UNIVERSITY OF CAMBRIDGE

Image-based 3D Reconstructions via Differentiable Rendering of Neural Implicit Representations

Tianhao Wu

Homerton College

This dissertation is submitted on 14, February, 2025 for the degree of Doctor of Philosophy

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted, or is being concurrently submitted, for any degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Abstract

Image-based 3D Reconstructions via Differentiable Rendering of Neural Implicit Representations

Tianhao Wu

Modeling objects in 3D is critical for various graphics and metaverse applications and is a fundamental step towards 3D machine reasoning, and the ability to reconstruct objects from RGB images only significantly enhances its applications. Representing objects in 3D involves learning two distinct aspects of the objects: geometry, which represents where the mass is located; and appearance, which affects the exact pixel colors to be rendered on the screen. While learning approximated appearance with known geometry is straightforward, obtaining correct geometry or recovering both simultaneously from RGB images alone has been a challenging task for a long period. The recent advancements in Differentiable Rendering and Neural Implicit Representations have significantly pushed the limits of geometry and appearance reconstruction from RGB images. Utilizing their continuous, differentiable, and less restrictive representations, it is possible to optimize geometry and appearance simultaneously from the ground truth images, leading to much better reconstruction accuracy and re-rendering quality. As one of the major neural implicit representations that have received great attention, Neural Radiance Field (NeRF) achieves clean and straightforward reconstruction of volumetric geometry and non-Lambertian appearance together from a dense set of RGB images. Various other forms of representations or modifications have also been proposed to handle specific tasks such as smooth surface modeling, sparse view reconstruction, or dynamic scene reconstruction. However, existing methods still make strict assumptions about the scenes captured and reconstructed, significantly constraining their application scenarios. For instance, current reconstructions typically assume the scene to be perfectly opaque with no semi-transparent effects, or assume no dynamic noise or occluders are included in the capture, or do not optimize rendering efficiency for high-frequency appearance in the scene.

In this dissertation, we present three advancements to push the quality of image-based 3D reconstruction towards robust, reliable, and user-friendly real-world solutions. Our improvements cover all of the representation, architecture, and optimization of image-based

3D reconstruction approaches. First, we introduce $\alpha Surf$, a novel implicit representation with decoupled geometry and surface opacity and a grid-based architecture to enable accurate surface reconstruction of intricate or semi-transparent objects. Compared to a traditional image-based 3D reconstruction pipeline that considers only geometry and appearance, it distinguishes the calculation of the ray-surface intersection and intersection opacity differently while maintaining both to be naturally differentiable, supporting decoupled optimization from photometric loss. Specifically, intersections on $\alpha Surf$ are found in closed-form via analytical solutions of cubic polynomials, avoiding Monte-Carlo sampling, and are therefore fully differentiable by construction, whereas additional grid-based opacity and radiance field are incorporated to allow reconstruction from RGB images only. We then consider the dynamic noise and occluders accidentally included in capture for static 3D reconstruction, as this is a common challenge encountered in the real world. This issue is particularly problematic for street scans or scenes with potential dynamic noises, such as cars, humans, or plants. We propose D^2NeRF, a method that reconstructs 3D scenes from casual mobile phone videos with all dynamic occluders decoupled from the static scene. This approach incorporates modeling of both 3D and 4D objects from RGB images and utilizes freedom constraints to achieve dynamic decoupling without semantic-based guidance. Hence, it can work on uncommon dynamic noises such as pouring liquid and moving shadows. Finally, we look into the efficiency constraint of 3D reconstruction and rendering, and specifically propose a solution for light-weight representation of scene components with simple geometry but high-frequency textures. We utilize a sparse set of anchors with correspondences from 3D to 2D texture space, enabling the high-frequency clothes on a forward-facing neural avatar to be modeled using 2D texture with neural deformation as a simplified and constrained representation.

This dissertation provides a comprehensive overview of neural implicit representations and various applications in 3D reconstruction from RGB images, along with several advancements for achieving more robust and efficient reconstruction in various challenging real-world scenarios. We demonstrate that the representation, architecture, and optimization need to be specifically designed to deal with challenging obstacles in the image-based reconstruction task due to the severely ill-posed nature of the problem. With the correct design of the method, we can reconstruct translucent surfaces, remove dynamic occluders in the capture, and efficiently model high-frequency appearance from only posed multiview images or monocular video.

Acknowledgements

Completing this PhD has been a significantly challenging yet deeply rewarding journey, and it would not have been possible without the guidance and support of many individuals. First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Cengiz Öztireli, for his unwavering support, expert guidance, and insightful feedback throughout my PhD journey. My heartfelt thanks also go to my colleagues and fellow researchers both within and outside of Cambridge, they are[1][2]: Prof. Andrea Tagliasacchi, Dr. Fangcheng Zhong, Dr. Forrester Cole, Dr. Gernot Riegler, Hanxue Liang, Prof. Jiankang Deng, Dr. Jing Yang, Jingyi Wan, Prof. Rafal Mantiuk, Shimon Vainer, Zhilin Guo

[1] names in alphabetical order and titles at the time of the completion of this dissertation.
[2] co-authors of the main publications during the course of this dissertation.

Contents

Chapter 1

Introduction

1.1 Overview

Accurately representing the 3D properties of objects using computational models is a foundational and ongoing challenge in all major fields and applications of 3D computer vision and graphics. Learning to estimate these properties from accessible scanning data, such as multiview RGB images, marks a significant advancement in improving the accessibility of 3D modeling applications. For most 3D modeling tasks, the properties to recover can be broadly categorized into two aspects: *geometry* and *appearance*. The geometry defines the precise spatial location where the object exists and interacts with its environment, while the appearance determines how the object is rendered visually and how it influences its surroundings through effects like occlusion and indirect illumination. Traditional 3D reconstruction pipelines typically treat geometry and appearance in separate processes. For instance, a Structure-from-Motion (SfM) pipeline is commonly used to obtain a sparse point cloud of the scene, followed by dense reconstruction with surface completion to recover the complete geometry. Once the geometry is obtained, appearance information, such as mesh textures, can be learned more easily from RGB observations. However, these traditional approaches struggle to capture accurate geometry for surfaces that lack distinctive textures. More critically, by treating geometry and appearance independently during the reconstruction process, the resulting scenes often lack photorealism when rendered from novel viewpoints.

Alternative to modeling geometry and appearance separately, simultaneously optimizing both using *differentiable rendering* has proven to be an effective approach for enhancing reconstruction quality. Differentiable rendering typically refers to the rendering processes that replace the binary, non-differentiable visibility term found in traditional rendering equations by introducing soft occlusions or approximating their gradients in the backward pass. This allows gradients to propagate from the photometric loss directly to the geometric representations, enabling the optimization of geometry using only RGB image supervision.

Compared to traditional pipelines, where geometry is reconstructed independently through methods, differentiable rendering refines geometric details based on RGB image data. This approach not only enhances the accuracy of the geometry but also ensures consistency between the reconstructed geometry and the input images.

With the advancement of differentiable rendering approaches, researchers have found that triangle meshes — one of the most widely used geometric representations in graphics due to their simplicity and rendering efficiency — may not be the most suitable choice for inverse rendering and image-based 3D reconstruction. Optimizing triangle meshes requires an initial shape with a pre-defined number of triangles and fixed connectivity, which severely constrains topology during differentiable rendering optimization and hence limits the possible shapes to be reconstructed. To address this limitation, various alternative representations have been proposed and revisited in the context of differentiable rendering and 3D reconstruction. Among these, the Neural Radiance Field (NeRF) has emerged as one of the most impressive and influential representations in the field. NeRF revisits a traditional implicit representation of computer graphics: the volumetric field for modeling of volumetric geometry with non-Lambertian appearance. The volumetric field is effectively a 5D plenoptic function that maps a spatial location and a 2D view direction to density and radiance, where the density corresponds to the probability of a ray being stopped at a given spatial location. NeRF leverages a Multilayer Perceptron (MLP) with positional encoding to model high-frequency volumetric fields, enabling the robust and high-quality reconstruction of both geometry and view-dependent appearances from multiview posed RGB images. Thanks to its simplicity and robustness across various real-world scenes, NeRF has inspired an explosion of 3D reconstruction methods, specializing in areas such as smooth surface reconstruction, few-shot reconstruction, and dynamic 3D object reconstruction.

This dissertation continues the exploration of image-based 3D reconstruction through various neural implicit representations, including NeRF and its successors. We address the limitations of existing representations and architectures for real-world 3D reconstructions, including ambiguities in geometric representation, challenges in handling dynamic noise, and restrictions on appearance that hinder efficient rendering. By improving the architecture, representation, and optimization, we tackle various real-world challenges in reconstructing 3D objects from real-world RGB scans.

We start by looking at a particularly challenging case for image-based reconstruction —- semi-transparent surfaces. Existing volumetric representations conflate two key properties: geometry (mass existence probability) and material (opacity), leading to artifacts when recovering surface representations that focus solely on geometry. To overcome this, we propose $\alpha Surf$ [122], a novel grid-based representation that decouples geometry and material into separate fields. To properly render and optimize such pure surface representation, we

need to deal with the non-differentiable intersection finding issue and ensure the surface representation can be correctly optimized with photometric loss. Existing methods rely on approximations, but we solve this by deriving a cubic polynomial equation for intersections. Using a closed-form algorithm for non-imaginary roots, we achieve accurate, efficient and differentiable intersection computation. This fully separates geometry and opacity, enabling accurate surface reconstruction for semi-transparent or thin structures.

We then present D^2NeRF [120], a method designed to decouple and remove dynamic noise in captured scenes. Real-world 3D reconstruction often faces challenges from dynamic occluders and noise, as multiview imaging typically involves moving a single camera through the scene. During this process, unavoidable dynamics such as moving cars, pedestrians, wind, or shadows can severely disrupt the capture. Our approach leverages a dynamic NeRF module to separate these dynamic elements using natural constraints on their degrees of freedom in a self-supervised manner, eliminating the need for external semantic segmentation tools. This makes D^2NeRF robust to various types of dynamic noise including liquid and shadows, ensuring reliable reconstruction in non-static environments.

Finally, we address the challenge of high-frequency texture modeling for real-time rendering. Existing efficient volume rendering approaches rely on specific architectures such as Gaussian Splatting [48], which contains a set of Gaussians where each carries a single spatial-invariant color. While effective for certain scenarios, this approach is ill-suited for objects with relatively simple geometry but high-frequency appearances. Accurately capturing such objects requires an excessively large number of Gaussians, leading to inefficiencies in both storage and rendering. One notable application scenario is the modeling of forward-facing neural upper-body avatars. The chest and shoulders in those avatars typically feature simpler motion and geometry but exhibit complex, high-frequency textures, such as detailed clothing patterns. To address this, we propose *Gaussian Head & Shoulders* [121], a hybrid representation designed to model high-frequency textures using a 2D neural texture combined with neural deformation. Anchor Gaussians are employed to ensure seamless and robust integration between the 2D and 3D components. This approach efficiently captures intricate details, such as clothing textures, while maintaining rendering speeds and stability superior to pure Gaussian architectures.

1.2 Geometry-Material Decoupling for Reconstruction of Translucent Surfaces

The modeling and reconstruction of translucent surfaces is an extremely challenging task for 3D reconstruction from RGB images. However, this is also a common issue in real-world scans. In addition to objects made with low-opacity materials, thin structures in real-world captures similarly exhibit semitransparent effects, because of the phenomena known as

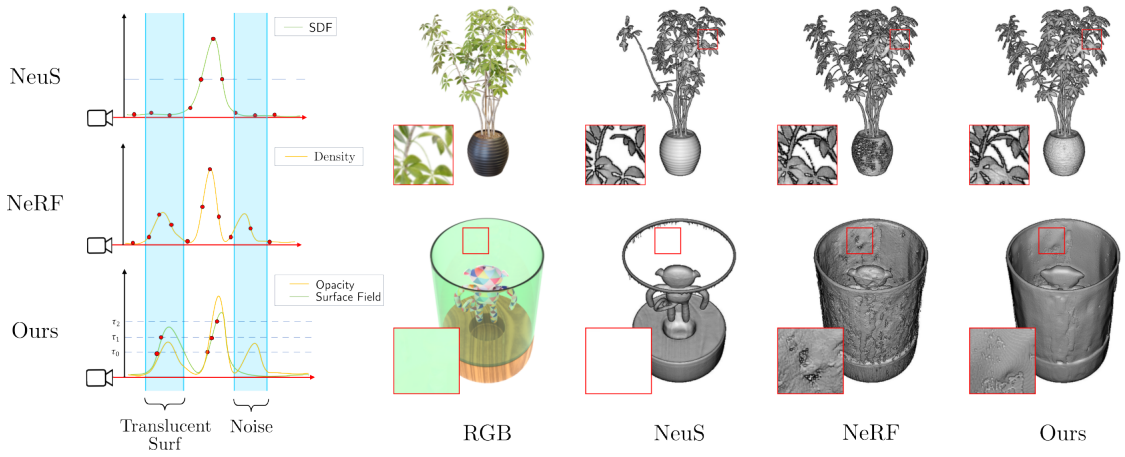


Figure 1.1: **Illustration of αSurf.** In comparison to existing surface modeling methods including NeuS [114], NeRF [70, 133], our method models decoupled surface and opacity fields, and can therefore accurately reconstruct surfaces exhibiting semi-transparency, including both translucent surfaces and thin structures due to blending effect.

the blending effect. Therefore, the ability to accurately recover surfaces for translucent surfaces is crucial for various real-world applications.

Although the volumetric representation in NeRF can effectively model translucent volume, extracting correct surfaces from it is a non-trivial problem due to various artifacts, including floating density artifacts and redundant inner volume. Several approaches have been proposed to mitigate those issues and improve the performance of surface reconstruction from RGB images. However, a core assumption made in their optimization is that all surfaces are assumed to be opaque with density approaching infinity in the end. We argue that, the incapability of existing representation in modeling translucent surfaces roots from the tightly coupled and ambiguous representation of geometry and material – opacity in particular. The density in traditional volumetric representation is incorrectly used to represent both properties: the probability of existence of mass, as well as the opacity of the mass. This ambiguity causes unavoidable artifacts when aiming to recover surface representation, which purely describes the geometry with no material involved. We therefore introduces a novel grid-based representation with geometry and material well-decoupled and represented in two different implicit fields. However, the difficulty further arises with the differentiable rendering of the surface representation. Since the ray-surface intersection is binary, finding the exact intersection locations in a differentiable approach is usually difficult, and existing approaches typically rely on approximation of the intersection. We resolve this issue by finding a cubic polynomial equation of the intersection. Therefore, the ray-surface intersection can be found by solving the valid non-imaginary roots of a cubic polynomial, for which there already exists an efficient and accurate closed-form root-solving algorithm. As such, intersection-finding

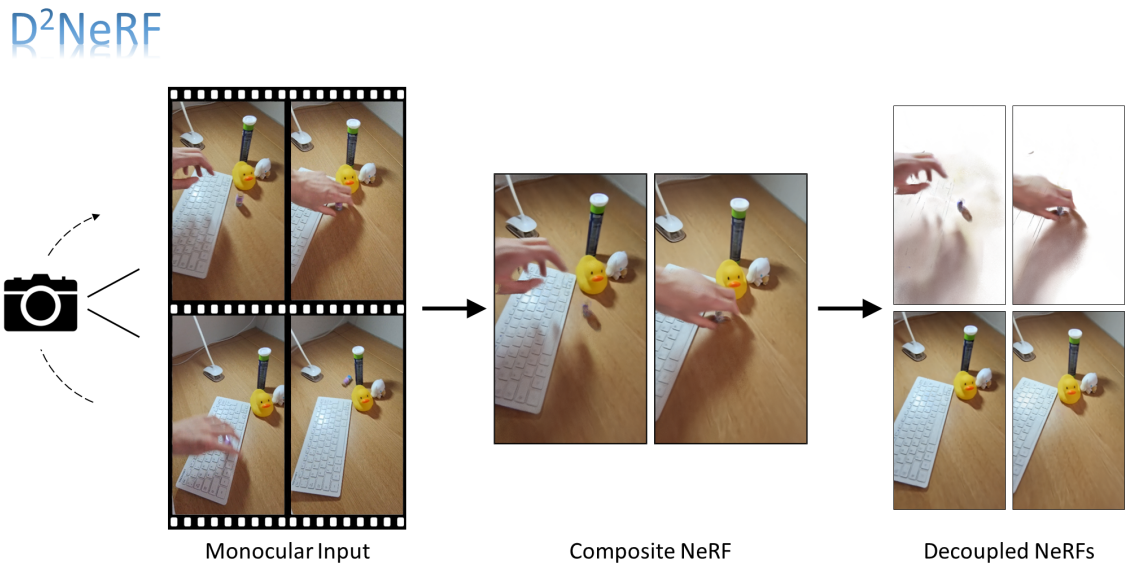


Figure 1.2: **D^{2}NeRF.** Using two composite volumetric fields and a dynamic shadow field, we achieve the decoupling of dynamic effects and static scenes in a monocular video.

becomes naturally differentiable, and hence the surface and opacity can be represented and optimized in two fully decoupled implicit fields. This allows us to accurately recover surfaces for objects with semi-transparent surfaces or thin structures, which can similarly exhibit semi-transparency due to their blending effect in RGB images; see Fig 1.1.

1.3 Static Scene Reconstruction with Dynamic Noise Removal

Another common real-world challenge is the presence of dynamic occluders and other transient noises in the capture. Since most existing 3D reconstruction approaches require multiview RGB images of the same scene, acquiring such data in real-world with a single camera would require the user to move the camera to multiple different locations for image capture. During this process, real-world scenes are rarely perfectly static, and disruptions such as moving cars, pedestrians, blowing leaves, shifting shadows, or other dynamic elements can interfere with the quality and accuracy of the reconstruction.

To encounter this challenge, We introduce D^2NeRF, a novel method that decouples and removes dynamic noise in real-world 3D scene reconstruction. Unlike conventional methods, which often rely on external tools such as semantic segmentation or manual preprocessing to isolate dynamic elements, our approach achieves this separation in a completely self-supervised manner. The method leverages natural constraints on the degrees of freedom of dynamic elements, allowing it to model and disentangle these motions without requiring additional labels or prior knowledge. This self-supervised

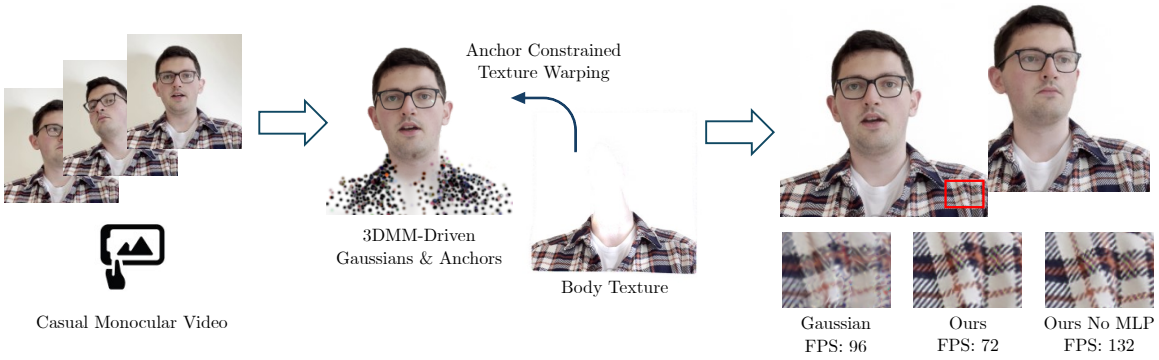


Figure 1.3: **Gaussian Head & Shoulders.** We reconstruct 3DMM-driven upper body avatars from casual monocular videos. By utilizing a high-frequency body neural texture which is warped using a neural texture warping field constrained by a set of sparse anchor Gaussians, we can learn sharp details of the cloth texture with highly efficient rendering speed.

approach ensures greater generality and adaptability, enabling the method to handle a wide range of dynamic noise types, such as flowing liquids or transient shadows, which are traditionally challenging to address with standard techniques. By robustly separating these dynamic components from the static elements of the scene, D^2NeRF significantly improves the quality of 3D reconstructions, even in environments with high levels of motion or variability. This makes it a highly effective solution for practical 3D reconstruction scenarios, where capturing fully static scenes is often impractical or impossible, see Fig 1.2.

1.4 Efficient High-Frequency Texture Modeling

In Chapter 5, we address the challenge of efficiently modeling high-frequency details in implicit representations. Our method builds upon Gaussian Splatting, a successor to Neural Radiance Fields (NeRF) that uses anisotropic Gaussians to model the volumetric field, enabling real-time rendering speeds. Gaussian Splatting has demonstrated promising performance and impressive rendering speed of over 100 FPS on a Nvidia A100 GPU. However, it encounters significant limitations when representing high-frequency textures, as each Gaussian in this representation is designed to carry a single color. Consequently, the number of Gaussians required increases dramatically for scenes or objects with intricate texture details, even when the underlying geometry is relatively simple.

A common example of this challenge is found in forward-facing neural human avatars. While the human body and clothing can often be modeled with smooth and simple geometric surfaces, the textures on clothing, such as fine patterns, embroidery, or intricate fabric designs, often contain extremely high-frequency details. To accurately capture these textures with Gaussian Splatting, the representation would require a prohibitively large

number of Gaussians, which significantly increases both computational cost and memory usage, thereby hindering efficiency.

To overcome this limitation, we propose Gaussian Head & Shoulders, a hybrid representation tailored for neural avatar modeling. In this method, the head and body are modeled using different strategies to optimize both geometric accuracy and texture detail. The head is represented using Linear Blend Skinning (LBS) driven Gaussian Splatting, which provides an accurate 3D representation of the head's geometry and appearance. On the other hand, the body is represented with a 2D neural texture that incorporates neural deformation to model complex textures. This hybrid approach effectively decouples the representation of geometry and texture, allowing for efficient and accurate modeling of high-frequency details without exponentially increasing the number of Gaussians. To ensure seamless integration and consistency between the head and body representations, we further introduce a sparse set of Anchor Gaussians in the body region. These Anchor Gaussians are simplified Gaussians with no anisotropic size or rotations, and their only purpose is to serve as 3D-2D constraints that regularize the 2D texture warping process with the 3D LBS. This effectively ensures that head and body remain connected and visually cohesive. This constraint mechanism addresses potential misalignments or artifacts that might arise from using separate representations for different parts of the avatar.

By combining the strengths of LBS-driven Gaussian Splatting for the head and neural texture mapping for high-frequency details, our method enables highly detailed and efficient avatar modeling. Compared to traditional Gaussian Splatting, our approach significantly reduces the computational overhead while accurately capturing fine texture details, such as the intricate patterns on clothing. This makes our method particularly well-suited for applications requiring detailed and realistic avatars, such as virtual reality, gaming, and digital content creation, where both efficiency and visual fidelity are critical. Additionally, our method extends the versatility of Gaussian Splatting, demonstrating its capability to handle more complex and high-resolution scenes beyond smooth surfaces, further pushing the boundaries of real-time implicit representations; see Fig 1.3.

1.5 Contribution

In this dissertation, we address several real-world challenges in image-based 3D reconstruction using neural implicit representations and provide a comprehensive analysis of why existing methods struggle to resolve these issues. To overcome these limitations, we propose specific modifications to the underlying representations, architectures, and optimization processes, enabling them to tackle these issues effectively. First, we address the challenge of reconstructing accurate and complete surfaces with semi-transparent effects. This problem arises from a fundamental limitation in existing volumetric representations, where density

is used to represent both geometry and material opacity. To resolve this, we propose a novel representation that decouples geometry, opacity, and appearance. Through a differentiable surface-finding approach based on cubic-polynomial root solving, we optimize the surface independently of the material properties using only RGB image supervision. Next, we introduce an approach that separates and removes dynamic noise from the static 3D scene by leveraging constraints on the degree of freedom. Unlike prior methods, we do not rely on prior knowledge of dynamic objects through object semantics, enabling it to handle a wide range of noise types, including hands, shadows, and liquids. Finally, we enhance the modeling of high-frequency textures on simple geometry while maintaining rendering efficiency. Our method simplifies the high-frequency components into a 2D neural texture warped by a 2D neural deformation field. To ensure seamless integration, we use a set of anchor Gaussians with 3D-2D correspondences to constrain the deformation. This ensures accurate merging of the 2D neural texture with 3D components, even under extreme or novel poses. Our findings demonstrate that, due to the significantly ill-posed nature of the image-based 3D reconstruction, effectively addressing real-world challenges would require carefully designed modifications to all of the representation, architecture, and optimization to integrate task-specific priors into the reconstruction, enabling robust and efficient solutions across diverse scenarios.

In summary, this dissertation made the following contributions:

- Identifications of three major real-world challenges that existing image-based 3D reconstruction methods fail to address.
- A study on reasons behind the failure of existing methods and the limitations of their underlying representations.
- A method with decoupled surface and opacity representation, as well as naturally differentiable surface intersection finding algorithm, to support the reconstruction of accurate and complete translucent surfaces from RGB images only.
- A method with regularized dynamic radiance field and shadow field to model and decouple various dynamic noises in the scene, including hands, shadows, and liquid.
- A method that supports efficiency and stable rendering speed of neural upper body avatars, regardless of the frequency of the clothing.

1.6 Publication

The following works were produced and incorporated into the main chapters during the course of this dissertation:

- Tianhao Wu, Hanxue Liang, Fangcheng Zhong, Gernot Riegler, Shimon Vainer, Jiankang Deng, Cengiz Oztireli. 2024. AlphaSurf: Implicit Surface Reconstruction for Semi-Transparent and Thin Objects with Decoupled Geometry and Opacity. International Conference on 3D Vision 2025

- Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. 2024. D2NeRF: self-supervised decoupling of dynamic and static objects from a monocular video. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22) Curran Associates Inc., Red Hook, NY, USA, Article 2366, 32653-32666.

- Tianhao Wu, Jing Yang, Zhilin Guo, Jingyi Wan, Fangcheng Zhong, and Cengiz Oztireli. Gaussian head shoulders: High fidelity neural upper body avatars with anchor Gaussian guided texture warping, 2025. The Thirteenth International Conference on Learning Representations (ICLR)

The following works were produced and partially incorporated into the background chapter during the course of this dissertation:

- K. Greff et al., "Kubric: A scalable dataset generator," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 3739-3751, doi: 10.1109/CVPR52688.2022.00373.

- Fangcheng Zhong, Kyle Fogarty, Param Hanji, Tianhao Wu, Alejandro Sztrajman, Andrew Spielberg, Andrea Tagliasacchi, Petra Bosilj, and Cengiz Oztireli. 2024. Neural fields with hard constraints of arbitrary differential order. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23) Curran Associates Inc., Red Hook, NY, USA, Article 992, 22870-22895.

- Liang, Hanxue & Wu, Tianhao & Hanji, Param & Banterle, Francesco & Gao, Hongyun & Mantiuk, Rafal & Oztireli, Cengiz. (2023). Perceptual Quality Assessment of NeRF and Neural View Synthesis Methods for Front-Facing Views. Computer Graphics Forum 2024

Chapter 2

Background

2.1 Rendering

The task of 3D reconstruction from images involves recovering the geometry, and optionally the appearance, of a three-dimensional scene using a set of multiview RGB images. This process can be understood as the inverse of forward rendering. To accurately model this inverse rendering and optimize the 3D properties, a comprehensive understanding of the camera model employed in forward rendering is essential.

2.1.1 Perspective Projection Model

As one of the most commonly used camera models in 3D computer vision and graphics, the pinhole camera is a model that assumes an infinitesimal aperture which only allows rays to traverse through a single point. Although not physically realizable in the real-world, it provides the simplest form of image rendering under an ideal scenario. The pinhole camera is parameterized by a 3×4 matrix $\mathbf{\Lambda}$, which is the product of a camera intrinsics matrix $\mathbf{K}$ and a camera extrinsics matrix $\mathbf{P}$. Figure 2.1 shows an illustration of the pinhole camera model.

$$
\begin{aligned}
\mathbf{\Lambda} &= \mathbf{KP} \\
\mathbf{K} &= \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \\
\mathbf{P} &= \left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_u \\ r_{21} & r_{22} & r_{23} & t_v \\ r_{31} & r_{32} & r_{33} & t_w \end{array}\right) = [\mathbf{R}|\mathbf{T}]
\end{aligned}
\tag{2.1}
$$

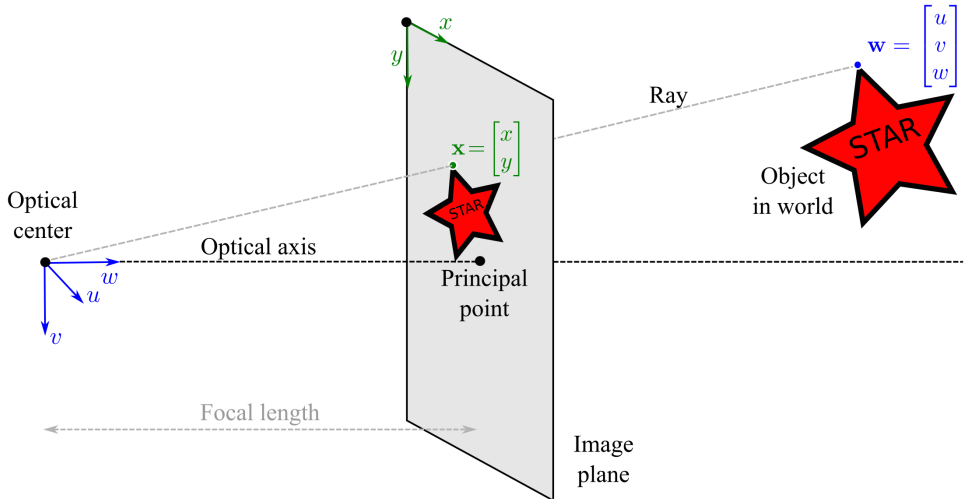


Figure 2.1: **The pinhole camera model [84].** The pinhole camera maps each spatial location in the 3D space $\mathbf{w}$ into a pixel coordinate $\mathbf{x}$.

where f_x, f_y are the focal lengths in vertical and horizontal directions respectively. c_x, c_y are the the coordinates of the camera principal point, which is the projection of the optical center on the image plane. s is the image skewness. The camera extrinsics matrix $\mathbf{P}$ can be further decomposed into a 3×3 camera rotation matrix $\mathbf{R}$ and a camera translation vector 1×3.

