).

[164] Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., et al., "Advances in Neural Rendering," in *CGF*, 2022 (cit. on p. 29).

[165] Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., and Nießner, M., "Face2Face: Real-time Face Capture and Reenactment of RGB Videos," in *CVPR*, 2016 (cit. on p. 11).

[166] Tretschk, E., Tewari, A., Golyanik, V., Zollhofer, M., Lassner, C., and Theobalt, C., "Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video," in *ICCV*, 2021 (cit. on pp. 12, 33).

[167] Trevithick, A. and Yang, B., "GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering," in *ICCV*, 2021 (cit. on p. 11).

[168] Troccoli, A. and Allen, P. K., "Relighting acquired models of outdoor scenes," in *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, IEEE, 2005, pp. 245–252 (cit. on p. 51).

[169] Vasconcelos, C. N., Oztireli, C., Matthews, M., Hashemi, M., Swersky, K., and Tagliasacchi, A., "CUF: Continuous Upsampling Filters," in *CVPR*, 2023, pp. 9999–10 008 (cit. on pp. 6, 76).

[170] Venter, H. and Ogterop, W., *Unreal Engine 5 Character Creation, Animation, and Cinematics: Create custom 3D assets and bring them to life in Unreal Engine 5 using MetaHuman, Lumen, and Nanite*. Packt Publishing Ltd, 2022 (cit. on p. 68).

[171] Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J., and Srinivasan, P., "Ref-NeRF: Structured View-dependent Appearance for Neural Radiance Fields," in *CVPR*, 2022, pp. 5481–5490 (cit. on p. 31).

[172] Vicini, D., Jakob, W., and Kaplanyan, A., "Non-exponential transmittance model for volumetric scene representations," *ToG*, vol. 40, no. 4, Jul. 2021, ISSN: 0730-0301 (cit. on p. 32).

[173] Wang, C., Chai, M., He, M., Chen, D., and Liao, J., "CLIP-NeRF: Text-and-Image Driven Manipulation of Neural Radiance Fields," in *CVPR*, 2022, pp. 3835–3844 (cit. on p. 32).

[174] Wang, D., Chandran, P., Zoss, G., Bradley, D., and Gotardo, P., "MoRF: Morphable Radiance Fields for Multiview Neural Head Modeling," in *SIGGRAPH*, ser. SIGGRAPH '22, Vancouver, BC, Canada: Association for Computing Machinery, 2022, ISBN: 9781450393379 (cit. on p. 33).

[175] Wang, L., Zhang, J., Liu, X., Zhao, F., Zhang, Y., Zhang, Y., Wu, M., Yu, J., and Xu, L., "Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-time," in *CVPR*, 2022, pp. 13 524–13 534 (cit. on p. 32).

[176] Wang, Y., Wang, J., and Qi, Y., "We-gs: An in-the-wild efficient 3d gaussian representation for unconstrained photo collections," *arXiv preprint arXiv:2406.02407*, 2024 (cit. on p. 52).

[177] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: from error visibility to structural similarity," *IEEE TIP*, vol. 13, no. 4, pp. 600–612, 2004 (cit. on pp. 58, 78).

[178] Wang, Z., Simoncelli, E. P., and Bovik, A. C., "Multiscale Structural Similarity for Image Quality Assessment," in *Asilomar Conference on Signals, Systems & Computers*, 2003 (cit. on pp. 22, 39).

[179] Wang, Z., Shen, T., Gao, J., Huang, S., Munkberg, J., Hasselgren, J., Gojcic, Z., Chen, W., and Fidler, S., "Neural Fields meet Explicit Geometric Representations for Inverse Rendering of Urban Scenes," in *CVPR*, Jun. 2023 (cit. on pp. 49, 51, 58).

[180] Weng, C.-Y., Curless, B., Srinivasan, P., Barron, J., and Kemelmacher-Shlizerman, I., "HumanNeRF: Free-viewpoint Rendering of Moving People from Monocular Video," in *CVPR*, 2022, pp. 16 210–16 220 (cit. on p. 32).

