Warsaw University of Technology

FACULTY OF ELECTRONICS AND INFORMATION TECHNOLOGY

PhD Thesis

in the discipline of Information and Communication Technology

Few-Shot Human Neural Rendering with Partial Information

Kacper Kania, M.Sc.

supervisor
Tomasz Trzciński, Prof. PhD DSc.

assistant supervisor
Marek Kowalski, PhD DSc.

WARSZAWA 2025

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Abstract

This thesis is a series of publications that introduce novel methods for human 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 human renderings. We propose techniques that use simple, interpretable inputs derived from sparse training data and extends these methods to perform effectively in few-shot learning scenarios.

We begin by examining the field of neural radiance fields, addressing limitations in existing approaches and presenting contributions to controllable radiance fields. By incorporating partial and sparse data during training, it leverages the smoothness of neural networks to produce controllable, high-quality human images.

To tackle the reliance on extensive, high-quality data annotations from multi-view videos, we introduce a new method for training neural radiance fields in few-shot, multi-view settings. This approach learns internal deformation templates, which blend smoothly during inference, significantly improving image quality compared to existing baselines and enabling effective human rendering from limited input images.

The work also addresses the need for adaptable computational efficiency during inference. It proposes a fine-to-coarse learning strategy for 3D Gaussian Splatting, which upscales a latent 2D grid that stores Gaussian representations. This strategy achieves competitive results while allowing deployment on various computational devices with minimal quality loss.

In addition, we develop a novel model for controlling radiance fields through environmental lighting. By incorporating precomputed radiance transfer, this model enables physically plausible scene relighting and provides users with intuitive control over lighting in reconstructed scenes.

This research advances the state of the art in controllable neural radiance fields and expands their application to few-shot learning scenarios. These innovations enhance the possibilities for human rendering 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

To jest streszczenie. To jest trochę za krótkie, jako że powinno zająć całą stronę.

Słowa kluczowe: A, B, C

Lay Summary

ok

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	CVPR 2025	Under Review

Contents

List of Abbreviations and Symbols

List of Figures

List of Tables

π	Stała matematyczna równa stosunkowi obwodu okręgu do jego średnicy
I	Natężenie prądu elektrycznego

Chapter 1

Introduction

With the advent of deep learning, research have been exploring varying ways to apply it to computer graphics. One of the most recent and promising approaches is neural rendering. Neural rendering is a field that combines deep learning and computer graphics to generate realistic images of 3D scenes. The neural radiance field (NeRF) is a popular neural rendering technique that represents a 3D scene as a continuous function that maps 3D coordinates to radiance values. NeRF has shown impressive results in generating photorealistic images of 3D scenes. However, NeRF has limitations in terms of memory and computational requirements, which makes it difficult to scale to large scenes.

To alleviate the problem, Kerbl *et al.* [30] proposed a new technique—3D Gaussian Splatting (3DGS). 3DGS is a neural rendering technique that represents a 3D scene as a set of 3D Gaussian that are splatted to an image space using algorithm proposed by Zwicker *et al.* [89]. In contrast to NeRF, 3DGS is more memory efficient and can be used to render large scenes. It can also render scenes with millions of points in real-time on a single GPU.

In this thesis, we focus on those two milestone techniques in neural rendering and address their fundamental problem—lack of controllability.

1.1 Motivation and challenges

NeRF and 3DGS are both impressive techniques that can generate realistic images. However, a single scene representation needs to be trained on a high-end GPU for hours or even days just to render a novel view at the inference time. However, any type of controllability is difficult to achieve with those models. That includes changing the lighting conditions, subject's attributes or even the scene itself. We see imbuing those models with controllability as a an important step towards making them more useful in practice. Our proposed models are designed to address this issue.

One may ask why the controllability is a feat sought after to be researched. We see the inspiration in how human artists work. Imagine an artist working on 3D game where they

need an asset, like a 3D mesh, to be created. Such a mesh takes much effort since it includes modeling, creating a UV map which can then be textured. After the process is finish, the artist's supervisor may task him to change the model to some extent which requires the artist to redo all the effort again. Such a process is not limited to 3D assets as meshes and could be applied to 3DGS or NeRF. However, 3DGS and NeRFs are volumetric in nature. Our exploited and well-established practices no longer apply to them since volumetric representations do not have the underlying surface representation. For that reason, we see a couple of avenues which we explore in this thesis.

Firstly, Park *et al.* [50] proposed NeRFies, a model that creates a volumetric representation of a person from a self-captured sequence with a phone camera. Since the inception of NeRFs [43], it was among the first works the achieved such a high quality of reconstructions from a casual videos. In its primal form, NeRFies were unable to control the avatar in any other way than by a linear interpolation of latent embeddings that embedded the video's time dimension. The follow-up work, HyperNeRF [51] handles this issue by projecting the learnable embeddings with D onto a lower-dimensional space $\mathbb{R}^d$ where $d \ll D$. After the assumption that the $d=2$ is enough to explain the sequence variability, that projected embedding becomes a 2-dimensional space that can be traversed in an interpretable way. However, that space is not intuitive since the projection is a non-linear operation and one cannot predict how values affect the results. To mitigate that issue, we propose to leverage smoothness of Multilayer Perceptrons (MLPs) [50, 72] to constrain the projection via sparse supervision. We realize our approach as a weakly-supervised MLP that out of many images from the sequence (we assume at least 100 frames in our work) only a few are provided with a coarse annotation. Such annotations denote what values a chosen attribute takes and where its effect spans in the image space. We show that our method, which we dubbed CoNeRF [28] and published at the CVPR 2022 conference, imbues NeRFs with a flexible editability feature without the lose of the rendering quality.

Secondly, approaches such as CoNeRF [28], EditNeRF [36] or FigNeRF [77] focus solely on static elements of the scene, hence their controllability is limited to changing colors or textures in general. HyperNeRF [51] arises as a potential solution due to its ability to model object deformations. However, our initial experiments showed that those changes cannot handle motions that affect a subject globally, *e.g.*, jumping jacks performed by a person. To solve the issue, Fang *et al.* [12] proposes to model the deformation via a multi-scale voxel structure which works well in the synthetic setting, such as the one proposed by Pumarola *et al.* [54].

There exists a plethora of works that approach the problem from the another angle—instead of modeling the motion purely from data, they use a template model in the form of a 3D mesh to canonicalize deformed points [88]. Such methods rely on the accuracy of the *registration*, *i.e.*, fitting the template mesh to subject. Since the registration methods [13, 87] are imperfect estimators, they inherently contain registration errors. Those deviations are exacerbated by learnable radiance field models which assume a perfectly calibrated scene. The authors of those approaches usually mitigate the issue with additional latent space [19, 28, 41] that requires

thousands of video frames to learn an avatar of high-fidelity that reacts correctly to deformations such as wrinkles on the forehead. At the same time, performing the registration on the large scale is costly [5]. In this thesis, we seek a remedy for those obstacles. We propose a method that is data-efficient, easy to improve with a minimal user input and can model realistic deformation dependent changes in the subject. Inspired by classical methods in character texturing [49] and motion modeling [34], we propose BlendFields [25], an *homage* to traditional blendshapes [34]. We build on VolTeMorph's [16] approach to point canonicalization to provide a data-efficient way to control the character. We further introduce a physically-based mixture of predefined, learned from data wrinkle templates that represent expression-dependent skin deformations. Our proposed was acknowledged by the reviewers and was accepted to the CVPR 2023 conference.

Thirdly, having the texture and coarse mesh-based controllability, we strive for control scene settings directly. The inverse rendering of 3D scenes is an ill-posed problem where many different lighting settings may explain the same light effects [52]. To facilitate solving the problem, many approaches use datasets of single object's images captured under different lighting conditions [6, 61, 80]. These approaches cannot decouple albedo from the lighting effects [6, 80] or need additional neural networks to predict correct shadows [61] which limits methods' practicality. We propose to use recently proposed 2D Gaussian Splatting [22] which exhibit remarkable quality of the surface reconstruction. Together with our precomputed radiance transfer from classical computer graphics approaches [57, 67], our LumiGauss achieves state-of-the-art reconstruction quality with the ability to render novel lighting conditions with high fidelity. Our work received positive reviews for the WACV 2025 conference.