In a simplified scenario where the object appearance has all been baked into the scene, i.e., additional computation of direct and indirect illuminations is already completed, rendering objects from 3D to 2D involves projecting the 3D geometry into the 2D image plane via the pinhole camera project matrix $\mathbf{\Lambda}$ and render the corresponding appearance into the pixel color. The depth information stored in the z-buffer is used to properly handle occlusion and remove projections of occluded geometry. This process, known as rasterization, is an extremely efficient rendering method specifically designed for explicit representation such as mesh triangles. Since only the view frustum of the camera needs to be considered for rendering in the actual implementation and application, most approaches instead compute and shoot camera rays for each pixel and search for the first intersection in the scene. Given a camera at $\mathbf{o} \in \mathbb{R}^3$ with optical axis pointing in direction $\mathbf{d}$, the camera ray responsible for pixel with coordinate u, v on the image plane is parameterized as $\mathbf{r}_{u,v}(t) = \mathbf{o} + t\mathbf{d}_{u,v}, t \in \mathbb{R}$, where $\mathbf{d}_{u,v} \in \mathbb{R}^3$ is computed as:

$$\mathbf{d}_{u,v} = \mathbf{R} \cdot [\frac{u - W/2}{f_x}, \frac{v - H/2}{f_y}, 1]^T \tag{2.2}$$

where W, H are the image width and height. This rendering approach can be further extended to path tracing, where camera rays are traced with reflections on intersections to

estimate realistic global illuminations.

2.1.2 Camera Parameter Estimation

It is very common that the multiview image set does not contain any information about camera intrinsics or extrinsics. Therefore, it is necessary to obtain an accurate estimation of the camera parameters before reconstructing geometry and appearance becomes possible. Structure-from-Motion (SfM) [25, 96] is a commonly used and reliable method for estimating camera parameters and obtaining sparse static reconstruction from a set of multiview RGB images. SfM mainly involves three stages: 1) Replying on the overlapping observations in the images, SfM first computes and matches visual features such as SIFT [68] or SURF [7] across each image pair. 2) The camera parameters, including potentially shared intrinsic and individual extrinsic for each frame, are then estimated through the feature correspondences and refined via global bundle adjustment. 3) 3D structure is then recovered to obtain a sparse point cloud reconstruction by applying triangulation to the matched 2D pixel features to compute their 3D coordinates.

2.2 3D Geometry Reconstruction

2.2.1 Multiview Stereo

The idea of Multiview Stereo (MVS) can be traced back to human stereopsis, where depth information is estimated and perceived via slight positional differences in human binocular vision. Compared to binocular stereo, MVS instead deal with more views with varying positions and directions for stronger robustness and reconstruction accuracy. A key property in MVS is the ability to deal with extremely large number of images and pixels, potentially in the order of millions. However, the key challenging problem is shared between two tasks – pixel matching across images to obtain 3D or depth information of the pixels. This is essentially a correspondence-solving problem, where for each pixel in every image, the corresponding pixels in other images must be found via photo-consistency measures. Note that since the camera parameters are known, the potential candidates can be much simplified through the geometric constraints from the cameras, known as epipolar geometry [36]. With known pixel correspondence, it is then very simple to determine depth information and extra depth maps for each view. The depth maps can then be extracted to 3D point cloud, which can then be converted to meshes or surfaces using surface completion and refinement methods.

As the most critical step that directly relates to the final reconstruction quality in MVS, several methods for the computation of the photo-consistency measures exist, such as sum of squared or absolute difference, normalized cross-correlation, and mutual information,

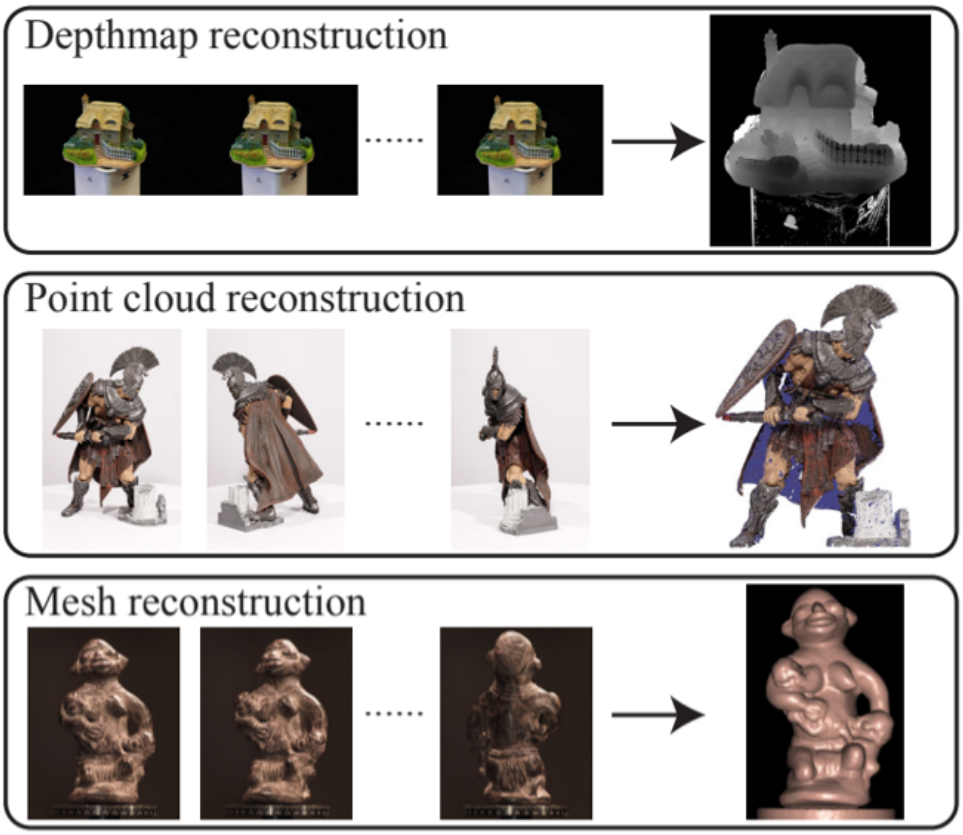


Figure 2.2: **Examples of MVS outputs from [26].** Based on different categories of the MVS algorithm, the pixel correspondences in multiview images are first determined through epipolar geometric constraint and photo-consistency measures, and eventually producing 3D representations including depth map, point cloud and mesh triangles.

yet all of them become less reliable as the complexity of the reconstruction increases. To properly establish correspondences, it is crucial to correctly define the regions of the scene where the appearance is both *unique* and *consistent* in different images, where the latter could further include illumination and view direction invariance. Therefore, the reconstruction quality of MVS methods is significantly limited in complex and realistic scenes, and they are not suitable for scenes with non-consistent lightning conditions and time-varying geometry. To overcome the limitations with hand-crafted prior, more recent methods incorporate neural networks and extract priors directly from data by learning an end-to-end mapping from RGB images to explicit 3D representations, including mesh [15, 72, 5, 113], point cloud [34, 81, 20, 1] or voxel [15, 72]. However, as they require ground truth 3D shape as training supervision, they tend to rely on the synthetic dataset for training and therefore contain an unavoidable performance gap for real-life scenes.

2.2.2 Differentiable Rendering

Neural networks have demonstrated effectiveness in both 2D and 3D understanding tasks, and their integration into traditional graphics rendering pipelines has garnered significant research interest in recent years. Yet, incorporating neural networks into these pipelines presents an unavoidable challenge: since they rely on backpropagation and gradient descent for end-to-end optimization, it becomes essential to derive a correct approximation of gradients during the forward rendering process, which generates images based on 3D properties such as geometry, material, and lighting.

Differentiable rendering encompasses a family of techniques designed to approximate gradients for various 3D representations, including meshes [61, 4, 67], voxels [109, 37, 44], and point clouds [93, 131, 119]. Despite their differences, these representations face a common challenge in achieving effective differentiable rendering. Taking triangle meshes as an example, the traditional graphics rendering pipeline renders only a single triangle for each pixel. Consequently, the gradients for all other triangles occluded or missed by the pixel ray are exactly zero, preventing those occluded triangles from contributing to the rendering or the optimization. Moreover, consider a triangle with a constant color. In such cases, the rendered color depends solely on the material and incoming light, instead of the barycentric coordinate of ray-triangle intersection. Hence, no gradient is generated for the vertice locations of the triangle mesh, and the geometry cannot be updated and will stay fixed during the whole optimization process.

Differentiable rendering methods can be categorized based on their approaches to gradient approximation. One category of methods focuses on directly estimating gradients for end-to-end optimization. For instance, OpenDR [67] is a general-purpose differentiable renderer that estimates pixel gradients using differentiable filters, such as the Sobel filter, applied to a neighborhood. Genova et al. [30] introduce the concept of negative barycentric coordinates for pixels outside a triangle, enabling the approximation of rasterization derivatives. Another category of methods modifies the forward rendering process itself to produce approximate, yet differentiable, outputs. Examples include defining a density parameter that decreases with distance from the mesh center [91], and employing soft rasterization, where the deterministic z-buffer-based occlusion process is replaced with a probabilistic approach [63]. Despite those innovative approaches, these methods can produce gradients that are incorrect or incomplete in certain scenarios, limiting their effectiveness in specific applications.

Due to the inherent challenges in obtaining accurate and complete gradients with traditional explicit representations, many recent studies have shifted focus to neural implicit representations or neural rendering. In this approach, neural networks are employed either to model traditional implicit representations, enabling them to be rendered in a fully differentiable manner, or to replace the entire rendering process with a forward pass

through a neural network, thus achieving full differentiability.

2.3 Implicit Representations

Implicit representations, such as signed distance functions (SDFs), Gaussian glob models, and volumetric fields, have long been foundational in computer graphics. These representations are continuous and differentiable functions that implicitly define the underlying scene by mapping each spatial sample to specific geometric properties, such as color, occupancy, or distance to the closest surface. By sufficiently sampling the 3D space, implicit representations can be converted into explicit forms with user-defined resolution.

Historically, implicit representations were less commonly used than explicit ones, primarily due to their inefficient and computationally demanding rendering. However, their continuous and differentiable nature makes them particularly well-suited for integration with machine learning and neural networks. In recent years, the use of neural networks to model implicit representations—commonly referred to as *neural implicit representations*—has become increasingly popular.

This chapter introduces several widely used implicit representations that have been adapted using neural networks for the task of 3D reconstruction.

2.3.1 Signed Distance Function

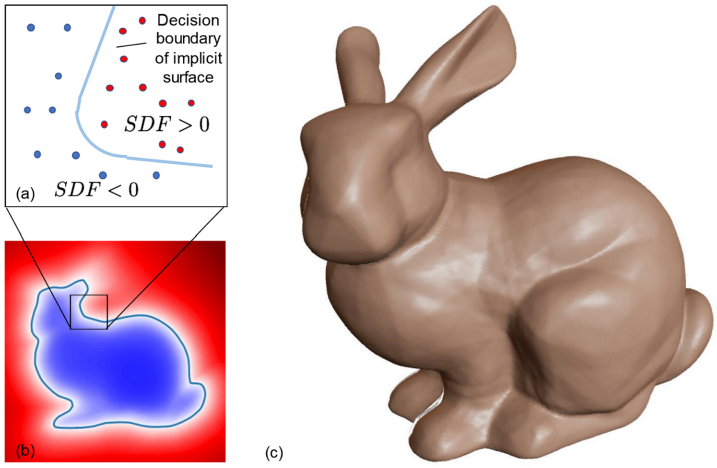


Figure 2.3: **The SDF of the Stanford Bunny.** (a) Signs of SDF outputs near the surface, which is implicitly represented by the zero level set. (b) A 2D slice of SDF values. (c) A rendering of the represented surface. Figure by Park et al. [76].

The Signed Distance Function (SDF) is an implicit representation that models a watertight surface as the zero-level set in 3D space. It is a fully continuous function that maps 3D coordinates to a scalar value, where the sign indicates whether the spatial sample

is inside or outside the object surface, and the magnitude represents the shortest distance to the surface. SDFs can be efficiently converted to explicit representations using the marching cubes algorithm [76].

SDF-based 3D reconstruction has demonstrated promising results. Although not leveraging differentiable rendering, DeepSDF [76] employs a simple multilayer perceptron (MLP) to map a single image to the SDF of a target object. The model generalizes to unseen objects using an auto-decoder architecture, where a latent code is optimized at test time for each reconstruction target and passed into the MLP decoder along with spatial coordinates to compute signed distances. DISN [125] enhances reconstruction quality by integrating a convolutional neural network (CNN) encoder to extract local features from small patches of the input image, capturing finer geometric details. Frodo [94] advances these approaches by detecting, segmenting, and grouping individual objects in dense image sets. It identifies and groups multiple views of the same object using pre-trained networks, merging them into a single latent code that encodes geometric information. However, these methods require ground truth 3D geometry for training supervision and do not reconstruct object appearance.

While SDF alone only encodes 3D geometry and cannot be supervised directly with photometric clues, it is possible to augment SDF with other appearance modeling representations and recover them simultaneously from image supervision only. Yariv et al. [129] build a bidirectional reflectance distribution rendering function (BRDF) based on SDF output to simultaneously reconstruct the geometry, albedo and lighting condition of a scene from a set of RGB images with masks. Yariv et al. [130] combines SDF with traditional volume rendering to represent the scene geometry as SDF and appearance as a radiance field, which is a 5D plenoptic function encoding non-Lambertian radiance. SDFSRN [60] combines SDF with a coloring function to render the view and compute the photometric loss to supervise the learning. It also uses the 2D object silhouette to further constrain the signed distance. This allows the model to be trained under limited condition, where the training dataset only contains a single image for each different object.

2.3.2 Structured Implicit Function

Structured implicit function (SIF) [32] is a 3D occupancy representation that can easily scale to objects with complex geometry. SIF consists of a set of independent anisotropic 3D Gaussian models, each describing the occupancy of local 3D space through a binary thresholding. The overall geometry is hence represented by the sum of all Gaussians as a zero level set. By directly inferring the SIF parameters through an autoencoder network from a set of posed depth images, the learned SIF model maintains a highly consistent shape template across different objects, where the same Gaussian model tends to describe the local occupancy with the same or similar semantic meaning (e.g., chair legs), leading to

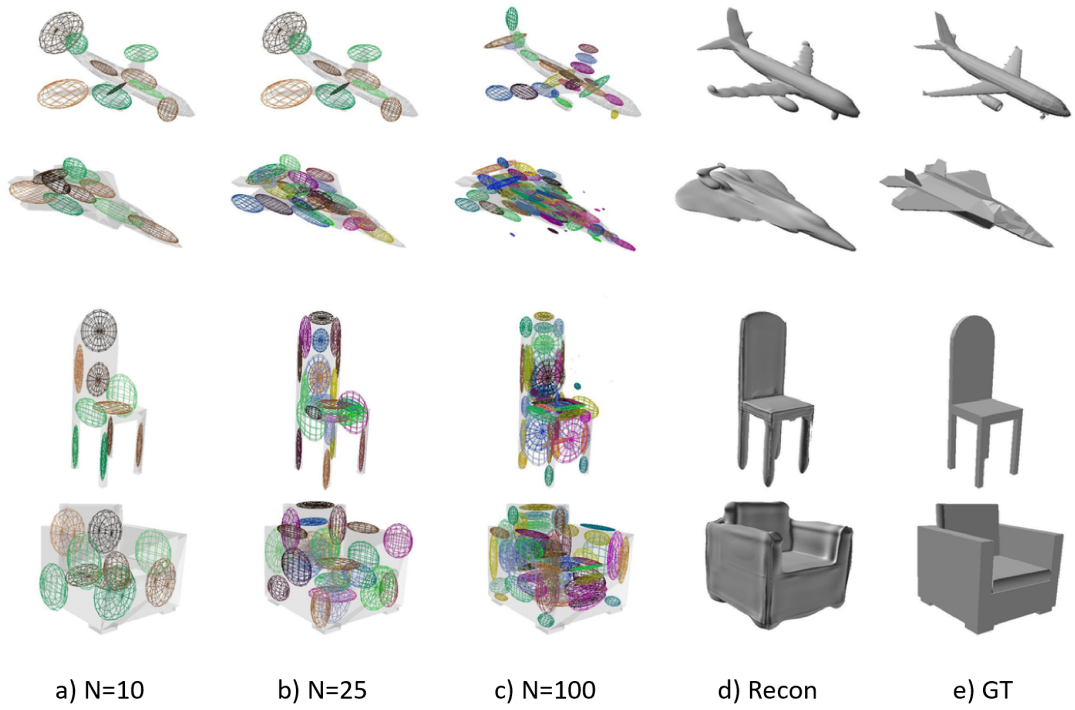


Figure 2.4: **Objects represented by SIF.** (a, b, c) SIF representation of different objects, where N indicates the total number of different 3D Gaussian distributions used. The same Gaussian tends to describe the local geometry with similar semantic meaning or position. (d) Renderings of surface reconstructed by SIF. (e) Renderings of the ground truth geometry. Image modified from [32].

many potential scene understanding applications; see Figure 2.4. LDIF [31] combines SIF with a local point feature decoder, where SIF is learned as a coarse model for geometry. Points are then sampled in the neighborhood of each Gaussian model, and are passed to a local network to predict more detailed geometry. Such coarse-to-fine prediction reduces the requirement in network size and improves training efficiency.

2.3.3 Light Field

Unlike other 3D representations, the light field models only the scene appearance, without explicitly accounting for geometry, and is primarily used for appearance modeling and novel view synthesis rather than full 3D reconstruction. A light field is a function that maps 4D camera rays, represented as a light slab [53] or two-sphere [8], to their corresponding radiance or pixel color, making it particularly efficient for image rendering. However, since the light field does not explicitly model scene geometry and cannot inherently ensure multi-view consistency, additional regularization techniques are necessary for achieving correct and reliable reconstruction. LFR [104] addresses this by constraining the learned light field multilayer perceptron (MLP) with weak geometric clues derived from epipolar lines. For each target ray, its epipolar lines in multiple reference views are identified, and

several pixels along these lines are sampled and aggregated into a view feature. This feature is then decoded through the light field MLP to predict the pixel color. RF-RSEN [3] takes a different approach by mapping each 4D ray into an embedding space and encouraging rays that terminate at the same spatial location to share similar embeddings. To implicitly model occupancy, the scene is subdivided into multiple local voxels, with each voxel learning a separate light field. Point Light Field [75] combines the light field representation with an explicit point cloud to enforce consistent geometry. Sparse point clouds, extracted using a pre-trained network or LiDAR sensor, serve as geometric anchors. Point features within a neighborhood are aggregated using multi-head attention and decoded via a light field network to obtain radiance. LFNS [103], on the other hand, incorporates meta-learning to extract data-driven priors from general scenes, enabling light field reconstruction from a single image. This approach demonstrates that limited geometric information is implicitly encoded in the color gradient of epipolar plane images for Lambertian surfaces and even extracts coarse depth maps from the light field. Despite these advancements, light field methods inherently struggle with scenes containing complex geometry and occlusions. In such cases, inconsistent multi-view geometry can introduce severe artifacts in novel view rendering, highlighting the limitations of this representation for comprehensive 3D reconstruction.

2.3.4 Volumetric Field

Volumetric field [70, 47], as illustrated in Fig 2.5, is an implicit volumetric representation that represents the scene via a 5D plenoptic function $\mathcal{F}$, which maps the spatial coordinate $\mathbf{x} \in \mathbb{R}^3$ and view direction $\mathbf{d} \in \mathbb{S}^3$ into the radiance $\mathbf{c}(\mathbf{x}, \mathbf{d}) \in \mathbb{R}^3$ and the density $\sigma(\mathbf{x}) \in \mathbb{R}$, which is correlated with the probability of stopping a ray at current location and therefore indicates the occupancy of spatial space:

$$\mathcal{F}\left(\mathbf{x}, \mathbf{d}\right) = \begin{cases} \sigma(\mathbf{x}) \\ \mathbf{c}(\mathbf{x}, \mathbf{d}) \end{cases} \tag{2.3}$$

Unlike commonly used explicit representations such as mesh and point cloud, the volumetric field is capable of modeling and rendering semi-transparent objects or objects with cloud effect due to its modeling of density and its unique volumetric rendering. As the density value $\sigma(\mathbf{x})$ indicates the occupancy of infinitesimal sample $\mathbf{x}$ and can be interpreted as the differential probability of terminating the camera ray by the sample, the pixel color $\hat{\mathbf{C}}_{u,v}(\mathbf{r})$ for pixel at ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}_{u,v}$ of a radiance field defined within bounds $[t_n, t_f]$ is therefore computed as:

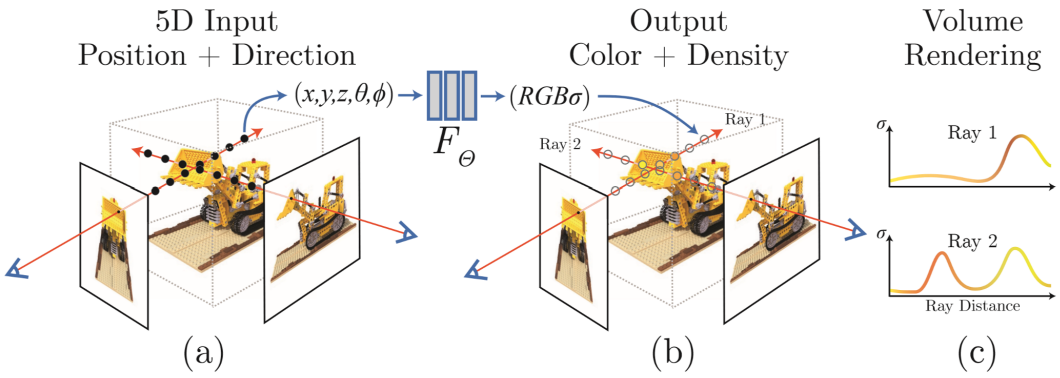


Figure 2.5: **The volumetric field.** (a, b) A volumetric field is a function that maps spatial coordinate and view direction into radiance and density. (c) The radiance and density can then be rendered into RGB images using volume rendering, which is a fully differentiable process. Illustration from [70].

$$\begin{aligned} \hat{\mathbf{C}}_{u,v}(\mathbf{r}) &= \int_{t_n}^{t_f} T\left(\mathbf{r}(t)\right) \cdot \sigma\left(\mathbf{r}(t)\right) \cdot \mathbf{c}\left(\mathbf{r}(t), \mathbf{d}_{u,v}\right)\, dt \\ T(\mathbf{r}(t)) &= exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds\right) \end{aligned} \tag{2.4}$$

where $T(\mathbf{r}(t))$ can be interpreted as the transmittance along the ray until current $\mathbf{r}(t)$, and is inversely correlated to the accumulated density along the ray. Intuitively, having large density objects in the former section of the ray is highly likely to terminate ray and occlude what is behind the objects, hence greatly reducing the radiance contribution of samples behind. It is notable that the rendering equation (2.4) is fully continuously and easily differentiable, allowing optimization of volumetric field representation with end-to-end photometric loss directly.

2.4 Neural and Non-Neural Architectures for Representations Modeling

In the last section, we introduced various implicit graphical representations for describing 3D scenes and properties. We then move to the methods or architectures for modeling those implicit representations as computational models, allowing them to be optimized end-to-end with gradient descent on photometric loss. Most of the architectures proposed involve neural networks, as their versatile capacity and natural smoothness can greatly enhance the quality of reconstruction. Additionally, some hybrid or pure-explicit architectures have also been proposed as an effective approach for achieving efficient optimization and

Figure 2.6: **2D image fitting with positional encoding of different frequencies** Gaussian-based positional encoding [106] is applied to fit a 2D RGB image with a small MLP, where σ indicates the frequency of the encoding. Higher frequency can significantly improve the level of details learned in the MLP, but using a value that is too large can also lead to over-fitting. Figure by Tancik et al. [106].

rendering.

2.4.1 Multilayer Perceptron

One commonly used architecture to model implicit representations such as SDF and volumetric field is Multilayer Perceptron (MLP), one of the most standard architectures for neural networks. For example, NeRF [70] uses a skip-connected MLP to model the volumetric field, where the spatial location $\mathbf{x}$ is passed to the network from the beginning to obtain view-invariant density $\sigma(\mathbf{x})$, and the view direction $\mathbf{d}$ is passed via skip connection to obtain radiance $\mathbf{c}(\mathbf{x}, \mathbf{d})$. In addition, NeRF incorporated a positional encoding function $\gamma_m(z)$ that lifts the dimensionality of the input variable:

$$\mathcal{F}_\theta\left(\gamma_m(\mathbf{x}), \gamma_n(\mathbf{d})\right) = \begin{cases} \sigma(\mathbf{x}) \\ \mathbf{c}(\mathbf{x}, \mathbf{d}) \end{cases} \tag{2.5}$$

Various forms of possible positional encoding function exists [106], where the most basic one is a mapping based on Fourier frequencies:

$$\gamma_m(z) = (z, sin(2^0\pi z), cos(2^0\pi z), ..., sin(2^{m-1}\pi z), cos(2^{m-1}\pi z)) \tag{2.6}$$

Tancik et al. [106] suggest positional encoding is capable of making the Neural Tangent Kernel (NTK) of the MLP shift-invariant and can tune kernel's spectrum via the frequency m, effectively controlling the range of frequencies that can be learned with the MLP. Therefore, incorporating positional encoding significantly improves the fitting and reconstruction of high-frequency details in the image; see 2.6.

Specific to the modeling of volumetric field and volume rendering, to calculate the nested double integrals in equation (2.4) with feasible computation in practice, Monte Carlo numerical quadrature is applied to approximate the integrals. To avoid inefficient sampling in empty space and to improve both computational efficiency and rendering

quality, NeRF adopts a *coarse-to-fine* hierarchical sampling strategy. A set of coarse samples is initially drawn in a stratified style, where the camera ray with bounds $[t_n, t_f]$ is divided into N_c equivalent sections, and a sample is drawn randomly and independently from each section. All coarse samples are used to query a coarse NeRF MLP to obtain coarse density and radiance predictions. It is then possible to identify sections that are more likely to contain concrete objects that contribute greatly to the final pixel color by inspecting the normalized volume rendering weights of coarse samples and drawing additional N_f fine samples from it:

$$\begin{aligned} P(t) &= \frac{\sum_{i=1}^{N_c-1} w(t_i)\mathbb{I}[t \in [t_i, t_{i+1}]]}{\sum_{i=1}^{N_c-1} w(t_i)} \\ t_{fine} &\sim P(t) \end{aligned} \tag{2.7}$$

The fine samples are fed to a separate fine NeRF MLP with the same architecture as the coarse network. With discrete samples, the volume rendering equation (2.4) is estimated as:

$$\begin{aligned} \hat{\mathbf{C}}_{u,v}(\mathbf{r}) &= \sum_{i=1}^{N} \hat{T}\left(\mathbf{r}(t_i)\right) \cdot \alpha\left(\mathbf{r}(t_i)\right) \cdot \mathbf{c}\left(\mathbf{r}(t_i), \mathbf{d}_{u,v}\right) \\ \hat{T}(t_i) &= exp\left(- \sum_{j=1}^{i-1} \sigma\left(\mathbf{r}(t_j)\right) \delta_j \right) \\ \alpha(t_i) &= 1 - exp\Big(- \sigma\left(\mathbf{r}(t_i)\right) \delta_i \Big) \end{aligned} \tag{2.8}$$

where $\delta_i = t_{i+1} - t_i$ represents the distance between the i-th and the subsequent samples, and could be evaluated to either 0 or infinity for the last sample depending on the hyperparameter.

At training time, coarse and fine MLPs are both volume-rendered and supervised with the photometric loss, whereas at inference time, only the fine MLP needs to be rendered and coarse MLP is queried to obtain coarse density predictions. However, it is notable that such an approach is only a technique for efficient rendering and training, and its reconstruction quality is upper-bounded by taking an extremely large number of samples uniformly. Also, some follow-up methods such as NeuS [114] do not model two separated MLPs, but apply both coarse and fine sampling on a single MLP.

2.4.2 Multi-Resolution Hash Grid

Although MLP with positional encoding can effectively model high-frequency details in the 3D reconstruction, their rendering speed becomes a major performance bottleneck that prevents real-time and interactive applications. To address this issue, grid-based hybrid neural networks have been proposed to provide over 100 times speed-up in rendering at the expense of increased storage and memory requirements. The multi-resolution hash grid architecture, first proposed in InstantNGP [71], is a particularly effective solution for modeling 2D images, 3D volumetric fields, SDFs, and various other implicit representations. It utilizes a novel encoding based on the multi-resolution hash grid, where a spatial coordinate $\mathbf{x}$ is mapped to a set of latent codes with a very small length (e.g., 2) $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L\}$, where L is the number of levels in the hash grid. The resolution of the grid starts at 16^3 at the first level, and grows with a factor of 2 as the level increases. Each level maintains a unique hash table, which maps the spatial coordinates of the grid vertices into a latent code. The latent code of $\mathbf{x}$ is then obtained using trilinear interpolation of the latent codes at 8 neighboring vertices. After obtaining latent codes at all levels, they are concatenated into a single latent code $\mathbf{z}$, typically of size 32. After optionally concatenating with other input such as the view direction $\mathbf{d}$, the latent code is passed to a small global MLP to decode into desired information such as distance or density and radiance.