[181] Whitted, T., "An improved illumination model for shaded display," in *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, 1979, p. 14 (cit. on p. 49).

[182] Wood, E., Baltrušaitis, T., Hewitt, C., Dziadzio, S., Cashman, T. J., and Shotton, J., "Fake It Till You Make It: Face analysis in the wild using synthetic data alone," in *CVPR*, 2021, pp. 3681–3691 (cit. on pp. 35, 38, 39).

[183] Wood, E., Baltrušaitis, T., Hewitt, C., Johnson, M., Shen, J., Milosavljević, N., Wilde, D., Garbin, S., Sharp, T., Stojiljković, I., et al., "3D Face Reconstruction with Dense Landmarks," in *ECCV*, Springer, 2022, pp. 160–177 (cit. on pp. 33, 38).

[184] Wu, C., Bradley, D., Gross, M., and Beeler, T., "An Anatomically-Constrained Local Deformation Model for Monocular Face Capture," *ACM TOG*, vol. 35, no. 4, pp. 1–12, 2016 (cit. on pp. 10, 11).

[185] Wuu, C.-h., Zheng, N., Ardisson, S., Bali, R., Belko, D., Brockmeyer, E., Evans, L., Godisart, T., Ha, H., Hypes, A., Koska, T., Krenn, S., Lombardi, S., Luo, X., McPhail, K., Millerschoen, L., Perdoch, M., Pitts, M., Richard, A., Saragih, J., Saragih, J., Shiratori, T., Simon, T., Stewart, M., Trimble, A., Weng, X., Whitewolf, D., Wu, C., Yu, S.-I., and Sheikh, Y., "Multiface: A Dataset for Neural Face Rendering," in *arXiv*, 2022 (cit. on pp. 37, 39, 40, 46).

[186] Xia, S., Yue, J., **Kania**, **K.**, Fang, L., Tagliasacchi, A., Yi, K. M., and Sun, W., "Densify Your Labels: Unsupervised Clustering with Bipartite Matching for Weakly Supervised Point Cloud Segmentation," *arXiv preprint arXiv:2312.06799*, 2023 (cit. on p. 8).

[187] Xian, W., Huang, J.-B., Kopf, J., and Kim, C., "Space-time Neural Irradiance Fields for Free-viewpoint Video," in *CVPR*, 2021, pp. 9421–9431 (cit. on p. 32).

[188] Xiangli, Y., Xu, L., Pan, X., Zhao, N., Rao, A., Theobalt, C., Dai, B., and Lin, D., "BungeeNeRF: Progressive Neural Radiance Field for Extreme Multi-scale Scene Rendering," in *ECCV*, 2022, pp. 106–122 (cit. on p. 31).

[189] Xie, C., Park, K., Martin-Brualla, R., and Brown, M., "FiG-NeRF: Figure-Ground Neural Radiance Fields for 3D Object Category Modelling," 2021 (cit. on pp. 3, 11).

[190] Xie, S., Zhu, H., Liu, Z., Zhang, Q., Zhou, Y., Cao, X., and Ma, Z., "DINER: Disorder-Invariant Implicit Neural Representation," in *CVPR*, 2023, pp. 6143–6152 (cit. on p. 73).

[191] Xing, G., Zhou, X., Peng, Q., Liu, Y., and Qin, X., "Lighting simulation of augmented outdoor scene based on a legacy photograph," in *Computer Graphics Forum*, Wiley Online Library, vol. 32, 2013, pp. 101–110 (cit. on p. 51).

[192] Xu, J., Mei, Y., and Patel, V. M., "Wild-GS: Real-Time Novel View Synthesis from Unconstrained Photo Collections," *arXiv preprint arXiv:2406.10373*, 2024 (cit. on pp. 52, 63).