Finally, volumetric representation are computationally intensive to render, compared to the traditional mesh representation. For NeRFs [43], it takes seven days on V100 NVidia GPU to train for single scene, and more than 60 seconds to render a single image—way beyond any practical applications. Although many approaches have been proposed to speed up the rendering process [17, 21, 45, 58, 82], they usually make a trade-off between memory requirements, quality, and rendering speed. 3D Gaussian Splatting [30] (3DGS) rose as an alternative to NeRF, offering both high-quality rendering at interactive frame rates. However, those frame rates could be achieved with the most advanced GPU units available at that time. As we see the potential in 3DGS to be a viable canonical representation for 3D data, akin to 3D meshes, a need for its adaptability to different computational resources exists. Meshes can be adapted easily with levels of detail (LoD) approaches that remove detail from meshes that do not affect the general object's perception if necessary.

1.2 Research objectives

In this thesis, we explore different avenues of radiance field controllability. With this goal in mind, we aim at answering the following research questions:

(**RQ 1**) Can we imbue a Neural Radiance Field (NeRF) with a controllability by providing sparse annotations to the training dataset? How many annotations suffice to learn smooth interpolation capabilities between controlled values?

(**RQ 2**) Are extreme facial expressions known from the literature sufficient to learn expression-dependent details that extrapolate to expressions unseen at the training time?

(**RQ 3**) Is it possible to learn an underlying radiance transfer function of a scene from images taken in "in-the-wild" setting? Can such a transfer function generalize to unknown environment maps?

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

Each of these questions is a representative of possible among many others controllability directions for radiance fields. In case of this thesis, we present summarize our methods as controls of: texture [25, 28], shape [25], lightning [24] and use of resources [26]. To answer (**RQ 1**), we describe our CoNeRF [28]. It is one of the pioneering works that uses sparsely annotated frames to continuously control the subject in a post-hoc manner. We leverage the fact that MLP used in NeRFs are smooth functions biased towards low frequency signals [72]. For that reason, NeRF can learn to interpolate smoothly the annotation signal between frames of high similarity. We show that this assumption is sufficient to obtain both novel view synthesis and novel attribute synthesis with a single model.

We further move towards answering (**RQ 2**). We introduce BlendFields [25], achieving two primary goals: ability to generalize unknown expressions via a predefined face mesh template, and a mixture model of training expressions that can produce spatially coherent, expression-dependent wrinkles on the face from as few as three expressions. In our work, we build on VolTeMorph [16] to achieve to former, and focus on the latter contribution. Inspired by texture maps in classical computer graphics pipelines [49], we define a set of learnable radiance fields, each being overfit to a particular extreme expression from the training set. We define an extreme expression as one of the possible expressions involving the most facial muscles. Building on VolTeMorph [16] allows us to use an underlying tetrahedral mesh to compute physical quantities such as the volume change of tetrahedra under a given expression. We use those quantities to linearly interpolate between the pretrained radiance fields. We mix the tetrahedra independently which makes rendering novel expressions possible. For example, BlendFields can render one of the eyebrows raised while maintaining the other in a natural position, which is a difficult expression to make for majority of people.

Our LumiGauss [24] answers (**RQ 3**). In contrary to common approaches [61], we posit that the radiance transfer function known among computer graphics researchers [57] can be learned from unconstrained photo collection under varying lighting conditions. To this end, we use 2DGS [22] which gives us a smooth shape representation, difficult to achieve when

using 3DGS [30]. With the use of contributed priors, we induce learning Gaussian's Spherical Harmonics such that they correctly react to changing environment maps. Not only our approach is fast to train, but renders realistic scenes and object's shadows even under novel lighting conditions.

All the contributions so far are affected by a specific disadvantage—a single model needs to be trained from scratch and it can be deployed only on high-end hardware. We then ask if we can train a single model that is adaptable at inference time to different hardware regimes (**RQ 4**). We propose CLoG [26] as a potential remedy. Our approach uses the fact that one can constrain the number of Gaussians in 3DGS [30] to a specific number, such that it can be formed as a 2D grid by simple reshape operation. With a specific training coarse-to-fine training protocol we contributed, the model learns a representation that converges to a high-quality volumetric structure. In the second training stage, we leverage the fact that Gaussians's can be spatially sorted in such a manner that their descriptors are placed next to each other if they are similar. That forms a low-frequency image which can be modulated with an off-the-shelf continuous upsampling architecture [73]. The architecture outputs a new grid of the given resolution. We show in our work that such an approach achieves remarkable results and can output any number of Gaussians at inference time with high quality.

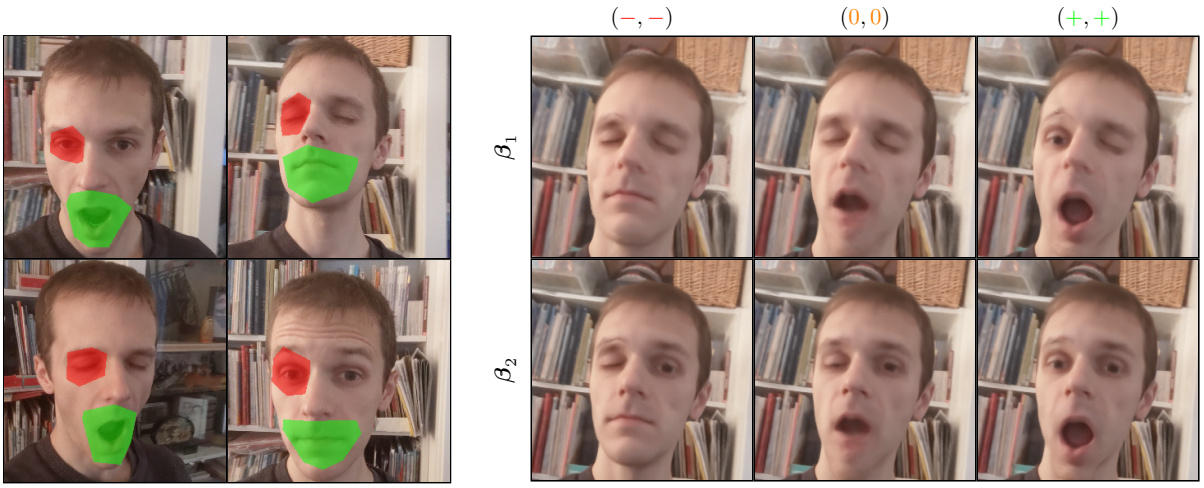
1.3 Contributions

Building on those questions, we structure this thesis in several chapters corresponding to the answers. An answer is in a form of a scientific article where we introduce the following:

- A novel approach for controlling trained radiance fields with the use of sparsely annotated images from a casually captured data.
- A new model capable of blending trained radiance fields from multi-view frames in an interpretable way and extrapolating to novel human expressions for the trained subject.
- The first use of Gaussian Splatting methods that learns a coherent shape representation of a subject and an ability to distill the varying lighting conditions in the data to a radiance transfer function.
- A novel paradigm for learning Gaussian Splatting models as 2D grids to achieve flexibility to adapt the model to different computational regimes at inference time.

1.3.1 Texture from Partial Information

Existing NeRF-based approaches in 2021 were simple models—they were overfit to a single subject, for a new subject the model needed to be retrained from scratch, and the editability



(a) annotated samples **(b)** renderings from the trained model

Figure 1. Example with annotations and controlled attributes – We show an example of how our CoNeRF is capable of controlling attributes selected sparse annotations at the training time. Top row shows possible value combinations (− denoting closed eye, 0 neutral position and + an open eye) and $\{\beta_1, \beta_2\}$ possible values of attributes that are not explicitly learned from the annotations but purely from data (please see Chapter 3 for further explanation).

capabilities were limited [36]. NeRFactor [86] could be considered a more sophisticated model. However, it worked only for simple scene with calibrated scenes and could not handle any motion.