Using hash tables to fetch latent codes at grid vertices significantly improves the query and essentially the rendering speed, in exchange for limited capacity and, as a result, a possibility for hash collision for levels with high resolutions. Hash collision is an event where two or more unique inputs accidentally map to the same hash in the table, and is unavoidable when the size of the hash table is smaller than the possible inputs. In the case of has-grid, the same latent code could be returned when encoding many different vertices. However, this does not raise a huge performance impact regarding the final reconstruction quality because of two main reasons: 1) Although hash collision exists at each level and each hash table, it is nearly impossible that exact same collision happens for all levels on a vertex. Therefore, only a few digits of $\mathbf{z}$ could collide with other vertices and the whole latent is still highly likely to be unique. 2) For the reconstruction of the 3D scenes in particular, the majority of the space would be empty, and the samples in the empty space would contribute a much smaller weight during gradient descent compared to samples on visible surfaces. As a result, the gradients will be dominated by the more important samples and the optimization of the hash grid will naturally reflect the requirements of samples with larger weights.

2.4.3 Non-Neural Gaussian

While neural networks provide great flexibility and smoothness for the reconstruction of various scenes, and the hash grid approach effectively enhances the training and rendering speed, they are still limited by fixed capacity and a lack of editability. As a result, non-neural parametric models have been revisited as adaptive and editable architectures, particularly for modeling the volumetric field as one of the primary implicit representations in 3D reconstruction. Aiming to represent the 5D plenoptic function for a scene, as described in Eq 2.3, Kerbl et al. [48] introduce Gaussian Splatting, a method that optimizes a set of 3D anisotropic Gaussians to represent the volumetric field. Unlike MLP and hash-grid architecture, the Gaussian architecture is specifically designed for modeling volumetric fields, along with a splitting-based rendering approach for efficient optimization and rendering. In this architecture, each Gaussian is described with four parameters: position (Gaussian mean) $\boldsymbol{\mu}$, 3D covariance matrix $\boldsymbol{\Sigma}$, opacity α and Spherical Harmonic (SH) coefficients $\mathbf{SH}$ for computing view-dependent RGB color. For ease of optimization, the covariance matrix is further decomposed into a scaling matrix $\mathbf{S}$, stored as a scaling vector $\mathbf{s}$, and a rotation matrix $\mathbf{R}$, which is stored as a quaternion vector $\mathbf{q}$. The covariance matrix can then be obtained as: $\boldsymbol{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T$.

To render 3D Gaussians to RGB images, their means are projected onto 2D image plane with standard projective transformation, while the projected covariance matrix is obtained by $\boldsymbol{\Sigma}' = \boldsymbol{JW\Sigma W}^T\boldsymbol{J}^T$, where $\mathbf{W}$ is the world to camera transformation and $\mathbf{J}$ is the Jacobian approximating the projective transformation [142]. The rendered RGB color at each pixel is then obtained through:

$$\mathbf{C}(\mathbf{x}) = \sum_{i \in N} \mathbf{c}_i \alpha_i^*(\mathbf{x}) \prod_{j=1}^{i-1} (1 - \alpha_j^*(\mathbf{x})), \tag{2.9}$$

$$\alpha_i^*(\mathbf{x}) = \alpha_i \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i')^T \boldsymbol{\Sigma}'^{-1} (\mathbf{x} - \boldsymbol{\mu}_i') \right), \tag{2.10}$$

where $\mathbf{x}$ is the 2D pixel coordinate, $\mathbf{c}_i$ is the view-dependent RGB radiance of i-th Gaussian on the ray obtained from SH function, α_i and $\boldsymbol{\mu}_i'$ are the opacity and projected 2D mean of the i-th Gaussian respectively.

It is notable that the number of Gaussians needed can be vastly different depending on the scale and level of details of the reconstructed scene. It is not a wise solution to use a fixed number of Gaussians. Therefore, as shown in Fig 2.7, Gaussian Splatting first initializes with the triangulated sparse point cloud from SfM, and then utilizes an adaptive and versatile density control optimization scheme. It is identified that, both under-reconstruction, meaning the existing Gaussians are too small to cover the desired geometry, and over-reconstruction, meaning the existing Gaussians are too large and fail to faithfully represent the desired fine details, are indicated by cumulatively large

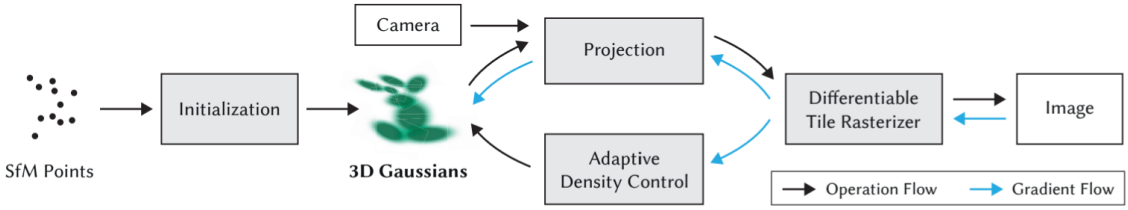


Figure 2.7: **Optimization pipeline of Gaussian Splatting** [48]. Starting optionally from the SfM sparse point cloud, Gaussian Splatting uses an adaptive density control step at fixed intervals to increase or decrease the number of Gaussians in the model, allowing representations of various scenes with both high quality and high efficiency. Figure by Kerbl et al. [48].

gradients on the Gaussian mean parameter $\boldsymbol{\mu}$ during backpropagation. Intuitively, since the existing Gaussians are simply insufficient to represent the scene, they would hence be constantly moved back and forth during each optimization step and could not reach a suitable minimum. By monitoring the cumulative gradients on each Gaussian's mean parameter, it is possible to identify the Gaussians that needed to be split or cloned to increase the number of Gaussians. Besides, all Gaussians will be reset to a low density value between fixed intervals, and the Gaussians that fail to bring the density higher during optimization will be removed to prevent unused Gaussians. This density control scheme effectively allows the number of Gaussians to freely increase or decrease during the optimization, and hence allowing the architecture to model scenes with varying scales and details.

2.5 3D Reconstruction for Specialized Applications

Apart from the general 3D reconstruction task, which aims to recover geometry and appearance from a set of multiview RGB images, there also exist various extensions for specialized applications, which would require tailored architecture and optimization.

2.5.1 Generalization

For typical 3D reconstruction using neural implicit representations, optimization is often conducted on a per-scene basis. This approach requires training the model from scratch for each reconstruction target, without leveraging prior knowledge from other scenes. Such limitations reduce reconstruction efficiency and hence demand a large set of multi-view captures to achieve reliable, high-quality results. To address this issue, many methods focus on generalized reconstruction by incorporating prior knowledge shared across different scenes. A common strategy is to integrate an encoder with 2D convolutional layers to process one or more input images into a compressed, scene-specific latent code. This latent

code is then passed to an MLP to decode the implicit representation [107, 132, 41]. By leveraging epipolar geometric constraints (discussed in Sec 2.2), it becomes possible to project a queried 3D location onto the input 2D images and extract local patch encodings, enabling the capture of finer details. However, this approach often limits the number of input images and requires extensive training data from a diverse set of scenes to generalize effectively.

Another promising direction leverages pre-trained models, such as CLIP [87], to extract rich semantic and visual information. These methods enforce strong semantic consistency across multiple views to ensure that the object remains consistent across all viewing angles and impose natural image constraints, where the renderings must align with the distribution of natural images. This enables high-quality 3D reconstructions from a few input images [40]. However, the effectiveness of these methods heavily depends on the quality and scope of the pre-trained models utilized.

Lastly, heuristic-based constraints have been designed to regularize implicit representations [89, 74, 49]. For example, assuming there exist no semi-transparent surfaces in the scene, then the weights of volume rendering described in Eq 2.4 should be concentrated on a single sample as the optimization converges and the volume rendering would approach surface rendering. Besides, the depth should be smooth apart from the clear edges of the objects, and the entropy of density distribution on a ray should be as low as possible to model concrete surfaces. However, such heuristic-based regularizations have clear limitations and cannot be applied to scenes with semi-transparent materials.

2.5.2 Dynamic Scene Modeling

While most 3D reconstruction methods a static scene during multiview scanning, this assumption often breaks down in real-world scenarios. Besides, the ability to model rigid or non-rigid dynamics within a scene opens new possibilities for applications in gaming, filmmaking, and metaverse environments.

The methods that extend neural implicit representations with the capability to represent time-varying geometry or appearance changes can be categorized based on the level of constraints incorporated. One group of methods directly learns the time-dependent geometry and appearance by conditioning neural networks on a per-frame time latent code. This approach captures dynamic effects such as the appearance and disappearance of objects or visual changes. While it provides flexibility for modeling complex dynamics, it is under-constrained due to the lack of correspondences between adjacent time steps. As a result, reconstructions may lack temporal smoothness, and interpolation for novel time stamps is often unreliable [57, 43]. Besides, time latent code conditioning is less effective for grid-based or Gaussian-based architectures, because the majority of the scene information is localized to allow efficient querying and rendering. For grid-based architecture, one

potential approach is to maintain a different set of hash grids at each time stamp, but it significantly increases the requirements on storage and memory and leads to poor interpolation. Gao et al. [29] resolves this issue by instead optimizing N set of hash grids as the basis, where N is much less than the number of total time stamps in training. The hash grid basis is composed through optimizable per-frame coefficients to form the grid at each time stamp, therefore achieving more storage-efficient solution.

Instead of directly learning the time-dependent effects, another set of works aims to recover the actual deformation in the scene and utilize this as a constraint to the optimization. They maintain a 3D reconstruction at a canonical time frame, which can correspond to a randomly selected time stamp in training data, or may not relate to any specific frame at all. During the optimization of the canonical 3D reconstruction, a time-dependent deformation field is also being optimized simultaneously. Such a deformation field is typically modeled using an MLP with encoded 3D spatial location and a time label as input, and it predicts the corresponding spatial location in the canonical space for each querying location at different time stamps, essentially re-organizing the canonical space to match the observation at different time stamp. To support better convergence in the deformation field, existing methods utilize additional clues such as depth, object mask, and optical flow predicted by pre-trained models to further constrain the optimization [85, 19, 123, 59, 77]. These methods learn high-quality dynamic objects in a canonical frame, but typically suffer from complex dynamic effects such as topological change or complete emergence and disappearance of visual effects such as flame particles. To overcome the limitations of both types of methods, Park et al. [78] combine both techniques from both categories and query a time-conditioned NeRF with deformed coordinates to best utilize the network capacity by introducing a spatially-dependent time latent warping into a hyperspace.

2.5.3 Surface Reconstruction

While volumetric fields represent both geometry and appearance, they often result in volumetric geometry that fails to accurately represent the actual and desirable surfaces. The recovered geometry in volumetric fields tends to be imprecise, noisy, and unsmooth, making it unsuitable for applications requiring high-quality surface reconstruction. In contrast, SDFs excel at modeling smooth and accurate surfaces, but they lack the capability to represent appearance and therefore cannot be trained solely with RGB images. To address this, several methods integrate traditional surface rendering techniques to jointly reconstruct geometry, albedo, and lighting conditions from multiview RGB images [129]. However, these approaches rely on precise mask supervision and struggle to recover sharp texture details.

A more robust method, introduced by NeuS [114], combines SDFs with volumetric fields

to achieve accurate and reliable surface reconstruction. NeuS proposes a mapping function that translates signed distance values in the SDF to density values in the volumetric field, satisfying two key requirements: 1) Unbiased Rendering: the volume rendering weights should peak for samples located on the surface, where the SDF value equals zero. 2) Occlusion Awareness: for two samples along a ray with the same absolute SDF values, the one closer to the camera should have a larger rendering weight due to occlusion effects. By deriving this mapping function, NeuS enables SDFs to be optimized via differentiable volumetric rendering, merging the strengths of both representations.

Despite these advancements, a critical limitation in current surface reconstruction methods is the assumption that scenes consist exclusively of opaque surfaces, where density approaches infinity. For surface-based rendering approaches, this assumption is inherent in the rendering pipeline and cannot be avoided. While volumetric rendering methods like NeuS are technically capable of modeling semi-transparent effects, they often perform poorly due to inherent ambiguities in the volumetric representation. Specifically, in volumetric fields, the density value—which quantifies the probability of a ray being stopped at an infinitesimal sample—can be interpreted in two conflicting ways:

- As the likelihood of a surface's existence (geometry).
- As the opacity of the material (material properties).

For example, a low but positive density value could either indicate an area where a surface is unlikely to exist or signify a translucent surface with low opacity. The original volumetric representation lacks the means to distinguish between these scenarios, making it unsuitable for reconstructing translucent surfaces.

2.5.4 Efficient Rendering

Due to the double integral and numerical quadrature estimation with Monte Carlo sampling in the volume rendering, the optimization and rendering of methods based on the volumetric field can be a very computationally expensive step. In NeRF which first utilizes an MLP architecture to model a volumetric field for 3D reconstruction, rendering of a $512x512$ image could take around 10 sections.

Apart from directly replacing the underlying architecture for neural implicit representations, various methods for efficient training and rendering have been proposed. Such efficiency improvement can be achieved in two distinct directions. One direction is based on the methodology behind hash grid architecture, where the scene information is localized, therefore both optimization and rendering involve less computation while achieving the same performance. Instead of directly incorporating a hash grid architecture, various methods have attempted sub-MLPs responsible for mutually exclusive blocks in the 3D

scene, or have utilized the traditional voxel grid to skip the empty space [90, 65, 62, 88]. Instead of utilizing hash table with latent code and the small global MLP, Yu et al. [**? ?**] directly incorporate an explicit volumetric grid, where each vertex stores explicit density and radiance values and trilinear interpolation is used for obtaining a smooth and continuous volumetric field. To model view-dependent radiance, the explicit volumetric grid models a set of Spherical Harmonic (SH) coefficients to form a low-dimensional SH function, which maps view direction into view-dependent radiance. Such explicit architecture enables straightforward editing and manipulation of the scene, but significantly increases the storage and memory requirement, and produces reconstruction with less smoothness compared to hash grid architecture.

Apart from sub-dividing and localizing the underlying architecture, another set of approaches focuses on reducing the number of samples required for estimating the double integral in volume rendering. Utilizing the theorem of importance sampling, significantly less number of samples would be required if they are drawn from a similar distribution to the rendering weight, therefore, the problem becomes identical to finding the first ray-surface intersections for each ray, assuming the surfaces are opaque. To do so, some works continue to refine and concentrate the sampling around estimated surface intersection during the optimization, and some directly train a sampling network that assists the sampling process [21, 16, 73]. However, the overall efficiency improvement through this approach is less significant than optimizing with localized architecture.

2.5.5 Neural Avatar

The modeling of dynamic neural avatars shares various commonalities with general dynamic scene reconstruction, but places a specialized focus on non-rigid deformation. This task requires capturing intricate details on deformations and high-frequency details, such as facial expressions, hair, and cloth textures, making it inherently more challenging than reconstructing general dynamic objects. Despite the challenge, one benefit that can be utilized in the neural avatar reconstruction is the constrained domain of the target object, where the rich human body or face priors can be incorporated to significantly constrain and simplify the optimization, making fine details reconstruction from limited viewpoints achievable. The most commonly used prior is the parametric 3D Morphable Models (3DMMs) of the body or face, where the human avatar is decomposed into expression/pose and identity, and blendshapes with the linear combination are used to represent the model in a fully differentiable way. Specifically, using the head 3DMM, FLAME [56], as an example, each mesh vertex comes with specific expression and pose blendshapes and LBS weights: $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{n_e \times 3}$ are the expression blendshapes, $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{n_p \times 9 \times 3}$ are the pose blendshapes, $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{n_j}$ are the LBS weights corresponding to each of the n_j bones. The standard

skinning function LBS can then be applied to obtain the vertex transformation matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \text{LBS}(\boldsymbol{B}_{\boldsymbol{\mathcal{P}}}(\theta; \boldsymbol{\mathcal{P}}) + \boldsymbol{B}_{\boldsymbol{\mathcal{E}}}(\psi; \boldsymbol{\mathcal{E}}), \mathbf{J}(\psi), \theta, \boldsymbol{\mathcal{W}}) \tag{2.11}$$

where $\mathbf{J}$ is the joint regressor in FLAME, and $\boldsymbol{B}_{\boldsymbol{\mathcal{P}}}$ and $\boldsymbol{B}_{\boldsymbol{\mathcal{E}}}$ are linear combination of blendshapes based on per-frame coefficients θ and ψ that control the head animation. Similarly, there also exists 3DMM for head (SMPL) [66], hand (MANO) [92], and full body (SMPLX) [80].

To utilize the 3DMMs for 3D reconstruction from RGB images, existing methods typically extend the mesh triangles in the original 3DMMs to be neural implicit representations such as volumetric field. Instead of predefined $\boldsymbol{\mathcal{E}}$, $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{W}}$, a shallow MLP is typically incorporated to predict them based on encoded sampling location and time indicator. This allows the modeling of intricate details such as hair, accessories, and specular reflectance in the eyes, which cannot be easily modeled with the 3DMMs due to their mesh representation and limited capacity. With human-specific constraints and priors from 3DMM, it is possible to reconstruct the head or body from a monocular video [112, 138, 27, 118, 50, 126, 29, 141, 126, 18, 124, 95, 13].

Chapter 3

Geometry-Material Decoupling

In Chapter 2, we introduced various implicit representations for 3D reconstruction and rendering, and specifically exploited the limitation and ambiguity in the volumetric field, a representation widely deployed for image-based 3D reconstruction and novel view synthesis: the volumetric field uses density σ as the main statistic for modeling geometry, with a meaning of the probability of a ray being stopped at an infinitesimal point on the ray. Regardless, such geometry representation is ambiguous as it tightly couples geometry (surface location) with one of the materials (opacity) - σ may signify the likelihood of a surface existing at a particular location or represent the transparency or opacity of the material at that point. This ambiguity does not cause any issue for geometry with high σ, as it can only indicate the existence of opaque surfaces with high confidence. However, it becomes problematic for area with low but non-zero σ values, particularly if reconstructing a scene where semi-transparent materials are likely to exist. In traditional volumetric representation, such σ value could either indicate a region where an opaque surface exists with a small probability, or a low opacity surface exists with a high probability, and there is no suitable approach within the original representation to distinguish between the two cases.

While semi-transparent materials may appear less frequently in general scenarios and hence handling them specifically might not be highly significant, the ability to correctly decouple geometry with material opacity and to model the semi-transparent effect in the image is also crucial for any object with sharp edges and thin shapes with sub-pixel silhouette. As illustrated in Figure 3.1, when rendering a thin structure that only partially occupies a pixel with an insufficient number of sampled rays, opaque foreground objects must be treated as semi-transparent in order to achieve an accurate pixel color blending the foreground and background. Methods that neglect this feature may fail to capture the correct reconstruction of thin structures.

In this chapter, we address the real-world challenge of accurate geometry acquisition for thin and translucent surfaces. Two key requirements must be fulfilled for this task: 1)

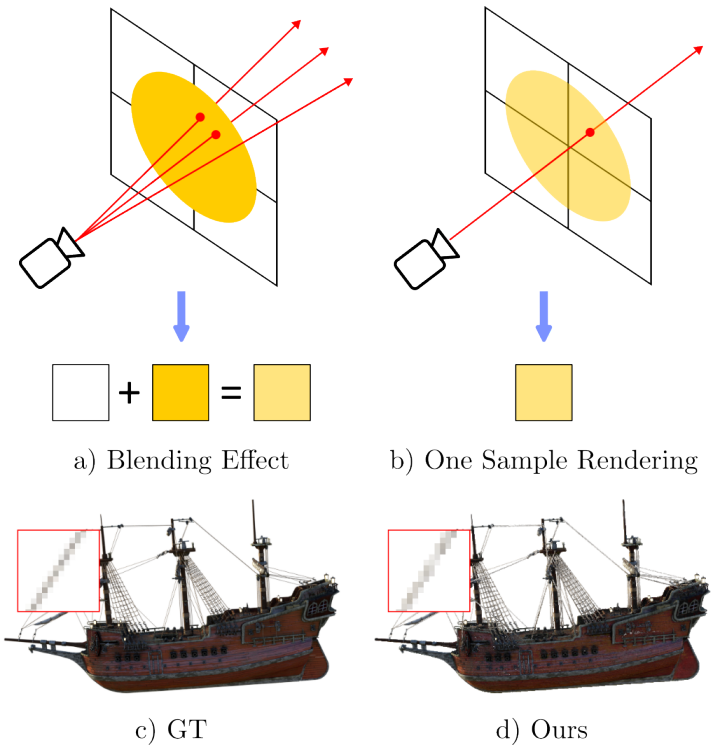


Figure 3.1: **Blending effect.** a) Real-world capture takes incoming light from multiple rays per pixel. Pixels that are partially occupied by an opaque object are therefore rendered as a mixture of the object and background color. b) With one-sample rendering in reconstruction, it becomes necessary for extremely tiny objects to be modeled as semi-transparent for the pixel color to match the ground truth. c), d) Our αSurf representation is fully capable of representing this phenomenon, leading to the accurate surface reconstruction of thin structures.

a representation that explicitly decouples geometry and materials; and 2) a differentiable rendering pipeline that considers more than the nearest ray-surface intersection, while also enabling gradient flows to both surface and opacity. To this end, we introduce αSurf, a novel surface representation based on a grid-based explicit architecture. We use separate values on the grid to represent geometry, opacity, and appearance. We define the surface as multiple level sets of a continuous scalar field, where the field itself is given by a trilinear interpolation of the grid values. Unlike previous methods, our representation does not require the scalar field to be an SDF subject to the Eikonal constraint. An important property of our approach is that the *exact* intersection points between a ray and all the surfaces, regardless of whether they are opaque or transparent, can be determined by analytically solving a cubic polynomial. The *closed-form* solution allows for full *differentiability* to both geometry and material in our forward rendering process, which simulates the semi-transparency effects via alpha compositing of intersection points.

αSurf utilizes a set of specifically designed initializations and optimizations that to facilitate efficient and accurate reconstruction. The total training time of our method

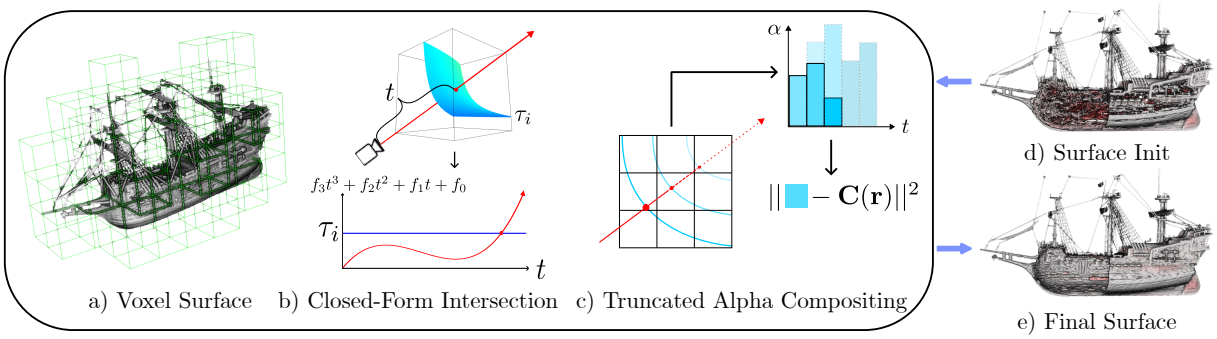


Figure 3.2: **Method.** a) Our surface representation is based on a voxel grid storing explicit values, without neural networks; see Section 3.1.1. b) We utilize a closed-form and differentiable method to compute ray-surface intersection. This is achieved by solving a cubic polynomial of depth t with known parameters $f_0, ..., f_3$ and τ_i; see Section 3.1.2. c) We incorporate surface-specific regularization such as truncated alpha compositing to obtain clean and accurate surfaces; see Section 3.1.3. d) We utilize a coarse initialization via Plenoxels [133] to start with roughly correct yet noisy surfaces. e) The optimization results in clean and complete surfaces in the end.

is around 30 minutes. We evaluate our method on an extended version of the NeRF synthetic dataset [70], which contains 8 original scenes and 16 additional objects with challenging thin structures or translucent materials. We show that our method is capable of reconstructing surfaces with better quality than the existing SDF and volumetric field methods.

In summary, our contributions are: 1) A novel grid-based scene representation for implicit surface reconstruction, specialized for translucent and thin objects. It incorporates a closed-form and differentiable evaluation of all surface intersections along the ray, and properly decouples surface geometry and opacity. 2) Initialization scheme utilizing fast Plenoxels [133] training and optimization with truncated volume rendering and surface smoothness constraint for efficient training and accurate reconstruction. 3) We show superior reconstruction quality compared to state-of-the-art methods on synthetic and real-world scenes featuring thin and translucent objects.

The work presented in this chapter produces the following publication:

- Tianhao Wu, Hanxue Liang, Fangcheng Zhong, Gernot Riegler, Shimon Vainer, Jiankang Deng, Cengiz Oztireli. 2024. AlphaSurf: Implicit Surface Reconstruction for Semi-Transparent and Thin Objects with Decoupled Geometry and Opacity. International Conference on 3D Vision 2025

3.1 Method

Given a set of multi-view posed RGB images, our goal is to recover an implicit geometry in the form of surface locations of the scene objects, particularly in cases involving translucent or thin surfaces. Towards this, we propose αSurf, a grid-based representation where the grid contains values pertaining to the surface's geometry and material properties (Figure 3.2a). This enables a closed-form evaluation of all the ray-surface intersections (Figure 3.2b), and thus a fully differentiable alpha composition (Figure 3.2c) to render the intersection points. Our method pre-trains Plenoxels [133] to efficiently initialize a coarse surface (Figure 3.2d), and through the optimization of a photometric loss and additional regularizations, we are able to reconstruct clean and accurate surfaces (Figure 3.2e).

3.1.1 Representation

Surface Following voxel-based architectures [133, 55], we represent the surface as the level sets of a continuous scalar field $\delta : \mathbb{R}^3 \rightarrow \mathbb{R}$. For each spatial coordinate $\mathbf{x}$ within a voxel $v_{\mathbf{x}}$, the value of the scalar field $\delta(\mathbf{x})$ is obtained by the trilinear interpolation of scalars stored at the voxel vertices:

$$\delta(\mathbf{x}) = \text{Trilearp}(\mathbf{x}, \{\hat{\delta}_i\}_{i=1}^{8}) \tag{3.1}$$

where $\{\hat{\delta}_i\}_{i=1}^{8}$ are the surface scalars stored at its eight adjacent vertices. The surface is then implicitly defined as level sets on the scalar field. Specifically, given a set of n constants $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{n}$ which we refer to as *level values*, the surface $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 | \exists \tau \in \boldsymbol{\tau} : \delta(\mathbf{x}) = \tau\}\,. \tag{3.2}$$

The cardinality n and values of $\boldsymbol{\tau}$ are determined through hyperparameter tuning. Note that this implicit surface field does not model distance to the closest surface as in SDF, therefore, it is not subject to the Eikonal constraint. The motivation behind the multi-level set is tightly related to our initialization strategy and is further explained in Sec 3.1.3.

Opacity and Appearance Within the same voxel grid, we also model the surface opacity denoted as $\alpha(\mathbf{x})$ and view-dependent appearance denoted as $\mathbf{c}(\mathbf{x}, \mathbf{d})$ in the same explicit style as in Plenoxels [133], which represents $\mathbf{c}(\mathbf{x}, \mathbf{d})$ as coefficients of the spherical harmonic (SH) function that maps view direction $\mathbf{d}$ to radiance. Trilinear interpolation is applied to obtain opacity and SH coefficients at surface locations. Note that although $\alpha(\mathbf{x})$ is essentially defined across all valid voxels in the 3D space, it only represents surface material property rather than geometry, and hence is only meaningful where surface exists.