[193] Xu, K. and Campeanuy, D., "Houdini engine: Evolution towards a procedural pipeline," in *Proceedings of the Fourth Symposium on Digital Production*, 2014, pp. 13–18 (cit. on p. 42).

[194] Xu, T.-X., Hu, W., Lai, Y.-K., Shan, Y., and Zhang, S.-H., "Texture-GS: Disentangling the Geometry and Texture for 3D Gaussian Splatting Editing," in *ECCV*, 2024 (cit. on p. 71).

[195] Xu, T. and Harada, T., "Deforming Radiance Fields with Cages," in *ECCV*, 2022 (cit. on pp. 31, 33).

[196] Xu, Y., Wang, L., Zhao, X., Zhang, H., and Liu, Y., "AvatarMAV: Fast 3D Head Avatar Reconstruction Using Motion-Aware Neural Voxels," in *SIGGRAPH*, 2023, pp. 1–10 (cit. on p. 34).

[197] Yang, B., Bao, C., Zeng, J., Bao, H., Zhang, Y., Cui, Z., and Zhang, G., "NeuMesh: Learning Disentangled Neural Mesh-based Implicit Field for Geometry and Texture Editing," in *ECCV*, 2022, pp. 597–614 (cit. on pp. 32, 33).

[198] Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., and Cui, Z., "Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering," in *ICCV*, 2021 (cit. on pp. 9, 11).

[199] Yang, S., Cui, X., Zhu, Y., Tang, J., Li, S., Yu, Z., and Shi, B., "Complementary intrinsics from neural radiance fields and cnns for outdoor scene relighting," in *CVPR*, 2023, pp. 16 600–16 609 (cit. on p. 51).

[200] Yang, Y., Zhang, S., Huang, Z., Zhang, Y., and Tan, M., "Cross-Ray Neural Radiance Fields for Novel-view Synthesis from Unconstrained Image Collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 15 901–15 911 (cit. on pp. 4, 50, 51).

[201] Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., and Quan, L., "BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1790–1799 (cit. on pp. 78–80, 85).

[202] Ye, V., Li, R., Kerr, J., Turkulainen, M., Yi, B., Pan, Z., Seiskari, O., Ye, J., Hu, J., Tancik, M., and Kanazawa, A., "gsplat: An Open-Source Library for Gaussian Splatting," *arXiv preprint arXiv:2409.06765*, 2024. arXiv: 2409.06765 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2409.06765 (cit. on p. 77).

[203] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A., "Plenoctrees for Real-time Rendering of Neural Radiance Fields," in *ICCV*, 2021, pp. 5752–5761 (cit. on pp. 4, 32, 33, 69).

[204] Yu, H., Julin, J., Milacski, Z. Á., Niinuma, K., and Jeni, L. A., "CoGS: Controllable Gaussian Splatting," in *CVPR*, 2024, pp. 21 624–21 633 (cit. on p. 88).

[205] Yu, H.-X., Guibas, L. J., and Wu, J., "Unsupervised Discovery of Object Radiance Fields," 2021 (cit. on p. 11).

[206] Yu, Y., Meka, A., Elgharib, M., Seidel, H.-P., Theobalt, C., and Smith, W. A., "Self-supervised outdoor scene relighting," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, Springer, 2020, pp. 84–101 (cit. on pp. 51, 58, 60).

[207] Yu, Z., Chen, A., Huang, B., Sattler, T., and Geiger, A., "Mip-Splatting: Alias-free 3D Gaussian Splatting," in *CVPR*, 2024, pp. 19 447–19 456 (cit. on p. 70).

[208] Yuan, Y.-J., Sun, Y.-T., Lai, Y.-K., Ma, Y., Jia, R., and Gao, L., "NeRF-Editing: Geometry Editing of Neural Radiance Fields," in *CVPR*, 2022, pp. 18 353–18 364 (cit. on pp. 31–33).