We approach those issues in our CoNeRF [28] published at the CVPR 2021 conference. Inspired by HyperNeRF [51], we propose to revisit the weak supervision in the context of radiance fields. Specifically, under an assumption that one can provide a few of sparse annotations to the dataset, we can leverage the smoothness of the neural networks to propagate the annotation across the dataset. The annotations also consist of what regions in the image space it refers to and hence we can train a semantic segmentation radiance fields that decouples attribute controls. We show an example in Fig. 1 where the left side represents the annotations present in the data and the right one possible manipulations at the inference time. We note that those annotations are easy to make in a matter of a few minutes for a single dataset. However, the complexity of the annotations grows with the number of attributes to control.

1.3.2 Expression from Few-Shot Learning

CoNeRF [28] is capable of rendering complex motions given sufficient amount of data and provided annotations. However, motions as the ones a person performs daily when speaking are infeasible in practice. We propose a solution to target that issue. We introduce BlendFields [25] from the CVPR 2023 proceedings which learns motion-dependent face deformation from data. We build on VolTeMorph [16] which uses existing face template models, such as FLAME [35], to learn a single canonical representation, akin to the canonicalization module in CoNeRF. Internally, BlendFields creates a tetrahedral cage around the face model. For each sample

along the ray in NeRF, it moves the points to a "neutral position", chosen once prior to the training procedure. Such a procedure comes insufficient to model realistic facial features, such as wrinkles. We contribute a novel approach to modeling those deformations. We compute a deformation gradient of each tetrahedra for a given expression which is a physically-based and easy to interpret quantity. The value serves us to smoothly transition face regions textures to appropriate colors. We obtain the colors from NeRFs branches, each overfit to particular expression. In principle, BlendFields predicts face bases which are conditioned on the face expression vector to output the final point color. Our framework works well even when only a few "extreme" expressions are provided such as grinning face with closed eyes and wide open mouth with open eyes.

1.3.3 Light from Unconstrained Images

In both approaches above, we tackle the problem of texture controllability. They assume an ideal case scenario where a capture can be taken in an idealized environment with constant camera exposure and lighting. Moreover, the produced colors are blended together, making the change of light impossible in practice. We then ask the questions if we can decouple an intrinsic color of the subject and change of that color stemming from the environment light. We answer that question with our LumiGauss [24] published at the WACV 2025 conference that, indeed, we can achieve that by learning the radiance transfer function directly from data. For that end, we train a 2DGS [22] model to obtain a smooth and spatially coherent surface of objects. On top of the other attributes known from 3DGS [30], we imbue our Gaussians with additional features corresponding to the radiance transfer, expressed as Spherical Harmonics [20]. We show in our experiments that such a formulation is sufficient to train a model that reacts to changing environment maps in a realistic manner and renders images of higher fidelity than prior approaches.

1.3.4 Levels of Detail in One Model

All the contributions above require considerable computation hardware to be trained on and then run inference in near real-time frame rates. Drawing an inspiration from the gaming industry where Levels of Details (LoD) for meshes is used heavily, we introduce CLoG [26]. Our approach trains a single Gaussian Splatting model such that it can be modulated at inference time and adapted to the target computational requirements with a minimal loss on the rendering quality. That allows us to democratize the use of 3DGS and deploy a single model even on handheld devices. We show later that even under a strict case where only 2,000 Gaussians[1] can be used, the object is still recognizable. The most important contribution of our model is that it is continuous by design while the existing baselines assume prior the training the model how many LoD the model should comprise at inference. To change the size of particular LoD in those

[1] Common 3DGS models can achieve from 10^5 to even 10^6 Gaussians.

baselines requires training the whole model from scratch, imposing a significant computational burden.

1.4 Thesis outline

This thesis is structured as follows. We start by introducing the preliminaries related to the Neural Radiance Fields and Gaussian Splatting in Chapter 2—a common theme in all works that appear later. We then move towards describing CoNeRF [28] in Chapter 3, our approach to control trained neural radiance fields by using sparse annotations. In Chapter 4, we describe BlendFields [25] that can produce realistic expression-dependent texture from just a few multi-view frames of the subjects. Chapter 5 introduces the LumiGauss [24], a Gaussian Splatting model that learns a radiance transfer function for novel lighting rendering capabilities. Finally, we bring a general method, CLoG [26], that uses Gaussian Splatting to learn continuous levels of detail while training only a single model. We conclude the thesis in Chapter 7 where we also explore possible future avenues that can be undertaken.

1.5 Publications not included in the thesis

We attach a list of articles that are related to and can be used to in neural rendering approaches:

- **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

Background

2.1 Neural Rendering

2.2 Neural Radiance Field

2.3 3D Gaussian Splatting

Chapter 3

CoNeRF: Controllable Neural Radiance Fields

Chapter 4

BlendFields: Few-Shot Example-Driven Facial Modeling

Chapter 5

LumiGauss: Relightable Gaussian Splatting in the Wild

Figure 2. 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*).

Chapter 6

CLoG: Leveraging UV Space for Continuous Levels of Detail

6.1 Abstract

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 work, 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 UV 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 UV map 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 [43], DTU [1] and AVA [42] at with various number of Gaussians, outperforming the state-of-the-art 3D Gaussian Splatting representations that support levels of detail.

6.2 Introduction

Since the introduction of Neural Radiance Fields (NeRF) [43], 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

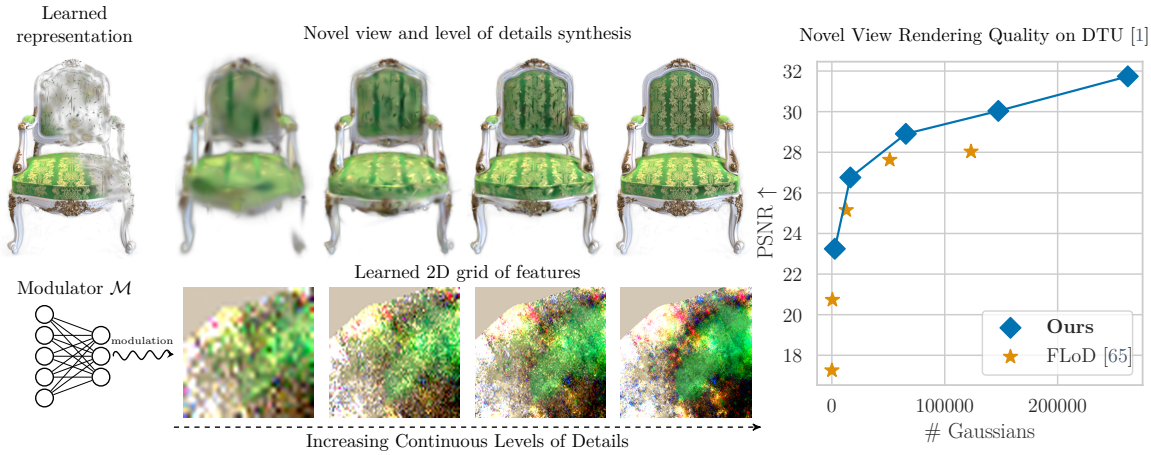


Figure 3. Teaser – We present a novel method to introduce continuous levels of detail (LoD) for Gaussian Splatting. We represent Gaussians as a set of vectors $\mathbf{\Lambda}$ embedded onto a 2D grid, which we can easily subsample to obtain a desired level of detail. To account for any artifacts that 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.

such as Multi-Layer Perceptrons (MLP) [43] and hash grids [45], and render them via volume rendering.

More recently, 3D Gaussian Splatting [30] 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 [10, 33, 46, 47], or concurrently to our work, via introducing Levels of Detail (LoD) [65, 66]. 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 [74] or streaming.