3.1.2 Differentiable rendering

A key feature in the rendering process of our representation is that it does not involve any Monte-Carlo sampling as in NeRF or sphere tracing as in SDF-based methods. Instead, it relies on ray-voxel traversal and directly takes samples at the ray-surface intersections found through a *closed-form* and fully *differentiable* function. Specifically, for each camera ray with origin $\mathbf{o}$ and direction $\mathbf{d}$, we first determine the set of voxels it traverses through and substitute the ray equation $\mathbf{r}(t) = \mathbf{x} = \mathbf{o} + t\mathbf{d}$ into Eq 3.4. for each voxel $v_{\mathbf{x}}$. Our aim is then to find the intersections between the ray and a level set surface with value τ_i within a voxel $v_{\mathbf{x}}$, which $\mathbf{r}(t)$ is guaranteed to hit. The value of the implicit surface field within $v_{\mathbf{x}}$ can be determined through the trilinear interpolation of eight surface scalars stored on the vertices of $v_{\mathbf{x}}$:

$$\begin{aligned}\delta(\mathbf{x}) =& \text{trilerp}(\mathbf{x}, \{\hat{\delta}_i\}_{i=1}^{8}) && (3.3)\\ =&(1-z)((1-y)((1-x)\hat{\delta}_1 + x\hat{\delta}_5)\\ &+ y((1-x)\hat{\delta}_3 + x\hat{\delta}_7))\\ &+ z((1-y)((1-x)\hat{\delta}_2 + x\hat{\delta}_6)\\ &+ y((1-x)\hat{\delta}_4 + x\hat{\delta}_8)) && (3.4)\end{aligned}$$

where $[x, y, z] = \mathbf{x} - \mathbf{l}$ are the relative coordinates within the voxel, and $\mathbf{l} = \text{floor}(\mathbf{x})$. Note that $x, y, z \in [0, 1]$. We first determine the near and far intersections t_n, t_f between the ray and voxel $v_{\mathbf{x}}$ through the ray-box AABB algorithm, and then redefine a new camera origin $\mathbf{o}' = \mathbf{o} + t_n\mathbf{d} - \mathbf{l}$. We hence directly have $[x, y, z] = \mathbf{o}' + t'\mathbf{d} \in [0, 1]$, where $t' = t - t_n$ without the need for calculating relative coordinates again. By denoting $\mathbf{o}' = [o'_x, o'_y, o'_z]$, $\mathbf{d} = [d_x, d_y, d_z]$, we substitute the above as well as $\delta(\mathbf{x}) = \tau_i$ into Equation 3.4:

$$\begin{aligned}\tau_i =&(1-(o'_z + t'd_z))((1-(o'_y + t'd_y))\\ &((1-(o'_x + t'd_x))\hat{\delta}_1 + (o'_x + t'd_x)\hat{\delta}_5)\\ &+ (o'_y + t'd_y)((1-(o'_x + t'd_x))\hat{\delta}_3 + (o'_x + t'd_x)\hat{\delta}_7))\\ &+ (o'_z + t'd_z)((1-(o'_y + t'd_y))\\ &((1-(o'_x + t'd_x))\hat{\delta}_2 + (o'_x + t'd_x)\hat{\delta}_6)\\ &+ (o'_y + t'd_y)((1-(o'_x + t'd_x))\hat{\delta}_4 + (o'_x + t'd_x)\hat{\delta}_8))\,. && (3.5)\end{aligned}$$

By re-arranging the equation, we obtain:

$$\tau_i = f_3 t'^3 + f_2 t'^2 + f_1 t' + f_0 \tag{3.6}$$

where

$$\begin{aligned}
f_0 =&(m_{00}(1-o'_y) + m_{01}(o'_y))(1-o'_x) \\
&+ (m_{10}(1-o'_y) + m_{11}(o'_y))(o'_x) \\
f_1 =&(m_{10}(1-o'_y) + m_{11}(o'_y))d_x + k_1(o'_x) \\
&- (m_{00}(1-o'_y) + m_{01}(o'_y))d_x + k_0(1-o'_x) \\
f_2 =&k_1 d_x + h_1(o'_x) - k_0 d_x + h_0(1-o'_x) \\
f_3 =&h_1 d_x - h_0 d_x
\end{aligned} \tag{3.7}$$

and

$$\begin{aligned}
m_{00} =&\hat{\delta}_1(1-o'_z) + \hat{\delta}_2(o'_z) \\
m_{01} =&\hat{\delta}_3(1-o'_z) + \hat{\delta}_4(o'_z) \\
m_{10} =&\hat{\delta}_5(1-o'_z) + \hat{\delta}_6(o'_z) \\
m_{11} =&\hat{\delta}_7(1-o'_z) + \hat{\delta}_8(o'_z) \\
k_0 =&(m_{01}d_y + d_z(\hat{\delta}_4 - \hat{\delta}_3)(o'_y)) \\
&- (m_{00}d_y - d_z(\hat{\delta}_2 - \hat{\delta}_1)(1-o'_y)) \\
k_1 =&(m_{11}d_y + d_z(\hat{\delta}_8 - \hat{\delta}_7)(o'_y)) \\
&- (m_{10}d_y - d_z(\hat{\delta}_6 - \hat{\delta}_5)(1-o'_y)) \\
h_0 =&d_y d_z(\hat{\delta}_4 - \hat{\delta}_3) - d_y d_z(\hat{\delta}_2 - \hat{\delta}_1) \\
h_1 =&d_y d_z(\hat{\delta}_8 - \hat{\delta}_7) - d_y d_z(\hat{\delta}_6 - \hat{\delta}_5)\,.
\end{aligned} \tag{3.8}$$

Therefore, we obtain a cubic polynomial with a single unknown t'. Note that here we only sketch the main idea. For the actual implementation, we refer to [35] which provides a more concise implementation that formulates the cubic polynomials with fewer operations through the use of fused-multiply-add.

We then incorporate Vieta's approach [83] to solve the real roots for t' in an analytic way. Namely, we first re-write the cubic polynomial as follows:

$$\tau_i = t'^3 + at'^2 + bt' + c \tag{3.9}$$

$$a = \frac{f_2}{f_3}, b = \frac{f_1}{f_3}, c = \frac{f_0}{f_3}\,. \tag{3.10}$$

Then, compute:

$$Q = \frac{a^2 - 3b}{9} \tag{3.11}$$

$$R = \frac{2a^3 - 9ab + 27c}{54}\,. \tag{3.12}$$

If $R^2 < Q^3$, we have three real roots given by:

$$\theta = \arccos(\frac{R}{\sqrt{Q^3}}) \tag{3.13}$$

$$t'_1 = -2\sqrt{Q}\cos(\frac{\theta}{3}) - \frac{a}{3} \tag{3.14}$$

$$t'_2 = -2\sqrt{Q}\cos(\frac{\theta - 2\pi}{3}) - \frac{a}{3} \tag{3.15}$$

$$t'_3 = -2\sqrt{Q}\cos(\frac{\theta + 2\pi}{3}) - \frac{a}{3} \tag{3.16}$$

where $t'_1 \leq t'_2 \leq t'_3$. This can be trivially seen from $0 \leq \theta \leq \pi$, $\sqrt{Q} \geq 0$ and $\cos(\frac{\theta}{3}) \geq \cos(\frac{\theta-2\pi}{3}) \geq \cos(\frac{\theta+2\pi}{3})$.

If $R^2 \geq Q^3$, we only have a single real root. First compute:

$$A = -\operatorname{sign}(R)\left(|R| + \sqrt{R^2 - Q^3}\right)^{1/3} \tag{3.17}$$

$$B = \begin{cases} Q/A, & \text{if } A \neq 0 \\ 0, & \text{otherwise} \end{cases}\,. \tag{3.18}$$

Then, the only real root can be obtained as:

$$t'_1 = (A + b) - \frac{a}{3}\,. \tag{3.19}$$

The intersection coordinate can therefore be determined as $\mathbf{r}(t_n + t')$. We then check the intersections against the bounding box of each voxel $v_{\mathbf{x}}$ to remove any samples outside of the voxels. Besides, the cubic polynomial might return multiple valid real roots within the voxel if $R^2 < Q^3$. If the roots are unique, that means the ray intersects with the same surface multiple times within the voxel, all the intersections are taken for rendering. However, if the roots are identical, we remove the redundant ones to prevent using the same intersection multiple times.

As both the formulation of cubic polynomials and Vieta's approach are fully differ-

entiable, we hence directly have gradients defined on our surface representation $\hat{\delta}$ from the photometric loss. The remaining valid intersections are ordered from near to far and taken as samples for rendering. For all intersection points $\{t_i\}$ along the ray, we obtain their surface opacity α and view-dependent radiance $\mathbf{c}$ through trilinear interpolation and evaluating the SH function, and then perform alpha compositing to render the pixel color:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_\alpha(t_i)\alpha(t_i)\mathbf{c}(t_i) \tag{3.20}$$

$$T_\alpha(t_i) = \prod_{j=1}^{i-1}(1 - \alpha(t_j)) \tag{3.21}$$

where we simplify our notation as $\alpha(t_i) \equiv \alpha(\mathbf{r}(t_i))$ and $\mathbf{c}(t_i) \equiv \mathbf{c}(\mathbf{r}(t_i), \mathbf{d})$.

3.1.3 Optimization

NeRF Initialization One advantage of our representation is that we can easily initialize coarse surfaces from a pre-trained NeRF. In practice, we use Plenoxels [133], a grid-based NeRF method that can be trained efficiently within 10 minutes. After fitting a NeRF, we obtain a density field $\sigma(\mathbf{x})$ from which we select a set of raw level values $\boldsymbol{\tau}_\sigma$ to define the initial surfaces. The values of $\boldsymbol{\tau}_\sigma$ are selected by first determining an upper bound of the initialized density field, then evenly dividing the range by n level values. The number of level sets n is determined by hyperparameter-tuning. We then normalize the density $\sigma(\mathbf{x})$ to be used as our initial surface scalars $\delta(\mathbf{x})$, and normalize the raw level values to be used as our level values:

$$\delta(\mathbf{x}) = \frac{\sigma(\mathbf{x}) - \tilde{\tau}_\sigma}{\overline{||\nabla\sigma||}}, \boldsymbol{\tau} = \left\{ \frac{\tau_\sigma - \tilde{\tau}_\sigma}{\overline{||\nabla\sigma||}} \mid \tau_\sigma \in \boldsymbol{\tau}_\sigma \right\} \tag{3.22}$$

where $\tilde{\tau}_\sigma$ is the median of the chosen raw level values, $\overline{||\nabla\sigma||}$ is the average over the norms of the finite difference gradient of the voxelated density field. $\overline{||\nabla\sigma||}$ is used to keep the initialized surface field within a constant range to make optimization easier. Note that $\boldsymbol{\tau}_\sigma$ is a hyperparameter and is defined only to make the selection of initial surface level values more convenient, whereas $\boldsymbol{\tau}$ is the actual level set values of the implicit surface field used throughout the optimization.

Although in theory, a single level set is sufficient to represent the surfaces, our multi-level set initialization scheme allows the geometric information from NeRF to be maximally preserved and inherited to the surface field – In NeRF, high density regions represent opaque geometries with high confidence, whereas low density regions can be either low-confidence surfaces or translucent surfaces and hence are ambiguous. Multi-level sets initialization therefore simultaneously captures high-confidence opaque geometry with

higher level values, and translucent geometry or low-confidence noise region with lower level values. Later optimization can then easily identify and remove those redundant noise surfaces. The effect of multi-level set initialization is also shown empirically in Sec 3.2.5.

We also initialize the opacity and SH field from the pre-trained Plenoxels to facilitate optimization. After taking the raw density values σ in the voxel grid, they are rescaled with a constant s_σ to be used as raw surface opacity σ_α, then mapped to opacity through a combined exponential-ReLU activation:

$$\sigma_\alpha = s_\sigma \, \sigma, \; \alpha = 1 - \exp\left(-\operatorname{ReLU}(\sigma_\alpha)\right) \tag{3.23}$$

Note that this activation resembles the mapping from discretized sample density to opacity in volume rendering where the step size term is replaced by the rescaling factor s_σ [70]. Unlike sigmoid which has a vanishing gradient towards 0, this activation can more easily encourage sparsity in the surface opacity to remove redundant surfaces. The rescaling s_σ is to ensure that initialized raw opacities do not map to α with too high values after removing the step size term, causing the gradients to be saturated. Note that, during training, we update σ_α as the training parameters rather than α. The SH coefficients are also initialized from the pre-trained Plenoxels and further optimized.

In comparison to our approach, SDF methods such as NeuS [114] cannot easily take the advantages of initializing from Plenoxels or other NeRF-based methods due to two key aspects: 1) initializing the weights of the network to become an SDF that matches the coarse shape learned by NeRF is difficult, as it requires additional optimization to fit the shape while satisfying the Eikonal constraint; and 2) even for explicit grid-based SDF methods [45, 111], algorithms such as fast sweeping [17, 137] are required to explicitly assign the grid values as distance to closest surface [2, 100, 99]. This process can be done efficiently, but it no longer preserves the information in the initialized density field, making the removal of redundant surfaces and recovery of missing surfaces more difficult.

Truncated Alpha Compositing A critical artifact in NeRF methods, including Plenoxels, is that they tend to learn low-density surfaces and inner volumes that are only visible from certain angles to represent high-frequency view-dependent appearance [110]. This leads to very noisy inner surfaces when initialized from Plenoxels; see Figure 3.2. To regularize those artifacts, we first remove ray intersections on backward-facing surfaces, and define the remaining set $\mathcal{T}$ of intersection points to be considered for rendering and regularization:

$$t_i \in \mathcal{T} \text{ if } \underbrace{\mathbf{n}(t_i) \cdot \mathbf{d} < 0}_{\text{back-face culling}} \tag{3.24}$$

where t_i is an original intersection, $\mathbf{n}(t_i) = \frac{-\nabla\delta(t_i)}{||\nabla\delta(t_i)||}$ is the surface normal, and $\mathbf{n}(t_i) \cdot \mathbf{d}$ checks whether the surface is facing backwards. This is similar to back-face culling in rendering. We then apply additional constraints by incorporating a truncated version of alpha compositing, where the later intersections along the ray are down-weighted with a truncated Hann window [77] to give less contribution to the rendering. Specifically, we obtain the re-weighted sample opacity as:

$$\alpha^*(t_i) = \gamma(i-1)\ \alpha(t_i) \tag{3.25}$$

$$\gamma(x) = \left(1 - \cos(\pi\,\mathrm{clamp}(a - x, 0, 1))\right)/2 \tag{3.26}$$

where i is the index of the intersection starting from 1. Note that this is the index with the back-face intersections excluded. During training, we linearly reduce a so that $\gamma(i-1)$ is only greater than zero for a smaller range of i. I.e., only the first ceil(a) intersections are kept and the rest are truncated. We now re-define our alpha compositing using this truncated version of opacity values: $\hat{C}^*(\mathbf{r}) = \sum_{t_i \in \mathcal{T}} T^*_\alpha(t_i)\alpha^*(t_i)\mathbf{c}(t_i)$, where $T^*_\alpha(t_i) = \prod_{j=1}^{i-1}(1 - \alpha^*(t_j))$.

Regularization We additionally apply regularizations to enforce smooth surfaces and mitigate the artifacts inherited from initialization. As previously mentioned, we initialize with multiple level sets to preserve geometry priors. However, this can lead to redundant surfaces and potentially deteriorate the reconstruction quality. We hence apply a surface convergence loss to each ray-surface intersection with non-trivial truncated opacity to converge different level surfaces together:

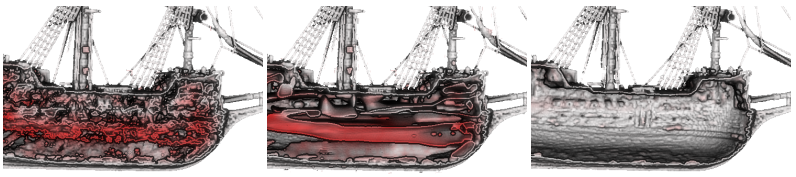
$$\mathcal{L}_c(\mathbf{r}) = \sum_{t_i \in \mathcal{T}} |\tilde{t} - t_i|\ \mathbb{I}[\alpha^*(t_i) > 10^{-8}] \tag{3.27}$$

where $\tilde{t}$ is the depth of the sample with the highest rendering weight $w(t_i) = T^*_\alpha(t_i)\alpha^*(t_i)$ on the ray, and $\mathbb{I}$ is the indicator function. This loss remedies the out-growing surfaces initialized from multi-level sets by encouraging them to move towards the actual surface location. The surface is also smoothed via a combination of an L1 and squared L2 normal smoothness loss and a total variation (TV) loss applied on the surface field:

$$\mathcal{L}_{\mathbf{n}_1} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{x}\in\mathcal{V}} |\nabla\mathbf{n}(\mathbf{x})|,\ \mathcal{L}_{\mathbf{n}_2} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{x}\in\mathcal{V}} ||\nabla\mathbf{n}(\mathbf{x})||^2 \tag{3.28}$$

$$\mathcal{L}_\delta = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{x}\in\mathcal{V}} ||\nabla\hat{\delta}(\mathbf{x})|| \tag{3.29}$$

where $\mathcal{V}$ contains the spatial coordinates of all voxels, $\nabla\hat{\delta}(\mathbf{x})$ is the surface gradient at voxel grids obtained using finite differences on the adjacent grid values. Note that



w/o $\mathcal{L}_\delta, \mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}$ w/o $\mathcal{L}_\delta$ Ours

Figure 3.3: Reconstruction without regularization tends to give an enormous amount of redundant inner surfaces. Encouraging consistent normals in the local neighborhood via $\mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}$ enforces smoother surfaces, but redundant surfaces still remain. With both normal regularization and TV loss applied on the surface scalar field, we can obtain clean and smooth reconstruction.

normal regularization is applied regardless of whether the voxel contains a valid surface or not, as it can help to smooth the overall surface field instead of just the actual surface. While the normal regularization $\mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}$ encourages smooth surfaces in a local scope, it alone struggles to remove redundant inner surfaces. $\mathcal{L}_\delta$ is more effective for regularizing redundant surfaces with large errors; see Figure 3.3. Note that the use of $\mathcal{L}_\delta$ is only possible because our implicit surface field is not an SDF and does not need to satisfy the Eikonal constraint.

Finally, we regularize the opacity field with a weight-based entropy loss adapted from [49] on each ray and an L1 sparsity loss:

$$\mathcal{L}_\mathcal{H}(\mathbf{r}) = -\sum_{t_i \in \mathcal{T}} \bar{w}(t_i) \log(\bar{w}(t_i)) \tag{3.30}$$

$$\text{where } \bar{w}(t_i) = \frac{T^*_\alpha(t_i)\alpha^*(t_i)}{\sum_{t_j \in \mathcal{T}} T^*_\alpha(t_j)\alpha^*(t_j)} \tag{3.31}$$

$$\mathcal{L}_\alpha = \frac{1}{|\mathcal{V}'|} \sum_{\mathbf{x} \in \mathcal{V}'} |\operatorname{ReLU}(\sigma_\alpha(\mathbf{x}))| \tag{3.32}$$

where $\mathcal{V}'$ is 10% of all existing voxels sampled uniformly at each iteration. They together encourage surfaces to have more concentrated and minimal opacity. Note that $\mathcal{L}_\mathcal{H}(\mathbf{r})$ does not always give beneficial regularization for scenes with semi-transparent materials, but we empirically found that having this term with a small weight can help remove the noise in the surface opacity.

Given a batch B of rays, the optimization target is:

$$\begin{aligned}\mathcal{L} = \frac{1}{|B|} \sum_{\mathbf{r} \in B} \Big(&||C(\mathbf{r}) - \hat{C}^*(\mathbf{r})||^2 + \lambda_c \mathcal{L}_c(\mathbf{r}) + \lambda_{\mathbf{n_1}} \mathcal{L}_{\mathbf{n_1}} \\ &+ \lambda_{\mathbf{n_2}} \mathcal{L}_{\mathbf{n_2}} + \lambda_\delta \mathcal{L}_\delta + \lambda_\mathcal{H} \mathcal{L}_\mathcal{H}(\mathbf{r}) + \lambda_\alpha \mathcal{L}_\alpha \Big)\end{aligned} \tag{3.33}$$

where $\lambda_c, \lambda_{\mathbf{n_1}}, \lambda_{\mathbf{n_2}}, \lambda_\delta, \lambda_\mathcal{H}, \lambda_\alpha$ are hyperparameters. The optimization targets include surface scalar δ, grid SH coefficients and raw opacity σ_α.

	Thin	Translucent	Avg
Plen	0.526	0.761	0.644
Mip360	1.445	3.063	2.254
NeuS	1.048	2.344	1.696
HFS	0.925	3.698	2.312
neuralangelo	0.424	1.125	0.774
NeRRF	2.349	2.086	2.218
Ours	0.284	0.624	0.454

Table 3.1: **Chamfer distance** $\downarrow \times 10^{-2}$ **on synthetic datasets.** We highlight the best methods.

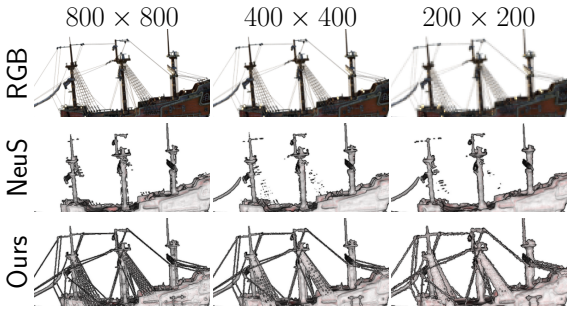


Figure 3.4: **Surface reconstruction with lower resolutions.** Our method can reconstruct thin surfaces under a low resolution, where thin structures such as ropes heavily blend with the background.

3.2 Evaluation

We quantitatively and qualitatively evaluate our method on an extended version of the NeRF synthetic dataset [70], with 8 additional objects with delicate and thin structures, and 8 objects with translucent materials. We also show a qualitative comparison on challenging real-world scenes containing translucent surfaces. We compare with recent SDF optimization methods including NeuS [114] and HFS [116], and NeRF-based methods including Plenoxels [133] and MipNeRF360 [6].

3.2.1 Datasets

Synthetic Thin & Translucent: We render 8 scenes with thin structures and 8 with translucent surfaces with Blender [14]. We sampled 100 different training views from a full sphere. For Thin dataset, we included the "ficus" and "ship" scenes from the original NeRF Synthetic dataset [70]. For each scene, we also rendered the depth and converted them to dense point clouds for quantitative geometry evaluation. This removes any invisible inner structures in the 3D assets. Except for the translucent scene "monkey" and "vase" where part of the object is completely surrounded by translucent surfaces, we hence directly extracted the scene meshes as the ground truth geometry. We will release all the datasets with reference geometry upon publication.

Real-World: We additionally captured a real-world scene with thin structures and two with translucent surfaces to qualitatively evaluate our performance. The real-world captures are processed with Colmap [97, 98] to obtain camera parameters. Note that since our work aims to resolve the geometry-material ambiguity in image-based neural reconstruction instead of handling complicated specular reflection or light transport, we hence focus on real-world thin translucent surfaces with less obvious view-dependent effects, such as cups.

Figure 3.5: **Qualitative evaluation on Thin datasets.** The red color indicates the L1 error in the reconstruction. Our method can accurately reconstruct thin surfaces missed in NeuS and HFS, while recovering more accurate and noise-free surfaces compared to neuralangelo and NeRF-based methods.

3.2.2 Baselines

We compared our method with the state-of-the-art SDF-based reconstruction methods, as well as NeRF-based methods with level set geometry. We compare with NeuS [114] HFS [116], GeoNueS [24] and neuralangelo [58]. Since GeoNueS [24] requires SfM points and visibility masks as input, we only compare with it qualitatively in real-world scenes. We also compare with Plenoxels [133] and MipNeRF 360 [6], a follow-up of NeRF with conical frustum sampling and weight regularization. We also compare with NeRRF [12], which reconstructs fully transparent objects with additional mask supervision.

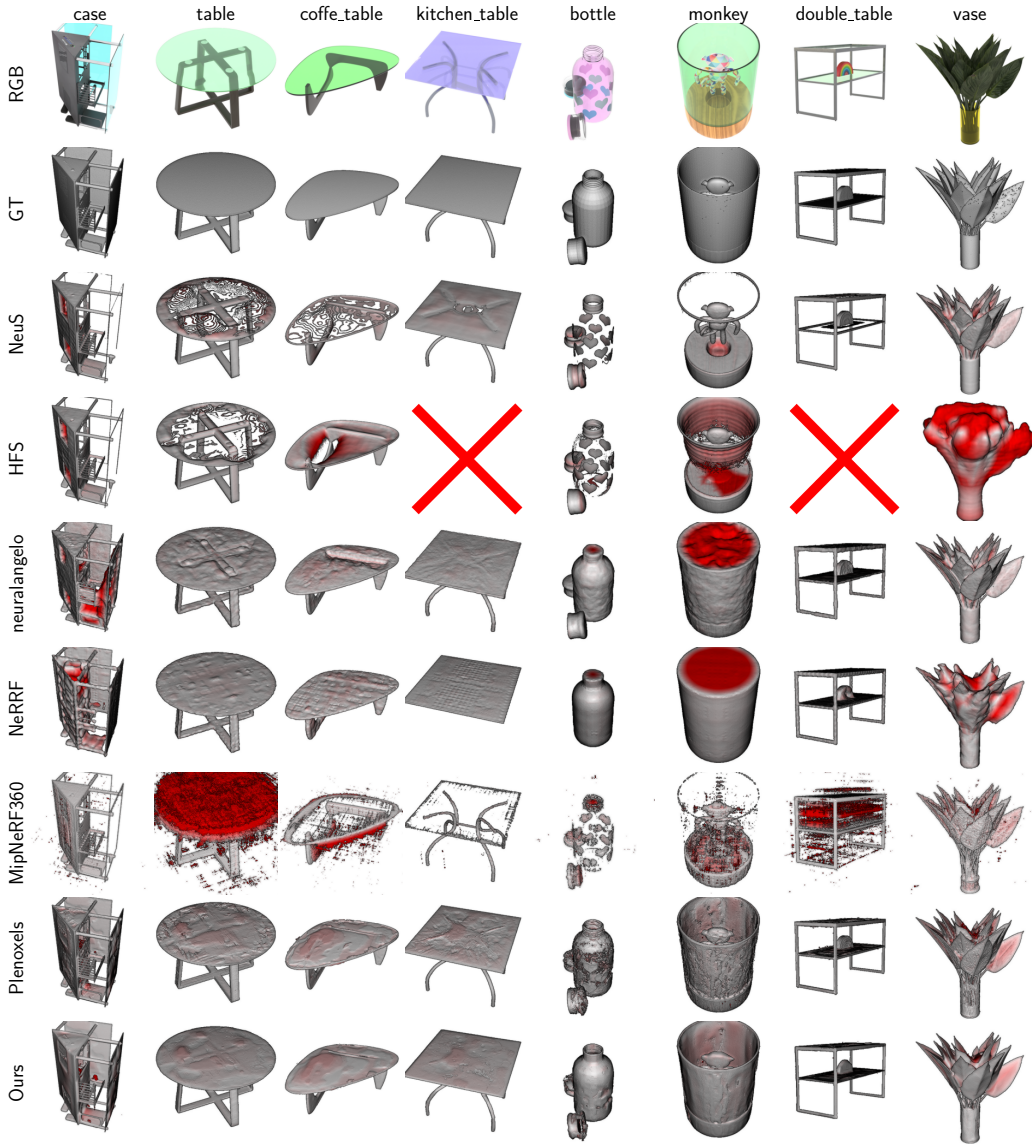


Figure 3.6: **Qualitative evaluation on Translucent datasets.** Our method can properly recover the translucent surfaces missed other methods.

3.2.3 Evaluation on Synthetic dataset

We quantitatively evaluate our method on the synthetic datasets and report the Chamfer-L1 distance as evaluated in [42] in Table 3.1. HFS failed and learned empty surfaces on two Translucent scenes, while neuralangelo failed on one scene in Thin Blender. In comparison, our method signifcantly outperforms the baselines, and the difference is particularly noticeable when compared to SDF-based methods on the translucent dataset, as the baselines cannot properly represent translucent surfaces. NeRF-based methods can potentially recover the translucent surfaces with a low density level set, but their ambiguity in representation leads to significant noise in the reconstruction.

We show the qualitative comparison in Figure 3.5 and Figure 3.6. While NeuS, HFS

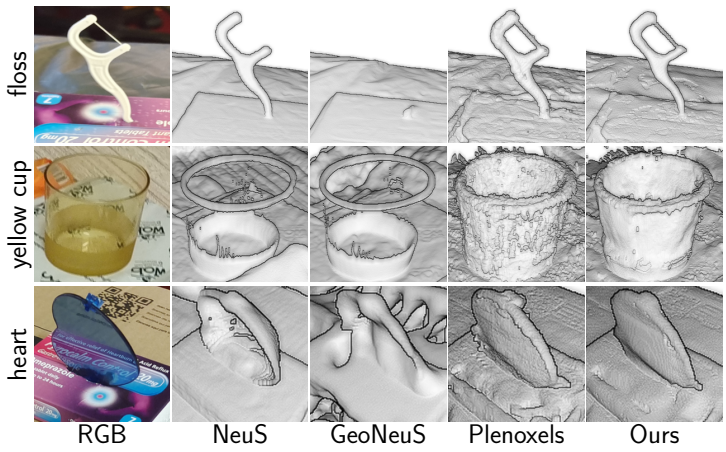


Figure 3.7: **Qualitative evaluation on translucent real-world scenes.** The images are not aligned perfectly due to the use of NDC during optimization.

name	ship	ficus	table	monkey
no tv	.317	.279	.373	.777
notvnorm	.373	.333	.404	.970
no trunc	.522	.837	.431	1.03
no conv	.287	.266	.412	.857
single lv	.624	.583	.470	.946
Ours	.277	.240	.373	.776

Table 3.2: **Quantitative results of ablation study.** We report the Chamfer distance $\times 10^{-2}$ on two scenes from each synthetic dataset.

and Neuralangelo are capable of reconstructing highly smooth surfaces, they cannot properly capture translucent or thin surfaces. Surfaces from Plenoxels and MipNeRF360 contain holes and floaters and are unsmooth. NeRRF is capable of reconstructing smooth translucent surfaces, but fails to model intricate structures. Our approach is capable of reconstructing surfaces with minimal artifacts. In Figure 3.4, we show surfaces reconstructed from 800, 400, and 200 resolution images on a scene with thin structures. Our method can reconstruct the thin surfaces despite the heavy blending effects in the low-resolution images.

3.2.4 Evaluation on Real World Dataset

We show the qualitative comparison on real-world scenes with thin and translucent surfaces in Figure 3.7. We show surfaces extracted Plenoxels using $\sigma = 10$ level sets, as it has the best qualitative results compared to $30, 50$ levels. Neus and GeoNueS fail to reconstruct the majority of the thin or translucent surface, while surfaces from Plenoxels are noisy and unsmooth. Our method mitigates the artifacts inherited from NeRF initialization and reconstructs surfaces with much higher quality. We would like to highlight that, due to the highly ill-posed nature of the problem and view-dependent appearance changes caused by global brightness shifts in an uncontrolled capture environment, it is nearly impossible to reconstruct the translucent surfaces perfectly. Our method achieves significant improvement upon baselines, validating the feasibility of our approach.