[209] Zeng, C., Chen, G., Dong, Y., Peers, P., Wu, H., and Tong, X., "Relighting Neural Radiance Fields with Shadow and Highlight Hints," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023 (cit. on p. 51).

[210] Zhang, D., Wang, C., Wang, W., Li, P., Qin, M., and Wang, H., "Gaussian in the wild: 3d gaussian splatting for unconstrained image collections," in *European Conference on Computer Vision*, Springer, 2025, pp. 341–359 (cit. on pp. 52, 53, 63).

[211] Zhang, J., Jiang, Z., Yang, D., Xu, H., Shi, Y., Song, G., Xu, Z., Wang, X., and Feng, J., "AvatarGen: A 3D Generative Model for Animatable Human Avatars," in *ECCV*, 2022, pp. 668–685 (cit. on p. 29).

[212] Zhang, K., Riegler, G., Snavely, N., and Koltun, V., "NeRF++: Analyzing and Improving Neural Radiance Fields," 2020 (cit. on pp. 9, 11, 31).

[213] Zhang, L., Han, Y., Lin, W., Ling, J., and Xu, F., *PRTGaussian: Efficient Relighting Using 3D Gaussians with Precomputed Radiance Transfer*, 2024. arXiv: 2408.05631 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.05631 (cit. on p. 52).

[214] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O., "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *CVPR*, 2018 (cit. on p. 22).

[215] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O., "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *CVPR*, 2018 (cit. on pp. 40, 78).

[216] Zhang, S., Moscovich, A., and Singer, A., "Product Manifold Learning," in *Inter. Conf. on Artif. Intell. and Stat.*, 2021 (cit. on p. 27).

[217] Zhang, X., Srinivasan, P. P., Deng, B., Debevec, P., Freeman, W. T., and Barron, J. T., "NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination," *ACM TOG*, vol. 40, no. 6, pp. 1–18, 2021 (cit. on pp. 5, 9, 11, 51).

[218] Zhao, F., Yang, W., Zhang, J., Lin, P., Zhang, Y., Yu, J., and Xu, L., "HumanNeRF: Efficiently Generated Human Radiance Field from Sparse Inputs," in *CVPR*, 2022, pp. 7743–7753 (cit. on p. 32).

[219] Zheng, Z., Yu, T., Liu, Y., and Dai, Q., "PaMIR: Parametric Model-Conditioned Implicit Representation for Image-based Human Reconstruction," *IEEE TPAMI*, 2021 (cit. on p. 12).

[220] Zhi, Y., Qian, S., Yan, X., and Gao, S., "Dual-Space NeRF: Learning Animatable Avatars and Scene Lighting in Separate Spaces," in *3DV*, IEEE, 2022, pp. 1–10 (cit. on p. 29).

[221] Zhuang, Y., Zhu, H., Sun, X., and Cao, X., "MoFaNeRF: Morphable Facial Neural Radiance Field," in *ECCV*, 2022, pp. 268–285 (cit. on p. 33).

[222] Zielonka, W., Bolkart, T., and Thies, J., "Towards Metrical Reconstruction of Human Faces," in *ECCV*, Springer, 2022, pp. 250–269 (cit. on p. 3).

[223] Zielonka, W., Bolkart, T., and Thies, J., "Instant Volumetric Head Avatars," in *CVPR*, 2023, pp. 4574–4584 (cit. on pp. 3, 34).

[224] Zins, P., Xu, Y., Boyer, E., Wuhrer, S., and Tung, T., "Data-Driven 3D Reconstruction of Dressed Humans From Sparse Views," 2021 (cit. on p. 12).

[225] Zwicker, M., Pfister, H., Van Baar, J., and Gross, M., "EWA volume splatting," in *Proceedings Visualization, 2001. VIS'01.*, IEEE, 2001, pp. 29–538 (cit. on pp. 1, 52, 72).

List of Figures

List of Tables