In this work, 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 UV mapping [14] of 3D Gaussians, representing them as uniform 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 UV 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 UV resolution. After training, because we rely on downsampling, which assumes nearby samples are alike, we sort [44] 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 UV 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 UV maps that support efficient subsampling for level-of-detail control.
- We develop a progressive trainin strategy that grows UV maps from noise, followed by spatial sorting for optimal structure. Additionally, we introduce a level-of-detail-conditioned modulator that refines downsampled UV maps for high-quality rendering.
- We demonstrate how our method dynamically adapts the underlying representation to accommodate the desired compute and memory budget.

6.3 Related Work

We first briefly discuss work on NeRFs [43] and 3D Gaussian Splatting [30], then discuss works that focus on levels of details.

Neural Radiance Fields. Neural Radiance Fields (NeRFs) [43] 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.

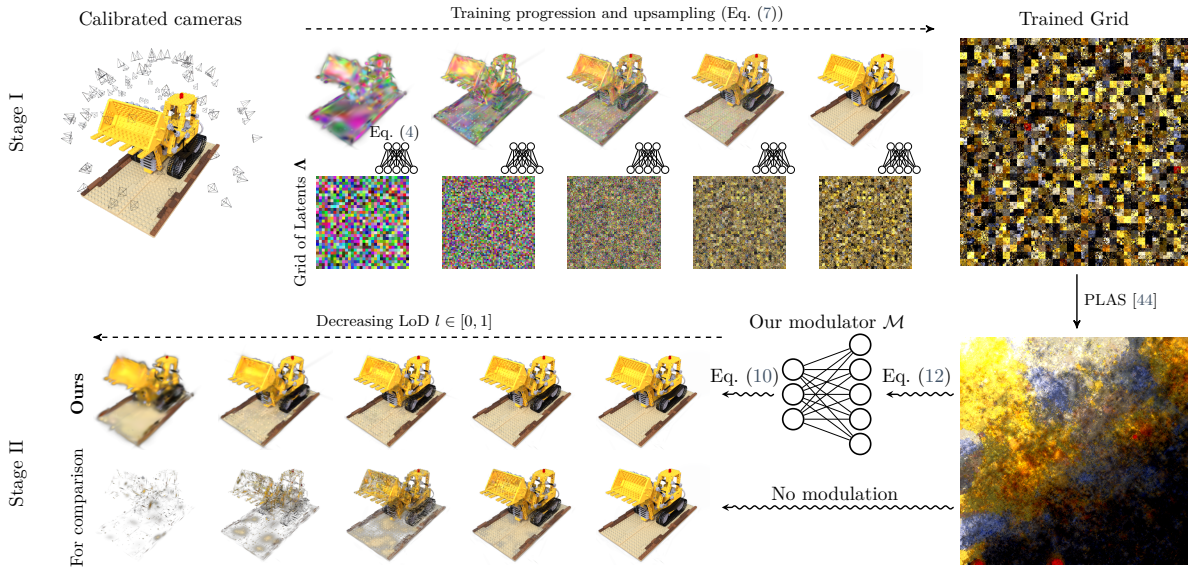


Figure 4. Pipeline – We represent the scene with a 2D UV map containing features that an MLP interprets 3D Gaussians. We first train the UV map at the highest resolution and sort it to ensure spatial coherence of similar features, enabling effective subsampling. For any desired level of detail (LoD), 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.

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 [17], using explicit octree representations [15, 82], and multi-level hashgrids [45] that eventually allowed NeRFs to run in real-time. Further enhancements, for example using quantized tables [70] were also suggested. Interestingly, this further allowed NeRF representations to be 'streamed' at desired resolutions [8, 59], effectively providing levels-of-detail, similarly to how Neural Geometric Level of Detail [71] provided LoDs for neural 3D shape representations.

Gaussian Splatting. Even with a hash grid-based implementation [45], the quality of NeRF renderings highly depends on how points are sampled along the light ray. On the other hand, 3D Gaussian Splatting (3DGS) [30] 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 [4, 18, 22, 23, 31, 37, 55, 63, 83], to distill Gaussians into 3D voxels [37, 39, 60] for efficient rendering, or to limit the number of Gaussians used [10, 11, 33, 44, 46, 47]. 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 [7, 40, 48]. 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 [71] proposed to produce a signed distance field in a continuous matter. In a similar spirit, Variable Bitrate Neural Fields [70] 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 [60] 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 [39], as they both start from points obtained from Structure-from-Motion [64] 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 [65], 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.

In our work, we draw inspiration from Mixed Volumetric Primitives (MVP) [38, 56] and Relightable Gaussian Codec Avatars (RGCA) [62] 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 was also explored in TextureGS [79], where this idea was used to disentangle texture from appearance. Here, we show that this 2D image can do much more—it can be used to group Gaussians together, creating a representation that can be used to provide continuous levels of detail for 3D Gaussian Splatting.

6.4 Method

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

6.4.1 Preliminary: 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [30] 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] $\boldsymbol{c} \in \mathbb{R}^3$ from a set of calibrated images $\{\mathbf{C}\}_{c \sim \mathcal{C}}$ with camera parameters $\mathcal{C}$. Kerbl *et al.* [30] use properties of splatting the 3D Gaussians on the image space introduced by Zwicker *et al.* [89]. 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:

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

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:

$$\ell_{\text{rec}} = (1 - \alpha_{\text{rec}})\ell_1 + \alpha_{\text{rec}}\ell_{\text{D-SSIM}}, \tag{2}$$

where ℓ_1 is the per-pixel L_1 loss, $\ell_{\text{D-SSIM}}$ is a differentiable SSIM [30] and α_{rec} is a loss weighting term. Please refer to the work of Kerbl *et al.* [30] for an in-depth derivation of the rendering equation used in 3DGS.

6.4.2 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 $l \in [0,1]$ [71]. Unlike previous work, which usually refers to the level of detail in frequency octaves of the input data [60, 65], 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. 4, 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 [2].

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

Additionally, as we use downsampled 2D map to enable LoDs, we use a learnable modulator $\mathcal{M}$ that uses the Gaussian parameters $\mathcal{G} = \{\mathbf{\Lambda}, \boldsymbol{x}^{\mathsf{3D}}\}$ and modulates them into $\tilde{\mathcal{G}} = \{\tilde{\mathbf{\Lambda}}, \boldsymbol{x}^{\mathsf{3D}}\}$. 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. (1) by applying the operator:

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

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.

6.4.3 Stage I – Training the Gaussian Representation

We parametrize our Gaussians as learnable D-dimensional latents $\mathbf{\Lambda} \in \mathbb{R}^{(W\times H)\times D}$, 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 $\mathbf{\Lambda}$ 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 $\boldsymbol{\lambda}_i \in \mathbf{\Lambda}$:

$$\begin{aligned}
\boldsymbol{\mu}_i(\boldsymbol{\lambda}_i) &= \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\mathbf{\Theta}), \quad \boldsymbol{q}_i(\boldsymbol{\lambda}_i) = \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\mathbf{\Theta}),\\
\boldsymbol{s}_i(\boldsymbol{\lambda}_i) &= \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\mathbf{\Theta}), \quad o_i(\boldsymbol{\lambda}_i) = \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\mathbf{\Theta}),\\
\boldsymbol{c}_i(\boldsymbol{\lambda}_i) &= \mathrm{MLP}(\boldsymbol{\lambda}_i\,;\,\mathbf{\Theta}).
\end{aligned} \tag{4}$$

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 [78]. We optimize $\{\mathbf{\Lambda}, \mathbf{\Theta}, \boldsymbol{x}^{\mathsf{3D}}\}$ by minimizing the objective:

$$\underset{\{\mathbf{\Lambda},\mathbf{\Theta},\boldsymbol{x}^{\mathsf{3D}}\}}{\arg\min}\ \ell_{\mathrm{rec}}(\mathbf{\Lambda}, \mathbf{\Theta}) + \Omega(\mathbf{\Lambda}, \mathbf{\Theta}), \tag{5}$$

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 $\mathbf{\Lambda}$ and parameters of the MLPs $\mathbf{\Theta}$, 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 [30, 39, 60, 63, 65], 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.* [32], 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 optimization as a function of their covariances [32]:

$$\tilde{\boldsymbol{\mu}}' = \alpha_{\text{lr}} \cdot \sigma\left(-k(o-t)\right) \cdot \Sigma\boldsymbol{\eta}, \tag{6}$$

where we adopt the hyperparameters from Kheradmand *et al.* [32].