3.2.5 Ablation

As shown in Table 3.2 and Figure 3.8, the truncated alpha compositing deals with the inner volume artifacts inherited from initialization, while the surface regularization $\mathcal{L}_{\mathbf{n}_1}, \mathcal{L}_{\mathbf{n}_2}, \mathcal{L}_\delta$

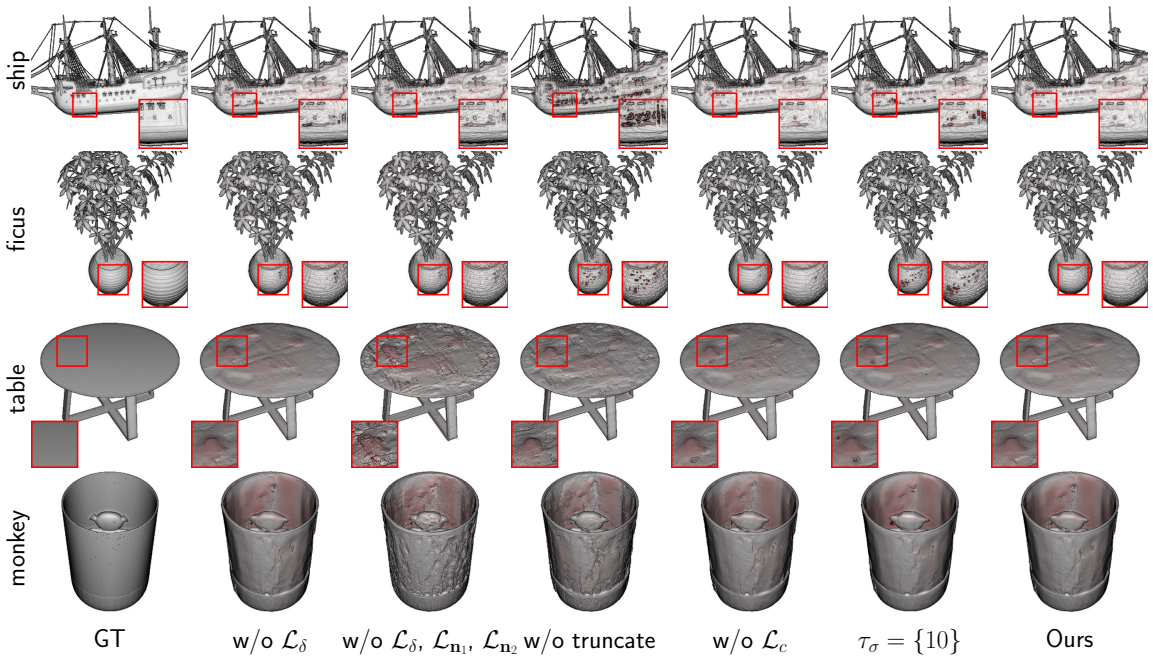


Figure 3.8: **Ablation Study.** We show the qualitative results of different ablations. Our full approach achieves the best quality overall.

and convergence loss $\mathcal{L}_c$ encourage smooth and accurate surfaces. Using a single level set $\boldsymbol{\tau}_\sigma = \{10\}$ to initialize the surface fails to capture all information in the pre-trained NeRF and causes artifacts in optimization. Ours with all techniques enabled achieves the best performance.

In addition, we show a comparison between the results after applying our TV surface regularization $\mathcal{L}_\delta$ and after applying the Eikonal constraint regularization used in most SDF optimization methods in Figure 3.9. Namely, in replace of TV surface regularization, we encourage the norm of the gradient of the surface field at every vertex to get close to 1 via mean squared error:

$$\mathcal{L}_{ek} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{x} \in \mathcal{V}} (||\nabla \hat{\delta}(\mathbf{x})||_2 - 1)^2 \,. \tag{3.34}$$

From Figure 3.9, it can be clearly seen that the Eikonal constraint is not sufficient to regularize and remove the noisy inner surfaces inherited from initialization. Moreover, it turns out to even harm the optimization by introducing additional surface floaters while trying to constrain the surface field into an SDF. This also shows that converting the surfaces extracted from a density field into proper SDF is a non-trivial task.

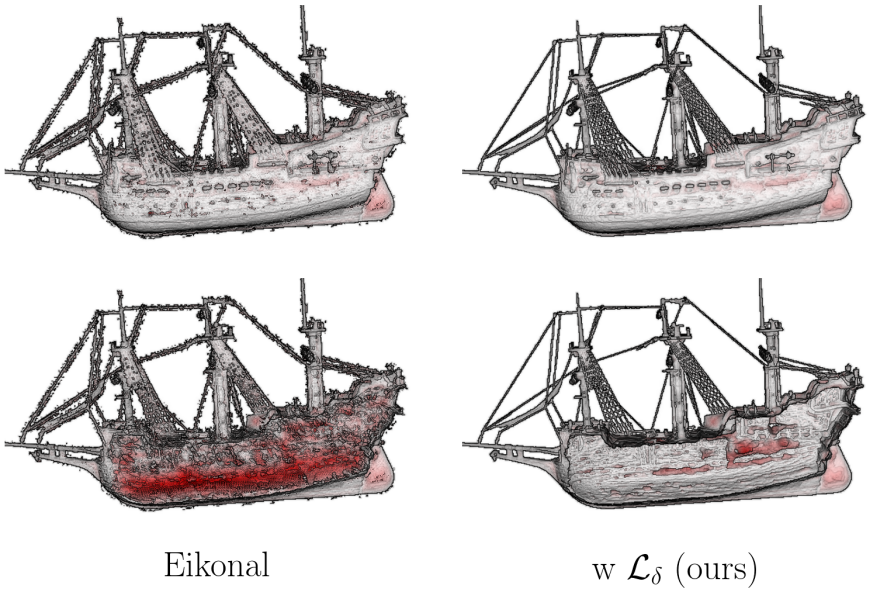


Figure 3.9: **Comparison with the Eikonal constraint regularization.** We visualize the out view (first column) and the inside view (second column) by cropping the surfaces along the y-axis. It can be clearly seen that the Eikonal constraint does not regularize the surface to be clean and smooth, but rather creates additional noises in optimization.

3.2.6 Novel View Synthesis

We show some novel view RGB renderings of our method in Figure 3.10. Although our method does not specialize in modeling appearance or synthesizing high-quality novel views, the illustrations show that our method can properly learn the translucent appearance for accurate image-based geometry recovery.

3.3 Summary

In this chapter, we present αSurf, a grid-based surface representation with decoupled geometry, opacity, and appearance. We develop closed-form intersection finding and differentiable alpha compositing to optimize the surface via photometric loss. Our representation utilizes initialization from efficient Plenoxels [133], and incorporates truncated rendering and additional surface regularizations to reconstruct high-quality surfaces for translucent objects and thin structures with heavy blending effects. This improved representation decouples the fundamental ambiguity in the volumetric geometry, hence enabling accurate reconstruction and modeling of surfaces with translucent effects.

Limitation: Compared to MLP-based SDF methods such as NeuS [114] and neuralangelo [58], our reconstructed surface tends to be less smooth due to the lack of spatial smoothness encoded in the MLP; see Figure 3.5. It presents a trade-off: stronger surface

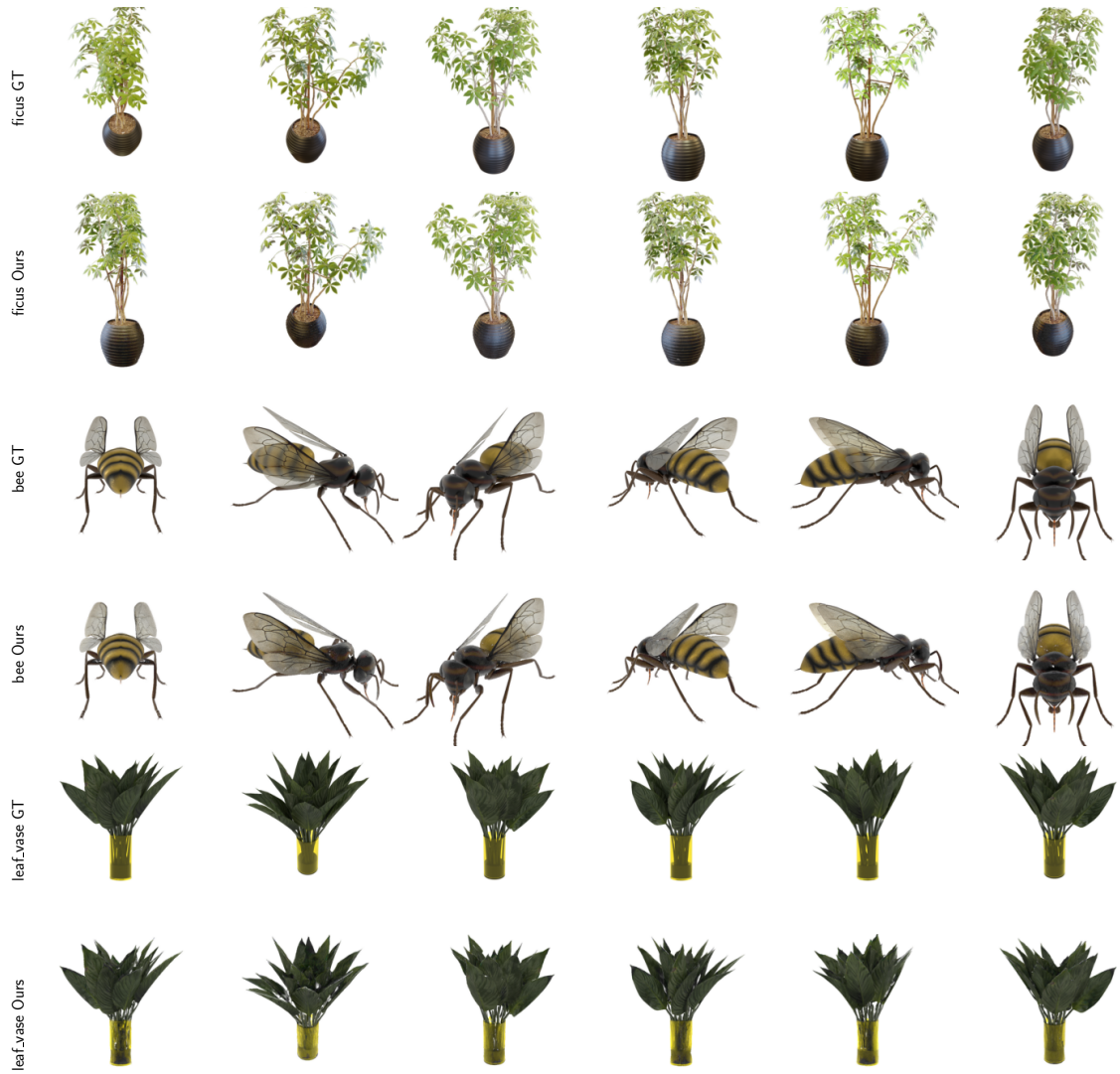


Figure 3.10: **RGB renderings of our methods on synthetic datasets.**

regularization can certainly give smoother surfaces, but can also destroy delicate thin structures in the reconstruction. In addition, we focus only on decoupling geometry and material ambiguity in existing volumetric representation, and do not specifically handle strong reflections as in Ref-NeRF [110]. Real-world reconstruction of highly specular translucent surfaces such as the "yellow cup" scene therefore still presents some artifacts. The quality can be further improved by incorporating reflection rendering similar to [110].

Chapter 4

Dynamic Noise Removal

Apart from intricate properties such as material opacity and high-frequency appearance that we want to model in our reconstruction, there also exist noises that we want to remove and exclude. One typical example is the dynamic effects or occluders. While most applications require a fully static 3D scene reconstruction, and therefore the majority of the methods only consider such scenarios, real-world multiview capture using a single camera cannot guarantee the data obtained is fully static without any time-dependent appearance or geometry changes. Applying naive 3D reconstruction methods on such data directly can cause deteriorated performance and significant artifacts. In this chapter, we look into approaches to mitigate this issue and allow static 3D reconstruction to be recovered from less restrictive and more casual multiview captures.

To do so, we adapt Neural Radiance Fields [70] (NeRF) and its extension HyperNeRF [78] to time-varying scenes by decoupling the dynamic and static components of the scene into separate volumetric fields; see Figure 1.2. Previous techniques that decouple dynamic and static scenes either rely on pre-trained object detection/segmentation modules [43, 59, 28, 52], or are limited to a single rigid object [134] or semi-static objects [108]. Our method instead learns dynamic and static components separately in a self-supervised fashion, using a novel skewed-entropy loss to encourage a clean separation of static and dynamic objects.

A crucial issue in creating a clean separation is *properly handling shadows*, as dynamic objects cast shadows that cause the radiance of the shadow receiver to vary with time. When the shadow receiver is part of the static component, this time-varying change in radiance cannot be directly modeled. We therefore relax the static component with a time-varying *shadow field* that modulates the radiance, allowing the shadows cast by moving objects to be captured while constraining density and color to be static.

Our method enables 3D scene decoupling and reconstruction from a monocular video captured from casual equipment such as a mobile phone, and can be readily extended to multi-view videos. By separately modeling the time-varying and time-independent

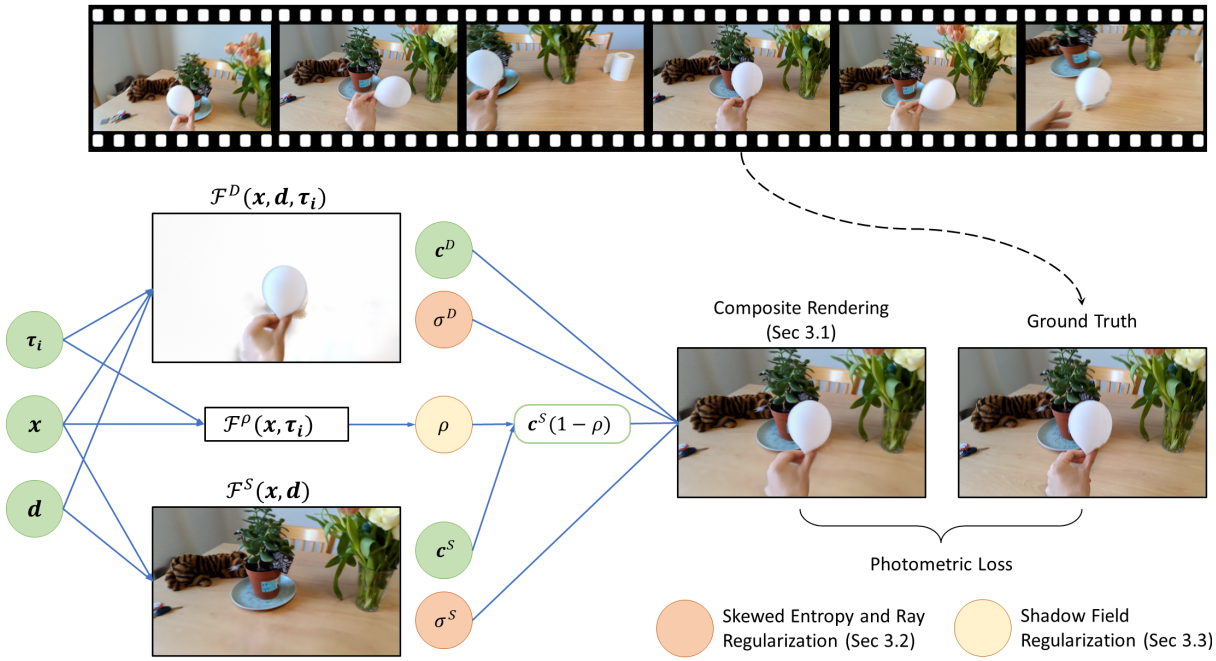


Figure 4.1: **Overview.** Given the ground truth view, camera pose and the time frame, our method reconstructs the underlying scene as a composite radiance field. Dynamic objects are represented by $\mathcal{F}^D$, while the static scene is represented by $\mathcal{F}^S$. The shadow-field $\mathcal{F}^\rho$ models non-static shadows within the input video.

targets in the video, our method can remove the dynamic occluders and their shadows, and synthesize a clean background from novel views.

We demonstrate the effectiveness of our method in *two aspects*: (i) the quality of novel view synthesis of the decoupled static background for monocular videos where the dynamic objects and shadows heavily occlude the scene, and (ii) the correctness of segmentation of dynamic objects and shadows on 2D images.

The work presented in this chapter produces the following publication:

- Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. 2024. D2NeRF: self-supervised decoupling of dynamic and static objects from a monocular video. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22) Curran Associates Inc., Red Hook, NY, USA, Article 2366, 32653-32666.

4.1 Method

Given a casual multiview capture with a single device, such as a *monocular* video captured from a freely *moving* hand-held camera, we aim to reconstruct a neural scene representation that decouples moving objects from the static environment, assuming a constant illumination and known camera poses (e.g. calibrated with COLMAP [96]). As illustrated

in Fig 4.1, our method achieves this by learning *separate* radiance fields for static and dynamic portions of the scene, and doing so in a fully self-supervised fashion. We describe our architecture (Sec 4.1.1), detail of our self-supervised losses (Sec 4.1.2), and describe how, while shadows are not explicitly modeled by NeRFs, a simple technique for their effective removal is attainable (Sec 4.1.3).

4.1.1 Composite Neural Radiance Field

The static component builds upon the traditional volumetric field and NeRF [70], which represents the scene as continuous spatial-dependent density σ and spatial-view-dependent radiance $\mathbf{c}$ using an multi-layer perceptron $\mathcal{F}^S$:

$$\left.\begin{aligned} \sigma^S(\mathbf{x}) &\in \mathbb{R} \\ \mathbf{c}^S(\mathbf{x}, \mathbf{d}) &\in \mathbb{R}^3 \end{aligned}\right\} = \mathcal{F}^S(\mathbf{x}, \mathbf{d}) \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the spatial coordinate, and $\mathbf{d} \in \mathbb{R}^3$, $\|\mathbf{d}\| = 1$ is the view direction. To model the dynamic component of a scene, we adapt HyperNeRF [78], which accurately captures scenes with *non-rigid* motion as well as *topological changes* by introducing additional degree of freedom and network capacity. For convenience, we denote it as a neural function $\mathcal{F}^D$:

$$\left.\begin{aligned} \sigma^D(\mathbf{x}, \boldsymbol{\tau}_i) &\in \mathbb{R} \\ \mathbf{c}^D(\mathbf{x}, \mathbf{d}, \boldsymbol{\tau}_i) &\in \mathbb{R}^3 \end{aligned}\right\} = \mathcal{F}^D(\mathbf{x}, \mathbf{d}, \boldsymbol{\tau}_i) \tag{4.2}$$

where $\boldsymbol{\tau}_i \in \mathbb{R}^m$ is the per-frame time latent code. Given a camera ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ originating from $\mathbf{o}$ and with direction $\mathbf{d}$, the two models are then composited to calculate the color $\hat{C}$ of the camera ray by integrating the radiance according to volumetric rendering within a pre-defined depth range $[t_n, t_f]$:

$$\hat{C}(\mathbf{r}, \boldsymbol{\tau}_i) = \int_{t_n}^{t_f} T(t) \left(\sigma^S(t) \cdot \mathbf{c}^S(t) + \sigma^D(t, \boldsymbol{\tau}_i) \cdot \mathbf{c}^D(t, \boldsymbol{\tau}_i)\right) \, dt \tag{4.3}$$

$$T(t) = exp\left(-\int_{t_n}^{t} (\sigma^S(s) + \sigma^D(s, \boldsymbol{\tau}_i)) \, ds\right) \tag{4.4}$$

where we simplify our notation as $\sigma(t) \equiv \sigma(\mathbf{r}(t))$ and $\mathbf{c}(t) \equiv \mathbf{c}(\mathbf{r}(t), \mathbf{d})$. Note that, with such an additive decomposition, samples from either fields are capable of terminating the camera ray and occluding the other.

4.1.2 Supervision Losses

To find the parameters of the static (Eq 4.1) and dynamic (Eq 4.2) NeRF networks, a photometric loss is applied to ensure that the output image sequences of the composite NeRF (Eq 4.3) align with the input video frames:

$$\mathcal{L}_p(\mathbf{r}, \boldsymbol{\tau}_i) = \|\hat{C}(\mathbf{r}, \boldsymbol{\tau}_i) - C(\mathbf{r}, \boldsymbol{\tau}_i)\|_2^2 \tag{4.5}$$

where $C(\mathbf{r}, \boldsymbol{\tau}_i)$ indicates the true color of camera ray $\mathbf{r}$ obtained from the i-th input video frame. However, note the dynamic component can naturally take over the static counterpart by incorrectly assigning occupancy of static objects to dynamic NeRF, and the photometric loss alone also does not guarantee a correct separation. In what follows, we design a collection of regularizers that promote such decoupling in a self-supervised fashion.

Dynamic vs. Static Factorization As physical objects cannot co-exist at the same spatial location, a physically realistic solution should have any position in space *either* occupied by a the static scene or by a dynamic object, but *not both*. To enforce this behavior we denote the spatial ratio of dynamic vs. static density as:

$$w(\mathbf{x}, \boldsymbol{\tau}_i) = \frac{\sigma^D(\mathbf{x}, \boldsymbol{\tau}_i)}{\sigma^D(\mathbf{x}, \boldsymbol{\tau}_i) + \sigma^S(\mathbf{x})} \in [0, 1] \tag{4.6}$$

and then penalize its deviation from a categorical $\{0, 1\}$ distribution via a binary entropy loss [134]:

$$\mathcal{L}_b(\mathbf{r}, \boldsymbol{\tau}_i) = \int_{t_n}^{t_f} H_b(w(\mathbf{r}(t), \boldsymbol{\tau}_i))\, dt \tag{4.7}$$

$$H_b(x) = -(x \cdot log(x) + (1 - x) \cdot log(1 - x)) \tag{4.8}$$

However, due to the strong expressive power of the dynamic networks (Eq 4.2), optimizing the loss (Eq 4.7) leads to the technique modeling parts of the scene as dynamic, regardless of whether they are dynamic or static; see Fig 4.2 (right). To overcome this issue, we propose a *skewed* entropy loss to bias our loss to *slightly favor* static explanations of the scene with skewness hyper-parameter k, that, as illustrated in Fig 4.2 (left, k>1), attains the desired behavior:

$$\mathcal{L}_s(\mathbf{r}, \boldsymbol{\tau}_i) = \int_{t_n}^{t_f} H_b(w(\mathbf{r}(t), \boldsymbol{\tau}_i)^k)\, dt \tag{4.9}$$

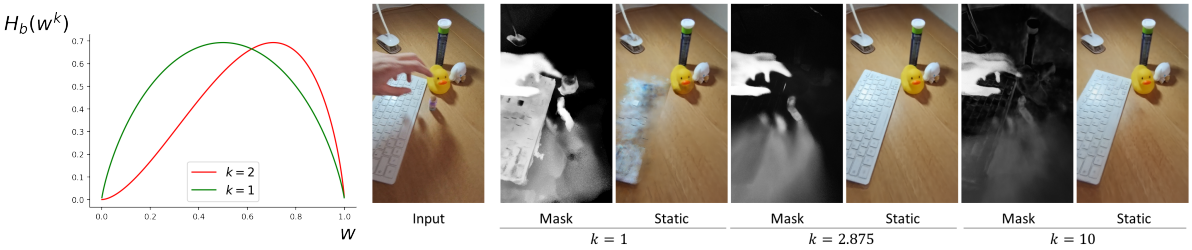


Figure 4.2: **Skewed entropy.** (left) The skewed ($k > 1$) and classical ($k = 1$) entropy losses. A skewed entropy encourages a wider range of w to decrease and has a larger gradient on values around 0.5, but its gradient vanishes when w approaches 0. (right) The decoupled alpha masks and static components when original, properly-skewed and over-skewed binary entropy losses are applied.

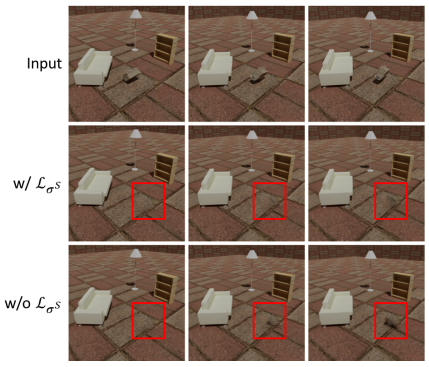


Figure 4.3: **Static regularization** – By encouraging a more concentrated density distribution along each camera ray in static component, the recovered background contains less view-dependent artifacts.

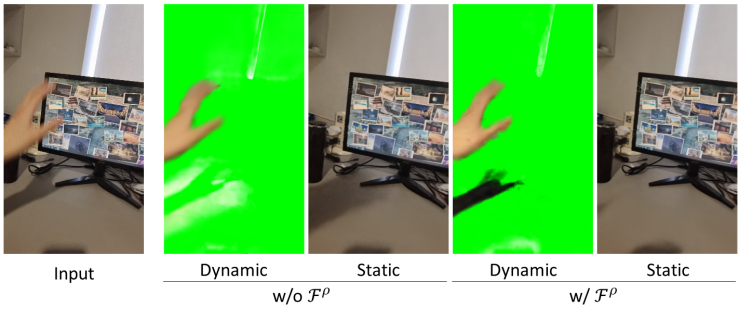


Figure 4.4: **Shadow ambiguity** – When shadows occur frequently in the input data, the average shadow gets integrated in the static component, and the dynamic component incorrectly learns the differential with respect to this average and appears as a brighter surface. This can be avoided by a more direct modeling of shadow effects as a dynamic darkening of static regions (i.e. the shadow field).

Ray Regularization Choosing a large value of skewness k causes the appearance of fuzzy floaters (low-density particles) in the static portion of the scene; see Fig 4.2 (right, k=10). As it can be intuitively understood from Fig 4.2 (left), this is caused by the small gradients of $H_b(x^k)$ as x approaches zero. To mitigate this effect, and reduce fuzziness in the reconstruction, we penalize the maximum of w along each camera ray:

$$\mathcal{L}_r(\mathbf{r}, \boldsymbol{\tau}_i) = \max_{t \in [t_n, t_f]} w(\mathbf{r}(t), \boldsymbol{\tau}_i) \tag{4.10}$$

Such loss can be intuitively interpreted as constraining the dynamic component to occupy as few pixels as possible while keeping minimal impact on the overall loss for all samples. Note that $\mathcal{L}_r$ only removes density floaters that sit along camera rays that *do not intersect* with any dynamic objects.

Static Regularization We empirically found that static component may abuse the camera pose as the hint for the current time frame and learn dynamic effects as sparse clouds that lead to high-frequency appearance changes; see Fig 4.3. The ambiguity comes from the fact that we are using monocular casual videos where the camera almost never visits the exact same position twice during the capture. That is, there exists a one-to-one mapping between camera pose and time variable. We solve this issue by imposing a prior on the distribution of density along a ray, penalizing density distributions that would cause cloud-like artifacts [49, 89]:

$$\mathcal{L}_{\sigma^S}(\mathbf{r}) = -\int_{t_n}^{t_f} p(t) \cdot log\, p(t)\, dt \quad \text{where} \quad p(t) = \frac{\sigma^S(\mathbf{r}(t))}{\int_{t_n}^{t_f} \sigma^S(\mathbf{r}(s))\, ds} \tag{4.11}$$

4.1.3 Shadow Fields

Neural radiance fields cannot faithfully model standalone shadows without significant changes to its architecture necessary to modeling materials and illumination; see NeRFactor [136]. In simple cases where shadows of the dynamic objects move rapidly, they could alternatively be learned by the dynamic radiance field as semi-transparent layers on top of the static surface. However, this tends to fail for shadows that do not move much, or that are highly correlated with the camera motion. As shadows are texture-less, understanding their movement is ambiguous, and representing them as a semi-transparent layer causes difficulties in the optimization; see Fig 4.4. To overcome this issue, and under the assumption of a direct illumination model (i.e. negligible global illumination effects), we make the observation that a cast shadow can be represented as a *pointwise reduction* in the radiance of the the static scene, and incorporate this within Eq 4.3 as:

$$\hat{C}(\mathbf{r}, \boldsymbol{\tau}_i) = \int_{t_n}^{t_f} T(t)((1 - \underbrace{\rho(\mathbf{r}(t), \boldsymbol{\tau}_i)}) \cdot \sigma^S(t) \cdot \mathbf{c}^S(t) + \sigma^D(t, \boldsymbol{\tau}_i) \cdot \mathbf{c}^D(t, \boldsymbol{\tau}_i))\, dt \tag{4.12}$$

$$\rho(\mathbf{x}, \boldsymbol{\tau}_i) \in [0, 1] = \mathcal{F}^\rho(\mathbf{x}, \boldsymbol{\tau}_i) \tag{4.13}$$

where $\rho(\mathbf{x}, \boldsymbol{\tau}_i)$ is a *shadow ratio* that scales-down the radiance of the static scene to incorporate the shadow. To avoid the shadow-ratio from over-explaining dark regions of the scene, we penalize its average squared magnitude along a ray:

$$\mathcal{L}_\rho(\mathbf{r}, \boldsymbol{\tau}_i) = \tfrac{1}{t_f - t_n} \int_{t_n}^{t_f} \rho(\mathbf{r}(t), \boldsymbol{\tau}_i)^2\, dt \tag{4.14}$$

Finally, note that shadows cast from dynamic objects onto other dynamic objects are *already* expressed from the radiance term of the dynamic branch, and do not need explicit modeling.

4.2 Experiment

4.2.1 Implementation details

We adopt the HyperNeRF [78] architecture as the dynamic component, which has a NeRF MLP network of 8 layers, each with 256 channels, and our static NeRF component has the same architecture. Similar to NeRF [70], we apply a hierarchical volume sampling with 64 coarse and 64 fine samples. The optimization takes $100k$ iterations with batch size 1024 and an exponentially decayed learning rate from 10^{-3} to 10^{-5}. This training procedure spans approximately two hours on four NVIDIA A100-SXM-80GB GPUs. The overall loss of our method is:

$$\mathcal{L}(\mathbf{r}, \boldsymbol{\tau}_i) = \mathcal{L}_p(\mathbf{r}, \boldsymbol{\tau}_i) + \lambda_s \mathcal{L}_s(\mathbf{r}, \boldsymbol{\tau}_i) + \lambda_r \mathcal{L}_r(\mathbf{r}, \boldsymbol{\tau}_i) + \lambda_{\sigma^S} \mathcal{L}_{\sigma^S}(\mathbf{r}) + \lambda_\rho \mathcal{L}_\rho(\mathbf{r}, \boldsymbol{\tau}_i) \tag{4.15}$$

where λ_s, λ_r, λ_{σ^S}, λ_ρ are the weights of the regularization terms respectively. For scenes with a mixture of dynamic objects and shadows, we apply shadow decay and set λ_ρ=0.1. We set λ_ρ=0.001 for scenes featuring view-correlated dynamic shadows only. We experimentally found that the optimal choice of the hyperparameters, especially λ_b, λ_r and the skewness k, are strongly influenced by the level of object motion, camera motion, and video length. Therefore, we performed a grid search on our synthetic and held-out real-world scenes, and some scenes from DAVIS [82], to establish a set of hyperparameters applicable to a variety of scenarios. We do not apply shadow field for evaluations on our synthetic scenes, as we empirically found that shadow field is not needed to learn correct shadows. We also disable the view direction input for synthetic scenes as they do not contain strong view-dependent effects. The training takes roughly 2 hours on 4x NVIDIA A100-SXM-80GB GPUs.

4.2.2 Evaluation

We demonstrate the performance of our method both quantitatively and qualitatively on three different tasks. Apart from the main objective of removing dynamic objects and recovering high-quality static 3D reconstructions, our method can also be evaluated in dynamic object segmentation correctness and dynamic scene novel view synthesis. We strongly encourage the readers to watch videos on the project page (https://d2nerf.github.io/) to better appreciate the results.

4.2.3 Datasets

In addition to the data obtained from HyperNeRF [78] and Nerfies [77], we acquire more complex datasets in the real-world, as well as design a synthetic dataset to enable quantitative comparisons.

	Car			Cars			Bag			Chairs			Pillow			Mean		
	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑	LPIPS↓	MS-SSIM↑	PSNR↑
NeRF-W [69]	.218	.814	24.23	.243	.873	24.51	.139	.791	20.65	.150	.681	23.77	.088	.935	28.24	.167	.819	24.28
NSFF [59]	.200	.806	24.90	.620	.376	10.29	.108	.892	25.62	.682	.284	12.82	.782	.343	4.55	.478	.540	15.64
NeuralDiff [108]	.065	.952	31.89	.098	.921	25.93	.117	.910	29.02	.112	**.722**	24.42	.565	.652	20.09	.191	.831	26.27
Ours	**.062**	**.975**	**34.27**	**.090**	**.953**	**26.27**	**.076**	**.979**	**34.14**	**.095**	.707	**24.63**	**.076**	**.979**	**36.58**	**.080**	**.919**	**31.18**

Table 4.1: **Scene decoupling (quantitative)** – We train on each scene with 200 frames, decouple the dynamic objects and shadows, and render the static component from 100 novel views to compare with ground truth. Note these are computed on the *synthetic* dataset, for which ground truth is available.

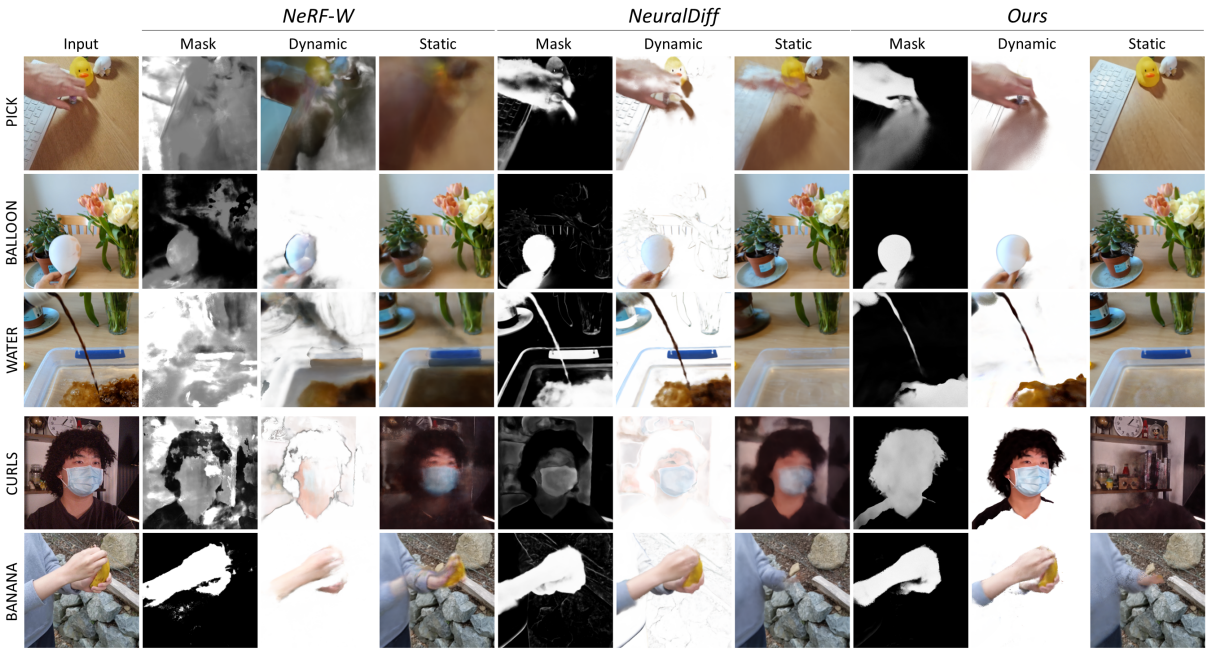


Figure 4.5: **Scene decoupling (qualitative)** – We visualize results on (top) our new real scenes, and on (bottom) scenes from HyperNeRF [78] and Nerfies [77]. To better illustrate the decoupled object and shadow, we render the dynamic component with a white background. Note that since our method only decouples dynamic targets, it does not include parts of objects that remain still throughout the capture, such as the body in "curls" and "banana" scenes.

Synthetic dataset We generate a synthetic dataset with ground-truth masks for moving objects and their shadows with Kubric [33]. This dataset consists of five scenes containing one or multiple dynamic objects from ShapeNet [10] with rigid or non-rigid motion, and the corresponding Kubric worker script is provided in our supplementary material. We move the virtual camera over 10 keyframes randomly sampled from azimuth $[2, 2 + \pi/4]$ and altitude $[1, 1.2]$ to generate a 200-frame video sequence for training. We also rotate the virtual camera around the center of all keyframes to generate 100 validation views with only the static background being visible. We additionally generate masks for both the dynamic objects and their shadows, allowing us to quantitatively study the performance of our algorithm. Note that shadows are usually absent in existing moving-objects segmentation benchmarks.

Figure 4.6: **Additional results on real-world scene decoupling and segmentation.** Similar to Fig 4.5, we show the dynamic alpha mask, dynamic part and static background respectively. Last two rows at bottom show the "broom" scene from HyperNeRF [78].

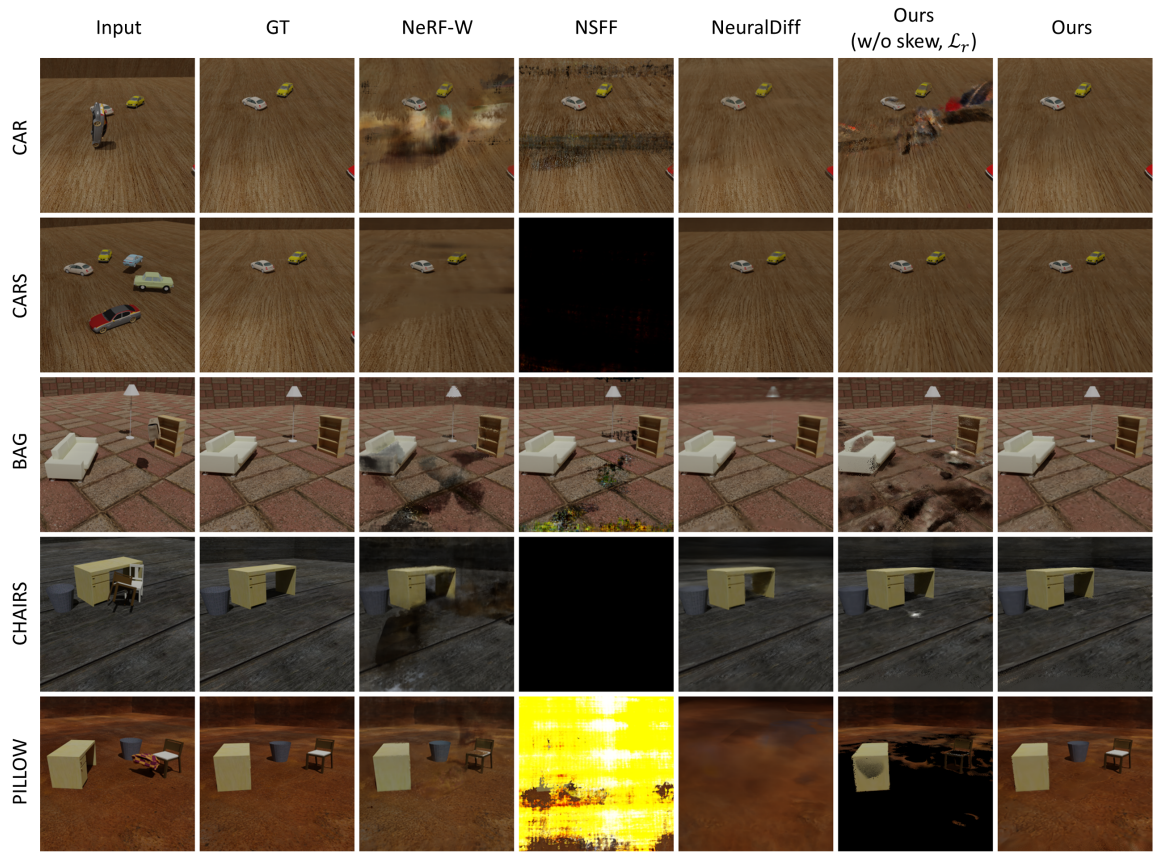


Figure 4.7: **Scene decoupling and novel view background recovery on synthetic scenes**. We train each method with videos containing various dynamic occluders and shadows, decouple the scene and render the background from unseen views to compare with the ground truth. Quantitative evaluation on corresponding scenes can be found at Table 4.1.

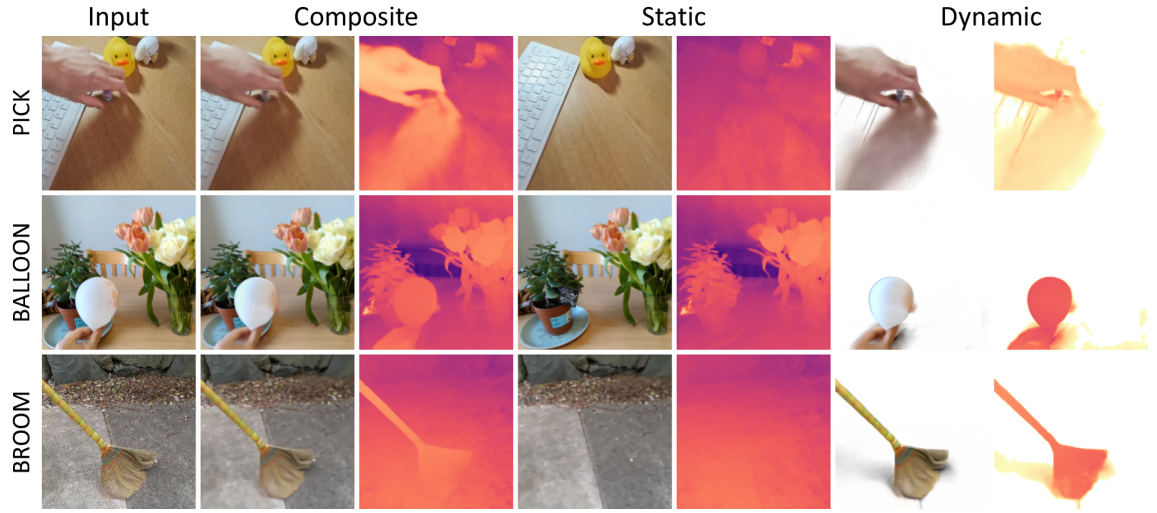


Figure 4.8: **Decoupled depth.** We show the disentangled geometry as depth maps for dynamic and static components respectively.

Figure 4.9: **Background novel view.** Our method learns the decoupled static background and can render it from unseen views, with the dynamic occluders cleanly removed.

Real-world dataset We also capture ten video sequences of real scenes to showcase our performance. Compared to HyperNeRF's, our dataset contains more challenging scenarios with rapid motion and non-trivial dynamic shadows. Note however that we *cannot* perform quantitative analysis for these datasets due to the absence of ground-truth views of a static scene or ground-truth masks. Five of the real scenes are captured with a similar setting as Nerfies, where we use a dual-hold phone rig and synchronize the capture based on audio. We use the images captured by one of the two phones as validation views for novel-view synthesis. To demonstrate the ability of fully self-supervised scene decoupling, we *do not* apply any masks when registering real-world images using COLMAP [96].

4.2.4 Scene Decoupling

We report the evaluation of our method on its ability to decouple dynamic objects and their shadows, while recovering the static background. We evaluate our performance against NeRF-W [69], NSFF [59] and NeuralDiff [108]. For NeRF-W, we used only transient embedding and disabled the appearance embedding that models variable lighting for evaluation on synthetic scenes, as they have constant illumination. For NSFF, we disabled the hard mining initialization via 2D masks input. For NeuralDiff, we disabled the actor component (as our input videos are not egocentric) and only use the transient component.

In the evaluation, we used each method to synthesize the static background from multiple validation views with the moving objects and their shadows removed; see Fig 4.5 and Fig 4.6 for qualitative results on real data, as well as Fig 4.7 for qualitative results on synthetic data. We compare the results with the ground truth on the synthetic data and report LPIPS [135], Multi-Scale SSIM [117], and PSNR as the metrics for novel view synthesis of the decoupled static background; see Table 4.1. Note that although NSFF similarly learns separate dynamic and static NeRFs, it is not designed to maximize the performance of scene decoupling and does not incorporate regularization on the dynamic NeRF. Therefore, it is extremely unstable and can fail completely by learning everything as dynamic, leaving static component as an empty scene.

In Fig 4.8, we show that the depth rendering of the scene is correctly decoupled based on the time-dependent geometry in the scene. In Fig 4.9, we show the novel view rendering of the static background with the dynamic occluders and shadows cleanly removed.

	Car		Cars		Bag		Chairs		Pillow		Mean	
	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$
MG [127]	.603	.743	.363	.474	.629	.738	.484	.613	.044	.080	.424	.529
NeRF-W [69]	.072	.132	.098	.162	.027	.052	.154	.254	.194	.314	.109	.183
NSFF [59]	.083	.152	.058	.104	.102	.182	.046	.087	.104	.188	.079	.143
NeuralDiff [108]	.806	.891	.508	.578	.080	.144	.368	.513	.097	.177	.372	.461
Ours (w/o skew)	.814	.896	**.807**	**.883**	.342	.483	.114	.198	.347	.511	.485	.594
Ours (w/o L_r)	.076	.139	.174	.261	.048	.089	.237	.367	.040	.078	.115	.187
Ours (w/o skew, L_r)	.077	.142	.376	.464	.043	.081	.315	.453	.027	.053	.168	.238
Ours	**.848**	**.917**	.790	.874	**.703**	**.818**	**.551**	**.687**	**.693**	**.818**	**.717**	**.822**

Table 4.2: **Video segmentation** – We report Jaccard index $\mathcal{J}$ and boundary measure $\mathcal{F}$ on training views. Our method performs well on "car" and "cars" scenes without the use of skewed entropy because the background is clearly distinguishable from the moving object.

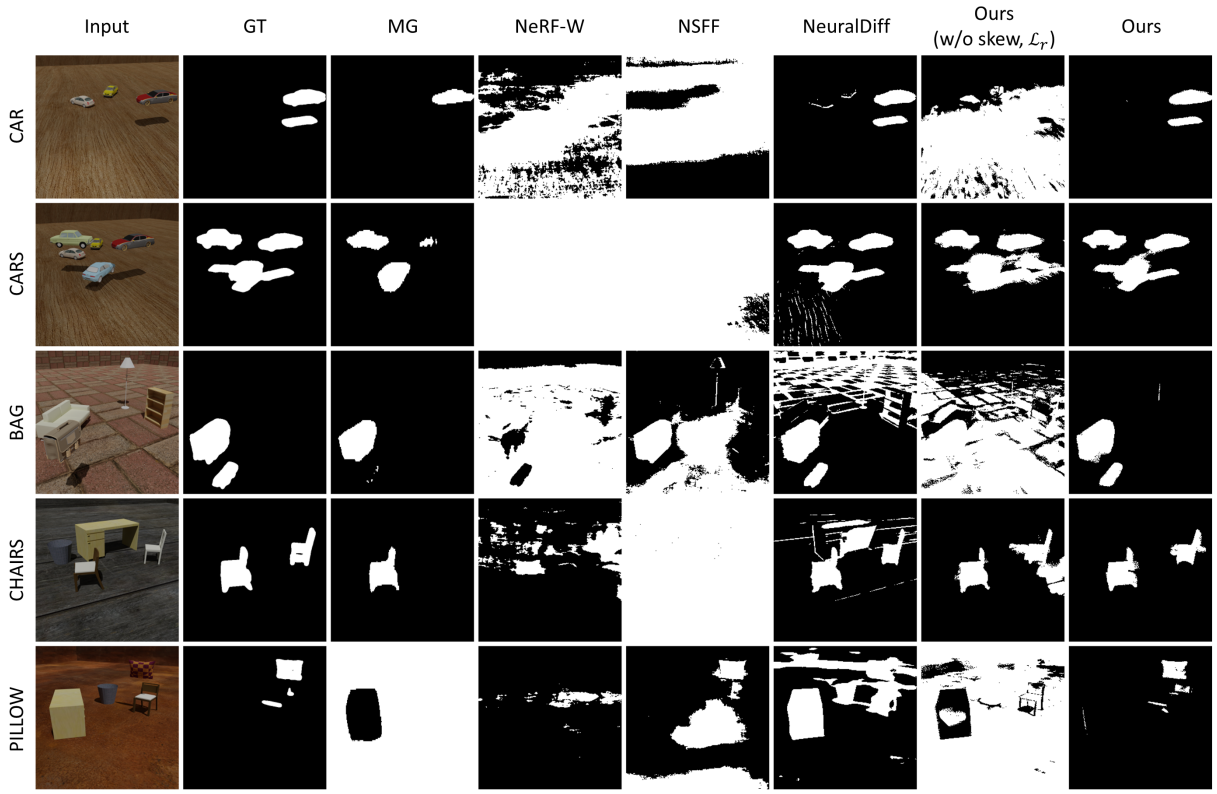


Figure 4.10: **Video segmentation (qualitative).** MG fails to identify the moving pillow and segments everything out except for the table due to its Hungarian setting. NeRF-W learns the transient component with severe cloud-like effects. While our method achieves best segmentation in all the scenes.

4.2.5 Video Segmentation

As our method learns a density distribution of the dynamic objects in the scene, we can further produce an alpha mask of the objects. We therefore also evaluate the correctness of object segmentation at the image level. Existing benchmarks on video segmentation [82,

115, 54] either contain too few video frames for reliable SfM reconstruction, or do not have correct ground truth masks for all of the dynamic objects and effects in the scene. Similarly, the dataset from NeuralDiff [108] focuses on egocentric videos and the over-exaggerated difference between frames in the videos is not suitable for HyperNeRF which we use as the dynamic component. Hence, to highlight the ability of our method to decouple dynamic objects and shadows from video sequences with large viewpoint shifts, we evaluate on our synthetic dataset. In addition to the NeRF-like baselines, we also compare with Motion Grouping (MG) [127], a motion-based 2D image segmentation method. We fine-tuned the pre-trained MG model on each scene for 5k iterations. For other NeRF-based methods, we used the same settings as in Section 4.2.4 and produced the alpha masks as the normalized radiance weights of the time-varying component, and then applied a threshold of 0.1 to obtain the binary masks. See Table. 4.2, Fig 4.11 for the results.

4.2.6 Novel View Synthesis

Although the aim of our method is not to improve the quality of time-varying scene reconstruction, as a by-product, we find that by introducing a static component to fully utilize the network capacity, our method achieves more robust reconstruction for both the dynamic objects and background. We therefore also evaluate our method on the ability to synthesize the whole scene from novel views. We compare several approaches for dynamic scene reconstruction, including NeuralDiff [108] and a baseline version of HyperNeRF [78], which has the same architecture as our dynamic component. Since our method uses an additional static component and naturally has more network parameters and capacity, we also compare with a fair version of HyperNeRF with roughly the same number of total parameters by extending the NeRF MLP width to 375. Unlike novel view experiments in [78], we do not interleave between two cameras, but use only the right camera as training view and the left camera as validation view. This presents greater challenges to all the methods. See Table. 4.4, Fig 4.12 for the results.

4.2.7 Ablations.

We quantitatively ablate our method on our synthetic dataset; see Table 4.5: where "skew" means skewness is applied in the binary entropy regularization, and "$\mathcal{L}_r$" stands for the density ratio ray regularization. We also qualitatively ablate it on a real scene; see Fig 4.13. We also qualitatively illustrate the ablation on shadow field network, which is necessary for decoupling shadows with large area, slow or repetitive motion, or shadows that are highly correlated with the camera view; see Fig 4.14.

	Pick2			Duck			Balloon			Water			Cookie		
	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑
NeuralDiff	**.208**	**.853**	**21.81**	.222	**.862**	21.92	.167	.836	20.13	.172	.811	18.36	.159	.875	**20.53**
HN (base)	.496	.413	13.06	.251	.830	20.64	.195	.803	17.81	.360	.483	15.06	.161	.836	19.93
HN (fair)	.486	.409	13.14	.253	.818	20.32	.187	.804	17.92	.361	.465	14.80	.162	.801	19.75
Ours	.253	.825	20.32	**.214**	.856	**22.07**	**.153**	**.858**	**20.92**	**.153**	**.849**	**21.63**	**.156**	**.877**	19.93

	Broom			Chicken			Banana			Mean		
	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑
NeuralDiff	.631	.468	17.75	.249	.822	21.17	.303	.748	19.43	.264	.784	20.14
HN (base)	.524	.636	19.65	.222	.878	23.94	.223	.818	21.20	.304	.712	18.91
HN (fair)	**.503**	.624	19.38	**.180**	.881	23.68	**.194**	**.832**	**21.52**	.291	.704	18.82
Ours	.565	**.712**	**20.66**	.204	**.890**	**24.27**	.260	.820	21.35	**.245**	**.836**	**21.39**

Table 4.4: **Novel view synthesis (quantitative).** We compare with NeuralDiff [108], a baseline version of HyperNeRF [78], denoted HN (base), and a fair version with extended network width to match the total number of parameters in our method, denoted HN (fair). Three scenes displayed in the bottom row are from HyperNeRF[78]

4.3 Summary

In this chapter, we present D^{2}NeRF, a method for self-supervised 3D scene decoupling and reconstruction from casual monocular videos. Our method decouples occluders and correlated shadows, recovers clean background representations, and enables high quality novel views synthesis. Our novel skewed entropy regularizer is critical to separate dynamic from static components of the scene, while our shadow-field allows for the removal of dynamic shadows without having to explicitly model the interaction between light and geometry. We demonstrate superior results for multiple tasks on existing datasets, as well as on two datasets that we introduce alongside our technique.

Limitations Similar to many NeRF-based methods, our approach relies on accurate camera registration to achieve success decoupling and reconstruction. Our approach also suffers from high frequency view-dependent radiance change, such as those caused by the presence of reflective surface within the scene. Due to the monocular moving camera setting, those effects could be misinterpreted as dynamic effects, resulting in incorrect decoupling. Removing texture-less target that repeatedly moves within a very small range is also difficult, as the motion clues are extremely ambiguous in this case; see Fig 4.15. We also note that the skewed binary entropy is not the only loss that suits the purpose of separating two NeRF components while suppressing one of them. Many losses such as the beta distribution from [64] and Laplacian distribution from [89] can behave similarly by incorporating an exponential coefficient. The main difference between our skewed entropy loss and beta or Laplacian distributions is that the former has a vanishing gradient when approaching 0, while the gradients of the latter become infinity. Choosing an appropriate form of loss could reduce the complexity of hyperparameters without affecting the decoupling accuracy.

Ambiguity between Dynamic Component and Shadow The shadow field network represents the density-less shadows in a more physically realistic way, and resolves the ambiguity in their motion. However, the aim of our method is to achieve decoupling of dynamic occluders from the static environment, and we do not over-extend to consider further decoupling between objects and shadows, which would require more priors related to environmental lighting conditions and background texture.

We empirically found that shadow field is not necessary for scenes with strong and fast-moving shadows, where they can be directly learned by the dynamic component as thin layers on top of the static geometry; see Fig 4.16. On the other hand, there exists unsolvable ambiguity between the shadow and dynamic object, especially for which with a similar or darker color to the background, and hence could be potentially explained as a moving shadow instead of an actual 3D shape due to our monocular camera setting; see Fig 4.17. As the later case causes failed dynamic geometry reconstruction, leading to a severe decrease in novel view synthesis performance, we deliberately choose a large value of λ_ρ to suppress the shadow field for scenes containing a mixture of dynamic objects and shadows. Although this potentially favors the former case and causes more shadows to be interpreted as thin-layers, we found that such setting has minimal impact on performance of both scene decoupling and reconstruction, and is still sufficient to achieve correct shadow decoupling, as the shadow field resolves the ambiguity in shadow in very early stage of the training.

	Car		Cars		Bag		Chairs		Pillow		Mean	
	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$
MG [127]	.603	.743	.363	.474	.629	.738	.484	.613	.044	.080	.424	.529
NeRF-W [69]	.072	.132	.098	.162	.027	.052	.154	.254	.194	.314	.109	.183
NSFF [59]	.083	.152	.058	.104	.102	.182	.046	.087	.104	.188	.079	.143
NeuralDiff [108]	.806	.891	.508	.578	.080	.144	.368	.513	.097	.177	.372	.461
Ours (w/o skew)	.814	.896	**.807**	**.883**	.342	.483	.114	.198	.347	.511	.485	.594
Ours (w/o L_r)	.076	.139	.174	.261	.048	.089	.237	.367	.040	.078	.115	.187
Ours (w/o skew, L_r)	.077	.142	.376	.464	.043	.081	.315	.453	.027	.053	.168	.238
Ours	**.848**	**.917**	.790	.874	**.703**	**.818**	**.551**	**.687**	**.693**	**.818**	**.717**	**.822**

Table 4.3: **Video segmentation.** We report Jaccard index $\mathcal{J}$ and boundary measure $\mathcal{F}$ on training views. Our method performs well on "car" and "cars" scenes without the use of skewed entropy because the background is clearly distinguishable from the moving object.

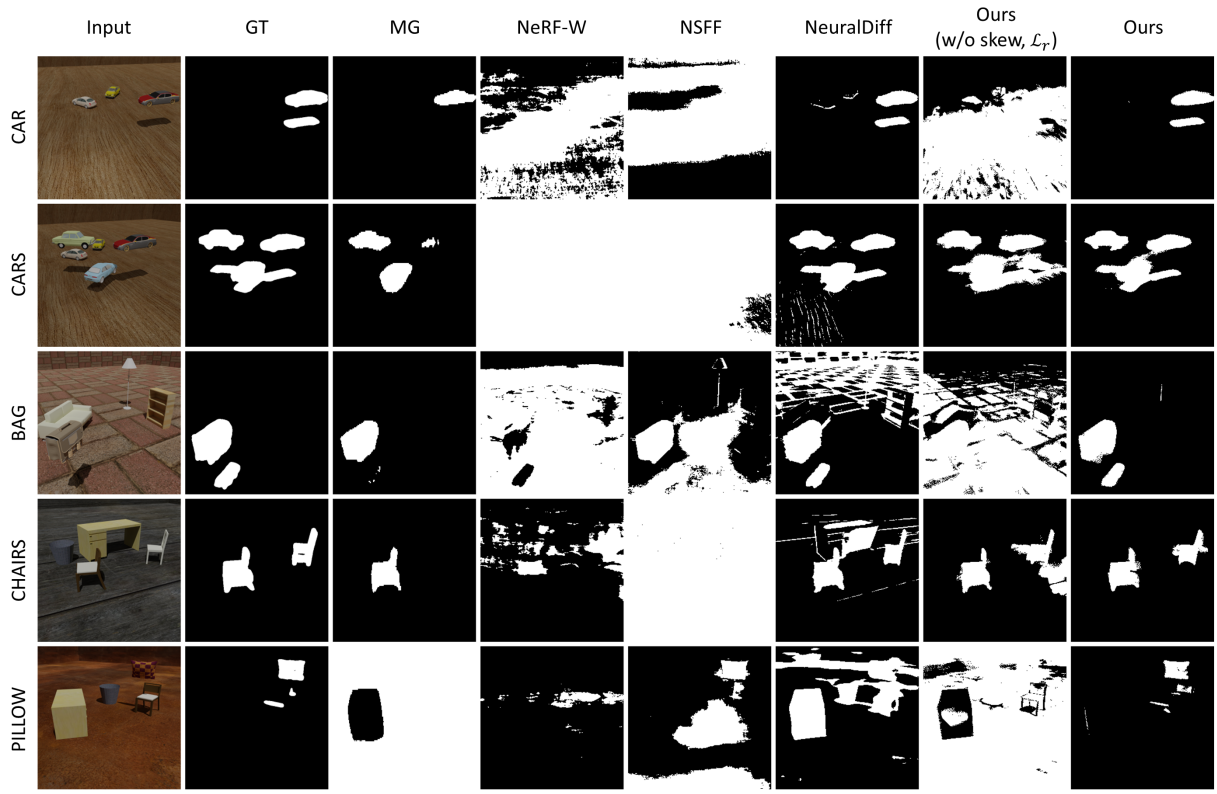


Figure 4.11: **Video segmentation (qualitative).** MG fails to identify the moving pillow and segments everything out except for the table due to its Hungarian setting. NeRF-W learns the transient component with severe cloud-like effects. Our method achieves the best segmentation in all the scenes.