Growing. 3DGS [30] 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:

$$\boldsymbol{x}^{\mathsf{3D}} \leftarrow \text{upsample}(\boldsymbol{x}^{\mathsf{3D}},\ 2) \quad \boldsymbol{\Lambda} \leftarrow \text{upsample}(\boldsymbol{\Lambda},\ 2), \tag{7}$$

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

Pruning (relocating). While 3DGS [30] removes low-opacity Gaussians due to their minimal rendering contribution, we adopt the relocation strategy of Kheradmand *et al.* [32]. We replace Gaussians with opacities $o < \tau_o$ by cloning more effective ones. Following Bulò *et al.* [4], 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. 6.4.2, we downsample it to generate 2D maps for lower LoDs. This process requires neighboring features to exhibit similarity. To achieve this, we apply the PLAS algorithm [44] to latents $\boldsymbol{\Lambda}$. Notably, we perform this sorting only *once* at the end of Stage-I training, unlike in [44] who apply it multiple times during training, which increases computational overhead. Fig. 5 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.

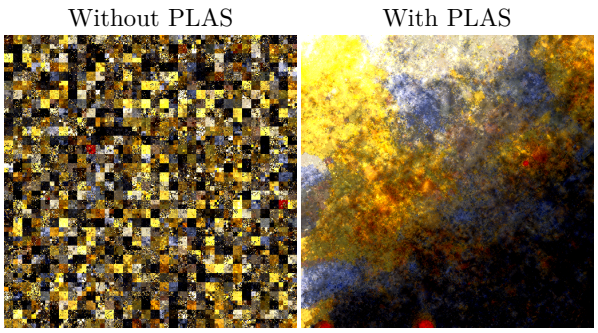


Figure 5. Effect of using PLAS [44] – We demonstrate the effect of sorting by visualizing the decoded Gaussian colors in the UV map, for the `lego` sequence from the NeRF Synthetic [43] dataset. After sorting, the UV map has smoother color transitions, confirming the successful spatial clustering of similar Gaussians.

6.4.4 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:

$$\begin{aligned} \mathbf{\Lambda}_{\downarrow} &= \text{downsample}(\mathbf{\Lambda}, l), \\ \boldsymbol{x}^{\mathsf{3D}}_{\downarrow} &= \text{downsample}(\boldsymbol{x}^{\mathsf{3D}}, l), \end{aligned} \tag{8}$$

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

$$W_{\downarrow} = \lfloor W \cdot l \rfloor \quad H_{\downarrow} = \lfloor H \cdot l \rfloor. \tag{9}$$

From here on, we denote Gaussian attributes decoded from $\boldsymbol{\lambda}_{\downarrow} \in \mathbf{\Lambda}_{\downarrow}$ 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 $\mathbf{\Lambda}_{\downarrow}$ need to be refined to compensate for missing Gaussians. We introduce a parametrized latent modulator $\mathcal{M}$ with parameters $\mathbf{\Theta}^{\mathcal{M}}$, 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:

$$\boldsymbol{\beta}(\boldsymbol{\lambda}), \boldsymbol{\gamma}(\boldsymbol{\lambda}) = \mathcal{M}(\boldsymbol{\lambda}_{\downarrow}, l\,;\, \mathbf{\Theta}^{\mathcal{M}}) \tag{10}$$

$$\tilde{\boldsymbol{\lambda}} = \sigma(\boldsymbol{\gamma}(\boldsymbol{\lambda})) \odot \boldsymbol{\lambda} + \boldsymbol{\beta}(\boldsymbol{\lambda}), \tag{11}$$

where $\sigma(\cdot)$ is a sigmoid activation function. Inspired by Continuous Upsampling Filters [73], 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 [43]. Overall, our $\mathcal{M}$ has

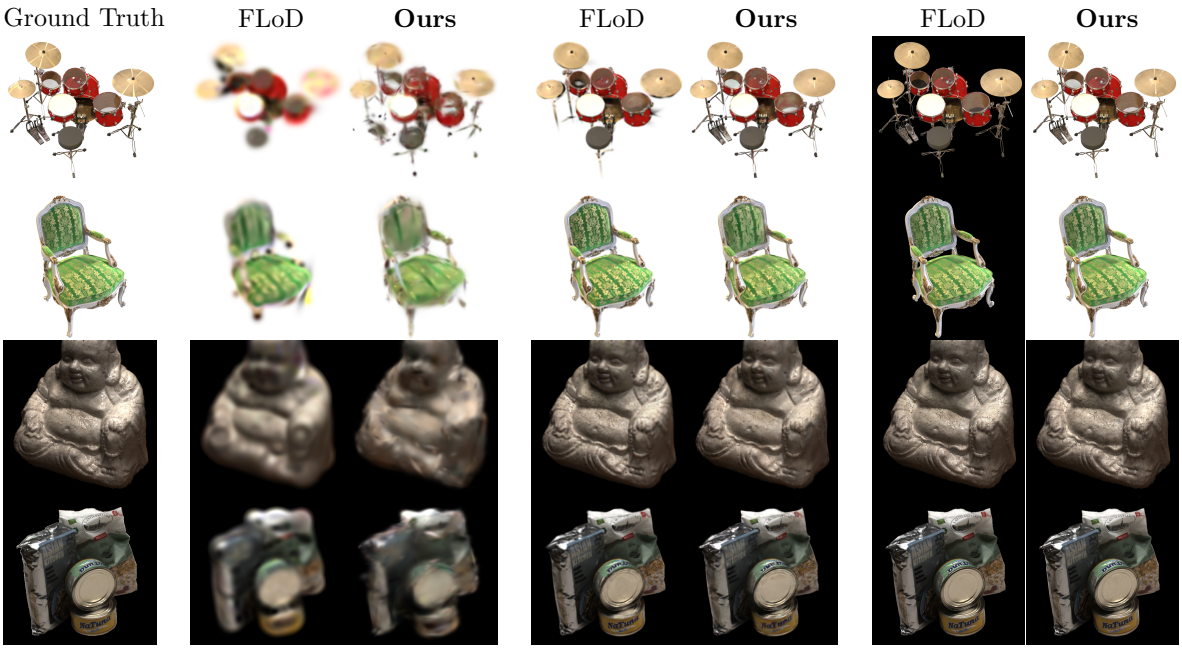


Figure 6. Qualitative highlight (varying levels of detail) – We provide example renderings for various levels of detail, denoted by the number of Gaussians used. As shown, our method is able to provide high-quality reconstructions at various levels, and provides a better tradeoff than FLoD [65]. We showcase 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.

the following form:

$$\mathcal{M}(\boldsymbol{\lambda}, l\,;\,\boldsymbol{\Theta}^{\mathcal{M}}) = \mathrm{MLP}(\boldsymbol{\lambda}, f(\boldsymbol{x}^{\mathsf{2D}}), f(l)\,;\,\boldsymbol{\Theta}^{\mathcal{M}}). \tag{12}$$

We optimize $\boldsymbol{\Theta}^{\mathcal{M}}$ by minimizing the reconstruction objective (as in Eq. (5)) and corresponding regularization terms:

$$\arg\min_{\boldsymbol{\Theta}^{\mathcal{M}}} \ell^{\mathcal{M}}_{\mathrm{rec}}(\boldsymbol{\Lambda}, \boldsymbol{\Theta}, \boldsymbol{\Theta}^{\mathcal{M}}) + \Omega^{\mathcal{M}}(\boldsymbol{\Lambda}, \boldsymbol{\Theta}^{\mathcal{M}}), \tag{13}$$

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.