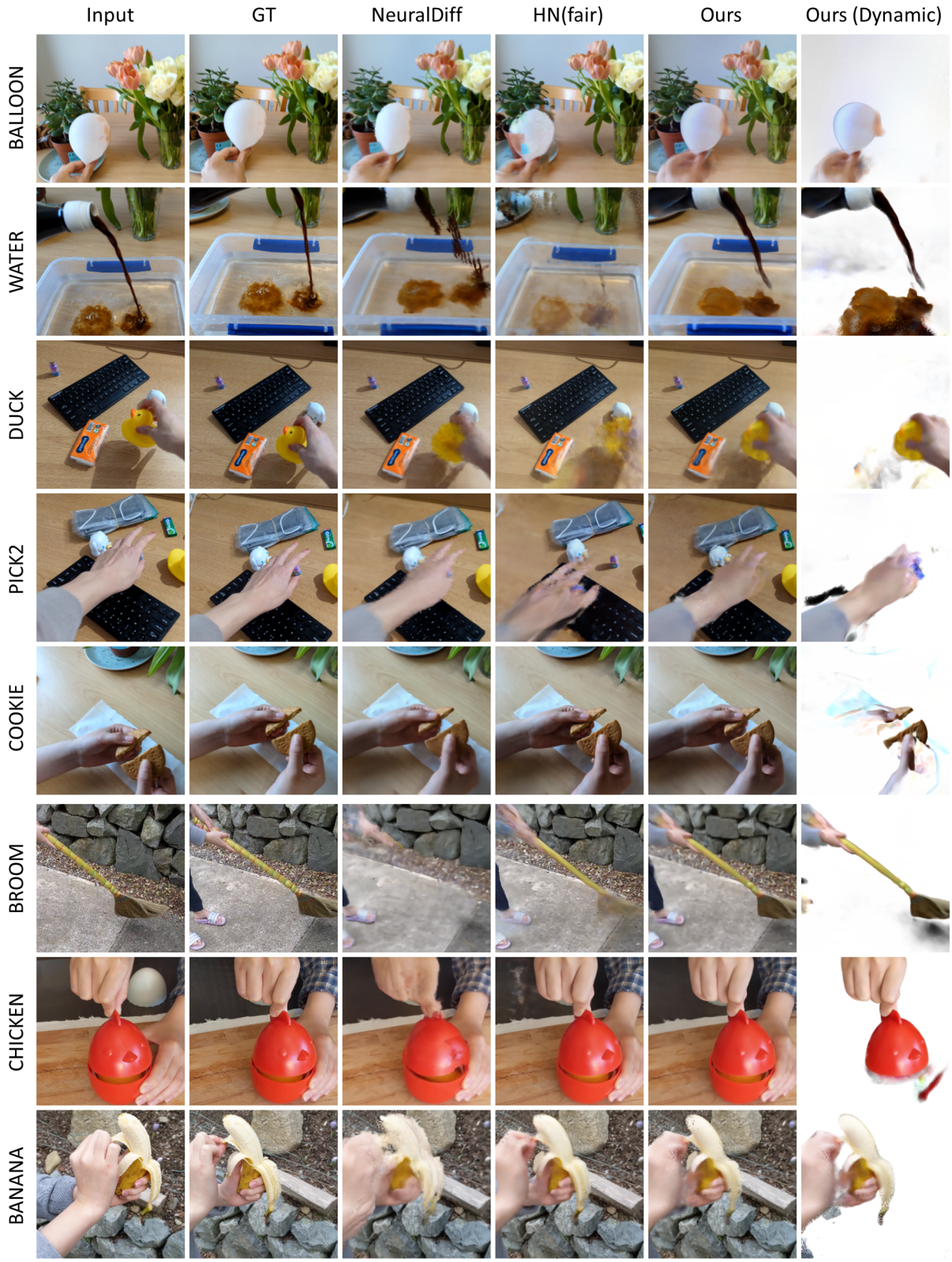


Figure 4.12: **Novel view synthesis (qualitative).** For challenging scenes such as "water" and "duck" where the dynamic object moves rapidly or training/validation views differ largely, HyperNeRF [78] fails to reconstruct a reasonable shape for the dynamic object, while ours might potentially predict a shifted object pose, but can still render the view with high fidelity. We additionally show the decoupled dynamic object from our method. The quality is slightly degraded compared to the decoupling results in Fig 4.5 and 4.6 as we are rendering from the more challenging novel views.

		Car			Cars			Bag			Chairs			Pillow			Mean		
skew	$\mathcal{L}_r$	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑	LPIPS↓	MS-SSIM↑	PNSR↑
☐	☐	.214	.834	26.26	.119	.943	26.10	.254	.666	19.96	.104	.698	24.42	.385	.671	14.24	.215	.762	22.20
☑	☐	.182	.865	25.89	.260	.803	22.47	.189	.893	28.38	.107	.693	24.44	.311	.770	15.27	.210	.805	23.29
☐	☑	.067	.973	34.06	.104	.948	26.19	.091	.955	31.55	.151	.653	22.92	.118	.940	28.17	.106	.894	28.58
☑	☑	**.062**	**.975**	**34.27**	**.090**	**.953**	**26.27**	**.076**	**.979**	**34.14**	**.095**	.707	**24.63**	**.076**	**.979**	**36.58**	**.080**	**.919**	**31.18**

Table 4.5: **Ablations (quantitative)** – We train on each scene with 200 frames, decouple the dynamic objects and shadows, and render the static component from 100 novel views for metric evaluations.

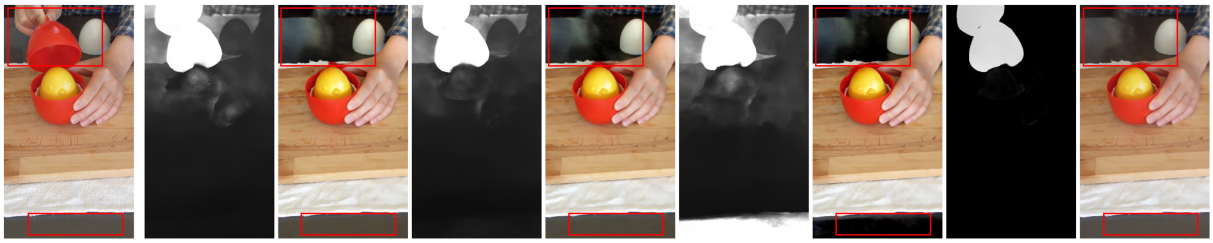


Figure 4.13: **Ablations (qualitative)** – For scenes with slow motion or strong view-dependent reflectance, $\mathcal{L}_r$ is used together with the skewed entropy to prevent the dynamic component from incorrectly decoupling the scene. In the scene above this appears as a slightly darkened color on the table.

Figure 4.14: **Ablations (shadows)** – Our method is able to remove large area of shadows, even if they are strongly correlated with the view direction (e.g., shadow cast by camera or the photographer). Note that the appearance embedding from NeRF-W [69] is not sufficient to remove shadow that is present throughout the capture; see the our website for additional qualitative results.

Figure 4.15: **Limitations.** Because the hand stays around the same position for the majority of the time during the video, our method is unable to fully decouple and remove the texture-less shadow.

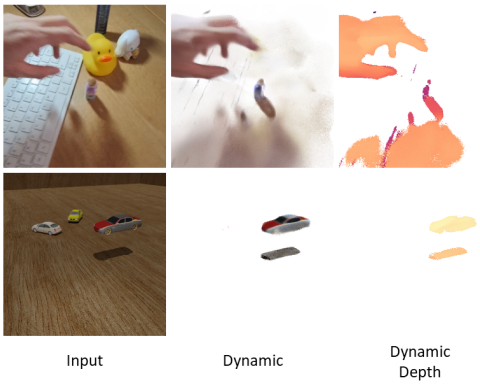


Figure 4.16: **Shadow as thin layer** – Although such representation could potentially represent more than just shadows, it still tends to exclude unnecessary texture from static background and learns only the darkening effects.

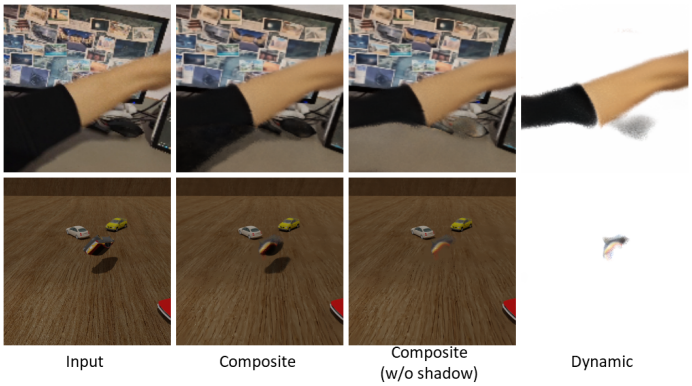


Figure 4.17: **Incorrect shadow.** The shadow field is incorrectly used to explain the black sleeve as well as the gray top of the moving car.

Chapter 5

Efficient High-Frequency Texture Modeling

Apart from the quality and robustness of the reconstructions, the efficiency in optimization and rendering is also a crucial factor in applications. As discussed in Sec 2.4, the most effective approach for improving efficiency is to adapt more explicit architecture such as hash grid, explicit grid, or Gaussians. Those architectures allow the scene information to be localized, hence both updating and rendering would only involve local area, effectively reducing the number of parameters affected. While grid-based architecture achieves significant speed-up and enables real-time rendering on high-end devices, their capacity is heavily constrained and can only be reduced through pruning rather than increased during optimization. Therefore, the explicit Gaussians has been proposed as a more adaptive and effective architecture for achieving versatile capacity control and real-time rendering even on mobile devices. However, one of the fundamental limitations in Gaussian architecture is that each Gaussian can represent only one color from a specific viewing angle. The lack of spatial-dependent appearance modeling heavily limits its capability to handle objects with relatively simple geometry but complex textures, such as high-frequency texture on cloth. To capture the detailed appearance of such objects, an excessively large number of Gaussians would be needed. This significantly increases memory requirement and slows down the rendering speed.

We argue for objects with relatively simple geometry, modeling them with implicit volumetric representation and the Gaussians architecture could be an over-complication of the problem. Instead, a more standard approach would be representing their appearance with a high-frequency texture. In a traditional texture-based rendering pipeline, the texture is first mapped to mesh geometry in the 3D world space via UV mapping, and then rasterized to the 2D image plane in the view space to obtain the pixel color. However, this approach requires a well-defined UV mapping and accurate mesh geometry. Such geometry can be easily obtained in some cases, such as a general scene or object reconstruction

from multiview images covering a wide range of viewpoints. However, for forward-facing scene reconstruction or neural avatar reconstruction from monocular video, it is extremely challenging to obtain reliable geometry for texture mapping due to the lack of multi-view correspondences.

Therefore, to better understand and evaluate the efficient and robust way to represent high-frequency appearance in image-based 3D reconstruction task, we choose a specific task of reconstructing animatable forward-facing neural avatars from monocular videos. This is a suitable example task for studying the general problem because the chest and shoulders of a human avatar are expected to have relatively simpler geometry compared to the head and face, and the clothing tends to contain high-frequency details. Besides, personalized and controllable 3D head avatar is a crucial asset for interactive Mixed Reality and metaverse applications, and modeling them efficiently and accurately is an important task in the field of 3D reconstruction. To deal with the high-frequency limitations, We propose to bypass the mapping from texture space to world space, and instead use a sparse set of Gaussians as "anchors" to define a direct neural warping field from a canonical 2D texture space, which consists of a coarse RGB texture and a fine neural texture, to the image plane. As the tracking of body 3DMM tends to be inaccurate due to the lack of landmarks, we only transform anchor Gaussians together with the head Gaussians via a head FLAME 3DMM [56] through Linear Blend Skinning (LBS). The transformed anchor Gaussians are used as soft constraints of the texture warping represented by a coordinate-based MLP, which is optimized together with the neural texture, regular Gaussians, and the anchor Gaussians. As the resolution of the neural texture is not limited by the number of Gaussians or the density control scheme, we can easily learn the high-frequency textures with sharp details on the clothes and avoid the common artifacts exhibited in Gaussian rendering under novel poses.

To maintain a competitive rendering speed with Gaussian architecture and enable real-time interactive applications, we additionally propose a method to remove the neural warping field and neural texture in the model and allow inference of reconstructed avatars at novel poses without any MLP queries. This accelerated inference effectively increases the rendering speed from 70 FPS to around 130 FPS, which surpasses the rendering speed of plain Gaussian Splatting avatars for subjects with high-frequency clothes.

We evaluate the proposed method with various casual monocular videos collected using smartphones or from the Internet. Compared to state-of-the-art methods which incorporate different representations and architectures, we show that our approach achieves better performance and robustness for both self-reenactment and cross-reenactment tasks. In summary:

- We propose a novel approach that maps intricate texture to the image plane via a sparse set of anchor Gaussians driven by LBS with 3DMM. This allows accurate

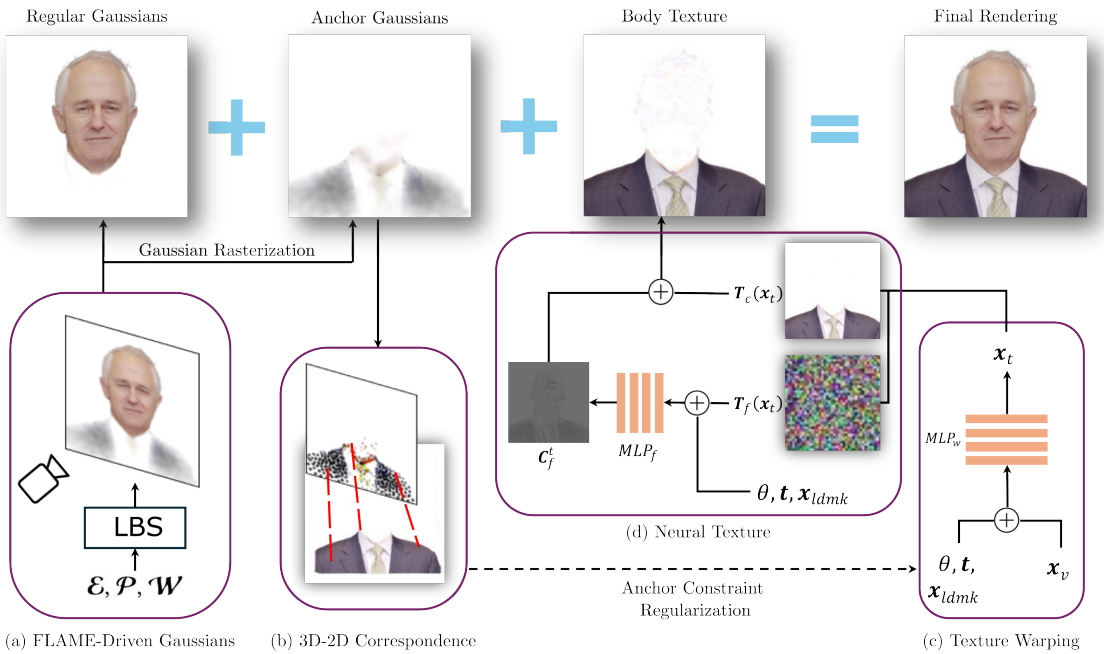


Figure 5.1: **Method.** (a) We utilize a set of standard head Gaussians and anchor Gaussians driven by LBS with the FLAME model. (b) Anchor Gaussians are initialized with a set of corresponding target coordinates in the texture space. This 3D-2D correspondence is used to constrain (c) a neural texture warping field that maps each pixel on the image plane $\mathbf{x}_v$ to a pixel in the texture space $\mathbf{x}_t$. (d) We then sample in the texture space to fetch the coarse texture $\mathbf{T}_c$ and latent texture $\mathbf{T}_f$, which is parsed by an MLP to obtain pose-dependent fine texture $\mathbf{C}_f^t$. Both coarse and fine textures are then combined to form a body texture, which is blended with other Gaussians through alpha compositing to form the final rendering.

and robust modeling of high-fidelity clothed chest and shoulders with less number of Gaussians.

- We propose a method to remove the MLP in our method at inference time to prevent any costly queries when rendering with novel poses and expressions and reach a rendering speed of around 130 FPS.

The work presented in this chapter produces the following publication:

- Tianhao Wu, Jing Yang, Zhilin Guo, Jingyi Wan, Fangcheng Zhong, and Cengiz Oztireli. Gaussian head shoulders: High fidelity neural upper body avatars with anchor gaussian guided texture warping, 2024. URL https://arxiv.org/abs/2405.12069.

5.1 Method

To study and evaluate the efficient and robust representation of high-frequency texture in image-based reconstructions, we define a specific application scenario as follows: given

a monocular video featuring a talking subject with various expressions and head poses, our goal is to reconstruct a high-fidelity and animatable avatar including the head and clothed upper body. As illustrated in Fig 5.1, our method jointly optimizes 1) a set of standard 3D Gaussians [48] which tightly follow the transformation of 3DMM via LBS to represent the head region, 2) a set of sparse anchor Gaussians spawning over the clothed body, and 3) a learnable neural texture with pose-dependent neural texture warping field constrained by the anchor Gaussians to represent the clothed body with sharp details and high robustness.

5.1.1 FLAME-Driven Head Gaussians

As the face region contains highly distinguishable characteristics and can be described accurately with parametric head 3DMM such as FLAME [56], we directly utilize standard 3D Gaussians that are deformed with parametric 3DMM via neural LBS to represent the head part [140, 138]. Specifically, we learn personalized FLAME expression and pose blendshapes and LBS weights through a small 3D coordinate-based MLP for each Gaussian: $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{W}} = \text{MLP}_d(\boldsymbol{\mu})$, where $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{n_e \times 3}$ are the expression blendshapes, $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{n_p \times 9 \times 3}$ are the pose blendshapes, $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{n_j}$ are the LBS weights corresponding to each of the n_j bones. Following [39], we use the standard skinning function LBS to obtain the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ for each Gaussian, and apply them to get the Gaussian mean $\boldsymbol{\mu}^d$ and covariance $\boldsymbol{\Sigma}^d$ in the 3D view space:

$$\boldsymbol{R}, \boldsymbol{T} = \text{LBS}(\boldsymbol{B}_{\boldsymbol{\mathcal{P}}}(\theta; \boldsymbol{\mathcal{P}}) + \boldsymbol{B}_{\boldsymbol{\mathcal{E}}}(\psi; \boldsymbol{\mathcal{E}}), \mathbf{J}(\psi), \theta, \boldsymbol{\mathcal{W}}), \tag{5.1}$$

$$\boldsymbol{\mu}^d = \boldsymbol{R}\boldsymbol{\mu} + \boldsymbol{T},\ \boldsymbol{\Sigma}^d = \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T, \tag{5.2}$$

where $\mathbf{J}$ is the joint regressor in FLAME, and $\boldsymbol{B}_{\boldsymbol{\mathcal{P}}}$ and $\boldsymbol{B}_{\boldsymbol{\mathcal{E}}}$ are linear combination of blendshapes based on per-frame coefficients θ and ψ that control the head animation. They can then be rendered with a standard Gaussian rasterization pipeline in Eq 2.9.

5.1.2 3D-2D Correspondence via Anchor Gaussians

3D Gaussian architecture based on the Gaussian Splatting method has shown promising performance and robustness in reconstructing 3D geometry and appearance from RGB images. However, they suffer from a significant constraint – each individual Gaussian can only represent a spatially invariant color under a fixed viewing direction, hence a vast number of Gaussians is required to represent objects with detailed textures, regardless of the actual complexity of the geometry. A naive application of Gaussian architecture therefore fails to capture the fine details of the upper body with complex textures and intricate deformation, and results in blurry details and floating artifacts under challenging

poses.

We hence propose to learn a high-frequency texture in canonical texture space, and use a sparse set of Gaussians as anchors to guide the warping between texture space and image plane. As such, we only need a small number of Gaussians and a texture with per-pose warping to represent a clothed body with arbitrarily complicated textures. Since anchor Gaussians themselves do not need to exactly represent the high-frequency appearance, we can model them as a simplified version of regular Gaussians: they only use view-independent RGB colors, are isotropic Gaussians with quaternion fixed at $(1, 0, 0, 0)$, and are excluded from the density control and therefore are not split, cloned, or pruned. To prevent them from becoming trivial in rendering, their opacity and size are clamped to be no smaller than 0.05 and 0.0001 respectively.

The anchor Gaussians are initialized as follows: after a short warm-up period that only trains plain Gaussian, we first reproject all Gaussian means onto the image plane of a canonical training frame, and filter out Gaussians that are located around the head region based on semantic masks. We then use farthest point sampling [86] to select $N_a = 1024$ Gaussians as anchor Gaussians. The first SH basis is converted to RGB values and the anchor scales in three directions are averaged to form a single scale for the anchor Gaussians. We then obtain a sparse set of anchor Gaussians, as well as their projected 2D means $\hat{\mathbf{x}}_i^v$ on the image plane (2D view space) of the canonical frame:

$$\hat{\mathbf{x}}_i^v = \mathbf{P}(\hat{\boldsymbol{\mu}}_i^d), \tag{5.3}$$

where $\mathbf{P}$ is the camera projective transformation, $\hat{\boldsymbol{\mu}}_i^d$ is the 3D Gaussian mean of the i-th anchor Gaussian transformed to 3D view space with LBS. To build the correspondence between anchor Gaussians and texture space coordinates, we assume that the mapping between the 2D image plane of the canonical frame and the texture space is an identity mapping. We can hence define a function $f_{anchor}(i)$ as a fixed correspondence between the i-th 3D anchor Gaussian mean and its target 2D pixel coordinate in texture space:

$$f_{anchor}(i) := \mathbf{I}(\hat{\mathbf{x}}_i^v), \tag{5.4}$$

where $\mathbf{I}$ is the identity function to map 2D image plane coordinates to texture space. Note that $f_{anchor}(i)$ is fixed after initialization and does not update with further optimization of $\hat{\boldsymbol{\mu}}_i$. Such correspondences will later be used to constrain the pose-dependent texture warping, as will be detailed in Sec 5.1.5.

5.1.3 Neural Texture and Texture Warping

We use a trainable neural texture in canonical space with a pose-dependent neural texture warping field to represent the part of the avatar with relatively simple overall geometry and complicated appearances, i.e., the clothed shoulder and chest. In a traditional textured mesh rendering pipeline, the texture is first mapped to the mesh triangles through a pre-defined UV mapping, and the meshes are then rasterized to find the first intersections with the camera rays. Those first intersections therefore establish a mapping between texture space and image plane. However, this approach is not applicable without accurate surfaces and well-defined UV mapping. We instead propose to bypass the intermediate step and learn a per-pose warping that directly maps pixel coordinates on image plane $\mathbf{x}_v$ to the texture coordinates $\mathbf{x}_t$ for texture fetching. Specifically, the warping field is represented using a coordinate-based MLP:

$$\Delta_{\mathbf{x}} = \text{MLP}_w \left(\gamma(\mathbf{x}_v), \gamma(\theta), \gamma(\mathbf{t}), \gamma(\mathbf{x}_{ldmk})\right), \tag{5.5}$$

where γ is the positional encoding [**?**], θ is the FLAME pose parameters containing head and neck rotations, $\mathbf{t}$ is the camera position, $\mathbf{x}_{ldmk}$ is 2D body landmarks for neck, left and right shoulders. The corresponding texture coordinate is obtained as $\mathbf{x}_t = \mathbf{x}_v + \Delta_{\mathbf{x}}$.

Our optimizable texture includes a coarse texture $\mathbf{T}_c$ with 3 channels and a latent texture $\mathbf{T}_f$ with D_t channels. Both textures have sizes of $[H + 2P, W + 2P]$, where H, W are the image height and width, P is the padding size which we empirically set to 50 to account for body parts that move in and out in the video sequence. The latent texture $\mathbf{T}_f$ is passed to an MLP to obtain pose-dependent appearances such as brightness changes on the clothes:

$$\mathbf{C}_f^t(\mathbf{x}_t) = \text{MLP}_f \left(\mathbf{T}_f(\mathbf{x}_t), \gamma(\theta), \gamma(\mathbf{t}), \gamma(\mathbf{x}_{ldmk})\right), \tag{5.6}$$

where $\mathbf{T}_c(\mathbf{x}_t), \mathbf{T}_f(\mathbf{x}_t)$ are coarse and latent texture sampled at 2D coordinate $\mathbf{x}_t$ via bilinear interpolation. The textured pixel color at the coordinate $\mathbf{x}_v$ is therefore obtained as $\mathbf{C}^t(\mathbf{x}_v) = \mathbf{T}_c(\mathbf{x}_t) + \mathbf{C}_f^t(\mathbf{x}_t)$.

By constraining with the correspondences between deformable anchor Gaussians and their fixed projections on 2D texture space, we can learn accurate and effective texture warping for various body movements including translation, rotation, and depth-based (in-and-out) motions; see Fig 5.2.

5.1.4 Rendering

To this end, we have a hybrid representation that includes 3D regular Gaussians that represent the head of the avatar, 3D anchor Gaussians that sparsely span over the body

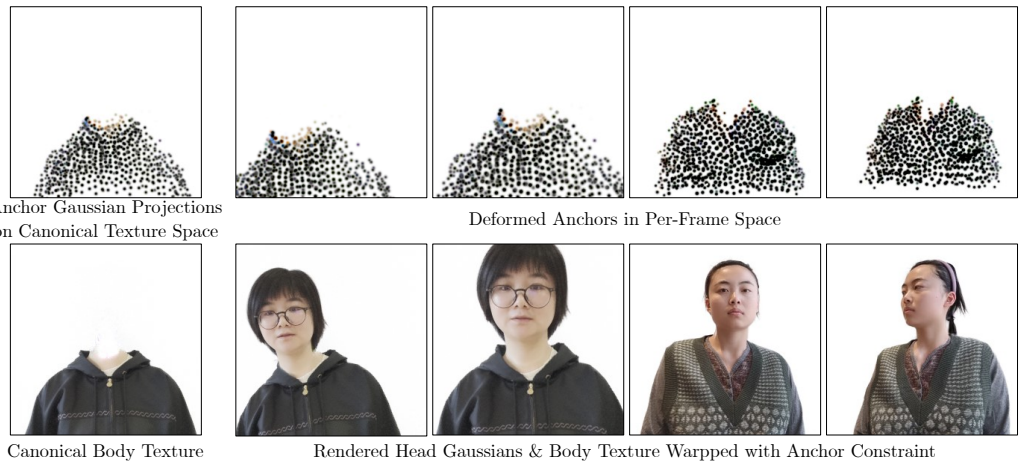


Figure 5.2: **Anchor Warping.** The anchors are initialized with corresponding projections on canonical texture space. When anchors are deformed via LBS to model the per-frame body movement, they map to the same projections in texture space and hence establish correspondences for body texture warping.

region, and a 2D neural texture for the body. To render all of them together for joint optimization, we simply use alpha blending:

$$\mathbf{C}^*(\mathbf{x}_v) = \underbrace{\hat{\mathbf{C}}(\mathbf{x}_v)}_{\text{Anchor Gaussians}} + \underbrace{(1-\hat{\alpha}(\mathbf{x}_v))\mathbf{C}(\mathbf{x}_v)}_{\text{Head Gaussians}} + \underbrace{(1-\hat{\alpha}(\mathbf{x}_v))(1-\alpha(\mathbf{x}_v))\mathbf{C}^t(\mathbf{x}_v)}_{\text{Body Texture}}, \tag{5.7}$$

where $\hat{\mathbf{C}}(\mathbf{x}_v), \mathbf{C}(\mathbf{x}_v)$ are the rendered RGB color of anchor Gaussian and regular Gaussian, $\hat{\alpha}(\mathbf{x}_v), \alpha(\mathbf{x}_v)$ are the total alpha of anchor Gaussian and regular Gaussian at pixel $\mathbf{x}_v$ respectively.

Note that our rendering process always renders anchor Gaussians in front of the regular Gaussians regardless of their actual positions. Though not physically realistic, we designed this rendering order so anchor Gaussians are always non-trivial and never occluded by regular Gaussians.

5.1.5 Optimization

The optimization is split into three different stages: anchor warm-up stage, main optimization stage, and texture refinement stage. In the anchor warm-up stage, neither anchor Gaussians nor body texture is applied, only the regular Gaussians are rendered and optimized. The purpose of this stage is to move Gaussians to roughly spawn over the area of interest including both head and body. At the end of this stage, we initialize anchor Gaussians from regular Gaussians using the method described in Sec 5.1.2. In the second stage, we render all of the regular Gaussians, anchor Gaussians, and the textured body with alpha compositing described in Eq 5.7 and jointly optimize them together. In the last stage, to recover faithful appearance for the body texture and enhance its robustness

under novel poses, we remove anchor Gaussians from the rendering pipeline, i.e., we set $\hat{\mathbf{C}}$ and $\hat{\alpha}$ to 0 in Eq 5.7., and freeze everything else except for the neural texture, texture warping field, and opacity and SH of regular Gaussians.

Following [140, 139], the training losses include standard MSE RGB loss $\mathcal{L}_{\boldsymbol{C}} = MSE(\mathbf{C}^* - \mathbf{C}^{GT})$, and a FLAME regularization that encourages the FLAME blendshapes and LBS weights predicted for each Gaussian stay close to the pseudo ground truth $\tilde{\boldsymbol{\mathcal{E}}}_i, \tilde{\boldsymbol{\mathcal{P}}}_i, \tilde{\boldsymbol{\mathcal{W}}}_i$ obtained from the nearest FLAME vertex:

$$\mathcal{L}_{flame} = \frac{1}{N} \sum_{i=1}^{N+N_a} (\lambda_{\boldsymbol{\mathcal{E}}} |\boldsymbol{\mathcal{E}}_i - \tilde{\boldsymbol{\mathcal{E}}}_i|_2 + \lambda_{\boldsymbol{\mathcal{P}}} |\boldsymbol{\mathcal{P}}_i - \tilde{\boldsymbol{\mathcal{P}}}_i|_2 + \lambda_{\boldsymbol{\mathcal{W}}} |\boldsymbol{\mathcal{W}}_i - \tilde{\boldsymbol{\mathcal{W}}}_i|_2). \tag{5.8}$$

During main optimization stage, we additionally include a VGG feature loss [46, 102] $\mathcal{L}_{VGG} = |\mathbf{F}_{vgg}(\mathbf{C}) - \mathbf{F}_{vgg}(\mathbf{C}^{GT})|$, and a head mask regularization to encourage regular Gaussians to stay only within the head region and allow the body texture to be trained properly without being occluded:

$$\mathcal{L}_{head} = MSE(max(\alpha - \alpha_{head}, 0)), \tag{5.9}$$

where α_{head} is the alpha mask of the head region obtained with matting pre-processing and semantic mask. We also include an L1 regularization on the 2D neural warping field to encourage a clean background to be learned in the neural texture, as well as an L1 loss to slowly decrease the opacity of anchor Gaussians to allow the body texture to be trained properly:

$$\mathcal{L}_{warp} = \frac{1}{HW} \sum_{i=1}^{HW} |\Delta_{\mathbf{x}_i}|, \; \mathcal{L}_{\hat{\alpha}} = \frac{1}{N_a} \sum_{i=1}^{N_a} |\hat{\alpha}_i|. \tag{5.10}$$

Finally, we include an anchor loss as a soft constraint of the per-pose texture warping:

$$\mathcal{L}_{anchor} = \frac{1}{N_a} \sum_{i=1}^{N_a} (f_{anchor}(i) - (\hat{\boldsymbol{x}}_i^v + \Delta_{\hat{\boldsymbol{x}}_i^v}))^2, \tag{5.11}$$

i.e., for each anchor Gaussian, we first transform it to 3D view space via LBS, and then project it onto the image plane to obtain its 2D mean $\hat{\boldsymbol{x}}_i^v$ via Eq 5.3. $\hat{\boldsymbol{x}}_i^v$ is then warped by the neural warping field MLP_w to obtain the corresponding coordinate in the texture space, which is optimized to match the anchor correspondence defined during initialization.

In the third stage, we remove the regularization losses including $\mathcal{L}_{head}, \mathcal{L}_{warp}$ and $\mathcal{L}_{\hat{\alpha}}$.

The total training objectives for each of the three stages are as follows:

$$\mathcal{L}_1 = \mathcal{L}_{\boldsymbol{C}} + \mathcal{L}_{flame}, \tag{5.12}$$

$$\mathcal{L}_2 = \mathcal{L}_1 + \lambda_{VGG}\mathcal{L}_{VGG} + \lambda_{head}\mathcal{L}_{head} + \lambda_{warp}\mathcal{L}_{warp} + \lambda_{\hat{\alpha}}\mathcal{L}_{\hat{\alpha}} + \lambda_{anchor}\mathcal{L}_{anchor},$$

$$\mathcal{L}_3 = \mathcal{L}_1 + \lambda_{VGG}\mathcal{L}_{VGG} + \lambda_{anchor}\mathcal{L}_{anchor}. \tag{5.13}$$

5.1.6 Accelerated Rendering with No MLP Queries

One of the main advantages of Gaussian architecture is its highly efficient rendering speed, which enables many real-time and interactive applications. To take full use of this advantage, we propose an accelerated version of our method that requires no MLP queries at inference time. Specifically, after training the model, we first cache the output of MLP_d for all head Gaussians and anchor Gaussians, then cache the view-dependent fine texture by querying the fine texture MLP MLP_f conditioned on the same canonical training frame which was previously used to initialize the anchor Gaussians. The queried fine texture colors are added to the coarse color to make a non-neural RGB texture. To deal with potential noise created by the fine texture MLP at the corners of the texture, we use an off-the-shelf background segmentation network [11] to compute a coarse mask and clean all the pixels outside of the mask; we show the necessity of this step in the supplementary. To replace the neural warping field MLP_w that warps image plane coordinates to texture space, we rely on the correspondence between anchor Gaussians and texture space coordinates to estimate a homography at inference time. Specifically, we first project all anchor Gaussians to the image plane of the canonical training frame, and then remove any anchor Gausians that go beyond the view frustum. To deal with any potential discrepancy between the neural warping field and the anchor correspondences, we update those correspondences based on the prediction of the neural warping field on the current frame:

$$f_{anchor}(i) := \hat{\mathbf{x}}_v^i + \Delta_{\hat{\mathbf{x}}_v^i}. \tag{5.14}$$

After that, we randomly select 100 training frames and use RANSAC [23] to estimate a homography between the image plane coordinates of anchor Gaussians and their corresponding texture space coordinates, and remove anchor Gaussians that are considered outliers by RANSAC. This effectively removes any anchor deformation that cannot be described by the rigid transformation. Finally, at inference time, we perform LBS on regular head Gaussians and anchor Gaussians. Based on the image plane coordinates of the anchor Gaussians $\hat{\mathbf{x}}_v^i$ and their correspondences f_{anchor}, we compute a homography with the least square error via singular value decomposition. The estimated transformation is applied to all pixels on the image plane to find the corresponding non-neural texture, which is then blended with the head Gaussians to form the final rendering. This accelerated

	full			head		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
INSTA	20.59	.781	.236	29.80	.936	.059
SplattingAvatar	21.00	.774	.274	28.29	.925	.069
PointAvatar	25.21	.851	.102	30.64	.943	.042
FlashAvatar	21.88	.808	.127	30.33	.946	.039
GS*	25.94	.854	.095	31.81	.947	.036
Ours	26.80	.875	.070	33.06	.959	.028
Ours (No MLP)	25.47	.859	.074	32.54	.956	.029

Table 5.1: **Quatitative evaluation of self-reenactment task** We color the best and second-best methods. Our full method achieves much better performance compared to existing baselines. While Ours (No MLP) achieves slightly lower PSNR, which is known to be over-sensitive to small misalignments and prefers blurry results [79], we show it achieves better LPIPS than existing methods.

	FPS	#GS	FPS	#GS
	003		004	
FlashAvatar	134	13453	137	13453
GS*	**141**	163830	**159**	125521
Ours	70	58701	71	39549
Ours (No MLP)	129	58701	134	39549
	005		007	
FlashAvatar	125	13453	126	13453
GS*	96	317968	**131**	191431
Ours	72	50708	69	52910
Ours (No MLP)	**132**	50708	127	52910

Table 5.2: **Performance measure.** We report rendering FPS and the number of Gausssians for each method. The rendering speed of our no MLP version surpasses pure Gaussian implementation for subjects wearing extremely high-frequency cloth.

inference approach effectively increases the rendering speed from around 70 FPS to 130 FPS.

5.2 Evaluation

5.2.1 Datasets

We evaluate different methods on 1 mobile phone sequence from PointAvatar [140], 2 internet sequences from Head2Head dataset [51], and 4 sequences captured with mobile phones. All sequences are preprocessed with DECA [22] and a slightly modified landmark fitting process from IMAvatar [139]. Additionally, we use DWPose [128] to predict 2D landmarks for nose, neck and shoulders, which are then smoothed with One Euro Filter [9].

5.2.2 Baselines

We compare our method with four neural head avatar methods based on various representations, including (1) INSTA [141], which employs a latent hash grid [71] combined with NeRF [70], (2) PointAvatar [140], which is based on isotropic point clouds, (3) SplattingAvatar [101], which utilizes Gaussian Splatting attached to local space of 3DMM meshes, and (4) GS*, a baseline we implemented by changing the point cloud representation in PointAvatar to Gaussian Splatting, which is similarly deformed via neural LBS.

5.2.3 Self-Reenactment

We show the quantitative and qualitative results of the self-reenactment task in Tab 5.1 and Fig 5.3. Our full version demonstrates superior reconstruction performance compared to existing baselines, especially for subjects with intricate cloth textures. Our No MLP version does not consistently achieve better PSNR when compared to existing baselines, as it is unable to render pose-dependent appearance changes and intricate cloth deformation. However, we note that it consistently achieves better LPIPS, demonstrating that our No MLP version can still generate realistic and faithful renderings. This discrepancy among different metrics arises because of the high sensitivity of PSNR to small misalignments in the cloth texture [79]. As a result, PSNR tends to prefer blurry reconstruction over sharp but slightly misaligned results. Notably, although we did not include specific treatments for the head region, better modeling of the body also leads to better face reconstruction. The qualitative evaluation in Fig 5.3 demonstrates that both versions of our method can learn sharper and more robust body texture compared to existing methods.

5.2.4 Cross-Reenactment

For the cross-identity reenactment task, we render the reconstruction of the original identity with FLAME expressions and poses from the source subject. With the full version of our method, we apply an additional Euclidean transformation after warping the image plane coordinates with the MLP. This is to ensure the body texture is always aligned with the head Gaussians under novel poses; see Fig 5.6. The Euclidean transformation is simply determined by fitting the MLP warped image plane coordinates of the anchor Gaussians and their target coordinates in the texture space. To deal with potential artifacts caused by coordinates warped to unseen corner parts in the texture, we apply the same appearance distillation process and remove the fine texture MLP. The No MLP version is applied the same way as in the self-reenactment task.

In addition to the improvement over the body texture, we observe that avatars reconstructed with our approach often give more accurate and faithful expression control, as shown in Fig 5.4. We deduce that this is because the 3DMM-driven Gaussians only need to model the head region, leading to a more accurate reconstruction of the head model and more reliable LBS weights and expression and pose blendshapes predicted by the LBS network.

5.2.5 Comparison with Full Body Avatars

To verify our choice of driving anchor Gaussians only with head 3DMM (FLAME), we select two subjects that show a larger portion of the upper body and compare our method with GSAvatar, a Gaussian Splatting based full body neural avatar methods that deform

	004			007		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
GSAvatar	17.08	.811	.178	16.64	.744	.143
Ours	26.82	.887	.094	23.87	.885	.052
Ours No MLP	26.70	.885	.094	22.43	.861	.056

Table 5.3: **Body Only Quantitative Comparison with Full Body Avatars.** We show that existing full body neural avatar methods that rely on SMPL deformation perform significantly worse than our methods. Metrics are computed after masking out the background and head regions.

the representation based on SMPL [38]. As the code release of GSAvatar only supports SMPL instead of SMPLX, we simply use semantic masks to remove the head region during the training and compare only the reconstruction quality of the body part. As shown in Tab 5.3 and Fig 5.5, since the existing SMPL tracking methods for monocular videos are developed only for views that include the whole body, the fitted SMPL is significantly misaligned with the GT [105], even after fine-tuning during Gaussian optimization. As a result, the clothed body reconstructed by GSAvatar presents several artifacts under novel poses and are significantly misaligned the GT. Our method is able to reconstruct the chest and shoulders with much better quality and accuracy. We would also like to note that, although we do not include body 3DMM in our method, due to the usage of virtual static bone, technically speaking, the effect is exactly the same as have a SMPLX 3DMM where the body and hand parts (SMPLX and MANO) are kept static during the whole sequences.

5.2.6 Ablation

We show the effectiveness of the anchor constraint $\mathcal{L}_{anchor}$, test-time Euclidean transformation and warp loss $\mathcal{L}_{warp}$ in Fig 5.6 and Table 5.4. Even for subjects with only slight movement in the upper body, anchor constraint is still needed to learn sharp and accurate cloth texture. Besides, without anchor Gaussians and test time Euclidean transformation, the body texture is unable to align with the head Gaussians under novel poses. The warp loss $\mathcal{L}_{warp}$ is needed to prevent the neural warping field from mapping the background pixel to an arbitrary white pixel in the texture space. As anchor Gaussians only exist within the body region, the additional Euclidean transformation computed from anchor correspondences would significantly distort the background pixels, causing severe artifacts as shown in Fig 5.6 (b).

	002			005			007		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
No Anchor Loss	24.96	.910	.088	22.91	.854	.117	19.30	.773	.134
No Warp Loss	32.86	.949	.041	24.19	.891	.081	22.74	.848	.076
Ours	31.98	.949	.042	24.48	.895	.074	23.26	.856	.074

Table 5.4: **Quatitative ablation.** We show the anchor constraint is necessary for learning sharp and correct body texture. While the warp loss might not necessarily improve the performance for the self-reenactment task, it is needed for cross-reenactment with out-of-distribution poses.

5.2.7 Rendering Efficiency

We report the number of Gaussians and the rendering speed for pure Gaussian implementation GS*, Ours, and Ours (No MLP) in Tab 5.2. The rendering speeds are tested on an RTX4080 Ti. For subjects wearing complicated clothes, the number of Gaussians required to model the high-frequency cloth texture significantly increases for pure Gaussian implementation, hence slowing down the rendering speed, whereas our method only models the head region with Gaussians and hence requires a much fewer number of Gaussians. The rendering speed of our no MLP version even surpasses pure Gaussian implementation for subject 005, who wears cloth with a very high-frequency texture.

5.3 Summary

In this chapter, we aim to study the problem of efficient and robust representation of high-frequency appearance in image-based 3D reconstruction. We present Gaussian Head & Shoulders, a method that reconstructs high-quality and animatable upper body avatars including head, chest, and shoulders as a specific task involving both high and low frequency appearance. By utilizing high-frequency neural texture to represent the clothed body, we are able to model sharp and robust high-frequency cloth details and significantly reduce the number of Gaussians needed to represent a subject. By constraining the texture warping with a sparse set of anchor Gaussians, the body texture is accurately mapped to the correct position even under unseen poses. By caching the neural texture and replacing the neural warping field with a projective transformation estimated using anchor correspondences, we significantly improve rendering speed and reach over 130 FPS at novel poses, surpassing the rendering speed of pure Gaussian implementation for subjects with complicated cloth textures.

Figure 5.3: **Qualitative comparison of self-reenactment task.** We show that both of our full version and No MLP version can recover a more accurate and robust body texture, even under extreme poses and high-frequency cloth textures.

Figure 5.4: **Qualitative evaluation of cross-identity reenactment.** Our method leads to both better cloth texture and more accurate expression, as LBS network only focuses on the head region in our approach.

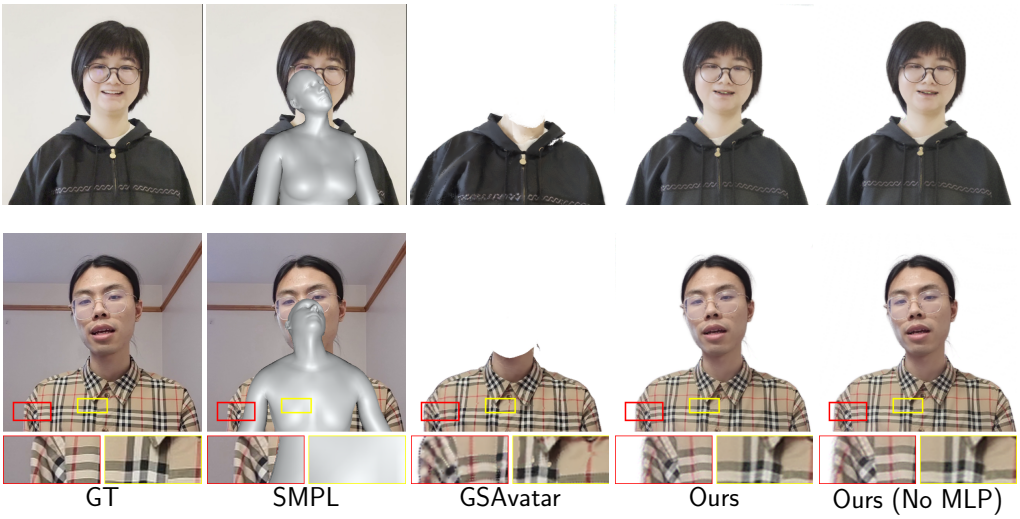


Figure 5.5: **Qualitative comparison with full body avatar methods.** Due to the limited landmarks available on the shoulders and chest, existing SMPL tracking methods fail to obtain correct SMPL parameters. Fully body neural avatars that rely on SMPL hence fail to learn accurate and robust body. While our method does not include SMPL 3DMM, the use of static virtual bone and neural texture warping allow us to learn the body texture accurately.

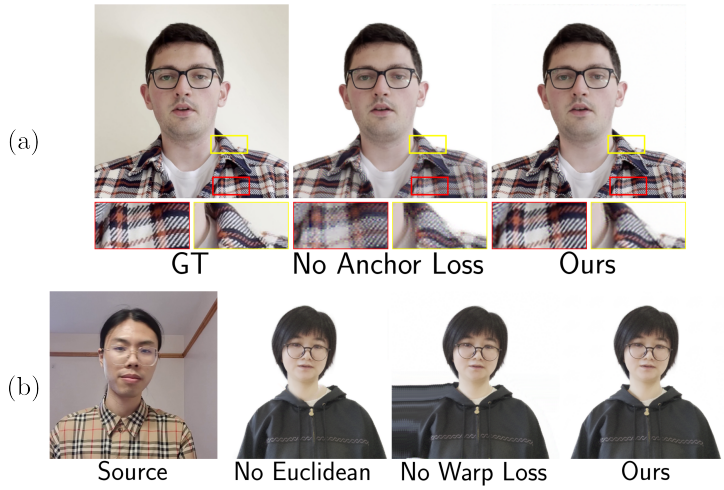


Figure 5.6: **Qualitative ablation for self-reenactment (a) and cross-reenactment (b).**

Chapter 6

Conclusion and Future Work

In this dissertation, we present a comprehensive overview of 3D reconstruction from RGB images with neural implicit representations and propose several advancements to achieve efficient and robust reconstruction in diverse and challenging real-world scenarios. Recent advancements in neural implicit representations and differentiable rendering have significantly improved the performance of image-based 3D reconstruction, positioning it as a key technology for accessible asset scanning in Metaverse and other graphical applications. Through this thesis, we demonstrate that with correctly and specifically designed priors encoded in all of the representation, architecture, and optimization, it is possible to address various challenging and severely ill-posed obstacles in real-world applications. These include reconstructing translucent surfaces, removing dynamic occluders in the capture, and efficiently modeling high-frequency appearance. From a research perspective, 3D reconstruction serves as a critical bridge between real-world observations and digital assets or computational models, forming the foundation for emerging fields such as scene understanding and the development of large-scale world foundation models. Continuous exploration of reliable and robust reconstruction approaches remains essential to further advancements in both application and research domains.

Given the complexities and numerous challenges inherent in 3D reconstruction, this dissertation represents merely the beginning of exploration in this field. Many research questions remain to be addressed in the future. While specific architecture designs have proven effective for modeling translucent surfaces, the explicit grid-based approaches often result in reconstructed surfaces that are less smooth compared to those produced by MLP-based or hash-grid-based methods. This disparity arises from the lack of spatial smoothness encoded as an inductive bias in the MLP architecture. As a result, extensive regularization terms are required to enhance surface smoothness, yet these methods still struggle to balance smoothness with the preservation of fine, thin structures. Developing more adaptive approaches to enforce surface smoothness while retaining thin and intricate structures remains an active area of research. Furthermore, our exploration into efficiently

representing high-frequency appearances has primarily focused on forward-facing scenes. While the proposed anchor Gaussians and neural warping effectively handle minor viewpoint changes and complex deformations in neural avatars, extending these techniques to 360-degree scenes poses additional challenges. One promising direction could involve leveraging existing surface reconstruction methods to simultaneously recover an SDF representation alongside Gaussian modeling. This approach could allow for the identification of simple surface structures, enabling their replacement with textured meshes for improved efficiency and accuracy. Such modifications could help bridge the gap between specialized applications and more generalizable solutions in 3D reconstruction.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds, 2018.

[2] David Adalsteinsson and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1995.1098. URL `https://www.sciencedirect.com/science/article/pii/S0021999185710984`.

[3] Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] Dejan Azinovic, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation, 03 2019.

[5] T. Bagautdinov, C. Wu, J. Saragih, P. Fua, and Y. Sheikh. Modeling facial geometry using compositional vaes. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3877–3886, 2018. doi: 10.1109/CVPR.2018.00408.

[6] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2007.09.014. URL `https://www.sciencedirect.com/science/article/pii/S1077314207001555`. Similarity Matching in Computer Vision and Multimedia.

[8] Emilio Camahort, Apostolos Lerios, and Donald Fussell. Uniformly sampled light fields. pages 117–130, 01 1998. doi: 10.1007/978-3-7091-6453-2_11.

[9] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 2527–2530, New York,

NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310154. doi: 10.1145/2207676.2208639. URL https://doi.org/10.1145/2207676.2208639.

[10] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[11] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. 06 2017.

[12] Xiaoxue Chen, Junchen Liu, Hao Zhao, Guyue Zhou, and Ya-Qin Zhang. Nerrf: 3d reconstruction and view synthesis for transparent and specular objects with neural refractive-reflective fields, 2023.

[13] Yufan Chen, Lizhen Wang, Qijing Li, Hongjiang Xiao, Shengping Zhang, Hongxun Yao, and Yebin Liu. Monogaussianavatar: Monocular gaussian point-based head avatar. *arXiv*, 2023.

[14] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

[15] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[16] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[17] Miles Detrixhe, Frédéric Gibou, and Chohong Min. A parallel fast sweeping method for the eikonal equation. *Journal of Computational Physics*, 237:46–55, 2013. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2012.11.042. URL https://www.sciencedirect.com/science/article/pii/S002199911200722X.

[18] Helisa Dhamo, Yinyu Nie, Arthur Moreau, Jifei Song, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. Headgas: Real-time animatable head avatars via 3d gaussian splatting, 2023.

[19] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[20] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image, 2016.

[21] Jiemin Fang, Lingxi Xie, Xinggang Wang, Xiaopeng Zhang, Wenyu Liu, and Qi Tian. Neusample: Neural sample field for efficient view synthesis. *arXiv:2111.15552*, 2021.

[22] Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. volume 40, 2021. URL `https://doi.org/10.1145/3450626.3459936`.

[23] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `https://doi.org/10.1145/358669.358692`.

[24] Qiancheng Fu, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *NeurIPS*, 2022.

[25] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2010. doi: 10.1109/CVPR.2010.5539802.

[26] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. ISSN 1572-2740. doi: 10.1561/0600000052. URL `http://dx.doi.org/10.1561/0600000052`.

[27] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8649–8658, June 2021.

[28] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[29] Xuan Gao, Chenglai Zhong, Jun Xiang, Yang Hong, Yudong Guo, and Juyong Zhang. Reconstructing personalized semantic facial nerf models from monocular video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 41(6), 2022. doi: 10.1145/3550454.3555501.

[30] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. Unsupervised training for 3d morphable model regression. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8377–8386, 2018.

[31] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape, 2019. URL `https://arxiv.org/abs/1912.06126`.

[32] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions, 2019. URL `https://arxiv.org/abs/1904.06447`.

[33] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[34] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[35] Herman Hansson Söderlund, Alex Evans, and Tomas Akenine-Möller. Ray tracing of signed distance function grids. *Journal of Computer Graphics Techniques (JCGT)*, 11 (3):94–113, September 2022. ISSN 2331-7418. URL `http://jcgt.org/published/0011/03/06/`.

[36] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[37] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9984–9993, 2019.

[38] Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[39] Shoukang Hu and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. *arXiv preprint arXiv:*, 2023.

[40] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis, 2021. URL `https://arxiv.org/abs/2104.00677`.

[41] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories, 2021. URL `https://arxiv.org/abs/2109.01750`.

[42] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[43] Zhang Jiakai, Liu Xinhang, Ye Xinyi, Zhao Fuqiang, Zhang Yanshun, Wu Minye, Zhang Yingliang, Xu Lan, and Yu Jingyi. Editable free-viewpoint video using a layered neural representation. In *ACM SIGGRAPH*, 2021.

[44] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1251–1261, 2020.

[45] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[46] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46475-6.

[47] James Kajiya and Brian von herzen. Ray tracing volume densities. *ACM SIGGRAPH Computer Graphics*, 18:165–174, 07 1984. doi: 10.1145/964965.808594.

[48] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

[49] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. *arXiv.org*, 2021.

[50] Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592455. URL https://doi.org/10.1145/3592455.

[51] M. Koujan, M. Doukas, A. Roussos, and S. Zafeiriou. Head2head: Video-based neural head synthesis. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020) (FG)*, pages 319–326, Los Alamitos, CA, USA, may 2020. IEEE Computer Society. doi: 10.1109/FG47880.2020.00048. URL https://doi.ieeecomputersociety.org/10.1109/FG47880.2020.00048.

[52] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[53] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 31–42, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237199. URL https://doi.org/10.1145/237170.237199.

[54] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.

[55] Hai Li, Xingrui Yang, Hongjia Zhai, Yuqian Liu, Hujun Bao, and Guofeng Zhang. Vox-surf: Voxel-based implicit surface representation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–12, 2022. doi: 10.1109/TVCG.2022.3225844.

[56] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. URL https://doi.org/10.1145/3130800.3130813.

[57] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2021. URL https://arxiv.org/abs/2103.02597.

[58] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023.

[59] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[60] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images, 2020.

[61] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.*, 37(6), December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275047. URL `https://doi.org/10.1145/3272127.3275047`.

[62] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields, 2021.

[63] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7708–7717, 2019.

[64] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019.

[65] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering, 2021. URL `https://arxiv.org/abs/2103.01954`.

[66] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[67] Matthew M. Loper and Michael J. Black. Opendr: An approximate differentiable renderer. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 154–169, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10584-0.

[68] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[69] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

[70] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[71] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL `https://doi.org/10.1145/3528223.3530127`.

[72] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pages 7220–7229. PMLR, 2020.

[73] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. ISSN 1467-8659. doi: 10.1111/cgf.14340. URL `https://doi.org/10.1111/cgf.14340`.

[74] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs, 2021. URL `https://arxiv.org/abs/2112.00724`.

[75] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[76] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[77] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[78] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021.

[79] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021.

[80] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[81] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34, 2021.

[82] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[83] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007. ISBN 0521880688. URL `http://www.amazon.com/Numerical-Recipes-3rd-Scientific-Computing/dp/0521880688/ref=sr_1_1?ie=UTF8&s=books&qid=1280322496&sr=8-1`.

[84] Simon J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, USA, 1st edition, 2012. ISBN 1107011795.

[85] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[86] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

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

[88] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[89] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. Lolnerf: Learn from one look. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL https://arxiv.org/abs/2111.09996.

[90] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021. URL https://arxiv.org/abs/2103.13744.

[91] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 765–773, 2015.

[92] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017.

[93] Riccardo Roveri, A Cengiz Öztireli, Ioana Pandele, and Markus Gross. Pointpronets: Consolidation of point clouds with convolutional neural networks. In *Computer Graphics Forum*, volume 37, pages 87–99. Wiley Online Library, 2018.

[94] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, and Richard Newcombe. Frodo: From detections to 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[95] Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. Relightable gaussian codec avatars. In *CVPR*, 2024.

[96] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[97] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[98] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[99] J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999. doi: 10.1137/S0036144598347059. URL https://doi.org/10.1137/S0036144598347059.

[100] James A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93 4:1591–5, 1996.

[101] Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting. In *Computer Vision and Pattern Recognition (CVPR)*, 2024.

[102] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

[103] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*, 2021.

[104] Mohammed Suhail, Leonid Sigal Carlos Esteves, and Ameesh Makadia. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[105] Yu Sun, Qian Bao, Wu Liu, Yili Fu, Black Michael J., and Tao Mei. Monocular, One-stage, Regression of Multiple 3D People. In *ICCV*, 2021.

[106] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[107] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering, 2020.

[108] Vadim Tschernezki, Diane Larlus, and Andrea Vedaldi. NeuralDiff: Segmenting 3D objects that move in egocentric videos. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2021.

[109] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[110] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022.

[111] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4): 125:1–125:18, July 2022. doi: 10.1145/3528223.3530139.

[112] Jie Wang, Jiu-Cheng Xie, Xianyan Li, Feng Xu, Chi-Man Pun, and Hao Gao. Gaussianhead: High-fidelity head avatars with learnable gaussian derivation, 2024.

[113] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images, 2018.

[114] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[115] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 06 2014. doi: 10.1109/CVPRW.2014.126.

[116] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. *arXiv preprint arXiv:2206.07850*, 2022.

[117] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216.

[118] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollh¨ofer. Learning compositional radiance fields of dynamic human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2021.

[119] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7467–7477, 2020.

[120] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D^2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *NeurIPS*, 2022.

[121] Tianhao Wu, Jing Yang, Zhilin Guo, Jingyi Wan, Fangcheng Zhong, and Cengiz Oztireli. Gaussian head shoulders: High fidelity neural upper body avatars with anchor gaussian guided texture warping, 2024. URL https://arxiv.org/abs/2405.12069.

[122] Tianhao Walter Wu, Hanxue Liang, Fangcheng Zhong, Gernot Riegler, Shimon Vainer, Jiankang Deng, and Cengiz Oztireli. Alphasurf: Implicit surface reconstruction for semi-transparent and thin objects with decoupled geometry and opacity.

[123] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021.

[124] Jun Xiang, Xuan Gao, Yudong Guo, and Juyong Zhang. Flashavatar: High-fidelity head avatar with efficient gaussian embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[125] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 492–502. Curran Associates, Inc., 2019.

[126] Yuelang Xu, Lizhen Wang, Xiaochen Zhao, Hongwen Zhang, and Yebin Liu. Avatarmav: Fast 3d head avatar reconstruction using motion-aware neural voxels. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.

[127] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *ICCV*, 2021.

[128] Zhendong Yang, Ailing Zeng, Chun Yuan, and Yu Li. Effective whole-body pose estimation with two-stages distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4210–4220, 2023.

[129] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction with implicit lighting and material, 2020.

[130] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[131] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[132] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020.

[133] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks, 2021. URL `https://arxiv.org/abs/2112.05131`.

[134] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13144–13152, 2021.

[135] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[136] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[137] Hongkai Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74: 603–627, 2004.

[138] Zhongyuan Zhao, Zhenyu Bao, Qing Li, Guoping Qiu, and Kanglin Liu. Psavatar: A point-based morphable shape model for real-time head avatar animation with 3d gaussian splatting, 2024.

[139] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. I M Avatar: Implicit morphable head avatars from videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[140] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J. Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[141] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant volumetric head avatars. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4574–4584, 2022. URL `https://api.semanticscholar.org/CorpusID:253761096`.

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