6.4.5 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. (4), 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 [45]. 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 [43]—we use 8 frequencies, increasing by a power of 2 starting from 1. Regarding hyperparemters, we set them identically

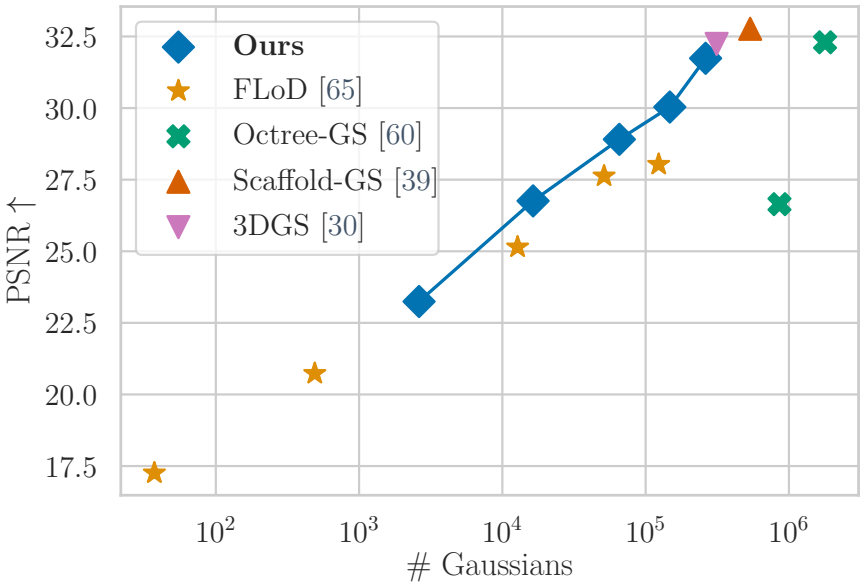


Figure 7. Rendering quality vs number of Gaussians for the DTU dataset [1] – We show the PSNR values for each method with respect to the number of Gaussians (level of detail). Our method is able to provide a continuous level of detail, and provides the best rendering quality for the same budget. Our finest level provides rendering quality that is on par with the traditional baselines.

Method	AVA [42]			Blender [43]			DTU [1]		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS [30]	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 [60]	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 [39]	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 [65]	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 1. 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 [43] dataset results for FLoD [65] should be interpreted with caution as it performs excessively poorly in some sequences.

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 $\|1-\boldsymbol{\beta}(\boldsymbol{\lambda})\|_2^2$ and $\|\boldsymbol{\gamma}(\boldsymbol{\lambda})\|_2^2$, all set empirically.

We initialize our 3D coordinates from a quasi-random Halton sequence from the $[-1,1]^3$ range. 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 for 10^5 steps so the training takes 2×10^5 steps in total. We implement our method using the `gsplat` [81] library.

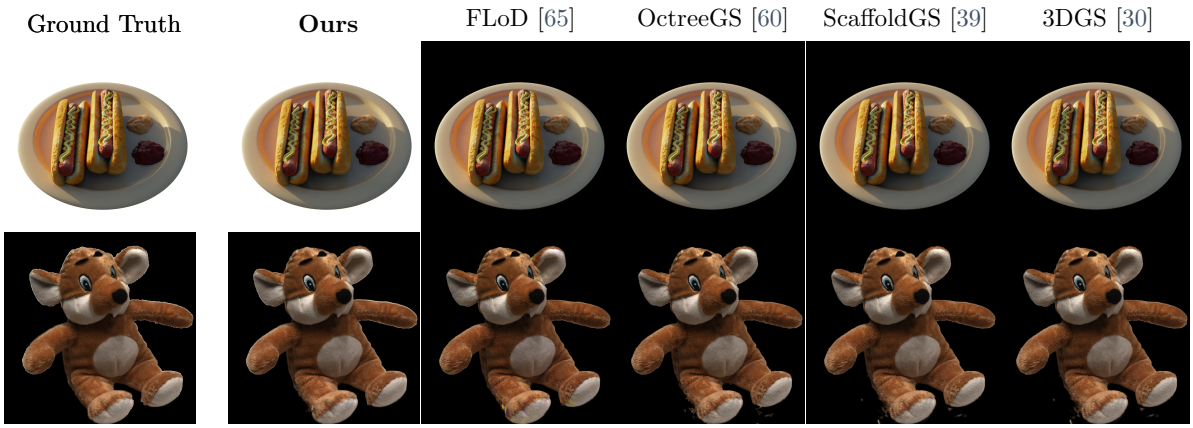


Figure 8. Qualitative examples (finest level of detail) – We show qualitative examples of our method, at the finest level of detail, compared to other baselines. Our method is able to provide comparable rendering quality at the finest level, and additionally allow continuous levels of detail as demonstrated in Fig. 6.

6.5 Results

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

6.5.1 Experimental setup

We discuss the datasets, the baselines, and metrics.

Datasets. We evaluate our method on three different datasets. We use the NeRF Synthetic [43] as this dataset is commonly used in previous work. We also use the DTU [1] dataset to evaluate performance on a non-synthetic dataset. Following the standard protocol [60], we use the provided foreground masks. Finally, we use the AVA [42] 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 [39] and Octree-GS [60], and vanilla 3DGS [30]. 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 [65], 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 [75] and LPIPS [84]. 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 [39] and Octree-GS [60], we measure the final number Gaussians

Method	Blender [43]				DTU [1]			
	PSNR ↑	SSIM ↑	LPIPS ↓	Number of Gaussians ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Number of Gaussians ↓
Octree-GS [60], 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 [60], 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 [65], 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 [65], 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 [65], 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 [65], 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 [65], 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 2. 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 [30] and also Octree-GS [60]. Note that NeRF Synthetic [43] dataset results for FLoD [65] should be interpreted with caution as it performs excessively poorly in some sequences.

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 [44]	23.326 ± 2.908	0.784 ± 0.097	0.297 ± 0.100
– resampling [4]	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 3. Ablation study – We compare our method without the modulator $\mathcal{M}$, exploration noise $\tilde{\boldsymbol{\mu}}'$, regularizers $\Omega(\cdot)$, sorting using PLAS [44], and resampling based on the method of Bulò *et al.* [4]. 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.

6.5.2 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. 6) and quantitatively (Fig. 7 and Tab. 2). As shown in Fig. 6, our method is able to render scenes at varying levels of detail. We additionally provide results for a concurrent work, FLoD [65], which is also able to provide levels of detail at pre-defined 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 [65] for the NeRF Synthetic [43] dataset should be interpreted with caution, as they performed particularly badly for `ficus` and `mic` sequences, despite our best efforts using the official implementation. We present metrics for each subject in the dataset and additional renders in `Supplementary`.

In Fig. 7, we summarize the quantitative results in Tab. 2 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 [65] 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.

6.5.3 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 Fig. 8 and a summary in Tab. 1. 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.

6.5.4 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. (6), the sorting with PLAS [44] in the feature space for Stage II training, the use of regularizers $\Omega(\cdot)$, and disabling error-based relocation [4]. 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. (1)). We present our results in Tab. 3.

6.6 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 UV 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.

CLoG: Leveraging UV Space for Continuous Levels of Detail

Supplementary Material

6.7 Potential Societal Impact

Our approach is a general framework for modeling Gaussian Splatting [30], and, therefore, inherits its potential misuses. The abuse includes topics like fake identity generation [85], using copy-righted content for own financial benefit, and detrimental effects of using GPU clusters for training on environment [3]. We admit that negative impact. However, we posit, that each of them can be addressed by deep-fake detection methods [53], improving existing methods that achieve better results on open-source material, and by training lighter representations [11, 33, 46].

6.8 Additional Results

We present additional, per-subject quantitative results in Tab. 4 and extended ablation studies over additional scales $l \in \{0.25, 0.5, 0.75, 1.0\}$ (for reference, Tab. 3 involves l=1.0) Tab. 5. We also include more reconstruction samples and comparison to baselines in Fig. 10 and the visualization the LoD progression has on the results in Fig. 9.

Method	AVA [42]																				
	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 [30]	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 [60]	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 [39]	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 [65]	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 [42]																				
	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 [30]	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 [60]	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 [39]	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 [65]	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 [43]																				
	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 [30]	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 [60]	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 [39]	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 [65]	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 [30]	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 [60]	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 [39]	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 [65]	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 [30]	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 [60]	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 [39]	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 [65]	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

Table 4. 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 [39] and FLoD [65] fails to produce crisp results and FLoD [65] outputs blank images on `ficus` and `mic` datasets from NeRF Synthetic [43], 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
– $\ddot{\mu}'$	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 [44]	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 [4]	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 5. 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 ■.

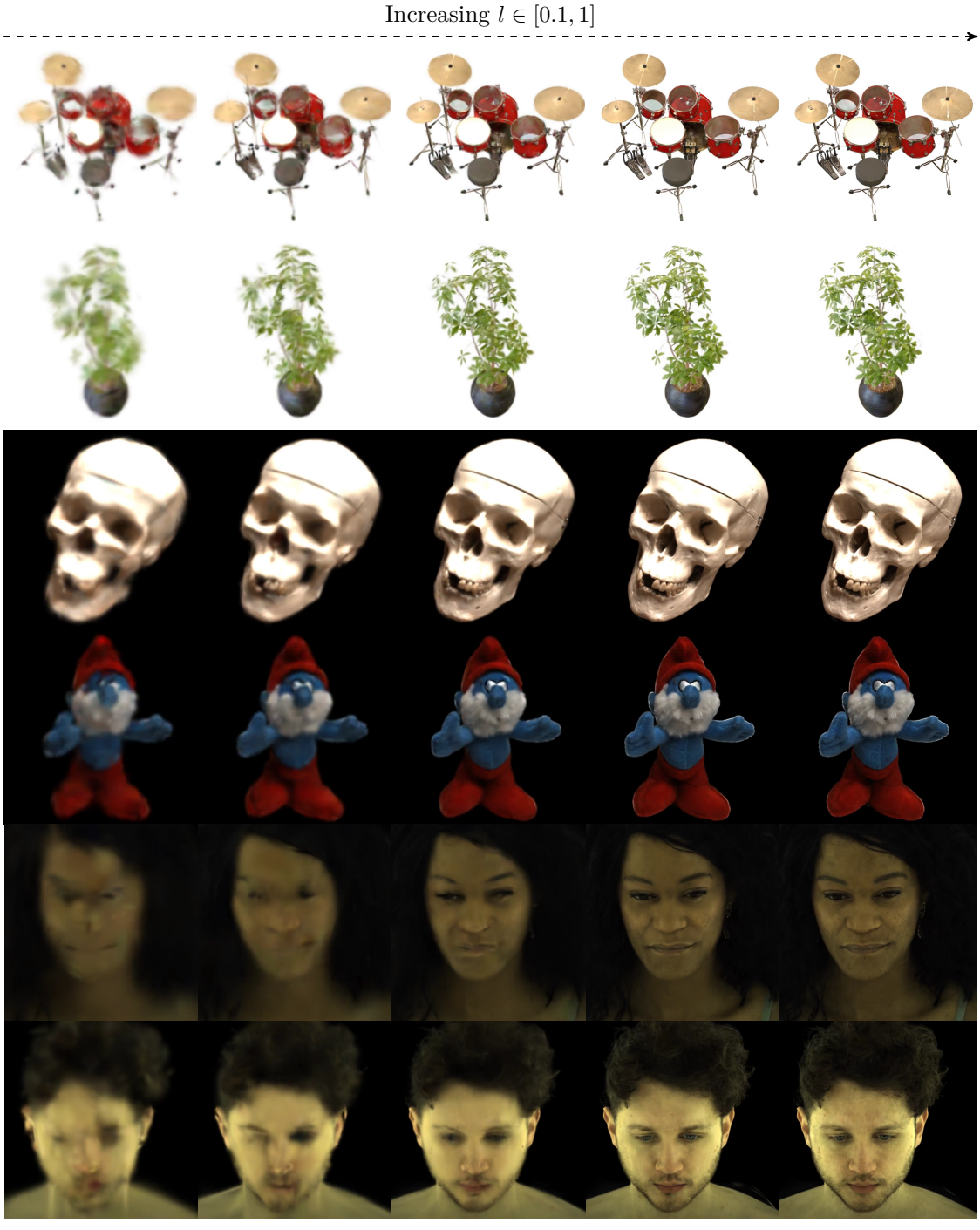


Figure 9. Additional qualitative examples (progressive level of details) – We showcase how our method reconstructs different samples under a given LoD l. The bottom row shows the number of Gaussians used to render an image. We use $l \in \{0.1, 0.28, 0.46, 0.82, 1.0\}$ levels od detail.

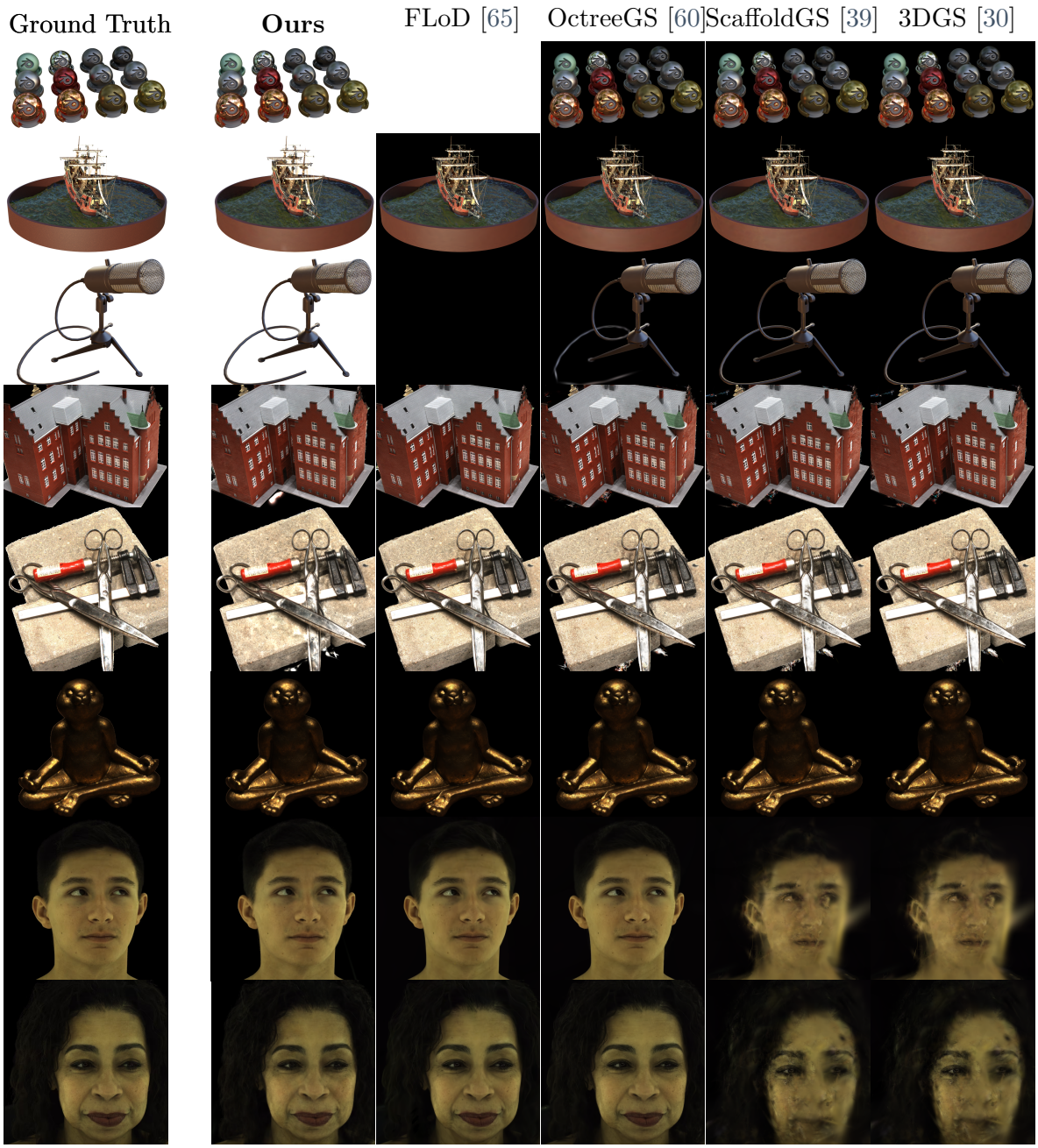


Figure 10. Additional qualitative examples (finest level of details) – We report additional qualitative results of our method and the baselines. Empty images for FLoD [65] denote subjects where the method failed to converge. However, it is worth noting that its quality is on par with OctreeGS [60], ScaffoldGS [39] and 3DGS [30] on all other samples. Please notice that ScaffoldGS [39] and 3DGS [30] 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.

Chapter 7

Final remarks and discussion

7.1 Conclusions

7.2 Future work

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. 21, 22, 31–33, 36).

[2] Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A., "Optimizing the Latent Space of Generative Networks," in *ICML*, 2017 (cit. on p. 26).

[3] Bouza, L., Bugeau, A., and Lannelongue, L., "How to estimate carbon footprint when training deep learning models? A guide and review," *Environmental Research Communications*, vol. 5, no. 11, p. 115 014, 2023 (cit. on p. 35).

[4] Bulò, S. R., Porzi, L., and Kontschieder, P., "Revising Densification in Gaussian Splatting," in *ECCV*, 2024 (cit. on pp. 24, 28, 33, 34, 36).

[5] 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 p. 7).

[6] 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 p. 7).

[7] 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. 25).

[8] 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. 24).

[9] 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. 12).

[10] 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. 22, 24).

[11] Fang, G. and Wang, B., "Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians," in *ECCV*, 2024 (cit. on pp. 24, 35).

[12] 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. 6).

[13] 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. 6).

[14] Flavell, L., "Uv mapping," in *Beginning Blender: Open Source 3D Modeling, Animation, and Game Design*, Springer, 2010, pp. 97–122 (cit. on p. 22).

[15] 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. 24).

[16] 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. 7, 8, 10).

[17] 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. 7, 24).

[18] Girish, S., Gupta, K., and Shrivastava, A., "EAGLES: Efficient Accelerated 3D Gaussians with Lightweight Encoding," in *ECCV*, 2024 (cit. on p. 24).

[19] 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 p. 6).

[20] Green, R., "Spherical harmonic lighting: The gritty details," in *Archives of the game developers conference*, vol. 56, 2003, p. 4 (cit. on p. 11).

[21] 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 p. 7).

[22] 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. 7, 8, 11, 24).

[23] 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. 24).

[24] Kaleta, J., Kania, K., Trzcinski, T., and Kowalski, M., "LumiGauss: High-Fidelity Outdoor Relighting with 2D Gaussian Splatting," 2025 (cit. on pp. 8, 11, 12).

[25] 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. 7, 8, 10, 12).

[26] 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. 8, 9, 11, 12).

[27] **Kania**, **K.**, Kowalski, M., and Trzciński, T., "TrajeVAE: Controllable Human Motion Generation from Trajectories," *arXiv preprint arXiv:2104.00351*, 2021 (cit. on p. 12).

[28] Kania, K., Yi, K. M., Kowalski, M., Trzciński, T., and Tagliasacchi, A., "CoNeRF: Controllable Neural Radiance Fields," in *CVPR*, 2022 (cit. on pp. 6, 8, 10, 12).

[29] **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. 12).

[30] 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. 5, 7, 9, 11, 22–24, 26–28, 31–33, 35, 36, 38).

[31] 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. 24).

[32] 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 p. 28).

[33] 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. 22, 24, 35).

[34] 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. 7).

[35] 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 p. 10).

[36] 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. 6, 10).

[37] 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. 24).

[38] Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., and Saragih, J., "Mixture of Volumetric Primitives for Efficient Neural Rendering," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–13, 2021 (cit. on p. 25).

[39] 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. 24, 25, 27, 31, 32, 36, 38).

[40] 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. 25).

[41] 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 *CVPR*, 2021, pp. 7210–7219 (cit. on p. 6).

[42] 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. 21, 31, 32, 36).

[43] 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. 6, 7, 21–23, 29–33, 36).

[44] Morgenstern, W., Barthel, F., Hilsmann, A., and Eisert, P., "Compact 3D Scene Representation via Self-Organizing Gaussian Grids," in *ECCV*, 2024 (cit. on pp. 23, 24, 28, 29, 33, 34, 36).

[45] 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. 7, 22, 24, 30).

[46] 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. 22, 24, 35).

[47] 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. 22, 24).

[48] 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. 25).

[49] 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 pp. 7, 8).

[50] 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 p. 6).

[51] 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," *ToG*, vol. 40, no. 6, 2021 (cit. on pp. 6, 10).

[52] 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. 7).

[53] Pei, G., Zhang, J., Hu, M., Zhai, G., Wang, C., Zhang, Z., Yang, J., Shen, C., and Tao, D., "Deepfake generation and detection: A benchmark and survey," *arXiv preprint arXiv:2403.17881*, 2024 (cit. on p. 35).

[54] 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 p. 6).

[55] 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. 24).

[56] 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. 25).

[57] 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. 7, 8).

[58] 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. 7).

[59] 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. 24).

[60] 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. 24–27, 31–33, 36, 38).

[61] 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. 7, 8).

[62] Saito, S., Schwartz, G., Simon, T., Li, J., and Nam, G., "Relightable Gaussian Codec Avatars," in *CVPR*, 2024, pp. 130–141 (cit. on p. 25).

[63] 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. 24, 27).

[64] Schonberger, J. L. and Frahm, J.-M., "Structure-from-Motion Revisited," in *CVPR*, 2016, pp. 4104–4113 (cit. on p. 25).

[65] 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. 22, 25–27, 30–34, 36, 38).

[66] 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. 22).

[67] 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 p. 7).

[68] 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. 12).

[69] 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. 12).

[70] 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. 24, 25).

[71] 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. 24–26).

[72] 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 pp. 6, 8).

[73] 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. 9, 29).

[74] 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. 22).

[75] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004 (cit. on p. 32).

[76] 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. 12).

[77] 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 p. 6).

[78] 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. 27).

[79] 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. 25).

[80] 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 p. 7).

[81] 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. 31).

[82] 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. 7, 24).

[83] 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. 24).

[84] 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. 32).

[85] Zhang, W., Yan, Y., Liu, Y., Sheng, X., and Yang, X., "E^3Gen: Efficient, Expressive and Editable Avatars Generation," *arXiv preprint arXiv:2405.19203*, 2024 (cit. on p. 35).

[86] 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," 2021 (cit. on p. 10).

[87] Zielonka, W., Bolkart, T., and Thies, J., "Towards Metrical Reconstruction of Human Faces," in *ECCV*, Springer, 2022, pp. 250–269 (cit. on p. 6).

[88] Zielonka, W., Bolkart, T., and Thies, J., "Instant Volumetric Head Avatars," in *CVPR*, 2023, pp. 4574–4584 (cit. on p. 6).

[89] 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. 5, 26).